DATA PROCESSING METHOD AND APPARATUS

TECHNICAL FIELD

[0001] This application relates to the field of artificial intelligence, and specifically, to a data processing method and an apparatus.

BACKGROUND

[0002] With rapid development of Internet technologies, an information overload (information overload) problem occurs. To resolve the information overload problem, a recommender system (recommender system, RS) emerges. Click-through rate (click-through rate, CTR) prediction is an important step in the recommender system. Whether to recommend a commodity needs to be determined based on a predicted CTR. In addition to a single feature, a feature interaction also needs to be considered during CTR prediction. To represent the feature interaction, a factorization machine (factorization machine, FM) model is proposed. The FM model includes feature interaction items of all interactions of single features. In a conventional technology, a CTR prediction model is usually built based on an FM.

[0003] A quantity of feature interaction items in the FM model increases exponentially with an order of a feature interaction. Therefore, with an increasingly higher order, the feature interaction items become numerous. As a result, there is an extremely large computing workload in FM model training. To resolve this problem, feature interaction selection (feature interaction selection, FIS) is proposed. Manual feature interaction selection is time-consuming and labor-intensive. Therefore, automatic feature interaction selection (automatic feature interaction selection, AutoFIS) is proposed in the industry.

[0004] In an existing automatic feature interaction selection solution, search space formed by all possible feature interaction subsets is searched for an optimal subset, to implement feature interaction selection. A search process consumes high energy and consumes a large amount of computing power.

SUMMARY

[0005] This application provides a data processing method and an apparatus, to reduce a computing workload and computing power consumption of feature interaction selection.

[0006] According to a first aspect, a data processing method is provided. The method includes: adding an architecture parameter to each feature interaction item in a first model, to obtain a second model, where the first model is an FM-based model, and the architecture parameter represents importance of a corresponding feature interaction item; performing optimization on architecture parameters in the second model, to obtain the optimized architecture parameters; obtaining, based on the optimized architecture parameters and the first model or the second model, a third model through feature interaction item deletion.

[0007] The FM-based model represents a model built based on the FM principle, for example, includes any one of the following models: an FM model, a DeepFM model, an IPNN model, an AFM model, and an NFM model.

[0008] The third model may be a model obtained through feature interaction item deletion based on the first model.

[0009] Alternatively, the third model may be a model obtained through feature interaction item deletion based on the second model.

[0010] A feature interaction item to be deleted or retained (in other words, selected) may be determined in a plurality of manners.

[0011] Optionally, in an implementation, a feature interaction item corresponding to an architecture parameter in the optimized architecture parameters whose value is less than a threshold may be deleted.

[0012] The threshold represents a criterion for determining whether to retain a feature interaction item. For example, if a value of an optimized architecture parameter of a feature interaction item is less than the threshold, it indicates that the feature interaction item is to be deleted. If a value of an optimized architecture parameter of a feature interaction item reaches the threshold, it indicates that the feature interaction item is to be retained (in other words, selected).

[0013] The threshold may be determined based on an actual application requirement. For example, a value of the threshold may be obtained through model training. A manner of obtaining the threshold is not limited in this application.

[0014] Optionally, in another implementation, if values of some architecture parameters change to zero after optimization is completed, feature interaction items corresponding to the optimized architecture parameters whose values are not zero may be directly used as retained feature interaction items, to obtain the third model.

[0015] Optionally, in still another implementation, if values of some architecture parameters change to zero after optimization is completed, a feature interaction item corresponding to an architecture parameter whose value is less than the threshold may be further deleted from feature interaction items corresponding to the optimized architecture parameters whose values are not zero, to obtain the third model.

[0016] In an existing automatic feature interaction selection solution, all possible feature interaction subsets are used as search space (search space), and a best candidate subset is selected from n randomly selected candidate subsets by using a discrete algorithm as a selected feature interaction. Training needs to be performed once for evaluating each candidate subset, resulting in a large computing workload and high computing power consumption.

[0017] In this application, the architecture parameters are introduced into the FM-based model, so that feature interaction item selection can be performed through optimization on the architecture parameters. In other words, in this application, provided that optimization on the architecture parameters is performed once, feature interaction item selection can be performed, and training for a plurality of candidate subsets in a conventional technology is not required. Therefore, this can effectively reduce a computing workload of feature interaction selection to save computing power, and improve efficiency of feature interaction selection.

[0018] In addition, an existing automatic feature interaction selection solution cannot be applied to a deep learning model with a long training period, because of the large computing workload and high computing power consumption.

[0019] In this application, feature interaction selection can be performed through an optimization process of the architecture parameters. Alternatively, feature interaction item selection can be completed through one end-to-end model training process, so that a period for feature interaction item selection (or search) may be equivalent to a period for one model training. Therefore, feature interaction selection can be applied to a deep learning model with a long training period.

[0020] In the FM model in the conventional technology, because all feature interactions need

to be enumerated, it is difficult to extend to a higher order.

[0021] In this application, the architecture parameters are introduced into the FM-based model, so that feature interaction selection can be performed through optimization on the architecture parameters. Therefore, in the solution of this application, the feature interaction item in the FM-based model can be extended to a higher order.

[0022] Optimization may be performed on the architecture parameters in the second model by using a plurality of optimization algorithms (or optimizers).

[0023] With reference to the first aspect, in a possible implementation of the first aspect, optimization allows the optimized architecture parameters to be sparse.

[0024] In this application, optimization on the architecture parameters allows the architecture parameters to be sparse, facilitating subsequent feature interaction item deletion.

[0025] Optionally, in an implementation in which optimization allows the optimized architecture parameters to be sparse, the obtaining, based on the optimized architecture parameters and the first model or the second model, a third model through feature interaction item deletion includes: obtaining, based on the first model or the second model, the third model by deleting a feature interaction item corresponding to an architecture parameter in the optimized architecture parameters whose value is less than a threshold.

[0026] In an implementation, in the first model, the third model is obtained by deleting a feature interaction item corresponding to an architecture parameter in the optimized architecture parameters whose value is less than a threshold.

[0027] In another implementation, in the second model, the third model is obtained by deleting a feature interaction item corresponding to an architecture parameter in the optimized architecture parameters whose value is less than a threshold.

[0028] It should be understood that the third model is obtained through feature interaction item deletion based on the second model, so that the third model has the optimized architecture parameters that represent importance of the feature interaction items. Subsequently, importance of the feature interaction items can be further learned through training of the third model.

[0029] With reference to the first aspect, in a possible implementation of the first aspect, optimization allows a value of an architecture parameter of at least one feature interaction item to be equal to zero after optimization is completed.

[0030] It is assumed that a feature interaction item corresponding to an architecture parameter

whose value is zero after optimization is completed is considered as an unimportant feature interaction item. That optimization allows a value of an architecture parameter of at least one feature interaction item to be equal to zero after optimization is completed may be considered as allowing the value of the architecture parameter of the unimportant feature interaction item to be equal to zero after optimization is completed.

[0031] Optionally, optimization is performed on the architecture parameters in the second model using a gRDA optimizer, where the gRDA optimizer allows the value of the architecture parameter of the at least one feature interaction item to tend to zero during an optimization process.

[0032] In embodiments of this application, optimization on the architecture parameters allows some architecture parameters to tend to zero, which is equivalent to removing some unimportant feature interaction items in an architecture parameter optimization process. In other words, optimization on the architecture parameters implements architecture parameter optimization and feature interaction item selection. This can improve efficiency of feature interaction selection and reduce a computing workload and computing power consumption.

[0033] In addition, in the architecture parameter optimization process, removing some unimportant feature interaction items can prevent noise generated by these unimportant feature interaction items. In this case, a model gradually evolves into an ideal model in the architecture parameter optimization process. In addition, prediction of other parameters (for example, architecture parameters and model parameters of an unremoved feature interaction item) in the model can be more accurate.

[0034] Optionally, in an implementation in which optimization allows the optimized architecture parameters to be sparse and allows a value of an architecture parameter of at least one feature interaction item to be equal to zero after optimization is completed, the obtaining, based on the optimized architecture parameters and the first model or the second model, a third model through feature interaction item deletion includes: obtaining the third model by deleting a feature interaction item other than feature interaction items corresponding to the optimized architecture parameters.

[0035] Optionally, in the first model, the third model is obtained by deleting the feature interaction item other than the feature interaction items corresponding to the optimized architecture parameters. In other words, the third model is obtained through feature interaction item deletion based on the first model.

[0036] Optionally, the second model obtained through architecture parameter optimization is used as the third model. In other words, the third model is obtained through feature interaction item deletion based on the second model.

[0037] It should be understood that the third model is obtained through feature interaction item deletion based on the second model, so that the third model has the optimized architecture parameters that represent importance of the feature interaction items. Subsequently, importance of the feature interaction items can be further learned through training of the third model.

[0038] Optionally, in an implementation in which optimization allows the optimized architecture parameters to be sparse and allows a value of an architecture parameter of at least one feature interaction item to be equal to zero after optimization is completed, the obtaining, based on the optimized architecture parameters and the first model or the second model, a third model through feature interaction item deletion includes: obtaining the third model by deleting a feature interaction item other than feature interaction items corresponding to the optimized architecture parameters and deleting the feature interaction item corresponding to the architecture parameter in the optimized architecture parameters whose value is less than the threshold.

[0039] Optionally, in the first model, the third model is obtained by deleting the feature interaction item other than the feature interaction items corresponding to the optimized architecture parameters and deleting the feature interaction item corresponding to the architecture parameter in the optimized architecture parameters whose value is less than the threshold.

[0040] Optionally, in the second model obtained through architecture parameter optimization, the third model is obtained by deleting a feature interaction item corresponding to an architecture parameter in the optimized architecture parameters whose value is less than a threshold.

[0041] It should be understood that the third model is obtained through feature interaction item deletion based on the second model, so that the third model has the optimized architecture parameters that represent importance of the feature interaction items. Subsequently, importance of the feature interaction items can be further learned through training of the third model.

[0042] With reference to the first aspect, in a possible implementation of the first aspect, the method further includes: performing optimization on model parameters in the second model, where optimization includes scalarization processing on the model parameters in the second model.

[0043] The model parameters indicate weight parameters other than the architecture parameters of the feature interaction item in the second model. In other words, the model

parameters represent an original parameter in the first model.

[0044] In an implementation, optimization includes performing batch normalization (batch normalization, BN) processing on the model parameters in the second model.

[0045] It should be understood that scalarization processing is performed on the model parameters of the feature interaction item, to decouple the model parameters from the architecture parameters of the feature interaction item. In this case, the architecture parameters can more accurately reflect importance of the feature interaction items, further improving optimization accuracy of the architecture parameters.

[0046] With reference to the first aspect, in a possible implementation of the first aspect, the performing optimization on architecture parameters in the second model and the performing optimization on model parameters in the second model include: performing simultaneous optimization on both the architecture parameters and the model parameters in the second model by using same training data, to obtain the optimized architecture parameters.

[0047] In other words, in each round of training in an optimization process, simultaneous optimization is performed on both the architecture parameters and the model parameters based on a same batch of training data.

