Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2877430 A1 2013/12/27

(21) 2 877 430

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) **Date de dépôt PCT/PCT Filing Date:** 2013/06/21
(87) **Date publication PCT/PCT Publication Date:** 2013/12/27
(85) **Entrée phase nationale/National Entry:** 2014/12/19
(86) **N° demande PCT/PCT Application No.:** EP 2013/062982
(87) **N° publication PCT/PCT Publication No.:** 2013/190085
(30) **Priorité/Priority:** 2012/06/21 (US61/662,812)

(51) **Cl.Int./Int.Cl.** ***G06F 19/24*** (2011.01)

(71) **Demandeurs/Applicants:**
PHILIP MORRIS PRODUCTS S.A., CH;
XIANG, YANG, CH;
HOENG, JULIA, CH;
MARTIN, FLORIAN, CH

(72) **Inventeurs/Inventors:**
XIANG, YANG, CH;
HOENG, JULIA, CH;
MARTIN, FLORIAN, CH

(74) **Agent:** GOWLING LAFLEUR HENDERSON LLP

(54) **Titre : SYSTEMES ET PROCEDES POUR GENERER DES SIGNATURES DE BIOMARQUEURS AU MOYEN D'ENSEMBLES DOUBLES INTEGRES ET DE TECHNIQUES D'ANNELAGE SIMULEES**

(54) **Title: SYSTEMS AND METHODS FOR GENERATING BIOMARKER SIGNATURES WITH INTEGRATED DUAL ENSEMBLE AND GENERALIZED SIMULATED ANNEALING TECHNIQUES**

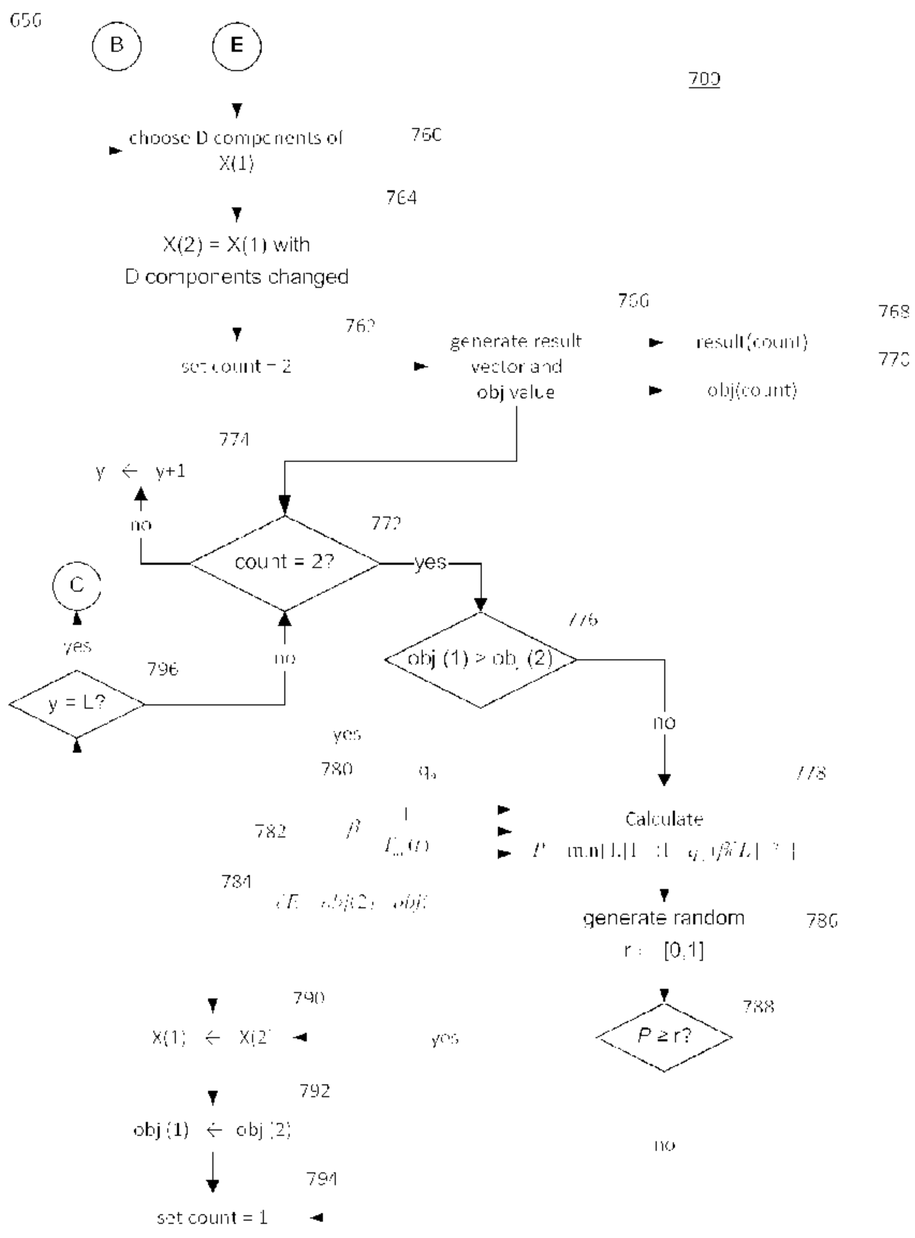

FIG. 7

(57) **Abrégé/Abstract:**
Described herein are systems and methods for classifying a data set using an ensemble classification technique. Classifiers are iteratively generated by applying machine learning techniques to a training data set, and training class sets are generated by

(21) **2 877 430**

(13) **A1**

(57) **Abrégé(suite)/Abstract(continued):**
classifying the elements in the training data set according to the classifiers. Objective values are computed based on the training class sets, and objective values associated with different classifiers are compared until a desired number of iterations is reached, and a final training class set is output.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 December 2013 (27.12.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/190085 A1

(51) International Patent Classification:
G06F 19/24 (2011.01)

(21) International Application Number:
PCT/EP2013/062982

(22) International Filing Date:
21 June 2013 (21.06.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/662,812 21 June 2012 (21.06.2012) US

(71) Applicant *(for all designated States except US)*: **PHILIP MORRIS PRODUCTS S.A.** [CH/CH]; Quai Jeanrenaud 3, CH-2000 Neuchatel (CH).

(72) Inventors; and

(71) Applicants : **XIANG, Yang** [SG/CH]; Rue du Rocher 24, CH-2000 Neuchâtel (CH). **HOENG, Julia** [DE/CH]; Grand-Rue 35, CH-2035 Corcelles (CH). **MARTIN, Florian** [CH/CH]; Chemin de l'Orée 1, CH-2034 Peseux (CH).

(74) Agent: **QUINLAN, Angela**; 27 Clyde Road, Dublin, 4 (IE).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(54) Title: SYSTEMS AND METHODS FOR GENERATING BIOMARKER SIGNATURES WITH INTEGRATED DUAL ENSEMBLE AND GENERALIZED SIMULATED ANNEALING TECHNIQUES

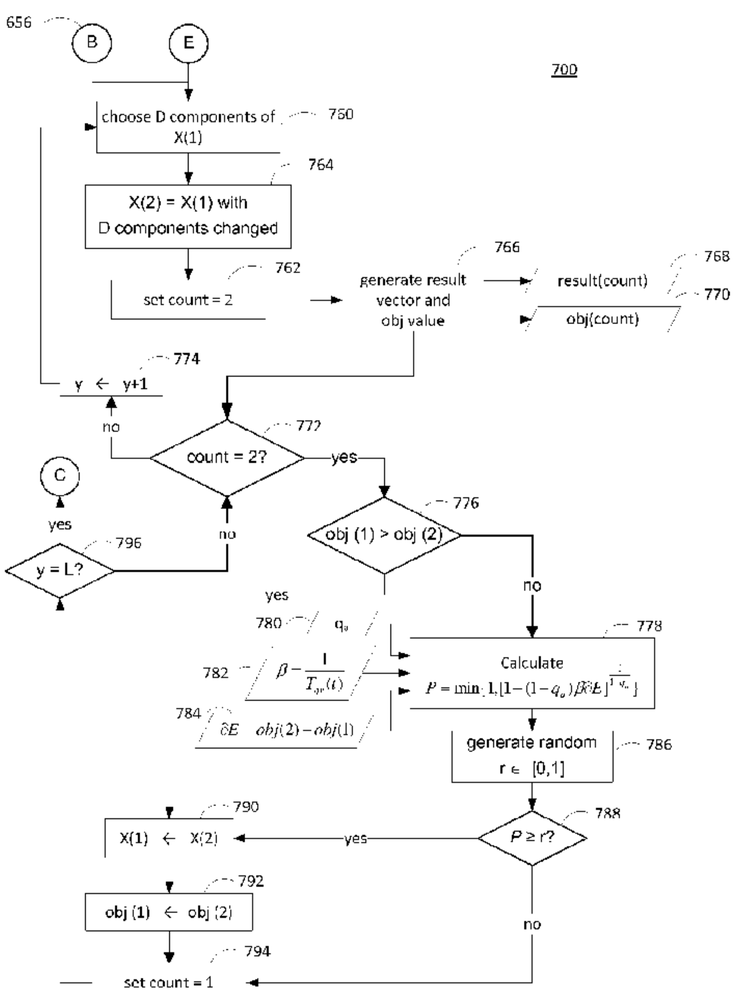

FIG. 7

(57) Abstract: Described herein are systems and methods for classifying a data set using an ensemble classification technique. Classifiers are iteratively generated by applying machine learning techniques to a training data set, and training class sets are generated by classifying the elements in the training data set according to the classifiers. Objective values are computed based on the training class sets, and objective values associated with different classifiers are compared until a desired number of iterations is reached, and a final training class set is output.

WO 2013/190085 A1

SYSTEMS AND METHODS FOR GENERATING BIOMARKER SIGNATURES WITH INTEGRATED DUAL ENSEMBLE AND GENERALIZED SIMULATED ANNEALING TECHNIQUES

Reference to Related Applications

This application claims priority under 35 U.S.C. § 119 to U.S. Provisional Patent Application No. 61/662,812, entitled "Systems and Methods for Generating Biomarker Signatures with Integrated Dual Ensemble and Generalized Simulated Annealing Techniques," filed June 21, 2012, which is incorporated herein in its entirety.

Background

In the biomedical field it is important to identify substances that are indicative of a specific biological state, namely biomarkers. As new technologies of genomics and proteomics emerge, biomarkers are becoming more and more important in biological discovery, drug development and health care. Biomarkers are not only useful for diagnosis and prognosis of many diseases, but also for understanding the basis for development of therapeutics. Successful and effective identification of biomarkers can accelerate the new drug development process. With the combination of therapeutics with diagnostics and prognosis, biomarker identification will also enhance the quality of current medical treatments, thus play an important role in the use of pharmacogenetics, pharmacogenomics and pharmacoproteomics.

Genomic and proteomic analysis, including high throughput screening, supplies a wealth of information regarding the numbers and forms of proteins expressed in a cell and provides the potential to identify for each cell, a profile of expressed proteins characteristic of a particular cell state. In certain cases, this cell state may be characteristic of an abnormal physiological response associated with a disease. Consequently, identifying and comparing a cell state from a patient with a disease to that of a corresponding cell from a normal patient can provide opportunities to diagnose and treat diseases.

These high throughput screening techniques provide large data sets of gene expression information. Researchers have attempted to develop methods for organizing these data sets into patterns that are reproducibly diagnostic for diverse populations of individuals. One approach has been to pool data from multiple sources to form a combined data set and then to divide the data set into a discovery/training set and a test/validation set. However, both transcription

profiling data and protein expression profiling data are often characterized by a large number of variables relative to the available number of samples.

Observed differences between expression profiles of specimens from groups of patients or controls are typically overshadowed by several factors, including biological variability or unknown sub-phenotypes within the disease or control populations, site-specific biases due to difference in study protocols, specimens handling, biases due to differences in instrument conditions (e.g., chip batches, etc), and variations due to measurement error. Some techniques attempt to correct to for bias in the data samples (which may result from, for example, having more of one class of sample represented in the data set than another class).

Several computer-based methods have been developed to find a set of features (markers) that best explain the difference between the disease and control samples. Some early methods included statistical tests such as LIMMA, the FDA approved mammaprint technique for identifying biomarkers relating to breast cancer, logistical regression techniques and machine learning methods such as support vector machines (SVM). Generally, from a machine learning perspective, the selection of biomarkers is typically a feature selection problem for a classification task. However, these early solutions faced several disadvantages. The signatures generated by these techniques were often not reproducible because the inclusion and exclusion of subjects can lead to different signatures. These early solutions also generated many false positive signatures and were not robust because they operated on datasets having small sample sizes and high dimensions.

Accordingly there is a need for improved techniques for identifying biomarkers for clinical diagnosis and/or prognosis, and more generally, for identifying data markers that can be used to classify elements in a data set into two or more classes.

Summary

Described herein are systems, computer program products and methods for identifying data markers that can be used to classify elements in a data set into two or more classes. In particular, Applicants have recognized that a combination of methods and gene set data can provide better prediction of test data than an individual method alone. The computer systems and computer program products described herein implement methods that include one or more such techniques for classifying elements into two or more classes. In particular, biomarker

signatures are generated using integrated dual ensemble and simulated annealing techniques. The techniques involve resampling a data set and predicting phenotypes using a dual ensemble method. In particular, the systems, computer program products, and methods described herein include forming a random vector indicative of a set of classification methods and data samples. . The random vector is iteratively perturbed, and different objective values are computed that correspond to the different perturbations.

In certain aspects, the systems and methods described herein include means and methods for classifying a data set into two or more classes executed by a processor. The methods may comprise receiving a training data set. The training data set may be determined by separating an aggregate data set into a discovery (training) set and a validation (test) set. For example, the aggregate data set may include data that is pooled together from multiple sources, and the aggregate data set may be randomly split into training and test data sets. The methods may further comprise generating a first classifier for the training data set by applying a first machine learning technique to the training data set. For example, the machine learning technique may correspond to support vector machines (SVM) or any suitable technique for feature selection. A first training class set is generated by classifying the elements in the training data set according to the first classifier. In particular, the first classifier may correspond to a classification rule that assigns each sample in a data set to a physiological state (such as diseased or disease-free, for example). The first classifier may combine multiple classification methods such as SVN, network-based SVMs, neural network-based classifiers, logistic regression classifiers, decision tree-based classifiers, classifiers using a linear discriminant analysis technique, a random-forst analysis technique, any other suitable classification method, or any combination thereof.

