

TRAINING METHOD AND APPARATUS FOR GRAPH NEURAL NETWORK

TECHNICAL FIELD

[0001] One or more implementations of the present specification relate to the field
5 of machine learning technologies, and in particular, to graph neural network.

BACKGROUND

[0002] A relational network graph is a description of a relationship between entities
in the real world, and is currently widely used in various service processing, for example,
social network analysis and chemical bond prediction. A graph neural network (GNN)
10 is applicable to various tasks on a relational network graph. However, performance of
the GNN largely depends on a quantity of labeled data, and generally, the performance
of the GNN rapidly decreases as labeled data decreases.

SUMMARY

[0003] The present specification provides a technical solution that breaks through
15 a limitation of insufficient labeled data during GNN training, and obtains a GNN
model with excellent performance, which effectively improves accuracy of a service
processing result.

[0004] One or more implementations of the present specification describe a training
method and apparatus for a graph neural network. Labeled data is expanded by using
20 unlabeled data, and an information gain is introduced to reduce the difference between
a training loss corresponding to distribution of original labeled data and a training loss
corresponding to distribution of expanded labeled data, which improves a training effect
of a GNN model.

[0005] According to a first aspect, a training method for a graph neural network is
25 provided, and relates to performing multiple rounds of iterative updating on a graph
neural network based on a user relational graph, where any round of the multiple rounds

includes: processing the user relational graph by using a current graph neural network, to obtain multiple classification prediction vectors corresponding to multiple user nodes in the user relational graph; allocating a corresponding pseudo classification label to a first quantity of unlabeled nodes in the multiple user nodes based on the multiple
5 classification prediction vectors; determining, for each of the first quantity of unlabeled nodes, an information gain generated by training the current graph neural network by using the unlabeled node; and updating a model parameter in the current graph neural network according to a classification prediction vector and a real classification label that are corresponding to each labeled node in the multiple user nodes, and a
10 classification prediction vector, a pseudo classification label, and an information gain that are corresponding to each unlabeled node.

[0006] In an example implementation, the multiple user nodes comprise a second quantity of unlabeled nodes, and classification prediction vectors comprise multiple prediction probabilities corresponding to multiple categories; where the allocating the
15 corresponding pseudo classification label to the first quantity of unlabeled nodes in the multiple user nodes based on the multiple classification prediction vectors includes: for each node in the second quantity of unlabeled nodes, in response to that a maximum prediction probability included in a classification prediction vector corresponding to the node reaches a predetermined threshold, classifying the node into the first quantity of
20 unlabeled nodes, and determining a category corresponding to the maximum prediction probability as a pseudo classification label of the node.

[0007] In an example implementation, the determining, for each of the first quantity of unlabeled nodes, the information gain generated by training the current graph neural network by using the unlabeled node includes: for a first unlabeled node of the first
25 quantity of unlabeled nodes, training the current graph neural network by using a first classification prediction vector and a pseudo classification label that are corresponding to the first unlabeled node, and determining a second classification prediction vector of the first unlabeled node based on a trained first graph neural network; determining first information entropy according to the first classification prediction vector; determining
30 second information entropy according to the second classification prediction vector; and obtaining the information gain based on a difference between the second information entropy and the first information entropy.

[0008] In an example implementation, the trained first graph neural network comprises multiple aggregation layers and an output layer; and the determining the

second classification prediction vector of the first unlabeled node based on the trained first graph neural network includes: performing, at an aggregation layer in the multiple aggregation layers, random zeroing processing on vector elements in multiple aggregation vectors for the multiple user nodes that are output by an upper aggregation layer, and determining, based on the multiple aggregation vectors after the random zeroing processing, multiple aggregation vectors that are output by the aggregation layer for the multiple user nodes; and processing, at the output layer, an aggregation vector output by a last aggregation layer for the first unlabeled user node, to obtain the second classification prediction vector.

[0009] In an example implementation, the trained first graph neural network comprises multiple aggregation layers and an output layer; and the determining the second classification prediction vector of the first unlabeled node based on the trained first graph neural network includes: performing, at an aggregation layer in the multiple aggregation layers, random zeroing processing on a matrix element in an adjacency matrix corresponding to the user relational graph, and determining, based on the adjacency matrix after the random zeroing processing and multiple aggregation vectors that are output by an upper aggregation layer for the multiple user nodes, multiple aggregation vectors for the multiple user nodes that are output by the aggregation layer; and processing, at the output layer, an aggregation vector output by a last aggregation layer for the first unlabeled user node, to obtain the second classification prediction vector.

[0010] Further, in a further example implementation, the determining the second classification prediction vector of the unlabeled node based on the trained first graph neural network includes: performing for multiple times an operation of determining the second classification prediction vector to correspondingly obtain multiple second classification prediction vectors; where the determining the second information entropy according to the second classification prediction vector includes: determining an average value of multiple pieces of information entropy respectively corresponding to the multiple second classification prediction vectors as the second information entropy.

[0011] In an example implementation, the updating the model parameter in the current graph neural network according to the classification prediction vector and the real classification label that are corresponding to each labeled node in the multiple user nodes, and the classification prediction vector, the pseudo classification label, and the information gain that are corresponding to each unlabeled node includes: determining

a first loss term according to the classification prediction vector and the real classification label that are corresponding to each labeled node; determining a second loss term for each unlabeled node according to the classification prediction vector and the pseudo classification label that are corresponding to each unlabeled node, and
5 weighting the second loss term by using the information gain corresponding to the unlabeled node; and updating the model parameter according to the first loss term and the weighted second loss term.

[0012] In an example implementation, the weighting the second loss term by using the information gain corresponding to the unlabeled node includes: normalizing the
10 information gain of each unlabeled node by using a first quantity of information gains corresponding to the first quantity of unlabeled nodes, to obtain a corresponding weighting coefficient; and performing weighting processing by using the weighting coefficient.

[0013] According to a second aspect, a training method for a graph neural network
15 is provided, and relates to performing multiple rounds iterative updating on a graph neural network based on a pre-constructed relational graph, where any round of the multiple rounds includes: processing the relational graph by using a current graph neural network, to obtain multiple classification prediction vectors corresponding to multiple service object nodes in the relational graph; allocating a corresponding pseudo
20 classification label to a first quantity of unlabeled nodes in the multiple service object nodes based on the multiple classification prediction vectors; determining, for each of the first quantity of unlabeled nodes, an information gain generated by training the current graph neural network by using the unlabeled node; and updating a model parameter in the current graph neural network according to a classification prediction
25 vector and a real classification label that are corresponding to each labeled node in the multiple service object nodes, and a classification prediction vector, a pseudo classification label, and an information gain that are corresponding to each unlabeled node.

[0014] According to a third aspect, a training apparatus for a graph neural network
30 is provided, where the apparatus performs, by using following units, any one of multiple rounds of iterative updating on a graph neural network according to a user relational graph: a classification prediction unit, configured to process the user relational graph by using a current graph neural network, to obtain multiple classification prediction vectors corresponding to multiple user nodes in the user relational graph; a pseudo label

allocation unit, configured to allocate a corresponding pseudo classification label to a first quantity of unlabeled nodes in the multiple user nodes based on the multiple classification prediction vectors; an information gain determining unit, configured to determine, for each of the first quantity of unlabeled nodes, an information gain generated by training the current graph neural network by using the unlabeled node; and a parameter updating unit, configured to update a model parameter in the current graph neural network according to a classification prediction vector and a real classification label that are corresponding to each labeled node in the multiple user nodes, and a classification prediction vector, a pseudo classification label, and an information gain that are corresponding to each unlabeled node.

[0015] According to a fourth aspect, a training apparatus for a graph neural network is provided, where the apparatus performs, by using following units, any one of multiple rounds of iterative updating on a graph neural network according to a pre-constructed relational graph: a classification prediction unit, configured to process the relational graph by using a current graph neural network, to obtain multiple classification prediction vectors corresponding to multiple service object nodes in the relational graph; a pseudo label allocation unit, configured to allocate a corresponding pseudo classification label to a first quantity of unlabeled nodes in the multiple service object nodes based on the multiple classification prediction vectors; an information gain determining unit, configured to determine, for each of the first quantity of unlabeled nodes, an information gain generated by training the current graph neural network by using the unlabeled node; and a parameter updating unit, configured to update a model parameter in the current graph neural network according to a classification prediction vector and a real classification label that are corresponding to each labeled node in the multiple service object nodes, and a classification prediction vector, a pseudo classification label, and an information gain that are corresponding to each unlabeled node.