[0048] Alternatively, the architecture parameters and the model parameters in the second model are considered as decision variables at a same level, and simultaneous optimization is performed on both the architecture parameters and the model parameters in the second model, to obtain the optimized architecture parameters.

[0049] In this application, one-level optimization processing is performed on the architecture parameters and the model parameters in the second model, to implement optimization on the architecture parameters in the second model, so that simultaneous optimization can be performed on the architecture parameters and the model parameters. Therefore, time consumed in an optimization process of the architecture parameters in the second model can be reduced, to further help improve efficiency of feature interaction item selection.

[0050] With reference to the first aspect, in a possible implementation of the first aspect, the method further includes: training the third model to obtain a click-through rate CTR prediction model or a conversion rate CVR prediction model.

[0051] According to a second aspect, a data processing method is provided. The method includes: inputting data of a target object into a click-through rate CTR prediction model or a

conversion rate CVR prediction model, to obtain a prediction result of the target object; and determining a recommendation status of the target object based on the prediction result of the target object.

[0052] The CTR prediction model or the CVR prediction model is obtained through the method in the first aspect.

[0053] Training of a third model includes the following step: train the third model by using a training sample of the target object, to obtain the CTR prediction model or the CVR prediction model.

[0054] Optionally, optimization on architecture parameters includes the following step: perform simultaneous optimization on both the architecture parameters and model parameters in a second model by using the same training data as that in the training sample of the target object, to obtain the optimized architecture parameters.

[0055] According to a third aspect, a data processing apparatus is provided. The apparatus includes the following units.

[0056] A first processing unit is configured to add an architecture parameter to each feature interaction item in a first model, to obtain a second model, where the first model is a factorization machine FM-based model, and the architecture parameter represents importance of a corresponding feature interaction item.

[0057] A second processing unit is configured to perform optimization on architecture parameters in the second model, to obtain the optimized architecture parameters.

[0058] A third processing unit is configured to obtain, based on the optimized architecture parameters and the first model or the second model, a third model through feature interaction item deletion.

[0059] With reference to the third aspect, in a possible implementation of the third aspect, the second processing unit performs optimization on the architecture parameters, to allow the optimized architecture parameters to be sparse.

[0060] With reference to the third aspect, in a possible implementation of the third aspect, the third processing unit is configured to obtain, based on the first model or the second model, the third model by deleting a feature interaction item corresponding to an architecture parameter in the optimized architecture parameters whose value is less than a threshold.

[0061] With reference to the third aspect, in a possible implementation of the third aspect, the

second processing unit performs optimization on the architecture parameters, to allow a value of an architecture parameter of at least one feature interaction item to be equal to zero after optimization is completed.

[0062] With reference to the third aspect, in a possible implementation of the third aspect, the third processing unit is configured to optimize the architecture parameters in the second model using a generalized regularized dual averaging gRDA optimizer, where the gRDA optimizer allows the value of the architecture parameter of the at least one feature interaction item to tend to zero during an optimization process.

[0063] With reference to the third aspect, in a possible implementation of the third aspect, the second processing unit is further configured to perform optimization on model parameters in the second model, where optimization includes scalarization processing on the model parameters in the second model.

[0064] With reference to the third aspect, in a possible implementation of the third aspect, the second processing unit is configured to perform batch normalization BN processing on the model parameters in the second model.

[0065] With reference to the third aspect, in a possible implementation of the third aspect, the second processing unit is configured to perform simultaneous optimization on both the architecture parameters and model parameters in a second model by using same training data, to obtain the optimized architecture parameters.

[0066] With reference to the third aspect, in a possible implementation of the third aspect, the apparatus further includes: a training unit, configured to train the third model, to obtain a CTR prediction model or a CVR prediction model.

[0067] According to a fourth aspect, a data processing apparatus is provided. The apparatus includes the following units.

[0068] A first processing unit is configured to input data of a target object into a click-through rate CTR prediction model or a conversion rate CVR prediction model, to obtain a prediction result of the target object.

[0069] A first processing unit is configured to determine a recommendation status of the target object based on the prediction result of the target object.

[0070] The CTR prediction model or the CVR prediction model is obtained through the method in the first aspect.

[0071] Training of a third model includes the following step: train the third model by using a training sample of the target object, to obtain the CTR prediction model or the CVR prediction model.

[0072] Optionally, optimization on architecture parameters includes the following step: perform simultaneous optimization on both the architecture parameters and model parameters in a second model by using the same training data as that in the training sample of the target object, to obtain the optimized architecture parameters.

[0073] According to a fifth aspect, an image processing apparatus is provided. The apparatus includes: a memory, configured to store a program; and a processor, configured to execute the program stored in the memory, where when the program stored in the memory is being executed, the processor is configured to perform the method in the first aspect or the second aspect.

[0074] According to a sixth aspect, a computer-readable medium is provided. The computer-readable medium stores program code to be executed by a device, and the program code is used to perform the method in the first aspect or the second aspect.

[0075] According to a seventh aspect, a computer program product including instructions is provided. When the computer program product runs on a computer, the computer is enabled to perform the method in the first aspect or the second aspect.

[0076] According to an eighth aspect, a chip is provided. The chip includes a processor and a data interface. The processor reads, through the data interface, instructions stored in a memory, to perform the method in the first aspect or the second aspect.

[0077] Optionally, in an implementation, the chip may further include a memory and the memory stores instructions; the processor is configured to execute the instructions stored in the memory, and when the instructions are executed, the processor is configured to perform the methods in the first aspect or the second aspect.

[0078] According to a ninth aspect, an electronic device is provided. The electronic device includes the apparatus provided in the third aspect, the fourth aspect, the fifth aspect, or the sixth aspect.

[0079] It can be learned from the foregoing description that, in this application, the architecture parameters are introduced into the FM-based model, so that feature interaction item selection can be performed through optimization on the architecture parameters. In other words, in this application, feature interaction item selection can be performed through optimization on the

architecture parameters, and training for a plurality of candidate subsets in the conventional technology is not required. Therefore, this can effectively reduce a computing workload of feature interaction selection to save computing power, and improve efficiency of feature interaction selection.

[0080] In addition, in the solution provided in this application, the feature interaction item in the FM-based model can be extended to a higher order.

BRIEF DESCRIPTION OF DRAWINGS

[0081] FIG. 1 is a schematic diagram of an FM model architecture;

[0082] FIG. 2 is a schematic diagram of FM model training;

[0083] FIG. 3 is a schematic flowchart of a data processing method according to an embodiment of this application;

[0084] FIG. 4 is a schematic diagram of an FM model architecture according to an embodiment of this application;

[0085] FIG. 5 is another schematic flowchart of a data processing method according to an embodiment of this application;

[0086] FIG. 6 is still another schematic flowchart of a data processing method according to an embodiment of this application;

[0087] FIG. 7 is a schematic block diagram of a data processing apparatus according to an embodiment of this application;

[0088] FIG. 8 is another schematic block diagram of a data processing apparatus according to an embodiment of this application;

[0089] FIG. 9 is still another schematic block diagram of a data processing apparatus according to an embodiment of this application; and

[0090] FIG. 10 is a schematic diagram of a hardware architecture of a chip according to an embodiment of this application.

DESCRIPTION OF EMBODIMENTS

[0091] The following describes technical solutions of this application with reference to accompanying drawings.

[0092] With rapid development of current technologies, there is an increasing amount of data. To solve an information overload (information overload) problem, a recommender system (recommender system, RS) is proposed. The recommender system sends historical behavior, interests, preferences, or demographic features of a user to a recommendation algorithm, and then uses the recommendation algorithm to generate a list of items that the user may be interested in.

[0093] In the recommender system, click-through rate (click-through rate, CTR) prediction (or further including conversion rate (conversion rate, CVR) prediction) is a very important step. Whether to recommend a commodity needs to be determined based on a predicted click-through rate. In addition to a single feature, a feature interaction also needs to be considered during CTR prediction. The feature interaction is very important for recommendation ranking. A factorization machine (factorization machine, FM) can reflect the feature interaction. The factorization machine may be referred to as an FM model for short.

[0094] Based on a maximum order of the feature interaction item, the FM model may be referred to as a *-order FM model. For example, an FM model whose feature interaction item has a maximum of a second order may be referred to as a second-order FM model, and an FM model whose feature interaction item has a maximum of a third order may be referred to as a third-order FM model.

[0095] An order of the feature interaction item indicates a specific quantity of features corresponding to the feature interaction item. For example, an interaction item of two features may be referred to as a second-order feature interaction item, and an interaction item of three features may be referred to as a third-order feature interaction item.

[0096] In an example, the second-order FM model is shown in the following formula (1):

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{m} w_i x_i + \sum_{i=1}^{m} \sum_{j=i+1}^{m} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$
$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle := \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f} \quad (1).$$

[0097] $\mathbf{x}$ indicates a feature vector, x_i indicates an ith feature, and x_j indicates a jth feature. m indicates a quantity of features, and may also be referred to as a feature field (field). w_0 indicates a global offset, and $w_0 \in R$. w_i indicates strength of the ith feature, and $\mathbf{w} \in R^m$. $\mathbf{v}_i$ indicates an auxiliary vector of the ith feature x_i. $\mathbf{v}_j$ indicates an auxiliary vector of the jth

feature x_j. k is a quantity of $\mathbf{v}_i$ and $\mathbf{v}_j$. A two-dimensional matrix $\mathbf{v} \in R^{m \times k}$.

[0098] $x_i x_j$ indicates a combination of the ith feature x_i and the jth feature x_j.

[0099] $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ indicates an inner product of $\mathbf{v}_i$ and $\mathbf{v}_j$, and indicates interaction between the ith feature x_i and the jth feature x_j. $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ may also be understood as a weight parameter of a feature interaction item $x_i x_j$, for example, $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ may be denoted as w_{ij}.

[00100] In this specification, $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ is denoted as a weight parameter of a feature interaction item $x_i x_j$.

[00101] Optionally, the formula (1) may also be expressed as the following formula (2):

$$l_{fm} = \langle w, x \rangle + \sum_{i=1}^{m} \sum_{j>i}^{m} \langle e_i, e_j \rangle \quad (2).$$

[00102] In the formula (2), $\langle e_i, e_j \rangle$ indicates $\langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$ in the formula (1), and $\langle w, x \rangle$ indicates $w_0 + \sum_{i=1}^{m} w_i x_i$ in the formula (1).

[00103] In another example, the third-order FM model is shown in the following formula (3):

$$l_{fm}^{3rd} = \langle w, x \rangle + \sum_{i=1}^{m} \sum_{j>i}^{m} \langle e_i, e_j \rangle + \sum_{i=1}^{m} \sum_{j>i}^{m} \sum_{t>j}^{m} \langle e_i, e_j, e_t \rangle \quad (3).$$

[00104] The FM model includes feature interaction items of all interactions of single features. For example, a second-order FM model shown in the formula (1) or the formula (2) includes feature interaction items of all second-order feature interactions of single features. For another example, the third-order FM model shown in the formula (3) includes feature interaction items of all second-order feature interactions of single features and feature interaction items of all third-order feature interactions of single features.