A first objective value is computed based on the training class set. In particular, a binary generalized simulated annealing method may be used to compute the objective value. A random vector may include as its elements a set of parameters that define the classification technique to be used. The technique defined by the random vector is used to compute the first objective value. Then, for a plurality of iterations, a second machine learning technique is applied to the training data set to generate a second classifier for the training data set, and a second training class set is generated by classifying the elements in the training data set according to the second classifier. In particular, the second classifier may be generated by randomly perturbing the random vector used to define the first classifier, and using the random perturbation of the random

vector to define the second classifier. Furthermore, a second objective value based on the second training class set is computed, and the first and second objective values are compared. Based on the comparison between the first and second objective values, the first training class set may be replaced with the second training class set, and the first objective value may be replaced by the second objective value, and the next iteration is begun. The iterations are repeated until a desired number of iterations is reached, and the first training class set is output.

In certain embodiments of the methods described above, the steps of the method are repeated for a plurality of training data sets, where each training data set in the plurality of training data sets is generated by bootstrapping an aggregate training data set. The bootstrapping may be performed with balanced samples or without balanced samples. Whether to bootstrap with balanced samples or without balanced samples may be determined by a binary element in the random vector, whose value may be updated when the random vector is perturbed. Other bootstrap parameters may be included as elements in the random vector, such as whether to sample a subset of samples from an aggregate set of samples with replacement or without replacement or a number of bootstraps. In certain embodiments of the methods, a sample is selected in a test data set, and the classifier corresponding to the output first training class set is used to predict a value associated with the selected sample. In certain embodiments of the methods, the second classifier is generated by applying a random vector to identify parameters for a classification scheme associated with the second classifier, the random vector including at least one binary value. In certain embodiments of the methods, the parameters of the random vector include a flag variable indicating whether to perform balanced bootstrapping, a number of bootstraps, a list of classification methods, a list of genes, or a combination thereof.

In certain embodiments of the methods, the step of computing the second objective value is based on a Matthew correlation coefficient. In particular, the objective value may correspond to a difference between 1 and the Matthew correlation coefficient of the results. The Matthew correlation coefficient is a performance metric that may be used as a composite performance score. In certain embodiments of the methods, the step of computing the second objective value comprises implementing a binary generalized simulated annealing method. In certain embodiments of the methods, the binary generalized simulated annealing method comprises locally perturbing one or more values of the random vector to identify parameters for the classification scheme. In certain embodiments of the methods, locally perturbing the one or

more values of the random vector comprises randomly updating each element of the random vector to obtain an updated random vector, computing an updated second objective value using the updated random vector, and accepting the updated second objective value based on a comparison between a probability value and a random number. In certain embodiments of the methods, locally perturbing the one or more values of the random vector comprises changing one element of the random vector for each iteration.

In certain embodiments of the methods, the step of replacing the first training class set with the second training class set and replacing the first objective value with the second objective value is based on a cooling formula. In particular, it may be desirable to decrease the objective value in a binary generalized simulated annealing method by performing major perturbations on the random vector. In simulated annealing, an artificial temperature value is gradually reduced to simulate cooling. A visiting distribution is used in simulated annealing to simulate a trial jump distance from one point (i.e., a first set of values for the random vector) to another point (i.e., a second set of values for the random vector). The trial jump is accepted based on whether the second objective value is less than the first objective value and on an acceptance probability. The binary generalized simulated annealing method is used to identify a global minimum to minimize the objective value. In certain embodiments of the methods, the second classifier is selected from the group comprising linear discriminant analysis, support vector machine-based methods, random forest methods, and k nearest neighbor methods.

The computer systems of the present invention comprise means for implementing the various embodiments of the methods, as described above. For example, a computer program product is described, the product comprising computer-readable instructions that, when executed in a computerized system comprising at least one processor, cause the processor to carry out one or more steps of any of the methods described above. In another example, a computerized system is described, the system comprising a processor configured with non-transitory computer-readable instructions that, when executed, cause the processor to carry out any of the methods described above. The computer program product and the computerized methods described herein may be implemented in a computerized system having one or more computing devices, each including one or more processors. Generally, the computerized systems described herein may comprise one or more engines, which include a processor or devices, such as a computer, microprocessor, logic device or other device or processor that is configured with hardware,

firmware, and software to carry out one or more of the computerized methods described herein. Any one or more of these engines may be physically separable from any one or more other engines, or may include multiple physically separable components, such as separate processors on common or different circuit boards. The computer systems of the present invention comprises means for implementing the methods and its various embodiments as described above. The engines may be interconnected from time to time, and further connected from time to time to one or more databases, including a perturbations database, a measurables database, an experimental data database and a literature database. The computerized system described herein may include a distributed computerized system having one or more processors and engines that communicate through a network interface. Such an implementation may be appropriate for distributed computing over multiple communication systems.

Brief Description of the Drawings

Further features of the disclosure, its nature and various advantages, will be apparent upon consideration of the following detailed description, taken in conjunction with the accompanying drawings, in which like reference characters refer to like parts throughout, and in which:

FIG. 1 depicts an exemplary system for identifying one or more biomarker signatures;

FIG. 2 is a graph depicting the classification of data samples and the determination of a classification rule;

FIG. 3 is a flow diagram of a dual ensemble method;

FIG. 4 is a flow diagram of a method for building data sets;

FIG. 5 is a flow diagram of a method for generating a result vector and objective value;

FIG. 6 is a flow diagram of a method for initializing a binary generalized simulated annealing method;

FIG. 7 is a flow diagram of a method for decreasing an objective value in a binary generalized simulated annealing method;

FIG. 8 is a flow diagram of a method for further decreasing an objective value in a binary generalized simulated annealing method;

FIG. 9 is a block diagram of a computing device, such as any of the components of the system of FIG. 1; and

FIG. 10 is a heatmap of a gene signature in a training data set.

Detailed Description

To provide an overall understanding of the systems and methods described herein, certain illustrative embodiments will now be described, including systems and methods for identifying gene biomarker signatures. However, it will be understood by one of ordinary skill in the art that the systems and methods described herein may be adapted and modified for other suitable applications, such as any data classification application, and that such other additions and modifications will not depart from the scope thereof. Generally, the computerized systems described herein may comprise one or more engines, which include a processor or devices, such as a computer, microprocessor, logic device or other device or processor that is configured with hardware, firmware, and software to carry out one or more of the computerized methods described herein.

The systems and methods described herein include techniques for generating biomarker signatures with integrated dual ensemble and simulated annealing techniques. The techniques involve resampling a data set and predicting phenotypes using a dual ensemble method. In particular, the systems and methods described herein include forming a random vector indicative of a set of classification methods, data samples, and iteratively perturbing the random vector and computing different objective values corresponding to the different perturbations.

FIG. 1 depicts an exemplary system 100 for identifying one or more biomarker signatures, in which the classification techniques disclosed herein may be implemented. The system 100 includes a biomarker generator 102 and a biomarker consolidator 104. The system 100 further includes a central control unit (CCU) 101 for controlling certain aspects of the operation of the biomarker generator 102 and the biomarker consolidator 104. During operation, data such as gene expression data is received at the biomarker generator 102. The biomarker generator 102 processes the data to generate a plurality of candidate biomarkers and corresponding error rates. The biomarker consolidator 104 receives these candidate biomarkers and error rates and selects a suitable biomarker having an optimal performance measure and size.

The biomarker generator 102 includes several components for processing data and generating a set of candidate biomarkers and candidate error rates. In particular, the biomarker generator 102 includes a data pre-processing engine 110 for splitting the data into a training data

set and a test data set. The biomarker generator 102 includes a classifier 114 for receiving the training data set and the test data set and classifying the test data set into one of two or more classes (e.g., disease data and non-diseased, susceptible and immune, etc.). The biomarker generator 102 includes a classifier performance monitoring engine 116 for determining the performance of the classifier as applied to the test data selected by the data pre-processing engine 110. The classifier performance monitoring engine 116 identifies candidate biomarkers based on the classifier (e.g., the components of the elements of the data set that are most important to the classification) and generates performance measures, which may include candidate error rates, for one or more candidate biomarkers. The biomarker generator 102 further includes a biomarker store 118 for storing one or more candidate biomarkers and candidate performance measures.

The biomarker generator may be controlled by the CCU 101, which in turn may be automatically controlled or user-operated. In certain embodiments, the biomarker generator 102 may operate to generate a plurality of candidate biomarkers, each time splitting the data randomly into training and test data sets. To generate such a plurality of candidate biomarkers, the operation of the biomarker generator 102 may be iterated a plurality of times. CCU 101 may receive one or more system iteration parameters including a desired number of candidate biomarkers, which in turn may be used to determine the number of times the operation of the biomarker generator 102 may be iterated. The CCU 101 may also receive other system parameters including a desired biomarker size which may be representative of the number of components in a biomarker (*e.g.*, the number of genes in a biomarker gene signature). The biomarker size information may be used by the classifier performance monitoring engine 116 for generating candidate biomarkers from the training data. The operation of the biomarker generator 102 and the classifier 114 in particular, are described in more detail with reference to FIGS. 2-8.

The biomarker generator 102 generates one or more candidate biomarkers and candidate error rates, which is used by the biomarker consolidator 104 for generating robust biomarkers. The biomarker consolidator 104 includes a biomarker consensus engine 128 which receives a plurality of candidate biomarkers and generates a new biomarker signature having the most frequently occurring genes across the plurality of candidate biomarkers. The biomarker consolidator 104 includes an error calculation engine 130 for determining an overall error rate

across the plurality of candidate biomarkers. Similar to the biomarker generator 102, the biomarker consolidator 104 may also be controlled by the CCU 101, which in turn may be automatically controlled or user-operated. The CCU 101 may receive and/or determine a suitable threshold values for the minimum biomarker size, and use this information to determine the number of iterations to operate both the biomarker generator 102 and the biomarker consolidator 104. In one embodiment, during each iteration, the CCU 101 decreases the biomarker size by one and iterates both the biomarker generator 102 and the biomarker consolidator 104 until the threshold is reached. In such an embodiment, the biomarker consensus engine 128 outputs a new biomarker signature and a new overall error rate for each iteration. The biomarker consensus engine 128 thus outputs set of new biomarker signatures each having a different size varying from the threshold value up to a maximum biomarker size. The biomarker consolidator 104 further includes a biomarker selection engine 126 which reviews the performance measure or error rate of each of these new biomarker signatures and selects the optimal biomarker for output.

The data preprocessing engine 110 receives one or more datasets. Generally, the data may represent expression values of a plurality of different genes in a sample, and/or a variety of a phenotypic characteristics such as levels of any biologically significant analyte. In certain embodiments, the data sets may include expression level data for a disease condition and for a control condition. As used herein, the term “gene expression level” may refer to the amount of a molecule encoded by the gene, e.g., an RNA or polypeptide, or the amount of a miRNA. The expression level of an mRNA molecule may include the amount of mRNA (which is determined by the transcriptional activity of the gene encoding the mRNA) and the stability of the mRNA (which is determined by the half-life of the mRNA). The gene expression level may also include the amount of a polypeptide corresponding to a given amino acid sequence encoded by a gene. Accordingly, the expression level of a gene can correspond to the amount of mRNA transcribed from the gene, the amount of polypeptide encoded by the gene, or both. Expression levels of a gene may be further categorized by expression levels of different forms of gene products. For example, RNA molecules encoded by a gene may include differentially expressed splice variants, transcripts having different start or stop sites, and/or other differentially processed forms. Polypeptides encoded by a gene may encompass cleaved and/or modified forms of polypeptides. Polypeptides can be modified by phosphorylation, lipidation, prenylation,

sulfation, hydroxylation, acetylation, ribosylation, farnesylation, addition of carbohydrates, and the like. Further, multiple forms of a polypeptide having a given type of modification can exist. For example, a polypeptide may be phosphorylated at multiple sites and express different levels of differentially phosphorylated proteins. The levels of each of such modified polypeptides may be separately determined and represented in the datasets.

The classifier 114 receives one or more sets of data from the data pre-processing engine 110. In certain embodiments, the classifier 114 generates a classification rule to classify the data. FIG. 2 depicts, graphically such a classification rule 200. The classifier 114 may apply the classification rule to assign the data sets to either one of two classes. For example, the classifier 114 may apply the classification to assign data sets to either disease or control.

In certain embodiments, as is described in relation to FIGS. 3 – 8, the classifier 114 uses a dual ensemble technique combined with a generalized simulated annealing method to generate a classification rule. In particular, the classifier 114 may combine multiple classification methods such as support vector machine (SVM), network based SVMs, neural network-based classifiers, logistic regression classifier, decision tree-based classifier, classifiers employing a linear discriminant analysis technique, and/or a random-forest analysis technique, or any other suitable classification method. The ensemble classification strategy may use a voting process across multiple diverse classification methods to identify an optimal classification. By incorporating multiple classification methods, ensemble techniques reduce the potential for overfitting to a small data set. In this way, small data sets may be used more efficiently by using an ensemble technique compared to other techniques. Furthermore, using an ensemble of multiple classification methods allows for enhanced classification compared to using a single classification method, especially when the multiple classification methods in the ensemble are diverse from one another.

In addition, the data received from the data pre-processing engine 110 may be perturbed to further increase overall diversity while providing better classification accuracy. Examples of perturbation of data are described in more detail in relation to FIGS. 4, 7, and 8.

As described herein, the classifier 114 uses an ensemble technique and a generalized simulating annealing method to generate a classification rule and are described in relation to applications in bioinformatics. However, the systems and methods described herein may

generally be applied to any large scale computational technique such as feature selection or extraction.