[0016] According to a fifth aspect, a computer readable storage medium that stores a computer program is provided, and when the computer program is executed on a computer, the computer is caused to perform the methods according to the first aspect or the second aspect.

[0017] According to a sixth aspect, a computing device is provided, including a memory and a processor, where the memory stores executable code, and when the

processor executes the executable code, the methods according to the first aspect or the second aspect are implemented.

5 [0018] According to the method and the apparatus provided in the implementations of the present specification, labeled data is expanded by using unlabeled data in a user relational graph, and an information gain is introduced to reduce the difference between a training loss corresponding to distribution of original labeled data and a training loss corresponding to distribution of expanded labeled data, so as to effectively improve a training effect of a GNN model, and further improve prediction accuracy of a trained GNN model on a user node.

10 BRIEF DESCRIPTION OF DRAWINGS

[0019] To describe the technical solutions in the implementations of the present disclosure more clearly, the following briefly introduces the accompanying drawings required for describing the implementations. Clearly, the accompanying drawings in the following description are merely some implementations of the present disclosure, and a person of ordinary skill in the field can still derive other drawings from these accompanying drawings without creative efforts.

[0020] FIG. 1 is a schematic diagram of a training framework of a graph neural network according to an implementation;

20 [0021] FIG. 2 is a schematic flowchart of a training method for a graph neural network according to an implementation;

[0022] FIG. 3 is a flowchart of a method for determining an information gain according to an implementation;

[0023] FIG. 4 is a schematic flowchart of a training method for a graph neural network according to another implementation;

25 [0024] FIG. 5 is a schematic structural diagram of a training apparatus for a graph neural network according to an implementation; and

[0025] FIG. 6 is a schematic structural diagram of a training apparatus for a graph neural network according to another implementation.

DESCRIPTION OF IMPLEMENTATIONS

[0026] The solutions provided in the present specification are described below with reference to the accompanying drawings.

[0027] As described above, the specification provides a solution that can break
 5 through a limitation of labeled data shortage during GNN training. The solution includes a way of self-training that solves, among others, the problem of scarcity of labeled data by making full use of abundant unlabeled data. For example, a model trained on an original labeled data set $\mathcal{V}_L$ is given as a teacher model, and prediction is performed on an unlabeled data set $\mathcal{V}_U$. Then, a pseudo label is marked on a
 10 corresponding unlabeled data subset $\mathcal{S}_U$ by using a prediction result with high confidence, so as to expand the original labeled data. Then, a student model is trained by using an expanded labeled data set $\mathcal{V}_L \cup \mathcal{S}_U$, and the teacher model is updated by using the trained student model. As such, iterations are repeated until the student model converges.

15 [0028] A key of the above self-training method is that pseudo labeling is performed on an unlabeled sample with high confidence, so as to expand labeled data. However, the inventor found through experiments and analysis that, compared with the original labeled data set $\mathcal{V}_L$, the expanded labeled data set $\mathcal{V}_L \cup \mathcal{S}_U$ obtained after expansion by using the unlabeled sample with high confidence undergoes distribution shift, which
 20 results in poor performance of a GNN model trained by using the expanded data set $\mathcal{V}_L \cup \mathcal{S}_U$, and it is difficult to obtain a decision boundary that is sufficiently clear and robust. Further, analysis is performed from the perspective of a loss function, data distribution obeyed by the original labeled data set $\mathcal{V}_L$ is denoted as P_{pop} , and a classifier f_θ whose parameter is denoted as θ is given. Therefore, optimal setting of
 25 a model parameter θ can be obtained by minimizing the loss function represented by the following equation (1).

$$\mathcal{L}_{pop} = \mathbb{E}_{(v_i, y_i) \sim P_{pop}(\mathcal{V}, \mathcal{Y})} l(y_i, \mathbf{p}_i) \quad (1)$$

[0029] In the above equation (1), v_i and y_i respectively represent a node feature and a node label of an i th labeled node that obeys P_{pop} distribution; $\mathbf{p}_i$
 30 represents a prediction result output by the classifier f_θ for the i th labeled node; and $l(\cdot, \cdot)$ represents multiple classification losses, for example, can be cross entropy losses.

[0030] Similarly, for the above self-training scenario in which distribution shift exists, the following loss function can be used to calculate a training loss:

$$\begin{aligned}\mathcal{L}_{st} = & \frac{|\mathcal{V}_L|}{|\mathcal{V}_L \cup \mathcal{S}_U|} \mathbb{E}_{(v_i, y_i) \sim P_{pop}(\mathcal{V}, \mathcal{Y})} l(y_i, \mathbf{p}_i) \\ & + \frac{|\mathcal{S}_U|}{|\mathcal{V}_L \cup \mathcal{S}_U|} \mathbb{E}_{(v_u, y_u) \sim P_{st}(\mathcal{V}, \mathcal{Y})} l(\bar{y}_u, \mathbf{p}_u)\end{aligned}\quad (2)$$

[0031] In the above equation (2), v_u and y_u respectively represent a node feature and a real node label (not actually obtained) of a u th unlabeled node that obeys P_{st} distribution; $\bar{y}_u$ indicates a pseudo label of the u th unlabeled node; and $\mathbf{p}_u$ indicates a prediction result output by the classifier f_θ for the u th unlabeled node.

[0032] By performing comparative analysis on the above equations (1) and (2), it can be understood that distribution shift in the self-training process severely affects training performance of a graph model, and further causes deterioration of generalization performance of the graph model in a prediction phase. Therefore, it is more ideal to optimize the classifier f_θ by using the training loss calculated by using the equation (1) than by using the equation (2). However, in practice, because labeled data is scarce, it is difficult to accurately restore real labeled data distribution, and only L_{st} calculated in equation (2) is available. Some implementations of the specification apply the following theorem, which reduce or eliminate the gap between L_{st} and L_{pop} :

[0033] Given the losses L_{pop} and L_{st} respectively defined in equations (1) and (2), assume that for each node v_u in a pseudo-labeled data set $\mathcal{S}_U$, $\bar{y}_u = y_u$. Then $L_{st} = L_{pop}$ is valid. When L_{st} can be denoted as having an additional weight coefficient $\gamma_u = \frac{P_{pop}(v_u, y_u)}{P_{st}(v_u, y_u)}$ as follows:

$$\begin{aligned}\mathcal{L}_{st} = & \frac{|\mathcal{S}_U|}{|\mathcal{V}_L \cup \mathcal{S}_U|} \mathbb{E}_{(v_u, y_u) \sim P_{st}(\mathcal{V}, \mathcal{Y})} \gamma_u l(\bar{y}_u, \mathbf{p}_u) \\ & + \frac{|\mathcal{V}_L|}{|\mathcal{V}_L \cup \mathcal{S}_U|} \mathbb{E}_{(v_i, y_i) \sim P_{pop}(\mathcal{V}, \mathcal{Y})} l(y_i, \mathbf{p}_i),\end{aligned}\quad (3)$$

[0034] the process of proof of the above theorem is as follows:

[0035] First, according to the assumption for the node v_u : $\bar{y}_u = y_u$, equation (1) can be rewritten in the following form:

$$\begin{aligned}\mathcal{L}_{pop} &= \frac{|\mathcal{S}_U|}{|\mathcal{V}_L \cup \mathcal{S}_U|} \mathbb{E}_{(v_u, y_u) \sim P_{pop}(\mathcal{V}, \mathcal{Y})} l(\bar{y}_u, \mathbf{p}_u) \\ &+ \frac{|\mathcal{V}_L|}{|\mathcal{V}_L \cup \mathcal{S}_U|} \mathbb{E}_{(v_i, y_i) \sim P_{pop}(\mathcal{V}, \mathcal{Y})} l(y_i, \mathbf{p}_i)\end{aligned}\quad (4)$$

[0036] It is noticed that:

$$\mathbb{E}_{(v_u, y_u) \sim P_{pop}(\mathcal{V}, \mathcal{Y})} l(\bar{y}_u, \mathbf{p}_u) = \mathbb{E}_{(v_u, y_u) \sim P_{st}(\mathcal{V}, \mathcal{Y})} \frac{P_{pop}(v_u, y_u)}{P_{st}(v_u, y_u)} l(\bar{y}_u, \mathbf{p}_u) \quad (5)$$