[00105] For example, in the industry, an operation of obtaining an auxiliary vector $\mathbf{v}_i$ of a feature x_i is referred to as embedding (embedding), and an operation of building a feature interaction item based on the feature x_i and the auxiliary vector $\mathbf{v}_i$ thereof is referred to as interaction (interaction). FIG. 1 is a schematic diagram of an FM model architecture. As shown in FIG. 1, an FM model may be considered as a neural network model, and includes an input (input) layer, an embedding (embedding) layer, an interaction (interaction) layer, and an output (output) layer. The input layer is used to generate a feature. A field 1, a field, ..., and a field m indicate a

quantity of features. The embedding layer is used to generate an auxiliary vector of the feature. The interaction layer is used to generate a feature interaction item based on the feature and the auxiliary vector of the feature. The output layer is used to output the FM model.

[00106] In the conventional technology, CTR prediction or CVR prediction is usually based on an FM.

[00107] In the current technology, an FM-based model includes an FM model, a DeepFM model, an IPNN model, an AFM model, an NFM model, and the like.

[00108] As an example instead of a limitation, a procedure of building an FM model is shown in FIG. 2.

[00109] S210: Enumerate and enter feature interaction items into the FM model.

[00110] For example, the FM model is built by using the formula (1) or the formula (3).

[00111] S220: Train the FM model until convergence, to obtain an FM model that can be put into use.

[00112] After FM model training is completed, online inference may be performed by using the trained FM model, as shown in step S230 in FIG. 2.

[00113] As described above, the FM model includes the feature interaction items of all interactions of single features. Therefore, FM model training has an extremely large computing workload and consumes a lot of time.

[00114] In addition, it can be learned from the formula (1) and the formula (3) that a quantity of feature interaction items in the FM model increases sharply with increases in the quantity of features and an order of feature interaction.

[00115] For example, in the formula (1), as the quantity m of features increases, the quantity of feature interaction items increases exponentially. For another example, as the order of feature interaction increases in a switch from a second-order FM model to a third-order FM model, the quantity of feature interaction items in the FM model increases greatly.

[00116] Therefore, increases in the quantity of features and the order of feature interaction results in a huge burden to an inference delay and a computing workload of the FM model. Therefore, a maximum quantity of features and the order of feature interaction that can be accommodated by the FM model are limited. For example, it is difficult to extend the FM model in the current technology to a higher order.

[00117] To resolve this problem, feature interaction selection (feature interaction selection, FIS)

is proposed.

[00118] In some conventional technologies, feature interaction selection is performed in a manual selection manner. To select good feature interactions may take many years of exploration by engineers. This manual selection manner consumes a large amount of manpower, and may miss an important feature interaction item.

[00119] To address a disadvantage of manual selection, an automatic feature interaction selection (automatic feature interaction selection, AutoFIS) solution is proposed in the industry. Compared with manual selection, valuable feature interactions can be selected through automatic feature interaction selection in a short period of time.

[00120] In the current technology, an automatic feature interaction selection solution is proposed. In the solution, all possible feature interaction subsets are used as search space (search space), and a best candidate subset is selected from n randomly selected candidate subsets by using a discrete algorithm as a selected feature interaction. Training needs to be performed once for evaluating each candidate subset, resulting in a large computing workload and high computing power consumption. In addition, when each candidate subset is evaluated, entire model training can improve evaluation accuracy, but may cause huge search energy consumption (search cost); mini-batch (mini-batch) training that is used for approximation may result in inaccurate evaluation. In addition, in this solution, as the order of the feature interaction increases, the search space increases exponentially, which increases energy consumption in a search process.

[00121] Therefore, the existing automatic feature interaction selection solution has a large computing workload, high energy consumption in a search process, and high computing power consumption.

[00122] For the foregoing problem, this application provides an automatic feature interaction selection solution. Compared with the conventional technology, this solution can reduce computing power consumption of automatic feature interaction selection, and improve efficiency of automatic feature interaction selection.

[00123] FIG. 3 is a schematic flowchart of a data processing method 300 according to an embodiment of this application. The method 300 includes the following steps: S310, S320, and S330.

[00124] S310: Add an architecture parameter to each feature interaction item in a first model, to obtain a second model.

[00125] The first model is a model based on a factorization machine (factorization machine, FM). In other words, the first model includes feature interaction items of all interactions of single features, or the first model enumerates feature interaction items of all interactions.

[00126] For example, the first model may be any one of the following FM-based models: an FM model, a DeepFM model, an IPNN model, an AFM model, and an NFM model.

[00127] As an example, the first model is a second-order FM model shown in the formula (1) or the formula (2), or the first model is a third-order FM model shown in the formula (3).

[00128] In this application, feature interaction item selection is performed, and the first model may be considered as a model on which a feature interaction item is to be deleted.

[00129] Adding an architecture parameter to each feature interaction item in the first model means adding a coefficient to each feature interaction item in the first model. In this application, the coefficient is referred to as an architecture parameter. The architecture parameter represents importance of a corresponding feature interaction item. A model obtained by adding the architecture parameter each feature interaction item in the first model is denoted as a second model.

[00130] In an example, assuming that the first model is a second-order FM model shown in the formula (1), the second model is shown in the following formula (4):

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{m} w_i x_i + \sum_{i=1}^{m} \sum_{j=i+1}^{m} \alpha_{(i,j)} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$
$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle := \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f} \quad (4).$$

[00131] $\mathbf{x}$ indicates a feature vector, x_i indicates an ith feature, and x_j indicates a jth feature. *m* indicates a feature dimension, and may also be referred to as a feature field (field). w_0 indicates a global offset, and $w_0 \in R$. w_i indicates strength of the ith feature, and $\mathbf{w} \in R^m$. $\mathbf{v}_i$ indicates an auxiliary vector of the ith feature x_i. $\mathbf{v}_j$ indicates an auxiliary vector of the jth feature x_j. *k* is a quantity of $\mathbf{v}_i$ and $\mathbf{v}_j$. A two-dimensional matrix $\mathbf{v} \in R^{m \times k}$.

[00132] $x_i x_j$ indicates a combination of the ith feature x_i and the jth feature x_j.

[00133] $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ indicates a weight parameter of a feature interaction item $x_i x_j$, and $\alpha_{(i,j)}$ indicates an architecture parameter of the feature interaction item $x_i x_j$.

[00134] $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ indicates an inner product of $\mathbf{v}_i$ and $\mathbf{v}_j$, and indicates interaction between the ith feature x_i and the jth feature x_j. $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ may also be understood as a weight parameter of a feature interaction item, for example, $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ may be denoted as w_{ij}.

[00135] Assuming that the first model is expressed as the second-order FM model shown in the formula (2), the second model may be expressed as the following formula (5):

$$l_{autoFIS} = \langle w, x \rangle + \sum_{i=1}^{m} \sum_{j>i}^{m} \alpha_{(i,j)} \langle e_i, e_j \rangle \quad (5).$$

[00136] $\alpha_{(i,j)}$ indicates an architecture parameter of a feature interaction item.

[00137] In another example, if the first model is a third-order FM model shown in the formula (3), the second model is shown in the following formula (6):

$$l_{autoFIS} = \langle w, x \rangle + \sum_{i=1}^{m} \sum_{j>i}^{m} \alpha_{(i,j)} \langle e_i, e_j \rangle + \sum_{i=1}^{m} \sum_{j>i}^{m} \sum_{t>j}^{m} \alpha_{(i,j,t)} \langle e_i, e_j, e_t \rangle \quad (6).$$

[00138] $\alpha_{(i,j)}$ and $\alpha_{(i,j,t)}$ indicate architecture parameters of feature interaction items.

[00139] For ease of understanding and description, the following is agreed in this specification: An original weight parameter (for example, $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ in the formula (4)) of the feature interaction item in the first model is referred to as a model parameter.

[00140] In other words, in the second model, each feature interaction item has two types of coefficient parameters: a model parameter and an architecture parameter.

[00141] FIG. 4 is a schematic diagram of feature interaction item selection according to an embodiment of this application. An embedding layer and an interaction layer in FIG. 4 have the same meanings as those of the embedding layer and the interaction layer in FIG. 1. As shown in FIG. 4, in this embodiment of this application, architecture parameters $\alpha_{(i,j)}$ ($\alpha_{(1,2)}$, $\alpha_{(1,m)}$, and $\alpha_{(m-1,m)}$ shown in FIG. 4) are added to feature interaction items at the interaction layer. The interaction layer in FIG. 4 may be considered as a first model, and the interaction layer to which the architecture parameters $\alpha_{(i,j)}$ are added to the feature interaction items may be considered as a second model.

[00142] S320: Perform optimization on architecture parameters in the second model, to obtain the optimized architecture parameters.

[00143] For example, optimization is performed on the architecture parameters in the second

model by using training data, to obtain the optimized architecture parameters.

[00144] For example, the optimized architecture parameters may be considered as optimal values α^* of the architecture parameters in the second model.

[00145] In embodiments of this application, the architecture parameter represents importance of a corresponding feature interaction item. Therefore, optimization on the architecture parameter is equivalent to learning importance of each feature interaction item or a contribution degree of each feature interaction item to model prediction. In other words, the optimized architecture parameter represents importance of the learned feature interaction item.

[00146] In other words, in embodiments of this application, contribution (or importance) of each feature interaction item may be learned by using the architecture parameters in an end-to-end manner.

[00147] S330: Obtain, based on the optimized architecture parameters and the first model or the second model, a third model through feature interaction item deletion.

[00148] The third model may be a model obtained through feature interaction item deletion based on the first model.

[00149] Alternatively, the third model may be a model obtained through feature interaction item deletion based on the second model.

[00150] A feature interaction item to be deleted or retained (in other words, selected) may be determined in a plurality of manners.

[00151] Optionally, in an implementation, a feature interaction item corresponding to an architecture parameter in the optimized architecture parameters whose value is less than a threshold may be deleted.

[00152] The threshold represents a criterion for determining whether to retain a feature interaction item. For example, if a value of an optimized architecture parameter of a feature interaction item is less than the threshold, it indicates that the feature interaction item is to be deleted. If a value of an optimized architecture parameter of a feature interaction item reaches the threshold, it indicates that the feature interaction item is to be retained (in other words, selected).

[00153] The threshold may be determined based on an actual application requirement. For example, a value of the threshold may be obtained through model training. A manner of obtaining the threshold is not limited in this application.

[00154] Still refer to FIG. 4. Assuming that the optimized architecture parameter $\alpha_{(1,2)}$ is less than the threshold, a feature interaction item corresponding to the architecture parameter $\alpha_{(1,2)}$ may be deleted. Assuming that the optimized architecture parameter $\alpha_{(1,m)}$ reaches the threshold, a feature interaction item corresponding to the architecture parameter $\alpha_{(1,m)}$ may be retained. A next-layer model is obtained by deleting the feature interaction item based on the optimized architecture parameter, as shown in FIG. 4. The third model in the embodiment in FIG. 3 is, for example, the next-layer model shown in FIG. 4.