The classifier performance monitoring engine 116 may analyze the performance of the classifier 114 using a suitable performance metric. In particular, when analyzing the performance of the classifier 114, the classifier performance monitoring engine 116 may be analyzing the robustness or performance of one or more candidate biomarkers. In certain embodiments, the performance metric may include an error rate. The performance metric may also include the number of correct predictions divided by the total predictions attempted. The performance metric may be any suitable measure without departing from the scope of the present disclosure. The candidate biomarker and the corresponding performance metric may be stored in biomarker store 118.

In certain embodiments the gene expression level in a cell or tissue may be represented by a gene expression profile. Gene expression profiles may refers to a characteristic representation of a gene's expression level in a specimen such as a cell or tissue. The determination of a gene expression profile in a specimen from an individual is representative of the gene expression state of the individual. A gene expression profile reflects the expression of messenger RNA or polypeptide or a form thereof encoded by one or more genes in a cell or tissue. An expression profile may generally refer to a profile of biomolecules (nucleic acids, proteins, carbohydrates) which shows different expression patterns among different cells or tissue. A data sample representing a gene expression profile may be stored as a vector of expression levels, with each entry in the vector corresponding to a particular biomolecule or other biological entity.

In certain embodiments, the data sets may include elements representing gene expression values of a plurality of different genes in a sample. In other embodiments, the data set may include elements that represent peaks detected by mass spectrometry. Generally, each data set may include data samples that each correspond to one of a plurality of biological state classes. For example, a biological state class can include, but is not limited to: presence/absence of a disease in the source of the sample (i.e., a patient from whom the sample is obtained); stage of a disease; risk for a disease; likelihood of recurrence of disease; a shared genotype at one or more genetic loci (e.g., a common HLA haplotype; a mutation in a gene; modification of a gene, such as methylation, etc.); exposure to an agent (e.g., such as a toxic substance or a potentially toxic

substance, an environmental pollutant, a candidate drug, etc.) or condition (temperature, pH, etc); a demographic characteristic (age, gender, weight; family history; history of preexisting conditions, etc.); resistance to agent, sensitivity to an agent (e.g., responsiveness to a drug) and the like.

Data sets may be independent of each other to reduce collection bias in ultimate classifier selection. For example, they can be collected from multiple sources and may be collected at different times and from different locations using different exclusion or inclusion criteria, i.e., the data sets may be relatively heterogeneous when considering characteristics outside of the characteristic defining the biological state class. Factors contributing to heterogeneity include, but are not limited to, biological variability due to sex, age, ethnicity; individual variability due to eating, exercise, sleeping behavior; and sample handling variability due to clinical protocols for blood processing. However, a biological state class may comprise one or more common characteristics (e.g., the sample sources may represent individuals having a disease and the same gender or one or more other common demographic characteristics).

In certain embodiments, the data sets from multiple sources are generated by collection of samples from the same population of patients at different times and/or under different conditions.

In certain embodiments, a plurality of data sets is obtained from a plurality of different clinical trial sites and each data set comprises a plurality of patient samples obtained at each individual trial site. Sample types include, but are not limited to, blood, serum, plasma, nipple aspirate, urine, tears, saliva, spinal fluid, lymph, cell and/or tissue lysates, laser microdissected tissue or cell samples, embedded cells or tissues (e.g., in paraffin blocks or frozen); fresh or archival samples (e.g., from autopsies). A sample can be derived, for example, from cell or tissue cultures in vitro. Alternatively, a sample can be derived from a living organism or from a population of organisms, such as single-celled organisms.

In one example, when identifying biomarkers for a particular cancer, blood samples for might be collected from subjects selected by independent groups at two different test sites, thereby providing the samples from which the independent data sets will be developed.

In some implementations, the training and test sets are generated by the data pre-processing engine 110, which receives bulk data and splits the bulk data into a training data set and a test data set. In certain embodiments, the data pre-processing engine 110 randomly splits the data into these two groups. Randomly splitting the data may be desirable for predicting

classes and generating robust gene signature. In other embodiments, the data pre-processing engine 110 splits the data into two or more groups based on the type or label of the data. Generally, the data can be split into a training data set and a test data set in any suitable way as desired without departing from the scope of the present disclosure. The training data set and the test data set may have any suitable size and may be of the same or different sizes. In certain embodiments, the data pre-processing engine 110 may discard one or more pieces of data prior to splitting the data into the training and test data sets. In certain embodiments, the data pre-processing engine 110 may discard one or more pieces of data from the training data set and/or the test data set prior to any further processing.

The classifier 114 may receive one or more candidate biomarkers and one or more sets of data from the data pre-processing engine 110. The classifier 114 may apply the classification rule to assign data sets to either one of two classes. For example, the classifier 114 may apply the classification to assign data sets to either disease or control. In certain embodiments, the classifier 114 may include a support vector machine (SVM) classifier, network based SVMs, neural network-based classifiers, logistic regression classifier, decision tree-based classifier, classifiers employing a linear discriminant analysis technique, and/or a random-forest analysis technique. The operation of the classifier 114 and its respective engines are described in more detail with reference to FIGS. 2-8.

The classifier performance monitoring engine 116 may analyze the performance of the classifier 114 using a suitable performance metric. In particular, when analyzing the performance of the classifier 114, the classifier performance monitoring engine 116 may be analyzing the robustness or performance of one or more candidate biomarkers. In certain embodiments, the performance metric may include an error rate. The performance metric may also include the number of correct predictions divided by the total predictions attempted. The performance metric may be any suitable measure without departing from the scope of the present disclosure. The candidate biomarker and the corresponding performance metric may be stored in biomarker store 118.

As noted earlier, the CCU 101 may also control the operation of the biomarker consolidator 104 for generating a suitable and robust biomarker based on the candidate biomarkers generated and stored in the biomarker generator 102. The biomarker consolidator 104 includes a biomarker consensus engine 128, which receives one or more

candidate biomarkers from the biomarker store 118. The biomarker consensus engine 128 may select frequently occurring genes within the one or more candidate biomarkers for a new biomarker signature. The new biomarker signature may include an N number of genes, where N is a desired size of the biomarker, the maximum allowed size of the biomarker, a minimum allowed size of the biomarker or a size between the maximum and minimum sizes. In certain embodiments, the number N may be user-selectable and may be adjustable as desired.

FIG. 3 is a flow diagram of a method 300 used by the classifier 114 to predict, using a voting method, a phenotype class. As shown, the method 300 includes the steps of building K data sets (step 302), identifying M classification methods (step 306), and identifying G samples in each of the K data sets (step 312). The method 300 further includes three iteration loops, including iterating over the K data sets, the M classification methods, and the G samples, where G is the sample size of the test data set. In particular, in each iteration, a classification method j is applied to a sample l in a dataset i to predict a phenotype (step 318), where i = 1, 2, . . . K, j = 1, 2, . . . M, and l = 1, 2, . . . G.

At step 302, the classifier 114 builds K data sets. The classifier may use the method depicted in FIG. 4 to build K data sets. In particular, the classifier 114 may use boot strapping aggregation methods to form multiple data sets of a complete data set. At step 304, a data set iteration parameter i, representative of a label applied to a data set, is initialized to 1.

At step 306, the classifier 114 identifies M classification methods. The classifier 114 may receive the classification methods from an external source, or the classification methods may be generated by the classifier 114 based on some input. As an example, the classifier 114 may identify the M classification methods based on a list of methods 308. Examples of methods include linear discriminant analysis, support vector machine-based methods, random forest methods (Breiman , *Machine Learning*, 45(1):5-32 (2001)), PAMR (Tibshirani et al., *Proc Natl Acad Sci USA*, 99(10):6567-6572 (2002)) or k nearest neighbor methods (Bishop, Neural Networks for Pattern Recognition, ed. O.U. Press, 1995). Any number of classification methods may be used and considered. At step 310, a method iteration parameter j, representative of a label applied to a classification method, is initialized to 1. At step 316, a sample iteration parameter l, representative of a label applied to a data sample, is initialized to 1. Each data sample may be representative of a person, a gene, or any other suitable data point.

At step 312, the classifier 114 selects the l^{th} samples in the test data set, and at step 318, the classifier 114 applies a classification method j to a data set i to build a classifier and predict a sample l in the test data. The prediction of sample l may correspond to the prediction of a phenotype. In some embodiments, the phenotype may be a flag variable (i.e., 1 if it is predicted that the person expresses the phenotype, and 0 otherwise). However, in general, the phenotype may take on any number of values. In particular, the phenotype prediction may be stored as a value in a three dimensional matrix P(i,j,l) 320.

At decision block 322, the classifier 114 determines whether the last data set has been considered, or equivalently, whether i=K. If i is less than K, the classifier 114 increments the data set iteration parameter i at step 324 and returns to step 318 to predict the phenotype for the new data set.

After all K data sets have been considered, the classifier 114 proceeds to decision block 326 to determine whether the last classification method has been applied, or equivalently, whether j=M. If j is less than M, the classifier 114 increments the method iteration parameter j at step 328 and returns to step 318 to predict the phenotype for the new classification method.

After all K data sets have been considered and all M classification methods have been applied, the classifier 114 has K×M phenotype predictions for the current data sample l. These phenotype predictions may be thought of as votes, and any sort of vote counting method may be used to arrive at a composite vote representative of the set of K×M phenotype predictions.

At decision block 332, the classifier determines whether all G data samples have been considered, or equivalently, whether l=G.

FIG. 4 is a flow diagram of a method 400 for building data sets, and may be used by the classifier 114 at step 302 in FIG. 3. In general, the method 400 provides a way to generate multiple data sets that are each subsets of a larger data set. A data subset may be formed by bootstrap aggregating ("bagging") methods, which involve randomly selecting a subset of samples in a large data set. The subset of samples may be selected with or without replacement. As shown, the method 400 includes the steps of receiving data (step 440) and determining whether it is desirable to perform bootstrapping without replacement (decision block 454). If so, a number W of samples may be randomly selected from each class (step 456) to form a data set. Alternatively, a number H of samples may be randomly selected with replacement from training data (steps 460 and 466) to form a data set. The value for H may correspond to a sample size of

the training data set. The above steps are repeated until each data set i described in relation to FIG. 3 has been considered.

At step 440, the classifier 114 receives data. The data may include samples sorted into two classes (i.e., class 1 samples 442 and class 2 samples 444), bootstrap parameters 446, and a ratio s 448, between the size of a resulting data set i (i.e., a data subset) and the size of a class (i.e., class 1 or class 2). As an example, the bootstrap parameters 446 may include a variable indicative of whether to bootstrap with or without replacement and the number of bootstrap data sets (i.e., K). The data 442, 444, 446, and 448 may be used by the classifier 114 to build the K data sets.

At step 452, a data set iteration parameter i is initialized to 1. The iteration parameter i is representative of a label applied to a data set.

At decision block 454, the classifier 114 determines whether it is desirable to bootstrap with balanced samples. In particular, the classifier 114 may use a variable such as a bootstrap parameter 446 to determine whether bootstrapping with balanced samples is desirable. In general, bootstrapping with balanced samples ensures that the total number of occurrences of each sample point across all K data sets is the same.

If balanced bootstrapping is desirable, the classifier 114 proceeds to step 450 to determine a data set size W. In particular, the size W may be dependent on the ratio s 448, such as W = min{size(class 1 samples), size(class 2 samples)} * s, for example. In particular, the ratio s may be a value between 0 and 1. At step 456, W samples from the training data set are randomly selected with balanced samples, forming a data set i 458. When the iteration parameter i is greater than 1, the selection of W samples at step 456 may be dependent on previously formed data sets, such that the bootstrapping is balanced.

Alternatively, if bootstrapping with balanced samples is not desirable, the classifier 114 proceeds to step 460 to randomly select H samples from the training data set with replacement. The selected samples form the data set i 464.

As is depicted in FIG. 4, balanced bootstrapping results in data sets with size W, while bootstrapping the data without balanced samples results in data sets with size H. However, in general, any suitable combination of methods may be used, such as bootstrapping without balanced samples for data sets with size W, or balanced bootstrapping for data sets with size H. In addition, bootstrapping without replacement methods may also be used.

After the current data set i has been formed, the classifier 114 proceeds to decision block 470 to determine whether the last data set has been formed, or equivalently, whether i=K. If not, at step 472, the data set iteration parameter i is incremented, and the classifier 114 proceeds to decision block 454 to begin forming the next data set.

FIG. 5 is a flow diagram of a method for generating a result vector and objective value. In general, the method 500 provides a way to calculate an objective value corresponding to a random vector X. As depicted in the method 500, the random vector X is a binary vector X and includes information regarding whether to bootstrap with replacement (506), the number of bootstraps (510), a list of the classification methods (514), and a list of data samples (518). Based on these data, a prediction matrix is formed (step 520), and a major class is determined (step 524). The classifier 114 iterates over the data samples until all data samples have been considered, and an objective value is computed based on the determined major classes for the data samples (step 532).

At step 502, the classifier 114 receives a binary random vector X. In an example, the vector X may be a list of binary values. The binary values may be indicative of whether to perform balanced bootstrapping, a number of bootstraps (i.e., K), a list of classification methods, and/or a list of genes. In particular, the number of bootstraps may take on either a zero value or a non-zero value (i.e., 60, for example). In this case, the binary value in the vector X corresponding to the number of bootstraps may be indicative of whether the number of bootstraps is zero or non-zero. The random values may be generated by a random value generator or any other suitable method for generating random values. As is described herein, the random vector X is a binary vector, meaning each value in the vector is one of two values (i.e., 0 or 1). However, in general, the values in the random vector X can be one of any number of values. The classifier 114 identifies various parameters based on the random values in the vector X. As examples, the classifier 114 identifies a value for a flag 506 indicative of whether to sample with balanced samples at step 504, a number of bootstraps 510 at step 508, a list of classification methods 514 at step 512, and a list of genes 518 at step 516.

Based on the identified various parameters, at step 520, the classifier 114 generates a prediction matrix.

At step 522, a sample iteration parameter l, representative of a label applied to a data sample, is initialized to 1.