[0037] Therefore, equation (4) can be rewritten as follows:

$$\begin{aligned}\mathcal{L}_{pop} &= \frac{|\mathcal{S}_U|}{|\mathcal{V}_L \cup \mathcal{S}_U|} \mathbb{E}_{(v_u, y_u) \sim P_{st}(\mathcal{V}, \mathcal{Y})} \frac{P_{pop}(v_u, y_u)}{P_{st}(v_u, y_u)} l(\bar{y}_u, \mathbf{p}_u) \\ &+ \frac{|\mathcal{V}_L|}{|\mathcal{V}_L \cup \mathcal{S}_U|} \mathbb{E}_{(v_i, y_i) \sim P_{pop}(\mathcal{V}, \mathcal{Y})} l(y_i, \mathbf{p}_i) \\ &= \frac{|\mathcal{S}_U|}{|\mathcal{V}_L \cup \mathcal{S}_U|} \mathbb{E}_{(v_u, y_u) \sim P_{st}(\mathcal{V}, \mathcal{Y})} \gamma_u l(\bar{y}_u, \mathbf{p}_u) \\ &+ \frac{|\mathcal{V}_L|}{|\mathcal{V}_L \cup \mathcal{S}_U|} \mathbb{E}_{(v_i, y_i) \sim P_{pop}(\mathcal{V}, \mathcal{Y})} l(y_i, \mathbf{p}_i)\end{aligned}\quad (6)$$

[0038] γ_u can be considered as the weight of the loss function $l(\bar{y}_u, \mathbf{p}_u)$ of the unlabeled node v_u .

[0039] Finally, recalling the loss function in the distribution shift case shown in equation (2), it can be found that, L_{st} in equation (2) can be denoted as a form of adding an additional weight coefficient γ_u to equation (1). In other words, as long as an appropriate coefficient γ_u can be added for each pseudo-labeled node in L_{st} , L_{st} can approximate L_{pop} .

[0040] However, because the labeled data distribution P_{pop} is usually difficult to solve, this means that the weight coefficient γ_u is difficult to solve accurately. Further, the inventor found, by using a means such as visualization, that the weight coefficient γ_u and an information gain $\mathbb{B}_u$ that are corresponding to the unlabeled node v_u have the same change trend. For example, farther from the decision boundary, smaller values of the two. In some implementations, the weight coefficient γ_u is approximated by solving the information gain $\mathbb{B}_u$. Simply, the information gain $\mathbb{B}_u$ is measurement of the contribution of the unlabeled node v_u to model optimization.

[0041] In some implementations, an implementation of weighting is conducted, by introducing the information gain $\mathbb{B}_u$, a loss term $l(\bar{y}_u, \mathbf{P}u)$ that is in the self-training loss L_{st} and that is for the unlabeled node v_u , so L_{st} approximates or equals L_{pop} . For ease of intuitive understanding, FIG. 1 is a schematic diagram of a training framework of a graph neural network according to an implementation; as shown in FIG. 1, graph data of a user relational network graph includes an original labeled sample set $\mathcal{V}_L$ and an unlabeled sample set $\mathcal{V}_U$. Based on this, in graph neural network iterative training of any round, a current GNN model is first used to process the relational network graph to obtain a classification result and confidence of each unlabeled node in the unlabeled sample set $\mathcal{V}_U$. An unlabeled node whose confidence is sufficiently high is selected and added to the unlabeled sample subset $\mathcal{S}_U$, an information gain of each unlabeled node v_u therein is determined, and then a training loss is determined by using the labeled sample set $\mathcal{V}_L$, the unlabeled sample subset $\mathcal{S}_U$, and the correspondingly determined information gain, so as to update the GNN model.

[0042] With reference to more implementations, the following describes steps for implementing example implementations of the technical solutions. FIG. 2 is a schematic flowchart of a training method for a neural network according to an implementation. The method can be performed by any apparatus, platform, or device cluster that has a calculation and processing capability.

[0043] The training method shown in FIG. 2 involves performing multiple rounds of iterative updating on a graph neural network based on a user relational graph (or referred to as a user relational network graph). For ease of understanding, the user relational graph is first described.

[0044] The user relational graph includes multiple user nodes corresponding to multiple users, and a connection edge formed by having an association relationship between the user nodes. A node feature of the user node can include a static feature (or a basic attribute feature) and a behavior feature of a corresponding user. In an implementation, the user static feature can include a user's gender, age, occupation, usual place of residence, interest, etc. In an implementation, the user behavior feature can include a consumption frequency, a consumption amount, a consumption period, a consumption category, graphic content published on a social networking site, social activity, etc.

[0045] The multiple user nodes include a small quantity of labeled nodes that carry a user category label, and a large quantity of unlabeled nodes that do not carry a label. Generally, the label carried in the labeled node is obtained by manually marking at high labor costs. The user category label is adapted to a specific prediction task. In one
5 implementation, the prediction task is user risk assessment. Correspondingly, the user category label can include a risky user and a risk-free user, or include a high-risk user, a low-risk user, and a medium-risk user, or include a defaulting user and a trustworthy user, or include a fraudulent user and a secure user. In an implementation, the prediction task is to divide consumption populations, and correspondingly, the user category label
10 can include a high consumption population and a low consumption population.

[0046] The user relational graph is described above. As shown in FIG. 2, iterative updating of any one of the above multiple rounds of iterative updating includes the following steps:

[0047] Step S210: Process the user relational graph by using a current graph neural
15 network, to obtain multiple classification prediction vectors corresponding to multiple user nodes in the user relational graph. Step S220: Allocate a corresponding pseudo classification label to a first quantity of unlabeled nodes in the multiple user nodes based on the multiple classification prediction vectors. Step S230: Determine, for each of the first quantity of unlabeled nodes, an information gain generated by training the current
20 graph neural network by using the unlabeled node. Step S240: Update a model parameter in the current graph neural network according to a classification prediction vector and a real classification label that are corresponding to each labeled node in the multiple user nodes, and a classification prediction vector, a pseudo classification label, and an information gain that are corresponding to each unlabeled node.

25 [0048] The above steps are described in detail as follows:

[0049] First, in step S210, process the user relational graph by using a current graph neural network, to obtain multiple classification prediction vectors corresponding to multiple user nodes in the user relational graph. In an implementation, this round of iteration is the first round. Correspondingly, the current graph neural network can be a
30 graph neural network obtained after parameter initialization setting is performed, or can be a graph neural network obtained after a graph neural network obtained after parameter initialization is trained by using multiple labeled nodes and labels carried by the multiple labeled nodes. In an implementation, the current round of iteration is not

the first round, and correspondingly, the current graph neural network can be a graph neural network obtained after updating by the previous round of iteration.

[0050] The current graph neural network includes multiple aggregation layers and an output layer, and the multiple aggregation layers are used to perform graph embedding processing on the user relational graph to obtain multiple node embedding vectors corresponding to the multiple user nodes. It should be understood that an input to the first aggregation layer in the multiple aggregation layers includes an original feature of a user node and/or a connection edge, and the multiple aggregation layers perform high-order representation of the node based on the original feature, so as to obtain a node representation vector (or referred to as a node embedding vector) with deep semantics. Further, the output layer is used to output a classification prediction result of a corresponding user node according to each node embedding vector.

[0051] In an implementation, the type of the current graph neural network is a graph convolutional neural network (GCN). Correspondingly, an output $\mathbf{H}^{(l)}$ of any an l th aggregation layer in the GCN can be calculated by using the following equation:

$$\mathbf{H}^{(l)} = \sigma(\Re(\mathbf{A})\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}) \quad (7)$$

[0052] In the above equation (7), $\mathbf{A}$ represents an adjacency matrix of the user relational graph, and is used to record a connection relationship between user nodes. For example, for any element A_{ij} in the adjacency matrix $\mathbf{A}$, when the value of the element is 1 or 0, respectively representing that a connection edge exists or does not exist between a user node i and a user node j . $\Re(\cdot)$ represents a normalization operator; $\mathbf{W}^{(l)}$ represents a parameter matrix at the l th aggregation layer, and $\mathbf{W}^{(l)} \in \mathbb{R}^{D_{l-1} \times D_l}$. $\mathbf{H}^{(l-1)}$ represents an output of an $((l-1))$ th aggregation layer. It should be understood that, $\mathbf{H}^{(0)} = \mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times D_v}$, where $\mathbf{X}$ represents a feature matrix formed by node features of the multiple user nodes, $\mathcal{V}$ represents a node set formed by the multiple user nodes, $|\mathcal{V}|$ represents a quantity of nodes in the node set, and D_v represents a dimension of the node feature. In addition, $\mathbf{H}^{(l)} \in \mathbb{R}^{|\mathcal{V}| \times D_l}$ a parameter of a GCN model $\theta = \{\mathbf{W}^{(l)}\}_{l=1}^L$, and L is a total quantity of the multiple aggregation layers.