[00155] As an example instead of a limitation, as shown in FIG. 4, an operation of determining, based on the optimized architecture parameter, whether to delete a corresponding feature interaction item may be denoted as a selection gate.

[00156] It should be noted that FIG. 4 is merely an example rather than a limitation.

[00157] Optionally, in another implementation, if values of some architecture parameters change to zero after optimization is completed, feature interaction items corresponding to the optimized architecture parameters whose values are not zero may be directly used as retained feature interaction items, to obtain the third model.

[00158] Optionally, in still another implementation, if values of some architecture parameters change to zero after optimization is completed, a feature interaction item corresponding to an architecture parameter whose value is less than the threshold may be further deleted from feature interaction items corresponding to the optimized architecture parameters whose values are not zero, to obtain the third model.

[00159] In this specification, a "model obtained through feature interaction item deletion" can be replaced with a "model obtained through feature interaction item selection".

[00160] As described above, in an existing automatic feature interaction selection solution, all possible feature interaction subsets are used as search space (search space), and a best candidate subset is selected from n randomly selected candidate subsets by using a discrete algorithm as a selected feature interaction. Training needs to be performed once for evaluating each candidate subset, resulting in a large computing workload and high computing power consumption.

[00161] In embodiments of this application, the architecture parameters are introduced into the FM-based model, so that feature interaction item selection can be performed through optimization on the architecture parameters. In other words, in this application, provided that optimization on

the architecture parameters is performed once, feature interaction item selection can be performed, and training for a plurality of candidate subsets in a conventional technology is not required. Therefore, this can effectively reduce a computing workload of feature interaction selection to save computing power, and improve efficiency of feature interaction selection.

[00162] In addition, in an existing automatic feature interaction selection solution, feature interaction selection is performed by searching for a candidate subset in search space (search space). It may be understood that, in the conventional technology, feature interaction selection is resolved as a discrete issue, in other words, a discrete feature interaction candidate set is searched for.

[00163] In this embodiment of this application, feature interaction selection is performed through optimization on the architecture parameters that are introduced into the FM-based model. It may be understood that, in this embodiment of this application, an existing problem of searching for the discrete feature interaction candidate set is continuous, in other words, feature interaction selection is resolved as a continuous issue. For example, the automatic feature interaction selection solution provided in this application may be expressed as a feature interaction search solution based on continuous search space. In other words, in this embodiment of this application, an operation of introducing the architecture parameters into the FM-based model may be considered as continuous modeling for automatic feature interaction item selection.

[00164] In addition, an existing automatic feature interaction selection solution cannot be applied to a deep learning model with a long training period, because of the large computing workload and high computing power consumption.

[00165] In embodiments of this application, feature interaction selection can be performed through an optimization process of the architecture parameters. Alternatively, feature interaction item selection can be completed through one end-to-end model training process, so that a period for feature interaction item selection (or search) may be equivalent to a period for one model training. Therefore, feature interaction selection can be applied to a deep learning model with a long training period.

[00166] As described above, in the FM model in the conventional technology, because all feature interactions need to be enumerated, it is difficult to extend to a higher order.

[00167] In embodiments of this application, the architecture parameters are introduced into the FM-based model, so that feature interaction selection can be performed through optimization on

the architecture parameters. Therefore, in the solution provided in embodiments of this application, the feature interaction item in the FM-based model can be extended to a higher order.

[00168] For example, an FM model built by using the solution provided in embodiments of this application may be extended to a third order or a higher order.

[00169] For another example, the DeepFM model built by using the solution provided in this embodiment of this application may be extended to a third order or a higher order.

[00170] In embodiments of this application, the architecture parameters are introduced into the conventional FM-based model, so that feature interaction selection can be performed through optimization on the architecture parameters. In other words, in embodiments of this application, the FM-based model that includes the architecture parameters is built, and feature interaction selection can be performed by performing optimization on the architecture parameters. A method for building the FM-based model that includes the architecture parameters is adding the architecture parameter before each feature interaction item in the conventional FM-based model.

[00171] As shown in FIG. 3, the method 300 may include step S340.

[00172] S340: Train the third model.

[00173] Step S340 may also be understood as performing model training again. It may be understood that the feature interaction item is deleted by using step S310, step S320, and step S330. In step S340, the model obtained through feature interaction item deletion is retrained.

[00174] In step S340, the third model may be directly trained, or the third model may be trained after a L1 regular term and/or a L2 regular term are/is added to the third model.

[00175] For example, an objective of training the third model may be determined based on an application requirement.

[00176] For example, assuming that a click-through rate (click-through rate, CTR) prediction model is to be obtained, the third model is trained by using the obtained CTR prediction model as the training objective, to obtain the CTR prediction model.

[00177] For another example, assuming that a conversion rate (conversion rate, CVR) prediction model is to be obtained, the third model is trained by using the conversion rate, CVR prediction model as the training objective, to obtain the CVR prediction model.

[00178] In step S320, for example, optimization may be performed on the architecture parameters in the second model by using a plurality of optimization algorithms (or optimizers).

[00179] A first optimization algorithm:

[00180] Optionally, in step S320, optimization is performed on the architecture parameters, to allow the optimized architecture parameters to be sparse.

[00181] For example, in step S320, optimization is performed on the architecture parameters in the second model by using least absolute shrinkage and selection operator (least absolute shrinkage and selection operator, Lasso) regularization (regularization).

[00182] That the formula (5) is used in the second model is used as an example. In step S320, the architecture parameters in the second model are optimized by using the following formula (7):

$$L_{search} = L_{\alpha,w}\left(y, \hat{y}_M\right) + \lambda \sum_{i,j>i} \left|\alpha_{(i,j)}\right| \quad (7).$$

[00183] $L_{\alpha,w}\left(y, \hat{y}_M\right)$ indicates a loss function. y indicates a model observed value. $\hat{y}_M$ indicates a model predicted value. λ indicates a constant, and its value may be assigned based on a specific requirement.

[00184] It should be understood that the formula (7) indicates a constraint condition for architecture parameter optimization.

[00185] The optimized architecture parameters are sparse, facilitating subsequent feature interaction item deletion.

[00186] Optionally, in an embodiment, in step S320, optimization on the architecture parameters allows the optimized architecture parameters to be sparse; in step S330, the third model is obtained, based on the first model or the second model, by deleting a feature interaction item corresponding to an architecture parameter in the optimized architecture parameters whose value is less than a threshold.

[00187] The threshold represents a criterion for determining whether to retain a feature interaction item. For example, if a value of an optimized architecture parameter of a feature interaction item is less than the threshold, it indicates that the feature interaction item is to be deleted. If a value of an optimized architecture parameter of a feature interaction item reaches the threshold, it indicates that the feature interaction item is to be retained (in other words, selected).

[00188] The threshold may be determined based on an actual application requirement. For example, a value of the threshold may be obtained through model training. A manner of obtaining the threshold is not limited in this application.

[00189] In embodiments of this application, optimization on the architecture parameters allows the architecture parameters to be sparse, facilitating feature interaction item selection.

[00190] It should be understood that the architecture parameters in the second model represent importance or a contribution degree of a corresponding feature interaction. If a value of an optimized architecture parameter is less than the threshold, for example, close to zero, it indicates that a feature interaction item corresponding to the architecture parameter is not important or has a very low contribution degree. Deleting (or referred to as removing or cutting) such feature interaction item can remove noise introduced by the feature interaction item, reduce energy consumption, and improve an inference speed of a model.

[00191] Therefore, deleting the feature interaction item corresponding to the architecture parameter in the optimized architecture parameters whose value is less than the threshold is an appropriate feature interaction selection operation.

[00192] A second optimization algorithm:

[00193] Optionally, in step S320, optimization is performed on the architecture parameters, so that the optimized architecture parameters are sparse and a value of an architecture parameter of at least one feature interaction item is equal to zero after optimization is completed.

[00194] It is assumed that the feature interaction item corresponding to the architecture parameter whose value is zero after optimization is completed is considered as an unimportant feature interaction item. Optimization on the architecture parameters in step S320 may be considered as allowing the value of the architecture parameter of the unimportant feature interaction item to be equal to zero after optimization is completed.

[00195] In other words, optimization on the architecture parameters allows the value of the architecture parameter of the at least one feature interaction item to tend to zero during an optimization process.

[00196] For example, in step S320, the architecture parameters in the second model are optimized using a generalized regularized dual averaging (generalized regularized dual averaging, GRDA) optimizer. The GRDA optimizer allows the architecture parameters to be sparse, and allows the value of the architecture parameter of the at least one feature interaction item to gradually tend to zero during an optimization process.

[00197] For example, in step S320, the architecture parameters in the second model are optimized by using the following formula (8):

$$\alpha_{t+1} = \arg\min_{\alpha}\left\{\alpha^{T}\left(-\alpha_{0} + \gamma\sum_{i=0}^{t}\nabla L\left(\alpha_{t}; y_{i+1}\right) + g\left(t,\gamma\right)\left\|\alpha\right\|_{1} + \frac{1}{2}\left\|\alpha\right\|_{2}^{2}\right)\right\} \quad (8).$$

[00198] γ indicates a learning rate. y_{i+1} indicates a model observation value. $g(t, \gamma) = c\gamma^{1/2}(t\gamma)^{\mu}$. c and μ represent adjustable hyperparameters. An objective of adjusting c and μ is to find a balance between a model accuracy and sparseness of an architecture parameter α.

[00199] It should be understood that the formula (8) indicates a constraint condition for architecture parameter optimization.

[00200] It should be further understood that, in this embodiment, in step S320, the second model obtained through architecture parameter optimization is a model obtained through feature interaction item selection.

[00201] In this application, optimization on the architecture parameters allows some architecture parameters to tend to zero, which is equivalent to removing some unimportant feature interaction items in an architecture parameter optimization process. In other words, optimization on the architecture parameters implements architecture parameter optimization and feature interaction item selection. This can improve efficiency of feature interaction selection and reduce a computing workload and computing power consumption.

[00202] In addition, in the architecture parameter optimization process, removing some unimportant feature interaction items can prevent noise generated by these unimportant feature interaction items. In this case, a model gradually evolves into an ideal model in the architecture parameter optimization process. In addition, prediction of other parameters (for example, architecture parameters and model parameters of an unremoved feature interaction item) in the model can be more accurate.

[00203] Optionally, in an embodiment, in step S320, optimization is performed on the architecture parameters, so that the optimized architecture parameters are sparse and a value of an architecture parameter of at least one feature interaction item is equal to zero after optimization is completed; in step S330, the third model may be obtained in the following plurality of manners.

[00204] Manner (1)

[00205] In step S330, feature interaction items corresponding to the optimized architecture parameters may be directly used as selected feature interaction items, and the third model is obtained based on the selected feature interaction items.

[00206] For example, in the first model, the feature interaction items corresponding to the

optimized architecture parameters are used as the selected feature interaction items, and remaining feature interaction items are deleted, to obtain the third model.