At step 524, the classifier 114 determines the major class P(., ., l). In particular, the classifier 114 may parse through the steps 302 – 330 in method 300 to identify K×M phenotype predictions, and take a majority vote on the K×M predictions to determine the major class P(., ., l). In general, any other suitable method for generating a composite prediction based on the set of K×M predictions may be used to determine the major class. The major class may be stored as an entry in a result vector 526.

At decision block 528, the classifier 114 determines whether the sample iteration parameter l is equal to the total number of data samples G. If not, the iteration parameter l is incremented at step 530, and the major class is determined for the next data sample.

After the major class has been determined for each sample in the set of G samples, the classifier 114 proceeds to step 532 to calculate an objective value. The objective value may be calculated based on the resultant set of entries in the result vector 526. In particular, a composite performance score may be an average of the performance metrics. As depicted in the method 500, the objective value 532 is calculated as the difference between 1 and the Matthew correlation coefficient (MCC) of the results. The MCC is a performance metric that may be used as a composite performance score. In particular, the MCC is a value between -1 and +1 and is essentially a correlation coefficient between the observed and predicted binary classifications. The MCC may be computed using the following equation:

$$MCC = \frac{\mathrm{TP} * \mathrm{TN} - \mathrm{FP} * \mathrm{FN}}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}}$$

where TP: true positive; FP: false positive; TN: true negative; FN: false negative. However, in general, any suitable technique for generating a composite performance metric based on a set of performance metrics may be used to calculate the objective value.

FIGS. 6 – 8 are flow diagrams of methods for parsing through the steps of a binary generalized simulated method. In general, the binary generalized simulated annealing method may be used to identify an optimal value (i.e., a global minimum) for the objective value as described in FIG. 5. As described herein, a binary generalized simulated annealing method is used in conjunction with the dual ensemble method described in FIG. 3. In particular, a random vector X as described in FIG. 5 is perturbed in various ways to identify an optimal objective value. FIG. 6 is a flow diagram for initializing the binary generalized simulated annealing method. FIG. 7 is a flow diagram for randomly perturbing various components of the random

vector X to decrease the objective value. FIG. 8 is a flow diagram for locally perturbing the random vector X to further decrease the objective value. In other words, the method depicted in FIG. 7 generates major perturbations of the random vector X, while the method depicted in FIG. 8 generates minor perturbations of the random vector X.

FIG. 6 is a flow diagram of a method 600 for initializing a binary generalized simulated annealing method. The method 600 initializes several parameters and generates a random binary vector X(1). In particular, at steps 640, 642, and 644, the classifier 114 initializes parameters t, y, and count to 1, respectively. The parameter t corresponds to a time interval, and is incremented when a suitable objective value is determined, as is described in relation to FIGS. 7 and 8. The iteration parameter y corresponds to a number of major perturbations that are performed, and is described in more detail in relation to FIG. 7. The parameter count corresponds to a parameter for keeping track of whether a perturbed version of the current vector X has been generated, and is described in more detail in relation to FIG. 7. At step 646, the classifier 114 generates a random binary vector X.

At step 648, a parameter D is set. The parameter D corresponds to a number of components in X that will be selected to be perturbed. In particular, at step 648, the parameter D is set to 0.2 * C, where C corresponds to the length of binary vector X.

At step 650, the classifier 114 generates a result vector and an objective value. In particular, the classifier 114 may use the method depicted in FIG. 5 to generate a result vector 526 and an objective value 534. However, in general, any suitable method to determine an objective value representative of a composite performance metric may be used. After generating an objective value, the classifier 114 proceeds to steps in FIG. 7 to decrease the objective value by perturbing the random vector X.

FIG. 7 is a flow diagram of a method for decreasing an objective value in a binary generalized simulated annealing method by performing major perturbations on a vector X. In the simulating annealing method, an artificial temperature is introduced (T(t=1)) and gradually reduced to simulate cooling. A visiting distribution is used in simulated annealing to simulate a trial jump distance from one point to a second point (i.e., from one random vector X(1) to another random vector X(2)). The trial jump is accepted based on whether the resulting objective value corresponding to the random vector X(2) is smaller than the objective value corresponding to the random vector X(1) and on an acceptance probability as is defined below.

As is described herein, a binary generalized simulated annealing method is used to locate a global minimum (i.e., to minimize the objective value). However, in general, any suitable algorithm may be used, such as steepest descent, conjugate gradient, simplex, and Monte Carlo methods.

After initializing the simulation using the method depicted in FIG. 6, the classifier 114 begins at step 760 to select D components of the vector X(1). The D components of the vector X(1) may be selected randomly, or any other suitable method of selecting D components of the vector X(1) may be performed. At step 762, the count variable is set to 2. At step 764, a second random binary vector X(2), corresponding to the original random vector X(1) with the D components changed, is generated.

At step 766, the classifier 114 generates a result vector 768 and an objective value 770 for the second vector X(2). In particular, the classifier 114 may use the method depicted in FIG. 5 to generate a result vector and an objective value. However, in general, any suitable method to determine an objective value representative of a composite performance metric may be used.

After generating the second result vector and the second objective value, the classifier determines that the count variable is equal to 2 at decision block 772 and proceeds to decision block 776 to compare the first objective value (i.e., corresponding to the random vector X(1)) and the second objective value (i.e., corresponding to the random vector X(2)).

If the second objective value is not smaller than the first objective value, this means that the first vector X(1) resulted in a better or equal correlation as the second vector X(2). In this case, the classifier proceeds to step 778 to compute a probability P. In particular, the probability P corresponds to a probability of accepting the second objective value, and is based on the equation:

$$P = \min\{1,[1-(1-q_a)\beta\partial E]^{\frac{1}{1-q_a}}\}$$

where $\partial E = obj(2) - obj(1)$

$$\beta = \frac{1}{T_{qv}(t)}$$

q_a is a control parameter for accepting the probability P and

T_{qv} is a temperature value.

As described herein, the probability P corresponds to a probability used in generalized simulated annealing methods, but in general, any suitable probability value may be used. At step 786, a random number r between 0 and 1, inclusive, is generated. The random number r may be generated from a uniform distribution, or any other suitable distribution, and r is compared to the probability P at decision block 788.

If P is greater than or equal to r, this means that the probability of accepting the second objective value is high, even though the second objective value was not smaller than the first objective value. In this case, the classifier 114 proceeds to step 790 and 792 to store the second vector X(2) and the second objective value as the first vector X(1) and the first objective value, respectively.

Alternatively, if at decision block 776, the classifier 114 determines that the second objective value is smaller than the first objective value, this means that the vector X(2) resulted in a better correlation, or better performance. Thus, the classifier proceeds directly to step 790 to update the vector X(1) with the vector X(2), and to step 792 to update the first objective value with the second objective value. At step 794, the classifier 114 sets the count variable to be equal to 1.

Alternatively, if at decision block 788, the classifier 114 determines that r is greater than P, this means that the probability of accepting the second objective value is low, such that the steps 790 and 792 are bypassed, and the vector X(1) and first objective value are not overwritten by the corresponding second values. In this case, the classifier 114 proceeds to step 794 and sets the count variable to be equal to 1.

After re-setting the count variable to 1, the classifier 114 proceeds to decision block 796, where the iteration parameter y is compared to a value L. The value L corresponds to a maximal number of major perturbations to be performed before proceeding to the method depicted in FIG. 8 to perform minor perturbations. If the iteration parameter y is not equal to L, the classifier 114 proceeds to decision block 772 and step 774 to increment the iteration parameter y and perform a major perturbation of the vector X at steps 760 – 764. The steps described above are repeated until a desired number of major perturbations L have been performed. As depicted in FIG. 7, the number of major perturbations to be performed is a fixed number L. However, the value for L may be dependent on any number of factors. For example, the classifier 114 may determine that the total number of major perturbations has been reached based on a convergence

of the objective values. In another example, the total number of major perturbations may be reached if no second objective values have been found to be less than the first objective value in a fixed number of recent comparisons at decision block 776. In general, any suitable method may be used to determine that the major perturbations are done, and that the classifier 114 may proceed to FIG. 8 to perform minor perturbations.

FIG. 8 is a flow diagram of a method for further decreasing an objective value in a binary generalized simulated annealing method by performing minor perturbations on the vector X. In particular, the method 800 begins at step 802 and sets a variable C equal to the length of the vector X(1). At step 804, the classifier 114 initializes an iteration parameter c to 1 and sets an improve flag variable to false.

At step 806, the classifier 114 performs a minor perturbation on the vector X(1) by flipping the c^{th} bit of X(1) to generate X_{temp}. In particular, X(1) is a binary vector of length C, and X_{temp} is nearly identical to X(1) except for the c^{th} bit.

At step 808, the classifier 114 generates a result vector 810 and an objective value 812 for the temporary vector X_{temp}. In particular, the classifier 114 may use the method depicted in FIG. 5 to generate a temporary result vector and a temporary objective value. However, in general, any suitable method to determine an objective value representative of a composite performance metric may be used.

At decision block 814, the first objective value is compared to the temporary objective value. If the temporary objective value is smaller than the first objective value, this means that perturbed version X_{temp} resulted in better performance than the original vector X(1). In this case, the classifier 114 proceeds to step 816 to overwrite the vector X(1) with the perturbed version X_{temp}, to step 818 to overwrite the first objective value with the temporary objective value, and to step 819 to set the improve flag variable to true.

At decision block 820, the classifier 114 determines whether each bit in the vector X(1) has been flipped at least once (i.e., at step 806), or equivalently, whether the iteration parameter c is equal to the size of X(1) C. If not, the classifier 114 proceeds to step 822 to increment the iteration parameter c and to step 806 to flip the c^{th} bit.

Otherwise, if the classifier 114 determines that the iteration parameter c is equal to the length of the vector X(1) C at decision block 820, the classifier 114 proceeds to decision block 822 to determine whether further improvement is desired. In particular, the classifier 114 may

identify the value of the improve flag variable to determine whether additional bit flipping is desirable. For example, if the improve flag variable is true, the classifier 114 returns to step 804 to re-initialize the iteration parameter c to 1 and the improve flag variable to false.

The depicted method in FIG. 8 uses the improve flag variable to determine when the process of performing minor perturbations (i.e., bit flipping) is complete. However, in general, any other suitable method may also be used to determine when the minor perturbations are complete. For example, the classifier 114 may require that the objective value is below some threshold, or that the difference between the objective value and the temporary objective value is below some threshold. If these requirements are not satisfied, the classifier 114 may return to step 806 to flip another bit of the vector X(1) to generate another temporary objective value.

After the classifier 114 has determined that the minimal objective value has been identified, the classifier 114 proceeds to steps 824 and 826 to increment the parameter t and decrement the parameter D, respectively.

At step 828, the classifier 114 computes a temperature T by the cooling formula commonly used in generalized simulated annealing. However, any suitable formula may be used.

$$T_{q_v}(t) \leftarrow T_{q_v}(1)\frac{2^{q_v-1}-1}{(1+t)^{q_v-1}-1}$$

where q_v is a parameter defining the curvature of the distribution function.

At decision block 830, the classifier 114 determines whether $T_{qv}(t)$ is less than T_L. The value for T_L represents a threshold value, where if the value for $T_{qv}(t)$ is below T_L, the method 800 ends, and the current random vector X(1) is used as the optimal classification.

Implementations of the present subject matter can include, but are not limited to, systems methods and computer program products comprising one or more features as described herein as well as articles that comprise a machine-readable medium operable to cause one or more machines (e.g., computers, robots) to result in operations described herein. The methods described herein can be implemented by one or more processors or engines residing in a single computing system or multiple computing systems. Such multiple computing systems can be connected and can exchange data and/or commands or other instructions or the like via one or more connections, including but not limited to a connection over a network (e.g. the Internet, a

wireless wide area network, a local area network, a wide area network, a wired network, or the like), via a direct connection between one or more of the multiple computing systems.

FIG. 9 is a block diagram of a computing device, such as any of the components of system 100 of FIG. 1 including circuitry for performing processes described with reference to FIGS. 2 – 8. Each of the components of system 100 may be implemented on one or more computing devices 900. In certain aspects, a plurality of the above-components and databases may be included within one computing device 900. In certain implementations, a component and a database may be implemented across several computing devices 900.

The computing device 900 comprises at least one communications interface unit, an input/output controller 910, system memory, and one or more data storage devices. The system memory includes at least one random access memory (RAM 902) and at least one read-only memory (ROM 904). All of these elements are in communication with a central processing unit (CPU 906) to facilitate the operation of the computing device 900. The computing device 900 may be configured in many different ways. For example, the computing device 900 may be a conventional standalone computer or alternatively, the functions of computing device 900 may be distributed across multiple computer systems and architectures. The computing device 900 may be configured to perform some or all of data-splitting, differentiating, classifying, scoring, ranking and storing operations. In FIG. 9, the computing device 900 is linked, via network or local network, to other servers or systems.

The computing device 900 may be configured in a distributed architecture, wherein databases and processors are housed in separate units or locations. Some such units perform primary processing functions and contain at a minimum a general controller or a processor and a system memory. In such an aspect, each of these units is attached via the communications interface unit 908 to a communications hub or port (not shown) that serves as a primary communication link with other servers, client or user computers and other related devices. The communications hub or port may have minimal processing capability itself, serving primarily as a communications router. A variety of communications protocols may be part of the system, including, but not limited to: Ethernet, SAP, SAS™, ATP, BLUETOOTH™, GSM and TCP/IP.

The CPU 906 comprises a processor, such as one or more conventional microprocessors and one or more supplementary co-processors such as math co-processors for offloading workload from the CPU 906. The CPU 906 is in communication with the communications

interface unit 1008 and the input/output controller 910, through which the CPU 906 communicates with other devices such as other servers, user terminals, or devices. The communications interface unit 908 and the input/output controller 910 may include multiple communication channels for simultaneous communication with, for example, other processors, servers or client terminals. Devices in communication with each other need not be continually transmitting to each other. On the contrary, such devices need only transmit to each other as necessary, may actually refrain from exchanging data most of the time, and may require several steps to be performed to establish a communication link between the devices.