[0053] In an implementation, the type of the current graph neural network can be a graph attention network (GAT), etc. It should be understood that there are a variety of existing types of graph neural networks, and the type can be selected as required in the implementations disclosed in the present specification, which is not specifically limited.

[0054] In an aspect, the output layer includes one or more fully connected network sublayers. By using the fully connected network sublayer, linear transformation and/or nonlinear transformation processing can be separately performed on each node embedding vector, so as to obtain a classification prediction vector of a corresponding user node, where multiple vector elements in the classification prediction vector are corresponding to multiple category probabilities.

[0055] Therefore, multiple classification prediction vectors corresponding to multiple user nodes can be obtained. Then, in step S220, allocate a corresponding pseudo classification label to a first quantity of unlabeled nodes in the multiple user nodes based on the multiple classification prediction vectors.

[0056] For description purposes, a quantity of all unlabeled nodes in the multiple user nodes is recorded as a second quantity. For example, in this step, a corresponding pseudo classification label can be allocated to some or all of the unlabeled nodes based on classification prediction vectors corresponding to the second quantity of unlabeled nodes.

[0057] In an implementation, for each node in the second quantity of unlabeled nodes, a category corresponding to the maximum prediction probability in a classification prediction vector corresponding to the node is determined as a pseudo classification label of the node. As such, a corresponding pseudo classification label can be allocated to the second quantity of unlabeled nodes. In this case, the first quantity is equal to the second quantity.

[0058] In an implementation, for each node in the second quantity of unlabeled nodes, in response to that a maximum prediction probability included in a classification prediction vector corresponding to the node reaches a predetermined threshold, the node is classified into the first quantity of unlabeled nodes, and a category corresponding to the maximum prediction probability is determined as a pseudo classification label of the node. In an example implementation, the predetermined threshold is that the maximum prediction probability is greater than a predetermined threshold (for example, 0.2). In an example implementation, the predetermined threshold is as follows: The maximum prediction probability corresponding to the node is top k ranked (for example, k = 1000) in a second quantity of maximum prediction probabilities. As such, it can be implemented that an unlabeled node with high confidence (the confidence is equal to the maximum prediction probability) is selected from a full quantity of unlabeled nodes,

and a pseudo label is marked for the unlabeled node. In this case, the first quantity is less than the second quantity.

[0059] The above can implement automatic marking on the first quantity of unlabeled nodes. For clarity of description, in this implementation of the present specification, an unlabeled subset formed by the first quantity of unlabeled nodes is denoted as $\mathcal{S}_U$.

[0060] Then, in step S230, determine, for each of the first quantity of unlabeled nodes, an information gain generated by training the current graph neural network by using the unlabeled node. It should be understood that, in probability theory or information theory, the information gain refers to a reduction in an amount of information about a random event after a specific variable value (for example, cloudy) is assigned to a random variable (for example, tomorrow's weather) in the random event (for example, whether it rains tomorrow). The information amount is usually obtained by calculating Shannon's entropy or called information entropy. According to the definition of the information gain, for any unlabeled node v_u in the unlabeled subset $\mathcal{S}_U$, the information gain $\mathbb{B}_u$ on the GNN model parameter θ can be calculated by using predictive distribution and a posterior parameter $P(\theta|\mathcal{G})$. For details, refer to the following equation:

$$\mathbb{B}_u(y_u, \theta | \mathbf{x}_u, \mathbf{A}, \mathcal{G}) = \mathbb{H}[\mathbb{E}_{P(\theta|\mathcal{G})}[y_u | \mathbf{x}_u, \mathbf{A}; \theta]] - \mathbb{E}_{P(\theta|\mathcal{G})}[H[y_u | \mathbf{x}_u, \mathbf{A}; \theta]] \quad (8)$$

[0061] In the above equation (8), the first term on the right is an expected value of information entropy of predictive distribution $\mathbf{p}_u(\mathbf{y}_u | \mathbf{x}_u, \mathbf{A}, \mathcal{G})$ under the posterior parameter $P(\theta|\mathcal{G})$, which is used to measure an information amount when a model parameter θ is not changed, $\mathcal{G}$ represents the above user relational graph, and $\mathbf{y}_u$ represents a category probability vector output by the GNN model f_θ . The second term is an average value (or expected value) of conditional entropy under a given node feature $\mathbf{x}_u$, and is used to capture an information amount of the model parameter θ after the model f_θ is optimized by using a node v_u . As such, the information gain brought by the unlabeled node v_u to the model parameter θ can be measured by calculating the difference between the two terms.

[0062] It can be observed from equation (8) that if the information gain $\mathbb{B}_u$ is calculated by using equation (8), the posteriori parameter $P(\theta|\mathcal{G})$ needs to be calculated. However, the posteriori parameter $P(\theta|\mathcal{G})$ is usually difficult to solve. In a

possible method, the posteriori parameter $P(\theta|G)$ can be calculated by using a conventional Bayesian network, but this causes huge calculation consumption.

[0063] Therefore, in an example method, a relatively accurate information gain calculation value can be obtained by using a small quantity of calculation. For example,
5 a dropout or dropedge algorithm is used to implement approximation of the posteriori parameter $P(\theta|G)$. With reference to FIG. 3, the following describes a method of determining the information gain based on the dropout algorithm or the dropedge algorithm. As shown in FIG. 3, the method includes the following steps:

[0064] Step S31: For any unlabeled node v_u (or referred to as a first unlabeled
10 node) in an unlabeled node subset S_U , train a current graph neural network by using a first classification prediction vector corresponding to the first unlabeled node in the multiple classification prediction vectors and a pseudo classification label corresponding to the first unlabeled node, to obtain a first graph neural network. For example, a training loss is calculated by using the first classification prediction vector
15 and the corresponding pseudo classification label, and then a parameter in the current graph neural network is optimized (or referred to as updated) by using the training loss, to obtain an updated first graph neural network.

[0065] Step S32: Determine a second classification prediction vector of the unlabeled node v_u based on the trained first graph neural network.

20 [0066] In an implementation, a dropout algorithm is introduced to perform random masking (or random zeroing) on a user node feature. For example, at an aggregation layer in multiple aggregation layers included in the first graph neural network, random zeroing processing is performed on vector elements in multiple aggregation vectors for the multiple user nodes that are output by an upper aggregation layer, and multiple
25 aggregation vectors that are output by the current aggregation layer for the multiple user nodes are determined based on the multiple aggregation vectors after the random zeroing processing.

[0067] In an example implementation, the aggregation layer can be pre-specified or randomly set by a worker. For example, the last aggregation layer in the multiple
30 aggregation layers can be specified to perform a dropout operation. In an example implementation, it is not limited to one aggregation layer for performing zeroing processing on the vector element, and there can be another quantity. For example, a dropout operation of a node feature can be performed on each aggregation layer.

[0068] In an aspect, in an example implementation, when the current graph neural network is a GCN, a process of processing, at the above aggregation layer based on the dropout algorithm, multiple aggregation vectors output at the upper layer to obtain an output at the current layer can be denoted as the following equation:

$$\mathbf{H}^{(l)} = \sigma(\mathfrak{N}(\mathbf{A})(\mathbf{H}^{(l-1)} \odot \mathbf{Z}^{(l)})\mathbf{W}^{(l)}) \quad (9)$$

[0069] In the above equation (9), $\mathbf{H}^{(l-1)}$ represents a matrix formed by multiple aggregation vectors output at the upper layer; $\mathbf{Z}^{(l)} \in \{0,1\}^{D_{l-1} \times D_l}$, where $D_{l-1} \times D_l$ matrix elements can be obtained by performing multiple times of sampling from Bernoulli distribution, and each matrix element indicates whether to set a matrix element at a corresponding location in the matrix $\mathbf{H}^{(l-1)}$ to zero; and the operator $\odot$ represents multiplication of elements at the same location between two matrices.