[00207] For another example, a model obtained through architecture parameter optimization on the second model is directly used as the third model.

[00208] Manner (2)

[00209] In step S330, the feature interaction item corresponding to the architecture parameter in the optimized architecture parameters whose value is less than the threshold is deleted from feature interaction items corresponding to the optimized architecture parameters, to obtain the third model.

[00210] The threshold may be determined based on an actual application requirement. For example, a value of the threshold may be obtained through model training. A manner of obtaining the threshold is not limited in this application.

[00211] For example, in the first model, the third model is obtained by deleting the feature interaction item other than the feature interaction items corresponding to the optimized architecture parameters and deleting the feature interaction item corresponding to the architecture parameter in the optimized architecture parameters whose value is less than the threshold.

[00212] For another example, in the second model obtained through architecture parameter optimization, the third model is obtained by deleting the feature interaction item corresponding to the architecture parameter in the optimized architecture parameters whose value is less than the threshold.

[00213] In embodiments of this application, optimization on the architecture parameters allows some architecture parameters to tend to zero, which is equivalent to removing some unimportant feature interaction items in an architecture parameter optimization process. In other words, optimization on the architecture parameters implements architecture parameter optimization and feature interaction item selection. This can improve efficiency of feature interaction selection and reduce a computing workload and computing power consumption.

[00214] It can be learned from the foregoing description of step S320 that, in step S330, an implementation of obtaining the third model through feature interaction item selection may be determined based on an optimization manner of the architecture parameters in step S320. The following describes implementations of obtaining the third model in the following two cases.

[00215] In a first case, in step S320, optimization is performed on the architecture parameters,

to allow the optimized architecture parameters to be sparse.

[00216] In step S330, the third model is obtained by deleting the feature interaction item corresponding to the architecture parameter in the optimized architecture parameters whose value is less than a threshold. For the threshold, refer to the foregoing description. Details are not described herein again.

[00217] As an example instead of a limitation, optimized architecture parameters obtained through architecture parameter optimization (namely, optimization convergence) are denoted as optimal values α^* of the architecture parameters. Based on the optimal values α^*, specific feature interaction items that are to be retained or deleted are determined. For example, if an optimal value $\alpha^*_{(i,j)}$ of an architecture parameter of a feature interaction item reaches the threshold, the feature interaction item should be retained; if an optimal value $\alpha^*_{(i,j)}$ of an architecture parameter of a feature interaction item is less than the threshold, the feature interaction item should be deleted.

[00218] For example, in the second model, for each feature interaction item, a selection gate $\psi_{(i.j)}$ indicating whether the feature interaction item is retained in a model is set. The second model may be expressed as the following formula (9):

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{m} w_i x_i + \sum_{i=1}^{m} \sum_{j=i+1}^{m} \alpha_{(i,j)} \psi_{(i.j)} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \tag{9}$$

[00219] A value of the switch item $\psi_{(i.j)}$ may be represented by using the following formula (10):

$$\psi_{(i.j)} = \begin{cases} 1 & \left|\alpha^*_{(i,j)}\right| \geq thr \\ 0 & \left|\alpha^*_{(i,j)}\right| < thr \end{cases} \tag{10}$$

[00220] thr indicates a threshold.

[00221] A feature interaction item whose switch item $\psi_{(i.j)}$ is 0 is deleted from the second model, to obtain the third model through feature interaction item selection.

[00222] In this embodiment, setting of the switch item $\psi_{(i.j)}$ may be considered as a criterion for determining whether to retain a feature interaction item.

[00223] Alternatively, the third model may be a model obtained through feature interaction item

deletion based on the first model.

[00224] For example, the feature interaction item whose switch item $\psi_{(i.j)}$ is 0 is deleted from the first model, to obtain the third model through feature interaction item selection.

[00225] Alternatively, the third model may be a model obtained through feature interaction item deletion based on the second model.

[00226] For example, the feature interaction item whose switch item $\psi_{(i.j)}$ is 0 is deleted from the second model, to obtain the third model through feature interaction item selection.

[00227] It should be understood that, in this embodiment, the third model has optimized architecture parameters that represent importance of feature interaction items. Subsequently, importance of the feature interaction items can be further learned through training of the third model.

[00228] In a second case, in step S320, optimization is performed on the architecture parameters, so that the optimized architecture parameters are sparse and a value of an architecture parameter of at least one feature interaction item is equal to zero after optimization is completed.

[00229] Optionally, in step S330, the third model is obtained by deleting the feature interaction item other than the feature interaction items corresponding to the optimized architecture parameters.

[00230] In an example, in step S330, the third model is obtained by deleting the feature interaction item other than the feature interaction items corresponding to the optimized architecture parameters in the first model. In other words, the third model is obtained through feature interaction item deletion based on the first model.

[00231] In another example, in step S330, the second model obtained through architecture parameter optimization is used as the third model. In other words, the third model is obtained through feature interaction item deletion based on the second model.

[00232] Optionally, in step S330, the third model is obtained by deleting the feature interaction item other than the feature interaction items corresponding to the optimized architecture parameters and deleting the feature interaction item corresponding to the architecture parameter in the optimized architecture parameters whose value is less than the threshold.

[00233] In an example, in step S330, the third model is obtained by deleting the feature interaction item other than the feature interaction items corresponding to the optimized architecture

parameters and deleting the feature interaction item corresponding to the architecture parameter in the optimized architecture parameters whose value is less than the threshold in the first model. In other words, the third model is obtained through feature interaction item deletion based on the first model.

[00234] In another example, in step S330, in the second model obtained through architecture parameter optimization, the third model is obtained by deleting a feature interaction item corresponding to an architecture parameter in the optimized architecture parameters whose value is less than a threshold. In other words, the third model is obtained through feature interaction item deletion based on the second model.

[00235] It should be understood that, in an embodiment in which the third model is obtained through feature interaction item deletion based on the second model, the third model has the optimized architecture parameters that represent importance of the feature interaction items. Subsequently, importance of the feature interaction items can be further learned through training of the third model.

[00236] It may be understood from the formula (4), the formula (5), or the formula (6) that the second model includes two types of parameters: an architecture parameter and a model parameter. The model parameters indicate weight parameters other than the architecture parameters of the feature interaction item in the second model. For example, in an example of the second model expressed in the formula (4), $\alpha_{(i,j)}$ indicates architecture parameters of the feature interaction items, and $\left\langle \mathbf{v}_i, \mathbf{v}_j \right\rangle$ indicates model parameters of the feature interaction items. For example, in an example of the second model expressed in the formula (5), $\alpha_{(i,j)}$ indicates architecture parameters of the feature interaction items, and $\left\langle e_i, e_j \right\rangle$ may indicate model parameters of the feature interaction items.

[00237] It may be understood that, an architecture parameter optimization process involves architecture parameter training and model parameter training. In other words, optimization on the architecture parameters in the second model in step S320 is accompanied by optimization on the model parameters in the second model.

[00238] For example, in the embodiment shown in FIG. 3, the method 300 further includes performing optimization on model parameters in the second model, where optimization includes scalarization processing on the model parameters.

[00239] In each round of training in the model parameter optimization process, scalarization processing is performed on the model parameters in the second model.

[00240] For example, scalarization processing is performed on the model parameters in the second model by performing batch normalization (batch normalization, BN) on the model parameters in the second model.

[00241] For example, in an example of the second model expressed in the formula (5), the architecture parameters in the second model are optimized by using the following formula (11):

$$\left\langle e_i, e_j \right\rangle_{BN} = \frac{\left\langle e_i, e_j \right\rangle_{\mathrm{B}} - \mu_{\mathrm{B}}\left(\left\langle e_i, e_j \right\rangle_{\mathrm{B}}\right)}{\sqrt{\sigma_{\mathrm{B}}^2\left(\left\langle e_i, e_j \right\rangle_{\mathrm{B}}\right) + \theta}} \tag{11}$$

[00242] $\left\langle e_i, e_j \right\rangle_{BN}$ indicates batch normalization of $\left\langle e_i, e_j \right\rangle$.

[00243] $\left\langle e_i, e_j \right\rangle_{\mathrm{B}}$ indicates mini-batch (mini-batch) data of $\left\langle e_i, e_j \right\rangle$.

[00244] $\mu_{\mathrm{B}}\left(\left\langle e_i, e_j \right\rangle_{\mathrm{B}}\right)$ indicates an average value of mini-batch (mini-batch) data of $\left\langle e_i, e_j \right\rangle$.

[00245] $\sigma_{\mathrm{B}}^2\left(\left\langle e_i, e_j \right\rangle_{\mathrm{B}}\right)$ indicates a variance of mini-batch (mini-batch) data of $\left\langle e_i, e_j \right\rangle$.

[00246] θ indicates disturbance.

[00247] Still refer to FIG. 4. Batch normalization (BN) shown in FIG. 4 indicates BN processing on the model parameters in the second model.

[00248] Scalarization processing is performed on the model parameters of the feature interaction items, to decouple the model parameters from the architecture parameters of the feature interaction items. In this case, the architecture parameters can more accurately reflect importance of the feature interaction items, further improving optimization accuracy of the architecture parameters. This is explained as follows.

[00249] It should be understood that e_i is continuously updated and changed in a model training process. After inner product is performed on e_i and e_j, in other words, $\left\langle e_i, e_j \right\rangle$, a scale (scale) of the inner product is constantly updated. It is assumed that $\alpha_{(i,j)}\left\langle e_i, e_j \right\rangle$ may be obtained through $\left(\frac{\alpha_{(i,j)}}{\eta}\right)\left(\eta \cdot \left\langle e_i, e_j \right\rangle\right)$, where a first term $\left(\frac{\alpha_{(i,j)}}{\eta}\right)$ is coupled to a second term $\left(\eta \cdot \left\langle e_i, e_j \right\rangle\right)$. If the second item $\left(\eta \cdot \left\langle e_i, e_j \right\rangle\right)$ is not scalarized, the first item $\left(\frac{\alpha_{(i,j)}}{\eta}\right)$ cannot

absolutely represent importance of the second item, causing great instability to a system.

[00250] In this embodiment of this application, scalarization processing is performed on the model parameters of the feature interaction item, so that $\alpha_{(i,j)}\langle e_i, e_j \rangle$ cannot be obtained through $\left(\frac{\alpha_{(i,j)}}{\eta}\right)\left(\eta \cdot \langle e_i, e_j \rangle\right)$, in other words, the model parameters of the feature interaction item can be decoupled from the architecture parameters.

[00251] The model parameters of the feature interaction item are decoupled from the architecture parameters, so that the architecture parameters can more accurately reflect importance of the feature interaction items, further improving optimization accuracy of the architecture parameters.

[00252] In other words, scalarization processing is performed on the model parameters of the feature interaction items, to decouple the model parameters from the architecture parameters of the feature interaction items, so that there is no coupling effect between the model parameters and the architecture parameters of the feature interaction items to cause large instability in the system.