The CPU 906 is also in communication with the data storage device. The data storage device may comprise an appropriate combination of magnetic, optical or semiconductor memory, and may include, for example, RAM 902, ROM 904, flash drive, an optical disc such as a compact disc or a hard disk or drive. The CPU 906 and the data storage device each may be, for example, located entirely within a single computer or other computing device; or connected to each other by a communication medium, such as a USB port, serial port cable, a coaxial cable, an Ethernet type cable, a telephone line, a radio frequency transceiver or other similar wireless or wired medium or combination of the foregoing. For example, the CPU 906 may be connected to the data storage device via the communications interface unit 908. The CPU 906 may be configured to perform one or more particular processing functions.

The data storage device may store, for example, (i) an operating system 1012 for the computing device 900; (ii) one or more applications 914 (e.g., computer program code or a computer program product) adapted to direct the CPU 906 in accordance with the systems and methods described here, and particularly in accordance with the processes described in detail with regard to the CPU 906; or (iii) database(s) 916 adapted to store information that may be utilized to store information required by the program. In some aspects, the database(s) includes a database storing experimental data, and published literature models.

The operating system 912 and applications 914 may be stored, for example, in a compressed, an uncompiled and an encrypted format, and may include computer program code. The instructions of the program may be read into a main memory of the processor from a computer-readable medium other than the data storage device, such as from the ROM 904 or from the RAM 902. While execution of sequences of instructions in the program causes the CPU 906 to perform the process steps described herein, hard-wired circuitry may be used in

place of, or in combination with, software instructions for implementation of the processes of the present invention. Thus, the systems and methods described are not limited to any specific combination of hardware and software.

Suitable computer program code may be provided for performing one or more functions in relation to performing classification methods as described herein. The program also may include program elements such as an operating system 912, a database management system and "device drivers" that allow the processor to interface with computer peripheral devices (e.g., a video display, a keyboard, a computer mouse, etc.) via the input/output controller 910.

A computer program product comprising computer-readable instructions is also provided. The computer-readable instructions, when loaded and executed on a computer system, cause the computer system to operate according to the method, or one or more steps of the method described above. The term "computer-readable medium" as used herein refers to any non-transitory medium that provides or participates in providing instructions to the processor of the computing device 900 (or any other processor of a device described herein) for execution. Such a medium may take many forms, including but not limited to, non-volatile media and volatile media. Non-volatile media include, for example, optical, magnetic, or opto-magnetic disks, or integrated circuit memory, such as flash memory. Volatile media include dynamic random access memory (DRAM), which typically constitutes the main memory. Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, any other magnetic medium, a CD-ROM, DVD, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, an EPROM or EEPROM (electronically erasable programmable read-only memory), a FLASH-EEPROM, any other memory chip or cartridge, or any other non-transitory medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to the CPU 906 (or any other processor of a device described herein) for execution. For example, the instructions may initially be borne on a magnetic disk of a remote computer (not shown). The remote computer can load the instructions into its dynamic memory and send the instructions over an Ethernet connection, cable line, or even telephone line using a modem. A communications device local to a computing device 900 (e.g., a server) can receive the data on the respective communications line and place the data on a

system bus for the processor. The system bus carries the data to main memory, from which the processor retrieves and executes the instructions. The instructions received by main memory may optionally be stored in memory either before or after execution by the processor. In addition, instructions may be received via a communication port as electrical, electromagnetic or optical signals, which are exemplary forms of wireless communications or data streams that carry various types of information.

Example

The following public datasets are downloaded from the Gene Expression Omnibus (GEO) (http://www.ncbi.nlm.nih.gov/geo/) repository:

a. GSE10106 (www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE10106)
b. GSE10135 (www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE10135)
c. GSE11906 (www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE11906)
d. GSE11952 (www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE11952)
e. GSE13933 (www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE13933)
f. GSE19407 (www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE19407)
g. GSE19667 (www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE19667)
h. GSE20257 (www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE20257)
i. GSE5058 (www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE5058)
j. GSE7832 (www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE7832)
k. GSE8545 (www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE8545).

The training datasets are on the Affymetrix platform (HGU-133 + 2). Raw data files are read by the ReadAffy function of the affy package (Gautier, 2004) belonging to Bioconductor (Gentleman, 2004) in R (R Development Core Team, 2007), and the quality is controlled by: generating RNA degradation plots (with the AffyRNAdeg function of the affy package), NUSE and RLE plots (with the function affyPLM (Brettschneider, 2008)), and calculating the MA(RLE) values; excluding arrays from the training datasets that fell below a set of thresholds on the quality control checks or that are duplicated in the above datasets; and normalizing arrays that pass quality control checks using the gcrma algorithm (Wu, 2004). Training set sample classifications are obtained from the series matrix file of the GEO database for each dataset. The output consists of a gene expression matrix with 54675 probesets for 233 samples (28 COPD samples and 205 control sample). To make a balanced data set, the COPD samples were multiple

time to obtain 224 COPD samples before the Duel Ensemble method as described in copending United States provisional application 61/662812 is applied. With a combined data set which contains 205 control and 224 COPD patients, a gene signature with 409 genes was built. 850 binary values were used in the random vectors. The classification methods used in the method included the following R packages: lda, svm, randomForest, knn, pls.lda and pamr. Maximum iteration was set to be 5000. The Matthew's Correlation Coefficient (MCC) and accuracy in cross validation process in training data set is 0.743 and 0.87 respectively. The heatmap of the gene signature in training data set is shown in FIG. 10. In the heatmap of FIG. 10, the gene expression value was centered by row. The colors of the heatmap may not be clearly shown in grey scale, but the data of FIG. 10 show that control data are shown on the left, and COPD data are shown on the right. The test data set is an unpublished data set obtained from a commercial supplier (Genelogic), which contains 16 control samples and 24 COPD samples. Without applying the transformation invariant method of the invention, the gene signature generated by Dual Ensemble correctly predicted 29 samples out of total 40 samples. The accuracy is 0.725, and the MCC is 0.527. The gene signature correctly predicted 15 out of 16 control samples and correctly predicted 14 out of 24 COPD samples.

While implementations of the invention have been particularly shown and described with reference to specific examples, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the disclosure.

Claims

1. A computer-implemented method of classifying a data set into two or more classes executed by a processor, comprising:

(a) receiving a training data set;

(b) generating a first classifier for the training data set by applying a first machine learning technique to the training data set;

(c) generating a first training class set by classifying the elements in the training data set according to the first classifier;

(d) computing a first objective value based on the training class set;

(e) for each of a plurality of iterations:

(i) generating a second classifier for the training data set by applying a second machine learning technique to the training data set;

(ii) generating a second training class set by classifying the elements in the training data set according to the second classifier;

(iii) computing a second objective value based on the training class set;

(iv) comparing the first and second objective values;

(v) replacing the first training class set with the second training class set and replacing the first objective value with the second objective value based on the comparison in step (iv), and return to step (i); and

(f) when a desired number of iterations has been reached, outputting the first training class set.

2. The method of claim 1, further comprising repeating the steps (a) through (f) for a plurality of training data sets, each training data set in the plurality of training data sets generated by bootstrapping an aggregate training data set.

3. The method of claim 2, wherein the bootstrapping is performed with balanced samples or without balanced samples.

4. The method of any of claims 1-3, further comprising selecting a sample in a test data set and using the classifier corresponding to the output first training class set to predict a value associated with the selected sample.

5. The method of any of claims 1-4, wherein the second classifier is generated by applying a random vector to identify parameters for a classification scheme associated with the second classifier, the random vector including at least one binary value.

6. The method of claim 5, wherein the parameters include a flag variable indicating whether to perform balanced bootstrapping, a number of bootstraps, a list of classification methods, a list of genes, or a combination thereof.

7. The method of any of claims 1-6, wherein the step of computing the second objective value is based on a Matthew correlation coefficient.

8. The method of any of claims 1-7, wherein the step of computing the second objective value comprises implementing a binary generalized simulated annealing method.

9. The method of claim 8, wherein the binary generalized simulated annealing method comprises locally perturbing one or more values of the random vector to identify parameters for the classification scheme.

10. The method of claim 9, wherein locally perturbing the one or more values of the random vector comprises randomly updating each element of the random vector to obtain an updated random vector, computing an updated second objective value using the updated random vector, and accepting the updated second objective value based on a comparison between a probability value and a random number.

11. The method of claim 9, wherein locally perturbing the one or more values of the random vector comprises changing one element of the random vector for each iteration.

12. The method of any of claims 1-11, wherein the step of replacing the first training class set with the second training class set and replacing the first objective value with the second objective value is based on a cooling formula.

13. The method of any of claims 1-12, wherein the second classifier is selected from the group comprising linear discriminant analysis, support vector machine-based methods, random forest methods, and k nearest neighbor methods.

14. A computer program product comprising computer-readable instructions that, when executed in a computerized system comprising at least one processor, cause the processor to carry out one or more steps of the method of any of claims 1-13

15. A computerized system comprising a processing device configured with non-transitory computer-readable instructions that, when executed, cause the processing device to carry out the method of any of claims 1-13.

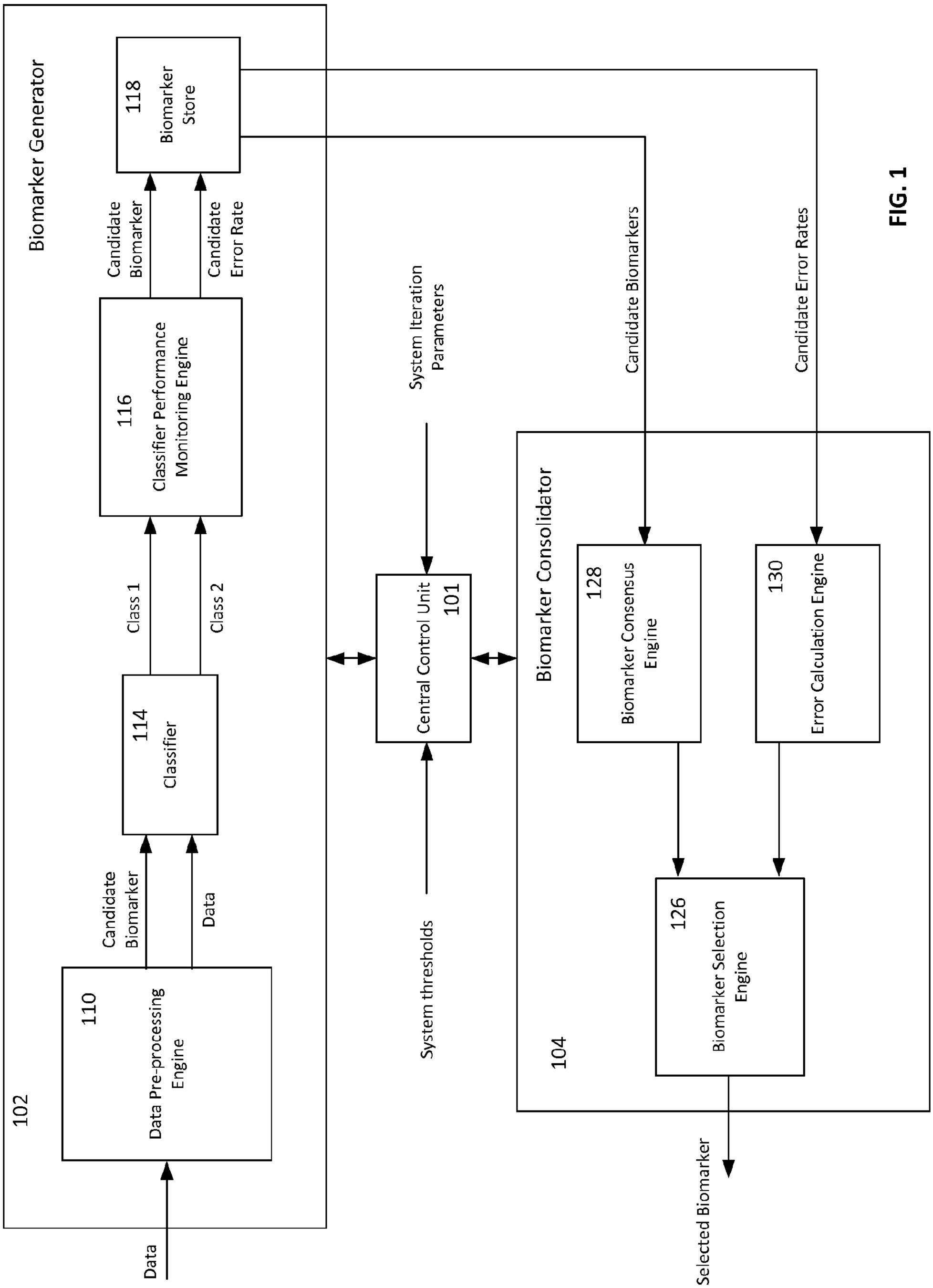
100
Biomarker Generator
102
110
Data Pre-processing Engine
Candidate Biomarker
Data
Data
104
Biomarker Consolidator
126
Biomarker Selection Engine
Selected Biomarker
114
Classifier
Class 1
Class 2
116
Classifier Performance Monitoring Engine
Candidate Biomarker
Candidate Error Rate
118
Biomarker Store
Central Control Unit 101
System Iteration Parameters
System thresholds
128
Biomarker Consensus Engine
130
Error Calculation Engine
Candidate Biomarkers
Candidate Error Rates