[0070] Further, at the output layer, processing is performed on the aggregation vector $\mathbf{H}^{(L)}$ output by the last aggregation layer for the unlabeled node v_u , to obtain the second classification prediction vector $\tilde{\mathbf{p}}_u = \tilde{\mathbf{p}}(\mathbf{y}_u | \mathbf{x}_u, \mathbf{A}; \tilde{\theta})$. Or average processing can be performed on multiple aggregation vectors $\{\mathbf{H}^{(l)}\}_1^L$ output by multiple aggregation layers for the unlabeled node v_u , to obtain the second classification prediction vector $\tilde{\mathbf{p}}_u$.

[0071] It is noted that the operation term $\mathbf{H}^{(l-1)} \odot \mathbf{Z}^{(l-1)}$ in equation (9) implements Bernoulli sampling on a node feature, which is equivalent to sampling in parameter distribution to which the posterior parameter $P(\theta | \mathcal{G})$ conforms. Therefore, to estimate the posterior parameter $P(\theta | \mathcal{G})$, the above prediction operation can be performed multiple times to perform sampling on the parameter distribution multiple times (denoted as T times), and correspondingly, a corresponding second classification prediction vector $\tilde{\mathbf{p}}_u^t = \tilde{\mathbf{p}}^t(\mathbf{y}_u | \mathbf{x}_u, \mathbf{A}; \tilde{\theta}_t)$ is obtained each time (denoted as a t th time). As such, T second classification prediction vectors $\{\tilde{\mathbf{p}}_u^t\}_{t=1}^T$ can be obtained based on the dropout algorithm.

[0072] In an implementation, a droppedge algorithm is introduced to randomly mask a connection edge between user nodes. For example, at an aggregation layer in the multiple aggregation layers included in the first graph neural network, random zeroing processing is performed on a matrix element in an adjacency matrix $\mathbf{A}$ corresponding to the user relational graph, and multiple aggregation vectors for the multiple user nodes that are output by the aggregation layer are determined based on the adjacency matrix

after the random zeroing processing and multiple aggregation vectors that are output by an upper aggregation layer for the multiple user nodes.

[0073] In an example implementation, the above aggregation layer can be pre-specified or randomly set by a worker. In practice, the above aggregation layer can be specified as the last aggregation layer in the multiple aggregation layers. In an example implementation, it is not limited to one aggregation layer for performing zeroing processing on the element of the adjacency matrix, and there can be another quantity. For example, a droppedge operation of an edge feature can be performed on each aggregation layer.

[0074] In an aspect, in an example implementation, when the current graph neural network is a GCN, a process of processing, at the above aggregation layer based on the droppedge algorithm, multiple aggregation vectors output at the upper layer to obtain an output at the current layer can be denoted as the following equation:

$$\mathbf{H}^{(l)} = \sigma(\mathfrak{N}(\mathbf{A} \odot \mathbf{Z}^{(l)})(\mathbf{H}^{(l-1)})\mathbf{W}^{(l)}) \quad (10)$$

[0075] In the above equation (10), $\mathbf{H}^{(l-1)}$ represents a matrix formed by multiple aggregation vectors output at the upper layer; $\mathbf{Z}^{(l)} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$, where $|\mathcal{V}| \times |\mathcal{V}|$ matrix elements can be obtained by performing multiple times of sampling from Bernoulli distribution, and each matrix element indicates whether to set a matrix element at a corresponding location in the adjacency matrix $\mathbf{A}$ to zero.

[0076] Further, at the output layer, processing is performed on the aggregation vector $\mathbf{H}^{(L)}$ output by the last aggregation layer for the unlabeled node v_u , to obtain the second classification prediction vector $\tilde{\mathbf{p}}_u = \tilde{\mathbf{p}}(\mathbf{y}_u | \mathbf{x}_u, \mathbf{A}; \tilde{\theta})$. Or average processing can be performed on multiple aggregation vectors $\{\mathbf{H}^{(l)}\}_1^L$ output by multiple aggregation layers for the unlabeled node v_u , to obtain the second classification prediction vector $\tilde{\mathbf{p}}_u$.

[0077] It is noted that the operation term $\mathbf{A} \odot \mathbf{Z}^{(l)}$ in equation (10) implements Bernoulli sampling on a connection edge, which is equivalent to sampling in parameter distribution to which the posterior parameter $P(\theta | \mathcal{G})$ conforms. Therefore, to estimate the posterior parameter $P(\theta | \mathcal{G})$, the above prediction operation can be performed multiple times to perform sampling on the parameter distribution multiple times (denoted as T times), and correspondingly, a corresponding second classification prediction vector $\tilde{\mathbf{p}}_u^t = \tilde{\mathbf{p}}^t(\mathbf{y}_u | \mathbf{x}_u, \mathbf{A}; \tilde{\theta}_t)$ is obtained each time (denoted as a t th time).

As such, T second classification prediction vectors $\{\tilde{\mathbf{p}}_u^t\}_{t=1}^T$ can be obtained based on the dropedge algorithm.

[0078] Therefore, T second classification prediction vectors $\{\tilde{\mathbf{p}}_u^t\}_{t=1}^T$ can be obtained based on the dropout algorithm or the dropedge algorithm.

5 [0079] Step S33: Subtract second information entropy determined based on the second classification prediction vector by using first information entropy determined based on the first classification prediction vector, to obtain an information gain of training the current graph neural network by using the first unlabeled node.

[0080] In an implementation, the obtained T second classification prediction
10 vectors $\{\tilde{\mathbf{p}}_u^t\}_{t=1}^T$ can be averaged to obtain an expectation for a prediction vector of the unlabeled node v_u :

$$\mathbf{p}_u^{\mathcal{G}} = \mathbf{p}(y_u | \mathbf{x}_u, \mathbf{A}, \mathcal{G}) = \frac{1}{T} \sum_{t=1}^T \tilde{\mathbf{p}}_u^t, \quad \tilde{\theta}_t \sim P(\theta | \mathcal{G}) \quad (11)$$

[0081] Therefore, the information gain $\mathbb{B}_u$ corresponding to the unlabeled node v_u can be calculated by using the following equation:

$$\mathbb{B}_u(y_u, \theta | \mathbf{x}_u, \mathbf{A}, \mathcal{G}) = - \sum_{d=1}^D p_{u,d}^{\mathcal{G}} \log p_{u,d}^{\mathcal{G}} + \frac{1}{T} \sum_{d=1}^D \sum_{t=1}^T \tilde{p}_{u,d}^t \log \tilde{p}_{u,d}^t \quad (12)$$

[0082] In the above equation (12), the first term on the right represents the first information entropy, and the reverse number of the second term represents the second information entropy. For example, D represents a dimension of the classification prediction vector, that is, a total quantity of categories; $p_{u,d}^{\mathcal{G}}$ represents a prediction probability corresponding to a d th category in the first classification prediction vector;
20 and $\tilde{p}_{u,d}^t$ represents a prediction probability corresponding to the d th category in the t th second classification prediction vector.

[0083] Therefore, the information gain $\mathbb{B}_u$ brought by the unlabeled node v_u to the model parameter can be determined based on the dropout algorithm or the dropedge
25 algorithm.

[0084] In addition, it is not preferable that in an implementation, in step S32, the dropout or dropedge algorithm may not be introduced, but the user relational graph is directly processed by using a parameter that is in the first graph neural network and that is not zeroed, to obtain the second classification prediction vector of the unlabeled node
30 v_u , so as to calculate the second information entropy according to the second

classification prediction vector. In an implementation, in step S32, parameter sampling times T in equation (12) can also be 1.

[0085] Therefore, an information gain $\mathbb{B}_u$ that can be brought by each unlabeled node v_u in the unlabeled subset $\mathcal{S}_U$ to the current GNN model parameter can be determined.

[0086] Then, in step S240, a model parameter in the current graph neural network is updated according to a classification prediction vector and a real classification label that are corresponding to each labeled node in the multiple user nodes, and a classification prediction vector, a pseudo classification label, and an information gain that are corresponding to each unlabeled node.