[00253] As described above, the second model includes two types of parameters: the architecture parameters and the model parameters. An architecture parameter optimization process involves architecture parameter training and model parameter training. In other words, optimization on the architecture parameters in the second model in step S320 is accompanied by optimization on the model parameters in the second model.

[00254] For ease of understanding and description, in the following description, an architecture parameter in the second model is denoted as α, and a model parameter in the second model is denoted as w (corresponding to $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ in the formula (4)).

[00255] Optionally, in the embodiment shown in FIG. 3, optimization processing on the architecture parameter α in the second model and optimization processing on the model parameter w in the second model include two-level (two-level) optimization processing on the architecture parameter α and the model parameter w in the second model.

[00256] In other words, in step S320, two-level (two-level) optimization processing is performed on the architecture parameter α and the model parameter w in the second model, to obtain the optimized architecture parameter α^*.

[00257] In this embodiment, the architecture parameter α in the second model is used as a

model hyperparameter for optimization, and the model parameter w in the second model is used as a model parameter for optimization. In other words, the architecture parameter α is used as a high-level decision variable, and the model parameter w is used as a low-level decision variable. Any value of the high-level decision variable α corresponds to a different model.

[00258] Optionally, when a model corresponding to any value of the high-level decision variable α is evaluated, an optimal model parameter woptimal is obtained through entire training of the model. In other words, each time a candidate value of the architecture parameter α is evaluated, entire training of a model corresponding to the candidate value is performed.

[00259] Optionally, when a model corresponding to any value of the high-level decision variable α is evaluated, wt+1 obtained by updating the model in one step by using mini-batch (mini-batch) data is used to replace the optimal model parameter woptimal.

[00260] Optionally, in the embodiment shown in FIG. 3, optimization processing on the architecture parameter α in the second model and optimization processing on the model parameter w in the second model include: simultaneous optimization on both the architecture parameter α and the model parameter w in the second model by using same training data.

[00261] In other words, in step S320, simultaneous optimization processing is performed on both the architecture parameter α and the model parameter w in the second model, to obtain the optimized architecture parameter α^* by using the same training data.

[00262] In this embodiment, in each round of training in an optimization process, simultaneous optimization is performed on both the architecture parameter α and the model parameter w based on a same batch of training data. Alternatively, the architecture parameter and the model parameter in the second model are considered as decision variables at a same level, and simultaneous optimization is performed on both the architecture parameter α and the model parameter w in the second model, to obtain the optimized architecture parameter α^*.

[00263] In this embodiment, optimization processing performed on the architecture parameter α and the model parameter w in the second model may be referred to as one-level (one-level) optimization processing.

[00264] For example, the architecture parameter α in the second model and the model parameter w freely explore their feasible fields in stochastic gradient descent (stochastic gradient descent, SGD) optimization until convergence.

[00265] For example, the architecture parameter α and the model parameter w in the second model are optimized by using the following formula (12):

$$\begin{aligned} \alpha_t &= \alpha_{t-1} - \eta_t \cdot \partial_\alpha L_{train}\left(w_{t-1}, \alpha_{t-1}\right) \\ w_t &= w_{t-1} - \delta_t \cdot \partial_w L_{train}\left(w_{t-1}, \alpha_{t-1}\right) \end{aligned} \quad (12).$$

[00266] α_t indicates an architecture parameter after optimization in step t is performed. α_{t-1} indicates an architecture parameter after optimization in step t–1 is performed. w_t indicates a model parameter after optimization in step t is performed. w_{t-1} indicates a model parameter after optimization in step t–1 is performed. η_t indicates an optimization rate of an architecture parameter during optimization in step t. δ_t indicates a learning rate of a model parameter during optimization in step t. $L_{train}\left(w_{t-1}, \alpha_{t-1}\right)$ indicates a loss function value of a loss function on a test set during optimization in step t. $\partial_\alpha L_{train}\left(w_{t-1}, \alpha_{t-1}\right)$ indicates a gradient of the loss function on the test set relative to the architecture parameter α during optimization in step t. $\partial_w L_{train}\left(w_{t-1}, \alpha_{t-1}\right)$ indicates a gradient of the loss function on the test set relative to the model parameter w during optimization in step t.

[00267] In this embodiment, one-level optimization processing is performed on the architecture parameters and the model parameters in the second model, to implement optimization on the architecture parameters in the second model, so that the architecture parameters and the model parameters can be simultaneously optimized. Therefore, time consumed in an optimization process of the architecture parameters in the second model can be reduced, to further help improve efficiency of feature interaction item selection.

[00268] After step S330 is completed, in other words, feature interaction item selection is completed, the third model is a model obtained through feature interaction item selection.

[00269] In step S340, the third model is trained.

[00270] The third model may be trained, or the third model may be trained after a L1 regular term and/or a L2 regular term are/is added to the third model.

[00271] An objective of training the third model may be determined based on an application requirement.

[00272] For example, assuming that a click-through rate (click-through rate, CTR) prediction

model is to be obtained, the third model is trained by using the obtained CTR prediction model as the training objective, to obtain the CTR prediction model.

[00273] For another example, assuming that a conversion rate (conversion rate, CVR) prediction model is to be obtained, the third model is trained by using the conversion rate CVR prediction model as the training objective, to obtain the CVR prediction model.

[00274] Alternatively, the third model is a model obtained through feature interaction item deletion based on the first model. For details, refer to the foregoing description of step S330. Details are not described herein again.

[00275] Alternatively, the third model is a model obtained through feature interaction item deletion based on the second model. For details, refer to the foregoing description of step S330. Details are not described herein again.

[00276] It should be understood that through feature interaction item deletion (or selection), the architecture parameters are retained in the model to train the model, so that importance of the feature interaction item can be further learned.

[00277] It can be learned from the foregoing description that, in embodiments of this application, the architecture parameters are introduced into the FM-based model, so that feature interaction item selection can be performed through optimization on the architecture parameters. In other words, in this application, feature interaction item selection can be performed through optimization on the architecture parameters, and training for a plurality of candidate subsets in the conventional technology is not required. Therefore, this can effectively reduce a computing workload of feature interaction selection to save computing power, and improve efficiency of feature interaction selection.

[00278] In addition, in the solution provided in this embodiment of this application, the feature interaction item in the FM-based model can be extended to a higher order.

[00279] FIG. 5 is another schematic flowchart of an automatic feature interaction selection method 500 according to an embodiment of this application.

[00280] First, training data is obtained.

[00281] For example, assuming that a quantity of features is m, the training data is obtained for features of m fields.

[00282] S510: Enumerate and enter feature interaction items into an FM-based model.

[00283] The FM-based model may be the FM model shown in the foregoing formula (1) or

formula (2), or may be any one of the following FM-based models: a DeepFM model, an IPNN model, an AFM model, and an NFM model.

[00284] The enumerating and entering feature interaction items into an FM-based model means building, based on all interactions of m features, feature interaction items based on an FM model.

[00285] It should be understood that when the feature interaction items are being built, auxiliary vectors of m features are involved.

[00286] For example, the embedding (embedding) layer shown in FIG. 1 or FIG. 3 may be used to obtain the auxiliary vectors of the m features. A technology of obtaining the auxiliary vectors of m features through the embedding (embedding) layer belongs to a conventional technology. Details are not described in this specification.

[00287] S520: Introduce architecture parameters to the FM-based model. Specifically, one coefficient parameter is added to each feature interaction item in the FM-based model, and the coefficient parameter is referred to as an architecture parameter.

[00288] Step S520 is corresponding to step S310 in the foregoing embodiment. For specific descriptions, refer to the foregoing description.

[00289] The FM-based model in the embodiment shown in FIG. 5 is corresponding to the first model in the embodiment shown in FIG. 3, and a model obtained by adding an architecture parameter to the FM-based model in the embodiment shown in FIG. 5 is corresponding to the second model in the embodiment shown in FIG. 3.

[00290] S530: Perform optimization on the architecture parameters until convergence, to obtain the optimized architecture parameters.

[00291] Step S530 is corresponding to step S320 in the foregoing embodiment. For specific descriptions, refer to the foregoing description.

[00292] S540: Perform feature interaction item deletion based on the optimized architecture parameters, to obtain a model through feature interaction item deletion.

[00293] Step S540 is corresponding to step S330 in the foregoing embodiment. For specific descriptions, refer to the foregoing description.

[00294] The model obtained through feature interaction item deletion in the embodiment shown in FIG. 5 is corresponding to the third model in the embodiment shown in FIG. 3.

[00295] S550: Train the model obtained through feature interaction item deletion until convergence, to obtain a CTR prediction model.

[00296] Step S550 is corresponding to step S340 in the foregoing embodiment. For specific descriptions, refer to the foregoing description.

[00297] After the CTR prediction model is obtained through training, online inference may be performed on the CTR prediction model.

[00298] For example, data of a target object is input into the CTR prediction model, and the CTR prediction model outputs a click-through rate of the target object. Whether to recommend the target object may be determined based on the click-through rate.

[00299] An automatic feature interaction selection solution provided in this embodiment of this application may be applied to any factorization machine (factorization machine, FM)-based model, for example, an FM model, a DeepFM model, an IPNN model, an AFM model, and an NFM model.

[00300] In an example, the automatic feature interaction selection solution provided in this embodiment of this application may be applied to an existing FM model.

[00301] For example, the architecture parameters are introduced into the existing FM model, so that importance of each feature interaction item is obtained through optimization on the architecture parameter. Then feature interaction selection is performed based on the importance of each feature interaction item, to finally obtain an FM model through feature interaction selection.

[00302] It should be understood that, the solution in this application is applied to the FM model, so that feature interaction item selection of the FM model can be efficiently performed, to support extending the feature interaction item of the FM model to a higher order.

[00303] In another example, the automatic feature interaction selection solution provided in this embodiment of this application may be applied to an existing DeepFM model.

[00304] For example, the architecture parameters are introduced into the existing DeepFM model, so that importance of each feature interaction item is obtained through optimization on the architecture parameter. Then feature interaction selection is performed based on the importance of each feature interaction item, to finally obtain a DeepFM through feature interaction selection.

[00305] It should be understood that, the solution in this application is applied to the DeepFM model, so that feature interaction item selection of the DeepFM model can be efficiently performed.

[00306] As shown in FIG. 6, this embodiment of this application further provides a data processing method 600. The method 600 includes the following steps: S610 and S620.

[00307] S610: Input data of a target object into a CTR prediction model or a CVR prediction model, to obtain a prediction result of a target object.

[00308] For example, the target object is a commodity.

[00309] S620: Determine a recommendation status of the target object based on the prediction result of the target object.

[00310] The CTR prediction model or the CVR prediction model is obtained through the method 300 provided in the foregoing embodiment, that is, the CTR prediction model or the CVR prediction model is obtained through step S310 to step S340 in the foregoing embodiment. Refer to the foregoing description. Details are not described herein again.

[00311] In step S340, a third model is trained by using a training sample of the target object, to obtain the CTR prediction model or the CVR prediction model.

[00312] Optionally, in step S320, simultaneous optimization is performed on both architecture parameters and model parameters in a second model by using the same training data as that in the training sample of the target object, to obtain the optimized architecture parameters.