FIG. 1

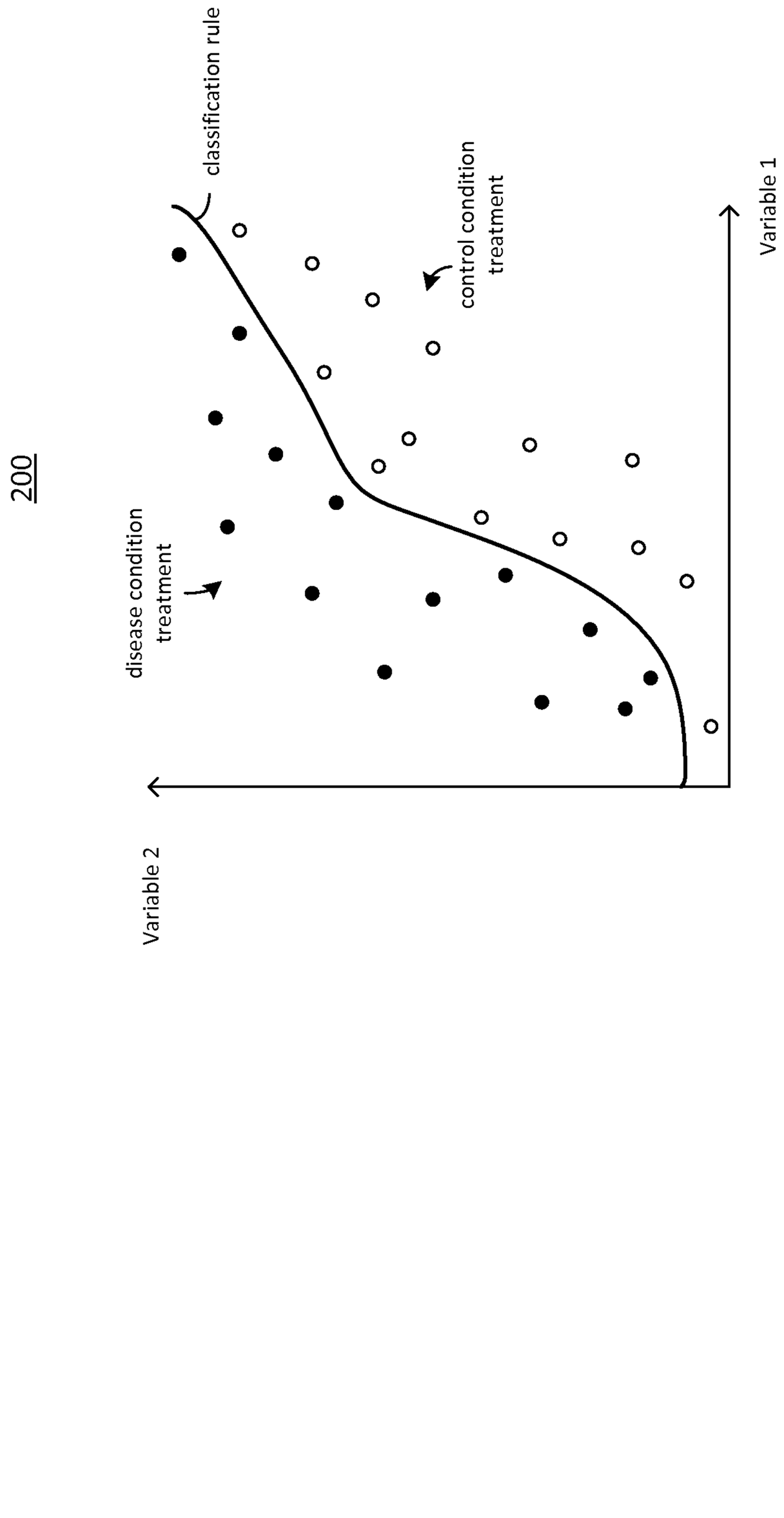
200
classification rule
control condition
treatment
disease condition
treatment
Variable 1
Variable 2

FIG. 2

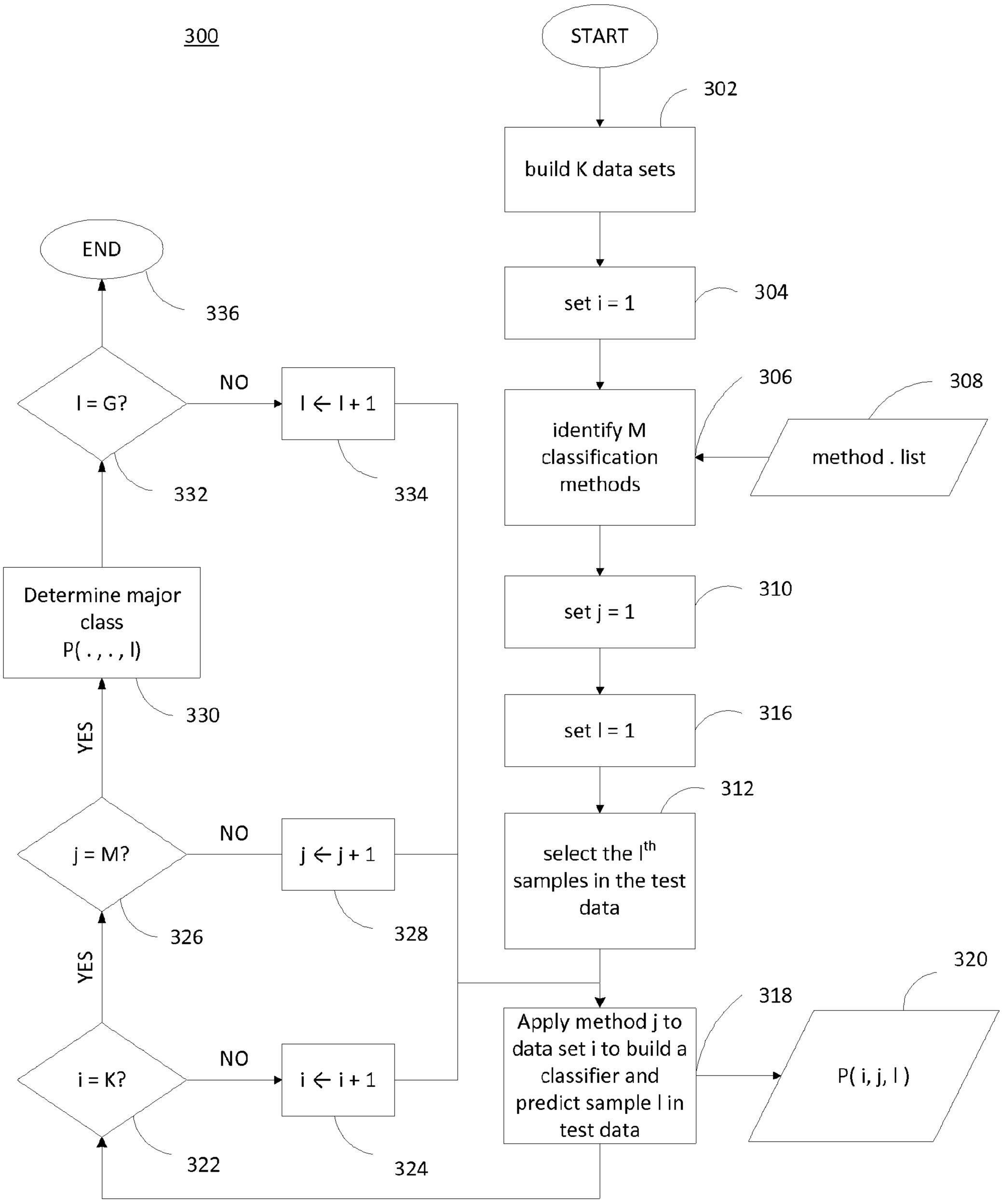
300
START
302
build K data sets
set i = 1
304
306
308
identify M classification methods
method . list
set j = 1
310
set l = 1
316
312
select the l^th samples in the test data
320
318
Apply method j to data set i to build a classifier and predict sample l in test data
P(i, j, l)
END
336
l = G?
NO
l ← l + 1
332
334
Determine major class P(. , . , l)
330
YES
j = M?
NO
j ← j + 1
326
328
YES
i = K?
NO
i ← i + 1
322
324

FIG. 3

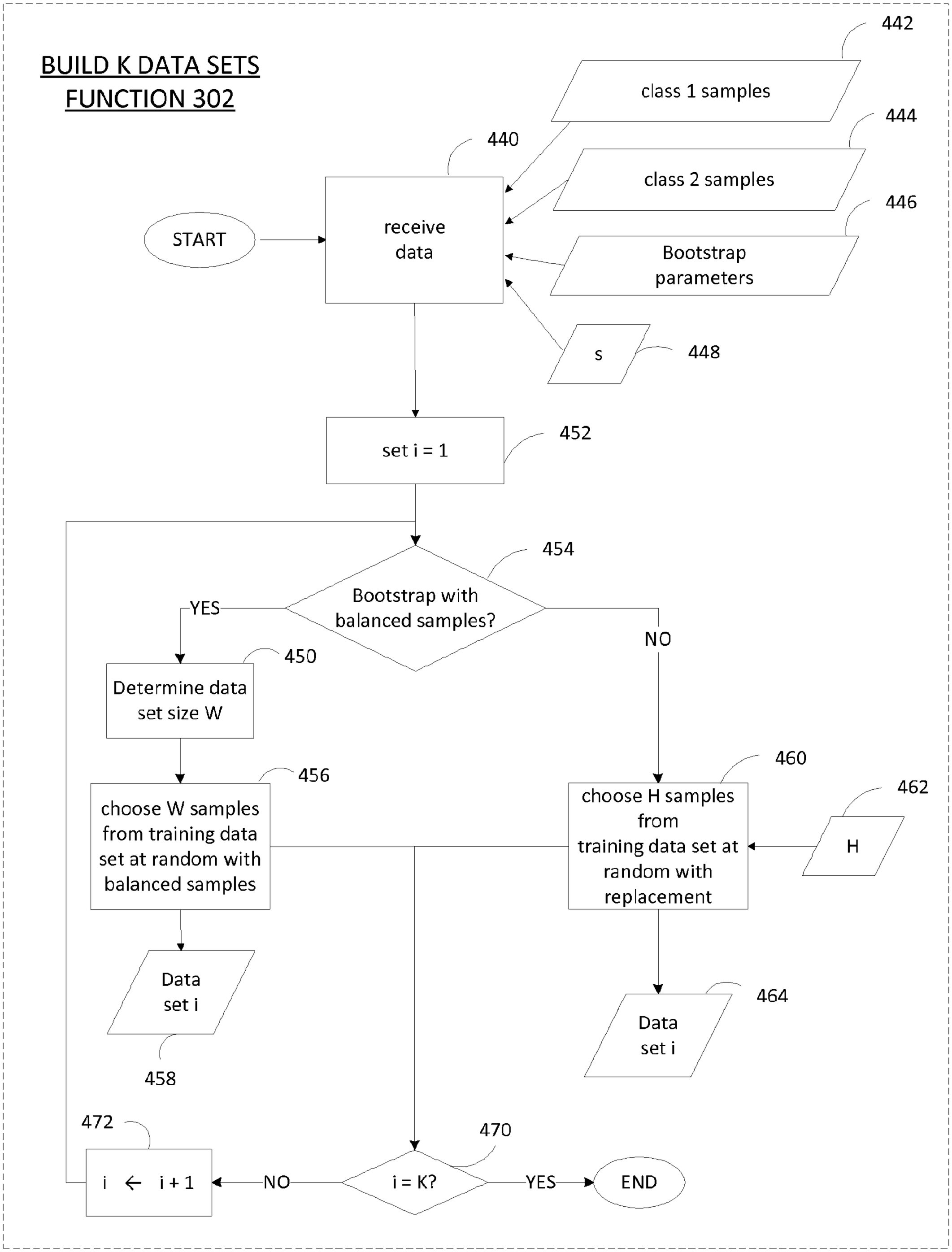
400
BUILD K DATA SETS FUNCTION 302
START
receive data
440
class 1 samples
442
class 2 samples
444
Bootstrap parameters
446
s
448
set i = 1
452
Bootstrap with balanced samples?
454
YES
NO
Determine data set size W
450
choose W samples from training data set at random with balanced samples
456
Data set i
458
choose H samples from training data set at random with replacement
460
H
462
Data set i
464
i ← i + 1
472
NO
i = K?
470
YES
END