[0087] For example, on one hand, a first loss term is determined according to the classification prediction vector and the real classification label that are corresponding to each labeled node. On the other hand, for each unlabeled node, a second loss term is determined according to the classification prediction vector and the pseudo classification label that are corresponding to the node, and weighted processing is performed on the second loss term by using the information gain corresponding to the node. Further, a comprehensive loss is determined according to the first loss term and the weighted second loss term, so as to update the model parameter in the current graph neural network according to the comprehensive loss.

[0088] In an implementation, the weighting processing includes: normalizing the information gain of each unlabeled node by using a first quantity of information gains corresponding to the first quantity of unlabeled nodes, to obtain a corresponding weighting coefficient; and performing weighting processing on the second loss term by using the weighting coefficient.

[0089] According to an example, the above comprehensive loss can be calculated by using the following equation:

$$\begin{aligned}\mathcal{L}_{st} &= \frac{|\mathcal{S}_U|}{|\mathcal{V}_L \cup \mathcal{S}_U|} \mathbb{E}_{(v_u, y_u) \sim P_{st}(\mathcal{V}, \mathcal{Y})} \bar{\mathbb{B}}_u l(\bar{y}_u, \mathbf{p}_u) \\ &\quad + \frac{|\mathcal{V}_L|}{|\mathcal{V}_L \cup \mathcal{S}_U|} \mathbb{E}_{(v_i, y_i) \sim P_{pop}(\mathcal{V}, \mathcal{Y})} l(y_i, \mathbf{p}_i) \\ \bar{\mathbb{B}}_u &= \frac{\mathbb{B}_u}{\beta \cdot \frac{1}{|\mathcal{S}_U|} \sum_i \mathbb{B}_i}\end{aligned}\tag{13}$$

[0090] In equation (13), $\mathbb{B}_i$ represents an information gain of an i th node in an unlabeled subset $\mathcal{S}_U$. As such, the weight coefficient γ_u of equation (3) can be

approximated by using the normalized result $\bar{\mathbb{B}}_u$ of the information gain $\mathbb{B}_u$, so as to obtain L_{st} of an approximation loss L_{pop} .

[0091] Further, a training gradient can be calculated by using the determined comprehensive loss, and then the model parameter in the current graph neural network model is updated by using a back propagation method according to the training gradient.

[0092] In conclusion, according to the training method for a graph neural network disclosed in this implementation of the present specification, labeled data is expanded by using unlabeled data in a user relational graph, and an information gain is introduced to reduce the difference between a training loss corresponding to distribution of original labeled data and a training loss corresponding to distribution of expanded labeled data, so as to effectively improve a training effect of a GNN model, and further improve prediction accuracy of a trained GNN model on a user node.

[0093] The above describes the training method for a graph neural network used to process a user relational network graph. Actually, the above method can be further extended to training a graph neural network that is associated with a relational network graph of another service object. FIG. 4 is a schematic flowchart of a training method for a neural network according to an implementation. The method can be performed by any apparatus, server, or device cluster that has a calculation and processing capability.

[0094] The training method shown in FIG. 2 involves performing multiple rounds of iterative updating on a graph neural network based on a relational graph, where the relational graph includes multiple object nodes corresponding to multiple service objects, and a connection edge formed by an association relationship existing between the object nodes. In an implementation, the multiple service objects are multiple products. Further, a feature of a product node can include: category, origin, cost, selling price, etc. A label involved in the product node can be a product popularity level, for example, a high-popularity product or a low-popularity product. In an implementation, the multiple service objects are multiple papers. Further, a feature of a paper node can include paper name, keyword, abstract, etc., and a label involved in the paper node can be a field to which the paper belongs, for example, biology, chemistry, physics, and a computer.

[0095] As shown in FIG. 4, the method includes the following steps:

[0096] Step S410: Process the relational graph by using a current graph neural network, to obtain multiple classification prediction vectors corresponding to multiple

service object nodes in the relational graph. Step S420: Allocate a corresponding pseudo classification label to a first quantity of unlabeled nodes in the multiple service object nodes based on the multiple classification prediction vectors. Step S430: Determine, for each of the first quantity of unlabeled nodes, an information gain
5 generated by training the current graph neural network by using the unlabeled node. Step S440: Update a model parameter in the current graph neural network according to a classification prediction vector and a real classification label that are corresponding to each labeled node in the multiple service object nodes, and a classification prediction vector, a pseudo classification label, and an information gain that are corresponding to
10 each unlabeled node.

[0097] It should be noted that for description of the method steps shown in FIG. 4, refer to the description of the method steps shown in FIG. 2 in the above implementation. Details are omitted herein for simplicity.

[0098] In conclusion, according to the training method for a graph neural network
15 disclosed in this implementation of the present specification, labeled data is expanded by using unlabeled data in a relational graph, and an information gain is introduced to reduce the difference between a training loss corresponding to distribution of original labeled data and a training loss corresponding to distribution of expanded labeled data, so as to effectively improve a training effect of a GNN model, and further improve
20 prediction accuracy of a trained GNN model on a service object node.

[0099] Corresponding to the above training method, an implementation of the present specification further discloses a training apparatus. FIG. 5 is a schematic structural diagram of a training apparatus for a graph neural network according to an implementation. The apparatus 500 performs, by using following units, any one of
25 multiple rounds of iterative updating on a graph neural network according to a user relational graph:

[0100] a classification prediction unit 510, configured to process the user relational graph by using a current graph neural network, to obtain multiple classification prediction vectors corresponding to multiple user nodes in the user relational graph; a
30 pseudo label allocation unit 520, configured to allocate a corresponding pseudo classification label to a first quantity of unlabeled nodes in the multiple user nodes based on the multiple classification prediction vectors; an information gain determining unit 530, configured to determine, for each of the first quantity of unlabeled nodes, an information gain generated by training the current graph neural network by using the

unlabeled node; and a parameter updating unit 540, configured to update a model parameter in the current graph neural network according to a classification prediction vector and a real classification label that are corresponding to each labeled node in the multiple user nodes, and a classification prediction vector, a pseudo classification label, and an information gain that are corresponding to each unlabeled node.

[0101] In an implementation, the multiple user nodes comprise a second quantity of unlabeled nodes, and classification prediction vectors comprise multiple prediction probabilities corresponding to multiple categories; and the pseudo label allocation unit 520 is, in some implementations, configured to: for each node in the second quantity of unlabeled nodes, in response to that a maximum prediction probability included in a classification prediction vector corresponding to the node reaches a predetermined threshold, classify the node into the first quantity of unlabeled nodes, and determine a category corresponding to the maximum prediction probability as a pseudo classification label of the node.

[0102] In an implementation, the information gain determining unit 530 includes: a training subunit 531, configured to: for a first unlabeled node of the first quantity of unlabeled nodes, train the current graph neural network by using a first classification prediction vector and a pseudo classification label that are corresponding to the first unlabeled node; a prediction subunit 532, configured to determine a second classification prediction vector of the first unlabeled node based on a trained first graph neural network; an information entropy determining subunit 533, configured to determine first information entropy according to the first classification prediction vector, and determine second information entropy according to the second classification prediction vector; and a gain determining subunit 534, configured to obtain the information gain based on a difference between the second information entropy and the first information entropy.

[0103] Further, in an example implementation, the trained first graph neural network includes multiple aggregation layers and an output layer. The prediction subunit 532 is, for example, configured to: perform, at an aggregation layer in the multiple aggregation layers, random zeroing processing on vector elements in multiple aggregation vectors for the multiple user nodes that are output by an upper aggregation layer, and determine, based on the multiple aggregation vectors after the random zeroing processing, multiple aggregation vectors that are output by the aggregation layer for the multiple user nodes; and process, at the output layer, an aggregation vector

output by a last aggregation layer for the first unlabeled user node, to obtain the second classification prediction vector.

[0104] In an example implementation, the trained first graph neural network includes multiple aggregation layers and an output layer. The prediction subunit 532 is, in some implementations, configured to: perform, at an aggregation layer in the multiple aggregation layers, random zeroing processing on a matrix element in an adjacency matrix corresponding to the user relational graph, and determine, based on the adjacency matrix after the random zeroing processing and multiple aggregation vectors that are output by an upper aggregation layer for the multiple user nodes, multiple aggregation vectors for the multiple user nodes that are output by the aggregation layer; and process, at the output layer, an aggregation vector output by a last aggregation layer for the first unlabeled user node, to obtain the second classification prediction vector.