[00313] Alternatively, the architecture parameters and the model parameters in the second model are considered as decision variables at a same level, and simultaneous optimization is performed on both the architecture parameters and the model parameters in the second model by using the training sample of the target object, to obtain the optimized architecture parameters.

[00314] Simulation testing: CTR prediction accuracy of online A/B testing is significantly improved, and inference energy consumption is greatly reduced.

[00315] As an example, simulation testing shows that when the feature interaction selection solution provided in this application is applied to the DeepFM model of a recommender system and online A/B testing is performed, a game download rate can be increased by 20%, a CTR prediction accuracy rate can be relatively improved by 20.3%, and a conversion rate (conversion rate, CVR) can be relatively improved by 20.1%. Therefore, a model inference speed can be effectively improved.

[00316] In an example, an FM model and a DeepFM model are obtained on a public dataset Avazu by using the solution provided in this application. Results of comparing performance of the FM model and the DeepFM model obtained by using the solution in this application with performance of another model in the industry are shown in Table 1 and Table 2. Table 1 indicates comparison of second-order models, and Table 2 indicates comparison of third-order models. In the second-order mode, the highest order of a feature interaction item in a model is second order. In the third-order mode, the highest order of a feature interaction item in a model is third order.

Table 1

Model (model)	Public dataset Avazu					
	AUC	Log loss	Top	Time (s)	Search+re-train cost (min)	Rel.Impr
FM	0.7793	0.3805	100%	0.51	0+3	0
FwFM	0.7822	0.3784	100%	0.52	0+4	0.37%
AFM	0.7806	0.3794	100%	1.92	0+14	0.17%
FFM	0.7831	0.3781	100%	0.24	0+6	0.49%
DeepFM	0.7836	0.3776	100%	0.76	0+6	0.55%
GBDT+LR	0.7721	0.3841	100%	0.45	8+3	–0.92%
GBDT+FFM	0.7835	0.3777	100%	2.66	6+21	0.54%
AutoFM (2nd)	0.7831	0.3778	29%	0.23	4+2	0.49%
AutoDeepFM (2nd)	0.7852	0.3765	24%	0.48	7+4	0.76%

Table 2

Model (model)	Public dataset Avazu					
	AUC	Log loss	Top	Time (s)	Search+re-train cost (min)	Rel.Impr
FM (3rd)	0.7843	0.3772	100%	5.70	0+21	0.64%
DeepFM (3rd)	0.7854	0.3765	100%	5.97	0+23	0.78%
AutoFM (3rd)	0.7860	0.3762	25%/2%	0.33	22+5	0.86%
AutoDeepFM (3rd)	0.7870	0.3756	21%/10%	0.94	24+10	0.99%

[00317] In Table 1 and Table 2, meanings of horizontal table headers are as follows.

[00318] AUC (area under curve) indicates an area under a curve. Log loss indicates log of a loss value. Top indicates a proportion of feature interaction items retained through feature interaction item selection. Time indicates a time period for a model to infer two million samples.

Search+re-train cost indicates a time period consumed for search and retraining, where a time period consumed for search indicates a time period consumed for step S320 and step S330 in the foregoing embodiment, and a time period consumed for retraining indicates a time period consumed for step S340 in the foregoing embodiment. Rel.Impr indicates a relative increase value.

[00319] In Table 1, meanings of vertical table headers are as follows.

[00320] FM, FwFM, AFM, FFM, and DeepFM represent FM-based models in the conventional technology. GBDT+LR and GBDT+FFM indicate models that use manual feature interaction selection in the conventional technology.

[00321] AutoFM (2nd) represents a second-order FM model obtained by using the solution provided in this embodiment of this application. AutoDeepFM (2nd) represents a third-order DeepFM model obtained by using the solution provided in this embodiment of this application.

[00322] In Table 2, meanings of vertical table headers are as follows.

[00323] FM (3rd) represents a third-order FM model in the conventional technology. DeepFM (3rd) represents a third-order DeepFM model in the conventional technology.

[00324] AutoFM (3rd) represents a third-order FM model obtained by using the solution provided in this embodiment of this application. AutoDeepFM (3rd) represents a third-order DeepFM model obtained by using the solution provided in this embodiment of this application.

[00325] It can be learned from Table 1 and Table 2 that, compared with the conventional technology, CTR prediction performed by using the FM model or the DeepFM model obtained in the solution provided in this embodiment of this application can significantly improve CTR prediction accuracy, and can effectively reduce an inference time period and energy consumption.

[00326] It can be learned from the foregoing description that, in embodiments of this application, the architecture parameters are introduced into the FM-based model, so that feature interaction item selection can be performed through optimization on the architecture parameters. In other words, in this application, provided that optimization on the architecture parameters is performed once, feature interaction item selection can be performed, and training for a plurality of candidate subsets in the conventional technology is not required. Therefore, this can effectively reduce a computing workload of feature interaction selection to save computing power, and improve efficiency of feature interaction selection.

[00327] In addition, in the solution provided in this embodiment of this application, the feature interaction item in the FM-based model can be extended to a higher order.

[00328] Embodiments described in this specification may be independent solutions, or may be combined based on internal logic. All these solutions fall within the protection scope of this application.

[00329] The foregoing describes the method embodiments provided in this application, and the following describes apparatus embodiments provided in this application. It should be understood that descriptions of apparatus embodiments correspond to the descriptions of the method embodiments. Therefore, for content that is not described in detail, refer to the foregoing method embodiments. For brevity, details are not described herein again.

[00330] As shown in FIG. 7, this embodiment of this application further provides a data processing apparatus 700. The apparatus 700 includes the following units.

[00331] A first processing unit 710 is configured to add an architecture parameter to each feature interaction item in a first model, to obtain a second model, where the first model is a factorization machine FM-based model, and the architecture parameter represents importance of a corresponding feature interaction item.

[00332] A second processing unit 720 is configured to perform optimization on architecture parameters in the second model, to obtain the optimized architecture parameters.

[00333] A third processing unit 730 is configured to obtain, based on the optimized architecture parameters and the first model or the second model, a third model through feature interaction item deletion.

[00334] Optionally, the second processing unit 720 performs optimization on the architecture parameters, to allow the optimized architecture parameters to be sparse.

[00335] In this embodiment, the third processing unit 730 is configured to obtain, based on the first model or the second model, the third model by deleting a feature interaction item corresponding to an architecture parameter in the optimized architecture parameters whose value is less than a threshold.

[00336] Optionally, the second processing unit 720 performs optimization on the architecture parameters, to allow a value of an architecture parameter of at least one feature interaction item to be equal to zero after optimization is completed.

[00337] For example, the third processing unit 730 is configured to optimize the architecture parameters in the second model using a gRDA optimizer, where the gRDA optimizer allows the value of the architecture parameter of the at least one feature interaction item to tend to zero during

an optimization process.

[00338] Optionally, the second processing unit 720 is further configured to perform optimization on model parameters in the second model, where optimization includes scalarization processing on the model parameters in the second model.

[00339] For example, the second processing unit 720 is configured to perform batch normalization BN processing on the model parameters in the second model.

[00340] Optionally, the second processing unit 720 is configured to perform simultaneous optimization on both the architecture parameters and model parameters in a second model by using same training data, to obtain the optimized architecture parameters.

[00341] Optionally, the apparatus 700 further includes a training unit 740, configured to train the third model.

[00342] Optionally, the training unit 740 is configured to train the third model, to obtain a CTR prediction model or a CVR prediction model.

[00343] The apparatus 700 may be integrated into a terminal device, a network device, or a chip.

[00344] The apparatus 700 may be deployed on a compute node of a related device.

[00345] As shown in FIG. 8, this embodiment of this application further provides an image processing apparatus 800. The apparatus 800 includes the following units.

[00346] A first processing unit 810 is configured to input data of a target object into a click-through rate CTR prediction model or a conversion rate CVR prediction model, to obtain a prediction result of the target object.

[00347] A second processing unit 820 is configured to determine a recommendation status of the target object based on the prediction result of the target object.

[00348] The CTR prediction model or the CVR prediction model is obtained through the method 300 or 500 in the foregoing embodiments.

[00349] Training of a third model includes the following step: train the third model by using a training sample of the target object, to obtain the CTR prediction model or the CVR prediction model.

[00350] Optionally, optimization on architecture parameters includes the following step: perform simultaneous optimization on both the architecture parameters and model parameters in a second model by using the same training data as that in the training sample of the target object, to obtain the optimized architecture parameters.

[00351] The apparatus 800 may be integrated into a terminal device, a network device, or a chip.

[00352] The apparatus 800 may be deployed on a compute node of a related device.

[00353] As shown in FIG. 9, this embodiment of this application further provides an image processing apparatus 900. The apparatus 900 includes a processor 910, the processor 910 is coupled to a memory 920, the memory 920 is configured to store a computer program or instructions, and the processor 910 is configured to execute the computer program or the instructions stored in the memory 920, so that the method in the foregoing method embodiments is performed.

[00354] Optionally, as shown in FIG. 9, the apparatus 900 may further include a memory 920.

[00355] Optionally, as shown in FIG. 9, the apparatus 900 may further include a data interface 930, where the data interface 930 is configured to transmit data to the outside.

[00356] Optionally, in a solution, the apparatus 900 is configured to implement the method 300 in the foregoing embodiment.

[00357] Optionally, in another solution, the apparatus 900 is configured to implement the method 500 in the foregoing embodiment.

[00358] Optionally, in still another solution, the apparatus 900 is configured to implement the method 600 in the foregoing embodiment.

[00359] An embodiment of this application further provides a computer-readable medium. The computer-readable medium stores program code to be executed by a device, and the program code is used to perform the method in the foregoing embodiments.

[00360] An embodiment of this application further provides a computer program product including instructions. When the computer program product is run on a computer, the computer is enabled to perform the method in the foregoing embodiments.

[00361] An embodiment of this application further provides a chip, and the chip includes a processor and a data interface. The processor reads, through the data interface, instructions stored in a memory to perform the method in the foregoing embodiments.

[00362] Optionally, in an implementation, the chip may further include a memory and the memory stores instructions; the processor is configured to execute the instructions stored in the memory, and when the instructions are executed, the processor is configured to perform the method in the foregoing embodiments.

[00363] An embodiment of this application further provides an electronic device. The electronic

device includes any one or more of the apparatus 700, the apparatus 800, or the apparatus 900 in the foregoing embodiments.

[00364] FIG. 10 is a schematic diagram of a hardware architecture of a chip according to an embodiment of this application. The chip includes a neural-network processing unit 1000. The chip may be disposed in any one or more of the following apparatuses or systems:

the apparatus 700 shown in FIG. 7, the apparatus 800 shown in FIG. 8, and the apparatus 900 shown in FIG. 9.

[00365] The method 300, 500, or 600 in the foregoing method embodiments may be implemented in the chip shown in FIG. 10.