FIG. 4

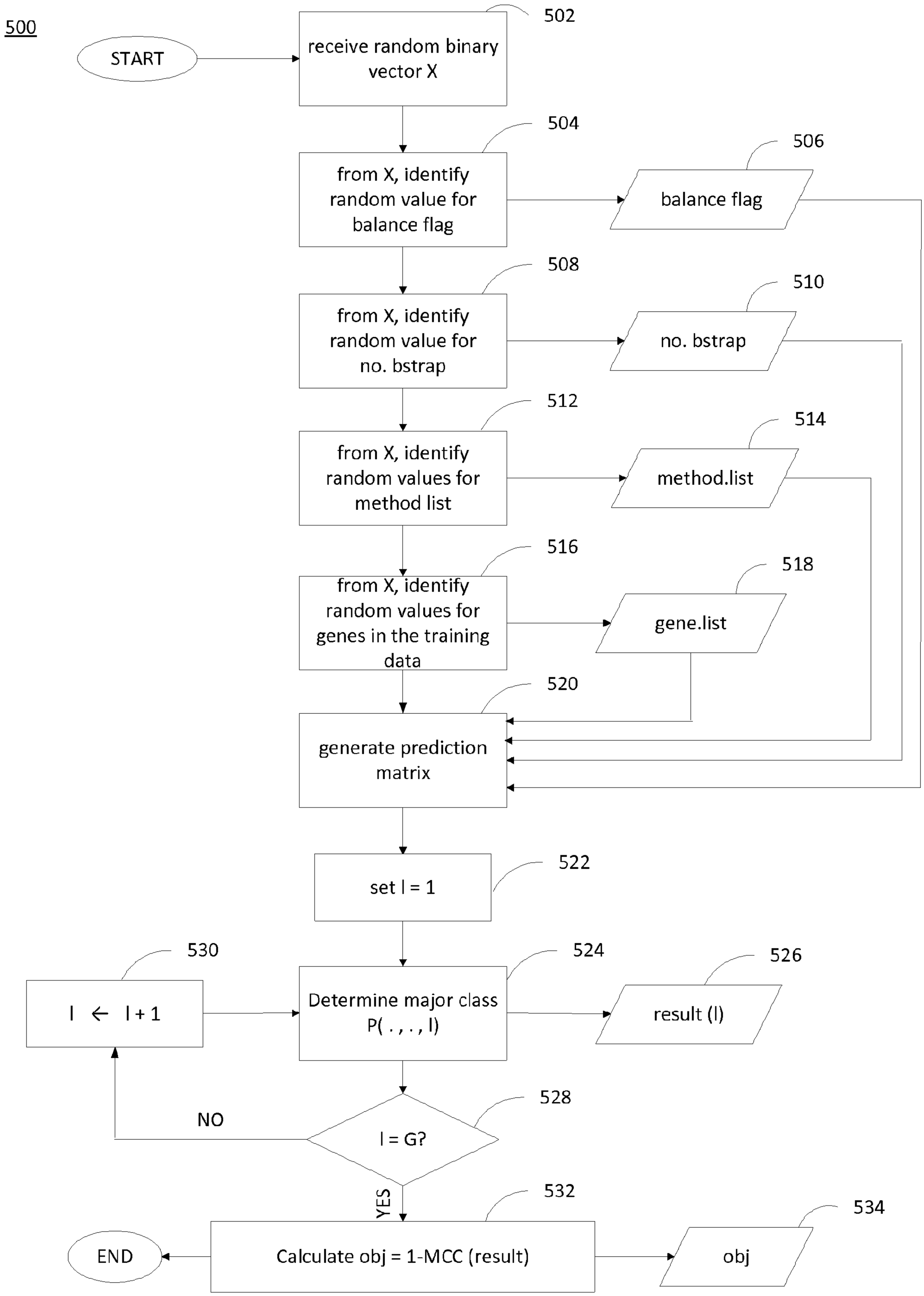
500
START
receive random binary vector X
502
from X, identify random value for balance flag
504
balance flag
506
from X, identify random value for no. bstrap
508
no. bstrap
510
from X, identify random values for method list
512
method.list
514
from X, identify random values for genes in the training data
516
gene.list
518
generate prediction matrix
520
set I = 1
522
I ← I + 1
530
Determine major class P(. , . , I)
524
result (I)
526
I = G?
528
NO
YES
Calculate obj = 1-MCC (result)
532
END
obj
534

FIG. 5

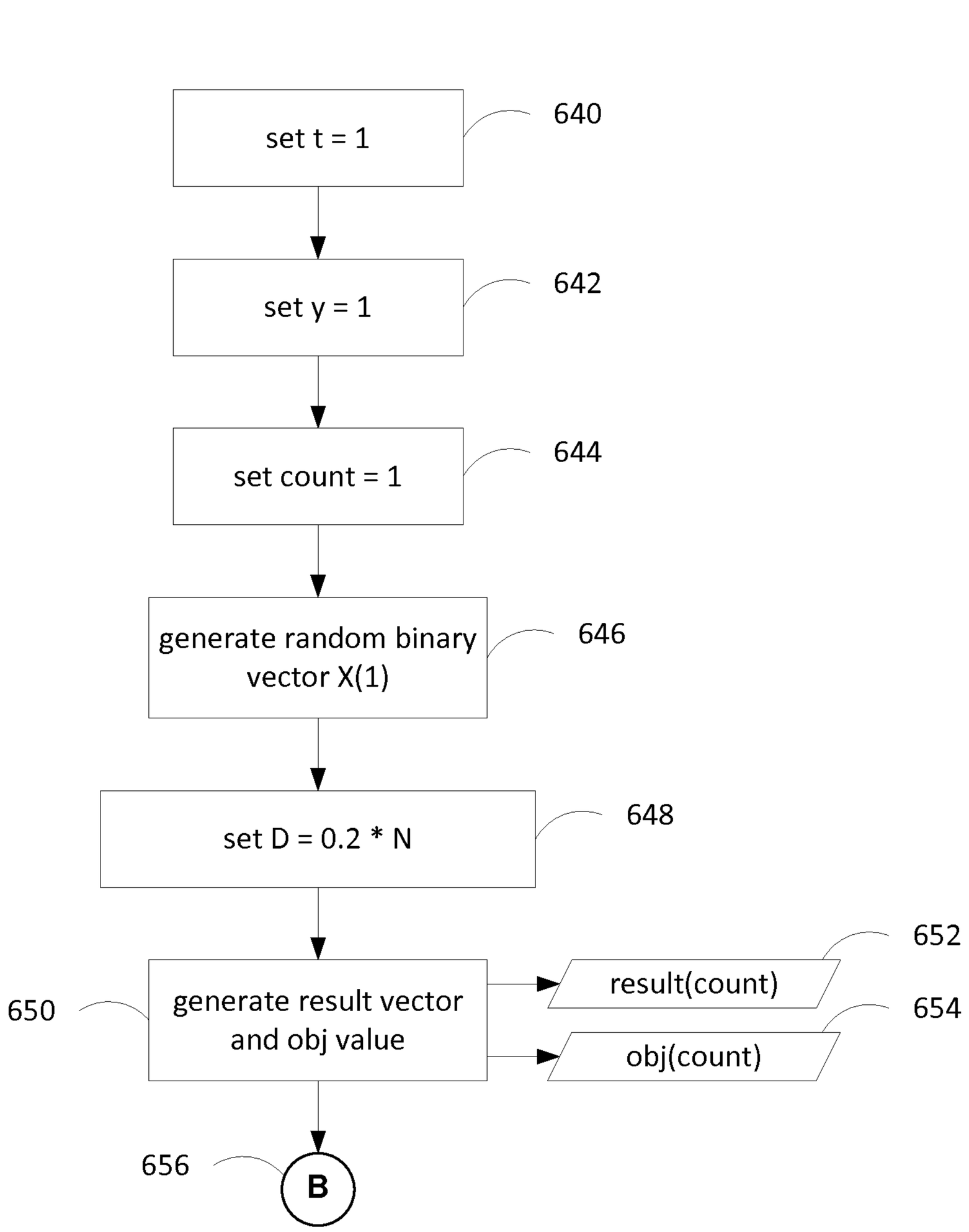
600
set t = 1
640
set y = 1
642
set count = 1
644
generate random binary vector X(1)
646
set D = 0.2 * N
648
generate result vector and obj value
650
result(count)
652
obj(count)
654
656
B

FIG. 6

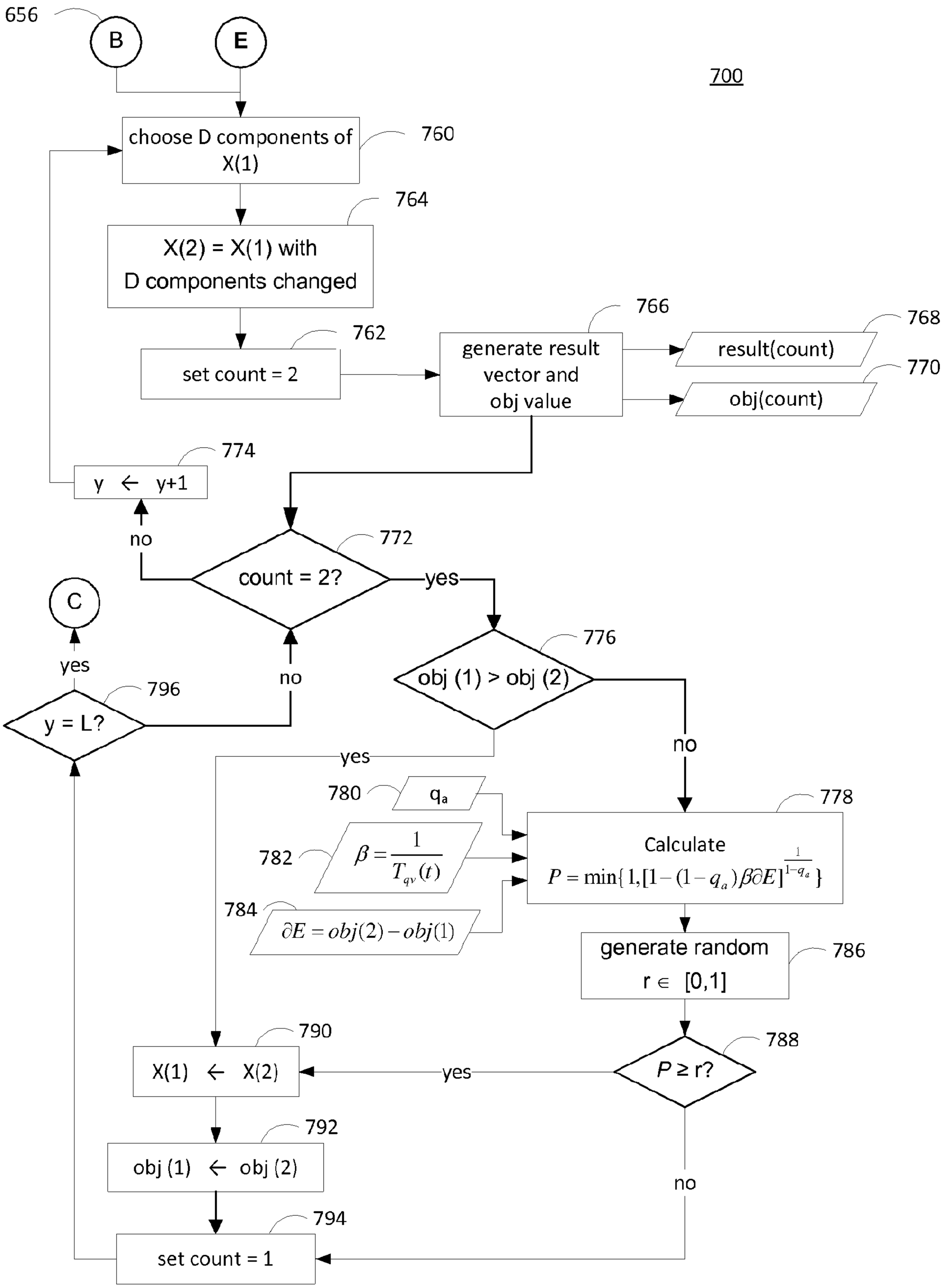
700
656
B
E
choose D components of X(1)
760
764
X(2) = X(1) with D components changed
762
set count = 2
766
generate result vector and obj value
768
result(count)
770
obj(count)
774
y ← y+1
no
772
count = 2?
yes
C
yes
796
y = L?
no
776
obj (1) > obj (2)
no
yes
780
q_a
782
$\beta = \frac{1}{T_{qv}(t)}$
784
$\partial E = obj(2) - obj(1)$
778
Calculate
$P = \min\{1, [1-(1-q_a)\beta\partial E]^{\frac{1}{1-q_a}}\}$
generate random
r ∈ [0,1]
786
788
P ≥ r?
yes
790
X(1) ← X(2)
792
obj (1) ← obj (2)
794
set count = 1
no

FIG. 7

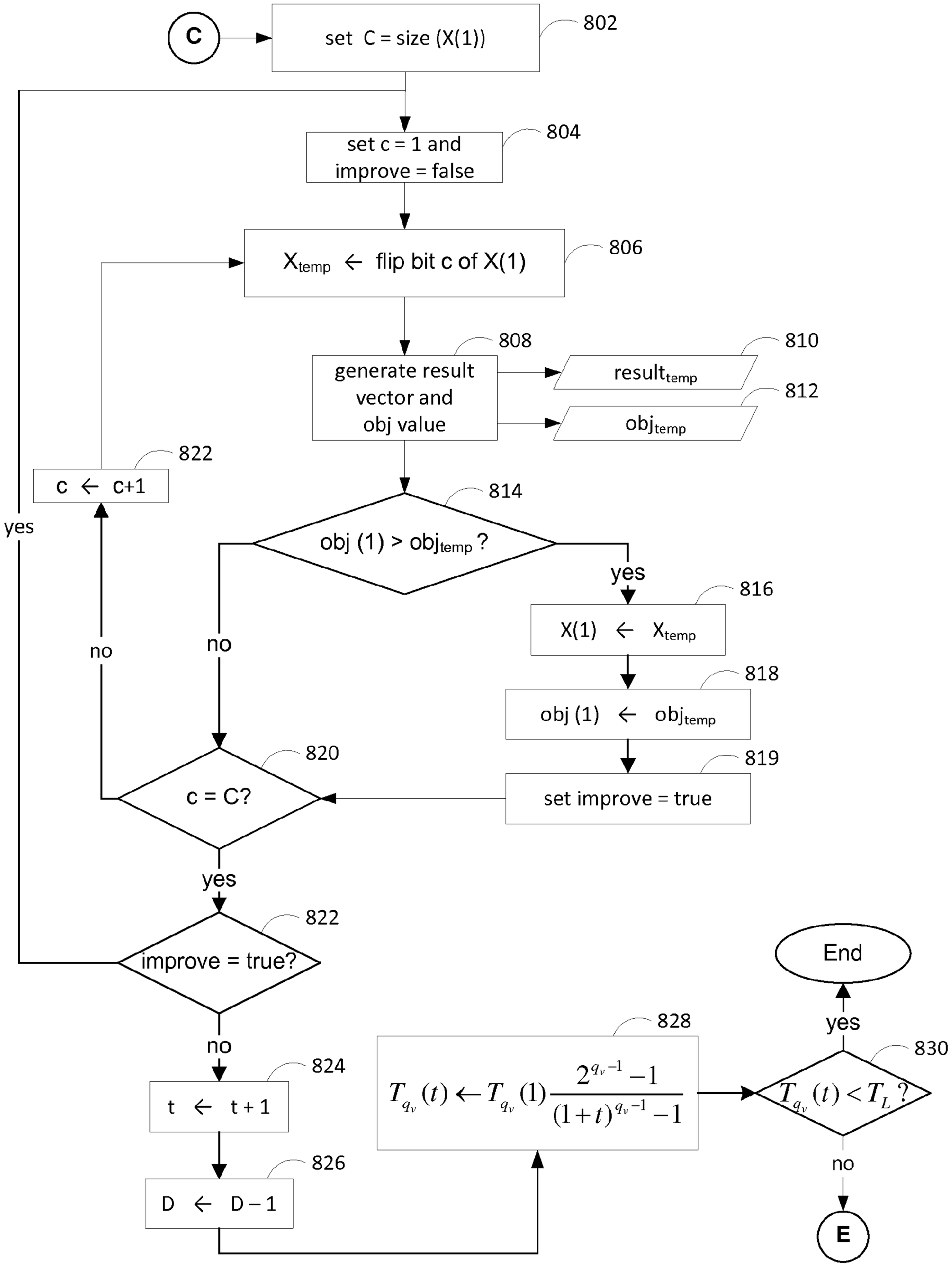
800
C
set C = size (X(1))
802
set c = 1 and
improve = false
804
X_{temp} ← flip bit c of X(1)
806
generate result
vector and
obj value
808
$result_{temp}$
810
obj_{temp}
812
obj (1) > obj_{temp} ?
814
yes
no
$X(1) \leftarrow X_{temp}$
816
$obj(1) \leftarrow obj_{temp}$
818
set improve = true
819
c = C?
820
yes
no
c ← c+1
822
improve = true?
822
yes
no
t ← t + 1
824
D ← D − 1
826
$T_{q_v}(t) \leftarrow T_{q_v}(1)\frac{2^{q_v-1}-1}{(1+t)^{q_v-1}-1}$
828
$T_{q_v}(t) < T_L$?
830
yes
no
End
E