[0105] Further, in a further example implementation, the prediction subunit 532 is further configured to: perform an operation of determining the second classification prediction vector for multiple times, and correspondingly obtain multiple second classification prediction vectors. The information entropy determining subunit 533 is, for example, configured to determine an average value of multiple pieces of information entropy respectively corresponding to the multiple second classification prediction vectors as the second information entropy.

[0106] In an implementation, the parameter updating unit 540 is configured to determine a first loss term according to the classification prediction vector and the real classification label that are corresponding to each labeled node; determine a second loss term for each unlabeled node according to the classification prediction vector and the pseudo classification label that are corresponding to each unlabeled node, and weight the second loss term by using the information gain corresponding to the unlabeled node; and update the model parameter according to the first loss term and the weighted second loss term.

[0107] In an example implementation, that the parameter updating unit 540 is configured to perform the above weighting processing includes: normalizing the information gain of each unlabeled node by using a first quantity of information gains corresponding to the first quantity of unlabeled nodes, to obtain a corresponding weighting coefficient; and performing weighting processing by using the weighting coefficient.

[0108] In conclusion, according to the training apparatus for a graph neural network disclosed in this implementation of the present specification, labeled data is expanded by using unlabeled data in a user relational graph, and an information gain is introduced to reduce the difference between a training loss corresponding to distribution of original
5 labeled data and a training loss corresponding to distribution of expanded labeled data, so as to effectively improve a training effect of a GNN model, and further improve prediction accuracy of a trained GNN model on a user node.

[0109] FIG. 6 is a schematic structural diagram of a training apparatus for a graph neural network according to an implementation. As shown in FIG. 6, the apparatus 600
10 performs, by using following units, any one of multiple rounds of iterative updating on a graph neural network according to a pre-constructed relational graph: a classification prediction unit 610, configured to process the relational graph by using a current graph neural network, to obtain multiple classification prediction vectors corresponding to multiple service object nodes in the relational graph; a pseudo label allocation unit 620,
15 configured to allocate a corresponding pseudo classification label to a first quantity of unlabeled nodes in the multiple service object nodes based on the multiple classification prediction vectors; an information gain determining unit 630, configured to determine, for each of the first quantity of unlabeled nodes, an information gain generated by training the current graph neural network by using the unlabeled node; and a parameter
20 updating unit 640, configured to update a model parameter in the current graph neural network according to a classification prediction vector and a real classification label that are corresponding to each labeled node in the multiple service object nodes, and a classification prediction vector, a pseudo classification label, and an information gain that are corresponding to each unlabeled node.

[0110] In an implementation, the multiple service object nodes comprise a second
25 quantity of unlabeled nodes, and classification prediction vectors comprise multiple prediction probabilities corresponding to multiple categories; and the pseudo label allocation unit 620 is, in some implementations, configured to: for each node in the second quantity of unlabeled nodes, in response to that a maximum prediction
30 probability included in a classification prediction vector corresponding to the node reaches a predetermined threshold, classify the node into the first quantity of unlabeled nodes, and determine a category corresponding to the maximum prediction probability as a pseudo classification label of the node.

[0111] In an implementation, the information gain determining unit 630 includes: a training subunit 631, configured to: for a first unlabeled node of the first quantity of unlabeled nodes, train the current graph neural network by using a first classification prediction vector and a pseudo classification label that are corresponding to the first
5 unlabeled node; a prediction subunit 632, configured to determine a second classification prediction vector of the first unlabeled node based on a trained first graph neural network; an information entropy determining subunit 633, configured to determine first information entropy according to the first classification prediction vector, and determine second information entropy according to the second classification
10 prediction vector; and a gain determining subunit 634, configured to obtain the information gain based on a difference between the second information entropy and the first information entropy.

[0112] Further, in an example implementation, the trained first graph neural network includes multiple aggregation layers and an output layer. The prediction
15 subunit 632 is, for example, configured to: perform, at an aggregation layer in the multiple aggregation layers, random zeroing processing on vector elements in multiple aggregation vectors for the multiple service object nodes that are output by an upper aggregation layer, and determine, based on the multiple aggregation vectors after the random zeroing processing, multiple aggregation vectors that are output by the
20 aggregation layer for the multiple service object nodes; and process, at the output layer, an aggregation vector output by a last aggregation layer for the first unlabeled service object node, to obtain the second classification prediction vector.

[0113] In an example implementation, the trained first graph neural network includes multiple aggregation layers and an output layer. The prediction subunit 632 is,
25 for example, configured to: perform, at an aggregation layer in the multiple aggregation layers, random zeroing processing on a matrix element in an adjacency matrix corresponding to the service object relational graph, and determine, based on the adjacency matrix after the random zeroing processing and multiple aggregation vectors that are output by an upper aggregation layer for the multiple service object nodes,
30 multiple aggregation vectors for the multiple service object nodes that are output by the aggregation layer; and process, at the output layer, an aggregation vector output by a last aggregation layer for the first unlabeled service object node, to obtain the second classification prediction vector.

[0114] Further, in a further example implementation, the prediction subunit 632 is further configured to: perform an operation of determining the second classification prediction vector for multiple times, and correspondingly obtain multiple second classification prediction vectors. The information entropy determining subunit 633 is, in some implementations, configured to determine an average value of multiple pieces of information entropy respectively corresponding to the multiple second classification prediction vectors as the second information entropy.

[0115] In an implementation, the parameter updating unit 640 is configured to determine a first loss term according to the classification prediction vector and the real classification label that are corresponding to each labeled node; determine a second loss term for each unlabeled node according to the classification prediction vector and the pseudo classification label that are corresponding to each unlabeled node, and weight the second loss term by using the information gain corresponding to the unlabeled node; and update the model parameter according to the first loss term and the weighted second loss term.

[0116] In an example implementation, that the parameter updating unit 640 is configured to perform the above weighting processing includes: normalizing the information gain of each unlabeled node by using a first quantity of information gains corresponding to the first quantity of unlabeled nodes, to obtain a corresponding weighting coefficient; and performing weighting processing by using the weighting coefficient.

[0117] In conclusion, according to the training apparatus for a graph neural network disclosed in this implementation of the present specification, labeled data is expanded by using unlabeled data in a service object relational graph, and an information gain is introduced to reduce the difference between a training loss corresponding to distribution of original labeled data and a training loss corresponding to distribution of expanded labeled data, so as to effectively improve a training effect of a GNN model, and further improve prediction accuracy of a trained GNN model on a service object node.

[0118] According to an implementation of an aspect, a computer readable storage medium on which a computer program is stored is further provided. When the computer program is executed in a computer, the computer is caused to perform the method described with reference to FIG. 2 or FIG. 3.

[0119] According to an implementation of still another aspect, a computing device is further provided and includes a memory and a processor. Executable code is stored

in the memory, and when executing the executable code, the processor implements the method described with reference to FIG. 2 or FIG. 3. A person skilled in the art should be aware that, in one or more of the above examples, the functions described in the present specification can be implemented by using hardware, software, firmware, or
5 any combination thereof. When these functions are implemented by software, they can be stored in a computer readable medium or transmitted as one or more instructions or code lines on the computer readable medium.

[0120] The example implementations mentioned above further describe the object, technical solutions and beneficial effects of the present disclosure. It should be
10 understood that the above descriptions are merely example implementations of the present disclosure and are not intended to limit the protection scope of the present disclosure. Any modification, equivalent replacement and improvement made based on the technical solution of the present disclosure shall fall within the protection scope of the present disclosure.

15

CLAIMS

What is claimed is:

1. A method for training a graph neural network, comprising:
 - 5 performing multiple rounds of updating on a graph neural network based on a relational graph, a round of the multiple rounds including:
 - processing the relational graph by using a current graph neural network, to obtain multiple classification prediction vectors corresponding to multiple nodes in the relational graph;
 - 10 allocating a corresponding pseudo classification label to a first quantity of unlabeled nodes in the multiple nodes based on the multiple classification prediction vectors;
 - determining, for each of the first quantity of unlabeled nodes, an information gain generated by training the current graph neural network by using the unlabeled node; and
 - 15 updating a model parameter in the current graph neural network according to a classification prediction vector and a real classification label that are corresponding to each labeled node in the multiple nodes, and a classification prediction vector, a pseudo classification label, and an information gain that are corresponding to each unlabeled node.
 - 20
2. The method according to claim 1, wherein the multiple nodes comprise a second quantity of unlabeled nodes, and classification prediction vectors comprise multiple prediction probabilities corresponding to multiple categories;
 - 25 wherein the allocating the corresponding pseudo classification label to the first quantity of unlabeled nodes in the multiple nodes based on the multiple classification prediction vectors includes:
 - for each node in the second quantity of unlabeled nodes, in response to that a maximum prediction probability included in a classification prediction vector
 - 30 corresponding to the node reaches a threshold, classifying the node into the first quantity of unlabeled nodes, and determining a category corresponding to the maximum prediction probability as a pseudo classification label of the node.