[00366] The neural-network processing unit 1000 serves as a coprocessor, and is disposed on a host CPU (Host CPU). The host CPU assigns a task. A core part of the neural-network processing unit 1000 is an operational circuit 1003, and a controller 1004 controls the operational circuit 1003 to obtain data in a memory (a weight memory 1002 or an input memory 1001) and perform an operation.

[00367] In some implementations, the operational circuit 1003 includes a plurality of processing engines (processing engine, PE). In some implementations, the operational circuit 1003 is a two-dimensional systolic array. Alternatively, the operational circuit 1003 may be a one-dimensional systolic array or another electronic circuit that can perform mathematical operations such as multiplication and addition. In some implementations, the operational circuit 1003 is a general-purpose matrix processor.

[00368] For example, it is assumed that there are an input matrix A, a weight matrix B, and an output matrix C. The operational circuit 1003 extracts corresponding data of the matrix B from a weight memory 1002, and buffers the corresponding data into each PE in the operational circuit 1003. The operational circuit 1003 fetches data of the matrix A from an input memory 1001, to perform a matrix operation on the matrix B, and stores an obtained partial result or an obtained final result of the matrix into an accumulator (accumulator) 1008.

[00369] A vector calculation unit 1007 may perform further processing such as vector multiplication, vector addition, an exponent operation, a logarithmic operation, or value comparison on output of the operational circuit 1003. For example, the vector calculation unit 1007 may be configured to perform network calculation, such as pooling (pooling), batch normalization (batch normalization), or local response normalization (local response

normalization) at a non-convolutional/non-FC layer in a neural network.

[00370] In some implementations, the vector calculation unit 1007 can store a processed output vector in a unified memory (which may also be referred to as a unified buffer) 1006. For example, the vector calculation unit 1007 may apply a non-linear function to the output of the operational circuit 1003, for example, a vector of an accumulated value, to generate an activation value. In some implementations, the vector calculation unit 1007 generates a normalized value, a combined value, or both a normalized value and a combined value. In some implementations, the processed output vector can be used as an activation input for the operational circuit 1003, for example, used in a subsequent layer in the neural network.

[00371] The method 300, 500, or 600 in the foregoing method embodiments may be performed by 1003 or 1007.

[00372] The unified memory 1006 is configured to store input data and output data.

[00373] For weight data, a direct memory access controller (direct memory access controller, DMAC) 1005 directly transfers input data in an external memory to the input memory 1001 and/or the unified memory 1006, stores weight data in the external memory in the weight memory 1002, and stores data in the unified memory 1006 in the external memory.

[00374] A bus interface unit (bus interface unit, BIU) 1010 is configured to implement interaction between the host CPU, the DMAC, and an instruction fetch buffer 1009 by using a bus.

[00375] The instruction fetch buffer (instruction fetch buffer) 1009 connected to the controller 1004 is configured to store an instruction used by the controller 1004.

[00376] The controller 1004 is configured to invoke the instruction cached in the instruction fetch buffer 1009, to control a working process of an operation accelerator.

[00377] In this embodiment of this application, the data herein may be to-be-processed image data.

[00378] Generally, the unified memory 1006, the input memory 1001, the weight memory 1002, and the instruction fetch buffer 1009 each are an on-chip (On-Chip) memory. The external memory is a memory outside the NPU. The external memory may be a double data rate synchronous dynamic random access memory (double data rate synchronous dynamic random access memory, DDR SDRAM), a high bandwidth memory (high bandwidth memory, HBM), or another readable and writable memory.

[00379] Unless otherwise defined, all technical and scientific terms used in this specification

have same meanings as that usually understood by a person skilled in the art of this application. The terms used in the specification of this application are merely for the purpose of describing specific embodiments, and are not intended to limit this application.

[00380] It should be noted that "first", "second", "third", or "fourth", and various numbers in this specification are merely used for differentiation for ease of description, and are not construed as a limitation to the scope of this application.

[00381] A person skilled in the art may be aware that units and algorithm steps in the examples described with reference to the embodiments disclosed in this specification can be implemented by electronic hardware or an interaction of computer software and electronic hardware. Whether the functions are performed by hardware or software depends on particular applications and design constraints of the technical solutions. A person skilled in the art may use different methods to implement the described functions for each particular application, but it should not be considered that the implementation goes beyond the scope of this application.

[00382] It may be clearly understood by a person skilled in the art that, for the purpose of convenient and brief description, for a detailed working process of the foregoing system, apparatus, and unit, refer to a corresponding process in the foregoing method embodiments, and details are not described herein again.

[00383] In the several embodiments provided in this application, it should be understood that the disclosed systems, apparatuses, and methods may be implemented in another manner. For example, the described apparatus embodiment is merely an example. For example, division into the units is merely logical function division and may be other division in actual implementation. For example, a plurality of units or components may be combined or integrated into another system, or some features may be ignored or not performed. In addition, the displayed or discussed mutual couplings or direct couplings or communications connections may be implemented through some interfaces. The indirect couplings or communications connections between the apparatuses or units may be implemented in an electrical form, a mechanical form, or other forms.

[00384] The units described as separate parts may or may not be physically separate. Parts displayed as units may or may not be physical units, and may be located in one position or may be distributed on a plurality of network units. Some or all of the units may be selected based on actual requirements to achieve the objectives of the solutions in embodiments.

[00385] In addition, functional units in embodiments of this application may be integrated into

one processing unit, each of the units may exist alone physically, or two or more units may be integrated into one unit.

[00386] When the functions are implemented in a form of a software functional unit and sold or used as an independent product, the functions may be stored in a computer-readable storage medium. Based on such an understanding, the technical solutions of this application essentially, or the part contributing to the conventional technology, or some of the technical solutions may be implemented in a form of a software product. The computer software product is stored in a storage medium, and includes several instructions for instructing a computer device (which may be a personal computer, a server, a network device, or the like) to perform all or some of the steps of the methods described in the embodiments of this application. The foregoing storage medium includes any medium that can store program code, such as a universal serial bus flash disk (USB flash disk, UFD) (UFD may also be referred to as a USB flash drive or a flash memory), a removable hard disk, a read-only memory (read-only memory, ROM), a random access memory (random access memory, RAM), a magnetic disk, or a compact disc. The UFD may also be briefly referred to as a USB flash drive or a USB flash drive.

[00387] The foregoing descriptions are merely specific implementations of this application, but are not intended to limit the protection scope of this application. Any variation or replacement readily figured out by a person skilled in the art within the technical scope disclosed in this application shall fall within the protection scope of this application. Therefore, the protection scope of this application shall be subject to the protection scope of the claims.

CLAIMS

What is claimed is:

1. A data processing method, comprising:

adding an architecture parameter to each feature interaction item in a first model, to obtain a second model, wherein the first model is a factorization machine FM-based model, and the architecture parameter represents importance of a corresponding feature interaction item;

performing optimization on architecture parameters in the second model, to obtain the optimized architecture parameters; and

obtaining, based on the optimized architecture parameters and the first model or the second model, a third model through feature interaction item deletion.

2. The method according to claim 1, wherein optimization allows the optimized architecture parameters to be sparse.

3. The method according to claim 2, wherein the obtaining, based on the optimized architecture parameters and the first model or the second model, a third model through feature interaction item deletion comprises:

obtaining, based on the first model or the second model, the third model by deleting a feature interaction item corresponding to an architecture parameter in the optimized architecture parameters whose value is less than a threshold.

4. The method according to any one of claims 1 to 3, wherein optimization allows a value of an architecture parameter of at least one feature interaction item to be equal to zero after optimization is completed.

5. The method according to claim 4, wherein optimization is performed on the architecture parameters in the second model using a generalized regularized dual averaging gRDA optimizer, wherein the gRDA optimizer allows the value of the architecture parameter of the at least one feature interaction item to tend to zero during an optimization process.

6. The method according to any one of claims 1 to 5, wherein the method further comprises:

performing optimization on model parameters in the second model, wherein optimization comprises scalarization processing on the model parameters.

7. The method according to claim 6, wherein optimization comprises batch normalization BN processing on the model parameters in the second model.

8. The method according to any one of claim 6, wherein the performing optimization on architecture parameters in the second model and the performing optimization on model parameters in the second model comprise:

performing simultaneous optimization on both the architecture parameters and the model parameters in the second model by using same training data.

9. The method according to any one of claims 1 to 8, wherein the method further comprises: training the third model to obtain a click-through rate CTR prediction model or a conversion rate CVR prediction model.

10. A data processing method, comprising:

inputting data of a target object into a click-through rate CTR prediction model or a conversion rate CVR prediction model, to obtain a prediction result of the target object;

determining a recommendation result of the target object based on the prediction result of the target object, wherein

the CTR prediction model or the CVR prediction model is obtained through the following process:

adding an architecture parameter to each feature interaction item in a first model, to obtain a second model, wherein the first model is a factorization machine FM-based model, and the architecture parameter represents importance of a corresponding feature interaction item;

performing optimization on architecture parameters in the second model, to obtain the optimized architecture parameters;

obtaining, based on the optimized architecture parameters and the first model or the second model, a third model through feature interaction item deletion; and

training the third model by using a training sample of the target object, to obtain the CTR prediction model or the CVR prediction model.

11. The method according to claim 10, wherein optimization allows the optimized architecture parameters to be sparse.

12. The method according to claim 11, wherein the obtaining, based on the optimized architecture parameters and the first model or the second model, a third model through feature interaction item deletion comprises:

obtaining the third model by deleting a feature interaction item corresponding to an architecture parameter in the optimized architecture parameters whose value is less than a threshold.

13. The method according to any one of claims 10 to 12, wherein optimization allows a value of an architecture parameter of at least one feature interaction item to be equal to zero after optimization is completed.

14. The method according to claim 13, wherein optimization is performed on the architecture parameters in the second model using a generalized regularized dual averaging gRDA optimizer, wherein the gRDA optimizer allows the value of the architecture parameter of the at least one feature interaction item to tend to zero during an optimization process.

15. The method according to any one of claims 10 to 14, wherein the process of obtaining the CTR prediction model or the CVR prediction model further comprises: performing optimization on model parameters in the second model, wherein optimization comprises scalarization processing on the model parameters.

16. The method according to claim 15, wherein optimization comprises batch normalization BN processing on the model parameters in the second model.

17. The method according to claim 15, wherein the performing optimization on architecture parameters in the second model and the performing optimization on model parameters in the second model comprise:

performing simultaneous optimization on both the architecture parameters and the model parameters in the second model by using the same training data as that in the training sample of the target object.

18. An image processing apparatus, comprising:

a memory, configured to store a program; and

a processor, configured to execute the program stored in the memory, wherein when the program stored in the memory is executed, the processor is configured to perform the method according to any one of claims 1 to 17.

19. A computer-readable storage medium, wherein the computer-readable medium stores program code to be executed by a device; and when the program code is executed, the device performs the method according to any one of claims 1 to 17.

20. A chip, comprising at least one processor and a data interface, wherein

the at least one processor is configured to invoke and run, by using the data interface, a computer program stored in a memory, to enable the chip to perform the method according to any one of claims 1 to 17.