FIG. 8

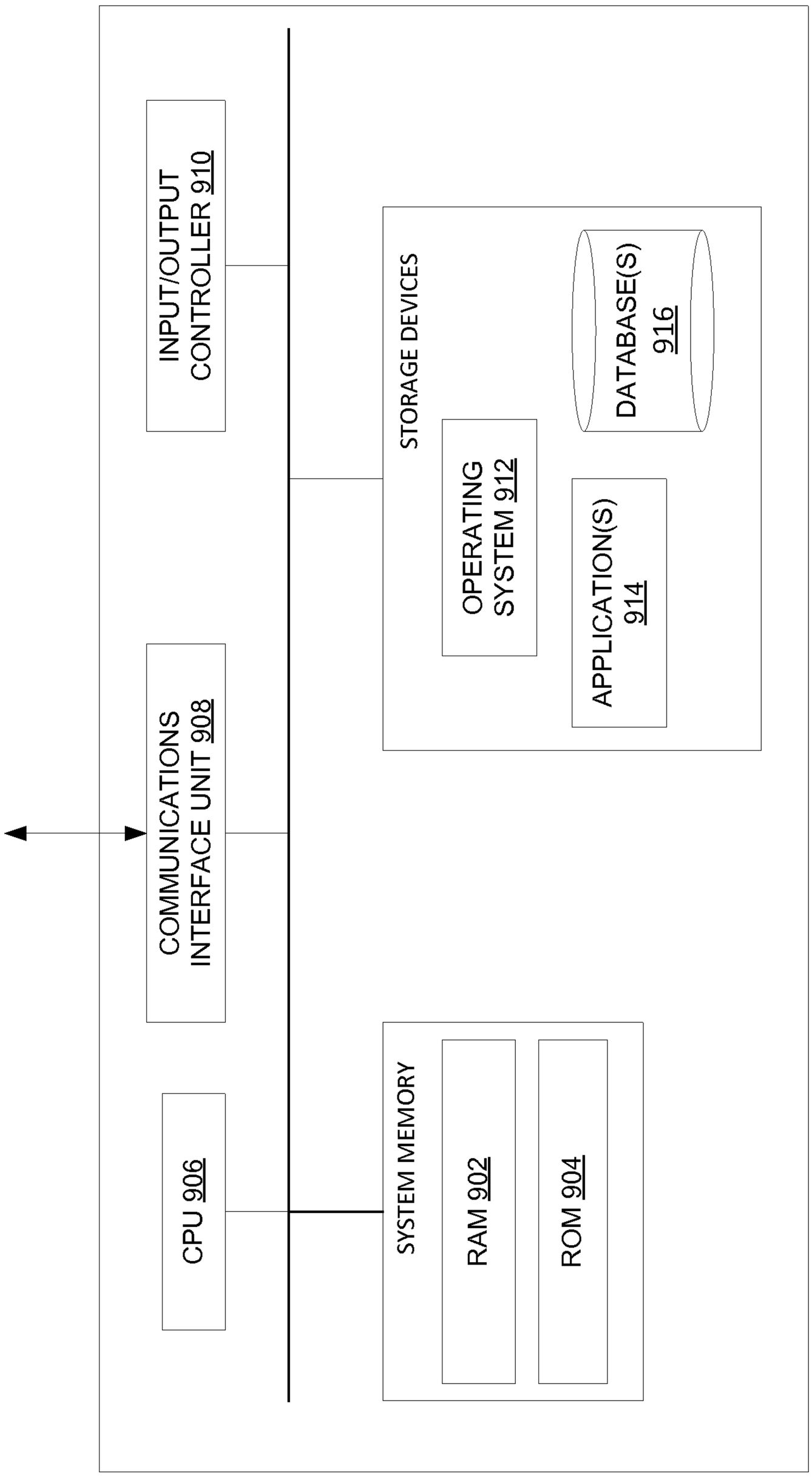
900
INPUT/OUTPUT CONTROLLER 910
COMMUNICATIONS INTERFACE UNIT 908
CPU 906
STORAGE DEVICES
OPERATING SYSTEM 912
APPLICATION(S) 914
DATABASE(S) 916
SYSTEM MEMORY
RAM 902
ROM 904

FIG. 9

1000

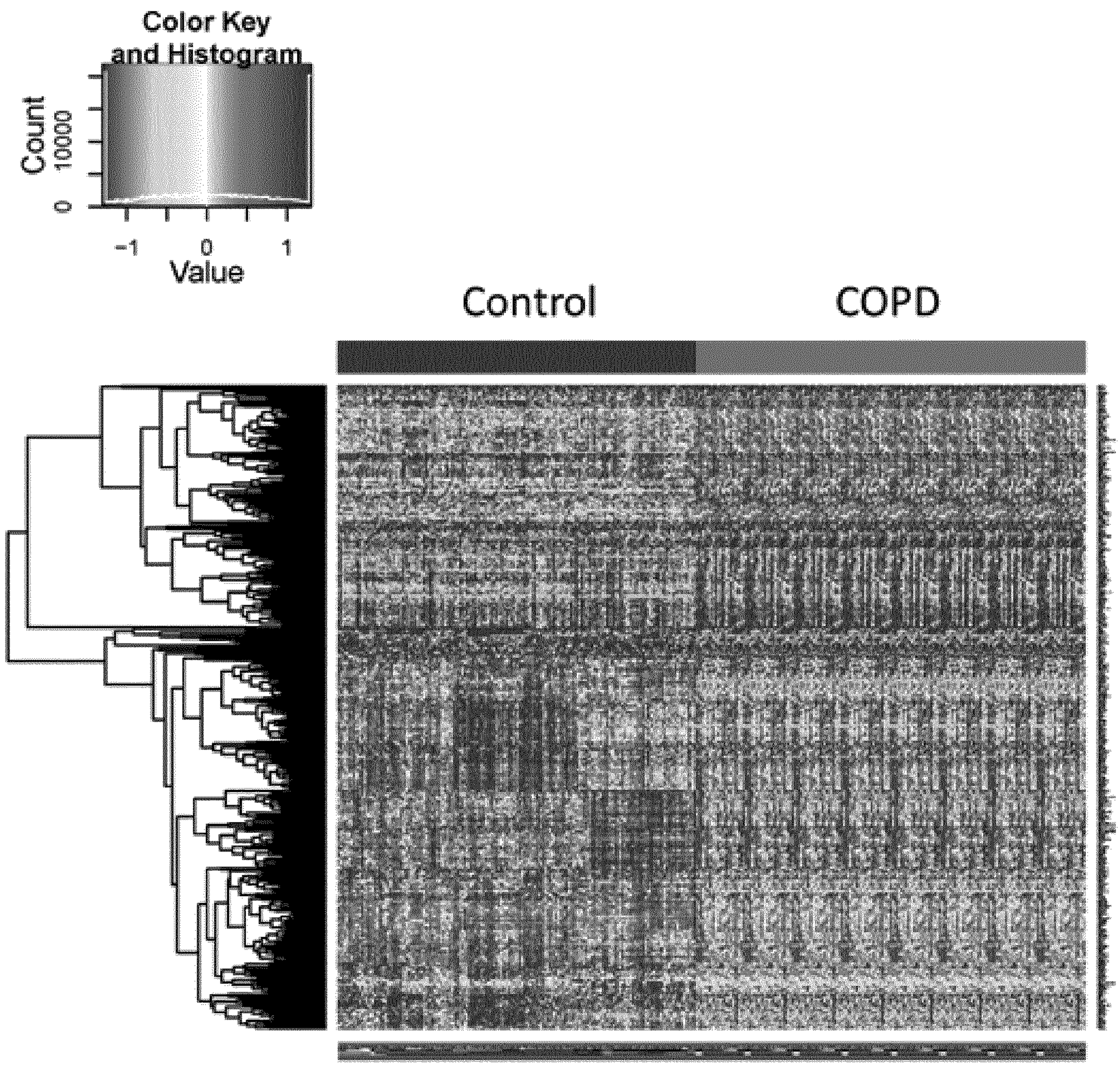
Color Key
and Histogram
Count
10000
0
-1
0
1
Value
Control
COPD

FIG. 10

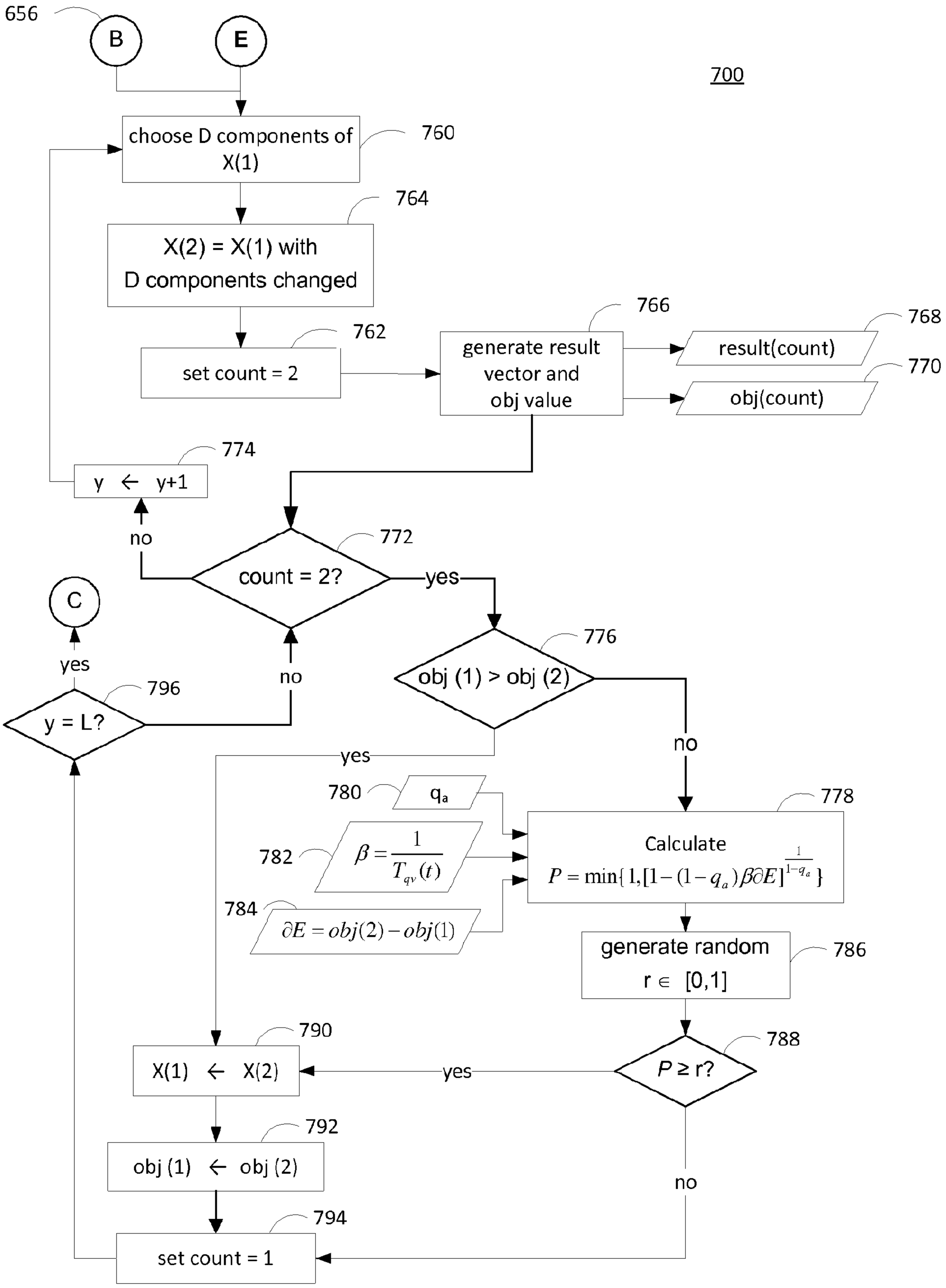
656
B
E
700
choose D components of X(1)
760
X(2) = X(1) with D components changed
764
set count = 2
762
generate result vector and obj value
766
result(count)
768
obj(count)
770
774
y ← y+1
no
count = 2?
772
yes
C
yes
796
y = L?
no
obj (1) > obj (2)
776
yes
no
780
q_a
782
$\beta = \frac{1}{T_{qv}(t)}$
784
$\partial E = obj(2) - obj(1)$
778
Calculate
$P = \min\{1, [1-(1-q_a)\beta\partial E]^{\frac{1}{1-q_a}}\}$
generate random
r ∈ [0,1]
786
788
$P \geq r$?
yes
no
790
X(1) ← X(2)
792
obj (1) ← obj (2)
794
set count = 1

FIG. 7