3. The method according to claim 1, wherein the determining, for each of the first quantity of unlabeled nodes, the information gain generated by training the current graph neural network by using the unlabeled node includes:

for a first unlabeled node of the first quantity of unlabeled nodes,

5 training the current graph neural network by using a first classification prediction vector and a pseudo classification label that are corresponding to the first unlabeled node, and determining a second classification prediction vector of the first unlabeled node based on a trained first graph neural network;

 determining first information entropy according to the first classification
10 prediction vector;

 determining second information entropy according to the second classification prediction vector; and

 obtaining the information gain based on a difference between the second information entropy and the first information entropy.

15

4. The method according to claim 3, wherein the trained first graph neural network comprises multiple aggregation layers and an output layer; and the determining the second classification prediction vector of the first unlabeled node based on the trained first graph neural network includes:

20 performing, at an aggregation layer in the multiple aggregation layers, random zeroing processing on vector elements in multiple aggregation vectors for the multiple nodes that are output by an upper aggregation layer, and determining, based on the multiple aggregation vectors after the random zeroing processing, multiple aggregation vectors that are output by the aggregation layer for the multiple nodes; and

25 processing, at the output layer, an aggregation vector output by a last aggregation layer for the first unlabeled node, to obtain the second classification prediction vector.

5. The method according to claim 3, wherein the trained first graph neural
30 network comprises multiple aggregation layers and an output layer; and the determining the second classification prediction vector of the first unlabeled node based on the trained first graph neural network includes:

 performing, at an aggregation layer in the multiple aggregation layers, random zeroing processing on a matrix element in an adjacency matrix corresponding to the

relational graph, and determining, based on the adjacency matrix after the random zeroing processing and multiple aggregation vectors that are output by an upper aggregation layer for the multiple nodes, multiple aggregation vectors for the multiple nodes that are output by the aggregation layer; and

5 processing, at the output layer, an aggregation vector output by a last aggregation layer for the first unlabeled node, to obtain the second classification prediction vector.

6. The method according to claim 4, wherein the determining the second
10 classification prediction vector of the unlabeled node based on the trained first graph neural network includes:

 performing for multiple times an operation of determining the second classification prediction vector to correspondingly obtain multiple second classification prediction vectors;

15 wherein the determining the second information entropy according to the second classification prediction vector includes:

 determining an average value of multiple pieces of information entropy respectively corresponding to the multiple second classification prediction vectors as the second information entropy.

20

7. The method according to claim 1, wherein the updating the model parameter in the current graph neural network according to the classification prediction vector and the real classification label that are corresponding to each labeled node in the multiple nodes, and the classification prediction vector, the pseudo classification label, and the
25 information gain that are corresponding to each unlabeled node includes:

 determining a first loss term according to the classification prediction vector and the real classification label that are corresponding to each labeled node;

 determining a second loss term for each unlabeled node according to the classification prediction vector and the pseudo classification label that are
30 corresponding to each unlabeled node, and weighting the second loss term by using the information gain corresponding to the unlabeled node; and

 updating the model parameter according to the first loss term and the weighted second loss term.

8. The method according to claim 7, wherein the weighting the second loss term by using the information gain corresponding to the unlabeled node includes:

normalizing the information gain of each unlabeled node by using a first quantity of information gains corresponding to the first quantity of unlabeled nodes, to

5 obtain a corresponding weighting coefficient; and

performing weighting processing by using the weighting coefficient.

9. A computing system, comprising at least one memory and at least one processor, the at least one memory storing executable instructions, which when
10 executed by the at least one processor enable the at least one processor to implement actions including:

performing multiple rounds of updating on a graph neural network based on a relational graph, a round of the multiple rounds including:

processing the relational graph by using a current graph neural network,
15 to obtain multiple classification prediction vectors corresponding to multiple nodes in the relational graph;

allocating a corresponding pseudo classification label to a first quantity of unlabeled nodes in the multiple nodes based on the multiple classification prediction vectors;

20 determining, for each of the first quantity of unlabeled nodes, an information gain generated by training the current graph neural network by using the unlabeled node; and

updating a model parameter in the current graph neural network according to a classification prediction vector and a real classification label that are
25 corresponding to each labeled node in the multiple nodes, and a classification prediction vector, a pseudo classification label, and an information gain that are corresponding to each unlabeled node.

10. The computing system according to claim 9, wherein the multiple nodes
30 comprise a second quantity of unlabeled nodes, and classification prediction vectors comprise multiple prediction probabilities corresponding to multiple categories;

wherein the allocating the corresponding pseudo classification label to the first quantity of unlabeled nodes in the multiple nodes based on the multiple classification prediction vectors includes:

for each node in the second quantity of unlabeled nodes, in response to that a maximum prediction probability included in a classification prediction vector corresponding to the node reaches a threshold, classifying the node into the first quantity of unlabeled nodes, and determining a category corresponding to the maximum prediction probability as a pseudo classification label of the node.

11. The computing system according to claim 9, wherein the determining, for each of the first quantity of unlabeled nodes, the information gain generated by training the current graph neural network by using the unlabeled node includes:

10 for a first unlabeled node of the first quantity of unlabeled nodes,

training the current graph neural network by using a first classification prediction vector and a pseudo classification label that are corresponding to the first unlabeled node, and determining a second classification prediction vector of the first unlabeled node based on a trained first graph neural network;

15 determining first information entropy according to the first classification prediction vector;

determining second information entropy according to the second classification prediction vector; and

obtaining the information gain based on a difference between the second

20 information entropy and the first information entropy.

12. The computing system according to claim 11, wherein the trained first graph neural network comprises multiple aggregation layers and an output layer; and the determining the second classification prediction vector of the first unlabeled node based

25 on the trained first graph neural network includes:

performing, at an aggregation layer in the multiple aggregation layers, random zeroing processing on vector elements in multiple aggregation vectors for the multiple nodes that are output by an upper aggregation layer, and determining, based on the multiple aggregation vectors after the random zeroing processing, multiple aggregation

30 vectors that are output by the aggregation layer for the multiple nodes; and

processing, at the output layer, an aggregation vector output by a last aggregation layer for the first unlabeled node, to obtain the second classification prediction vector.

13. The computing system according to claim 11, wherein the trained first graph neural network comprises multiple aggregation layers and an output layer; and the determining the second classification prediction vector of the first unlabeled node based on the trained first graph neural network includes:

5 performing, at an aggregation layer in the multiple aggregation layers, random zeroing processing on a matrix element in an adjacency matrix corresponding to the relational graph, and determining, based on the adjacency matrix after the random zeroing processing and multiple aggregation vectors that are output by an upper aggregation layer for the multiple nodes, multiple aggregation vectors for the multiple
10 nodes that are output by the aggregation layer; and

processing, at the output layer, an aggregation vector output by a last aggregation layer for the first unlabeled node, to obtain the second classification prediction vector.

15 14. The computing system according to claim 13, wherein the determining the second classification prediction vector of the unlabeled node based on the trained first graph neural network includes:

performing for multiple times an operation of determining the second classification prediction vector to correspondingly obtain multiple second classification
20 prediction vectors;

wherein the determining the second information entropy according to the second classification prediction vector includes:

determining an average value of multiple pieces of information entropy respectively corresponding to the multiple second classification prediction vectors as
25 the second information entropy.

15. The computing system according to claim 9, wherein the updating the model parameter in the current graph neural network according to the classification prediction vector and the real classification label that are corresponding to each labeled node in
30 the multiple nodes, and the classification prediction vector, the pseudo classification label, and the information gain that are corresponding to each unlabeled node includes:

determining a first loss term according to the classification prediction vector and the real classification label that are corresponding to each labeled node;

determining a second loss term for each unlabeled node according to the classification prediction vector and the pseudo classification label that are corresponding to each unlabeled node, and weighting the second loss term by using the information gain corresponding to the unlabeled node; and

5 updating the model parameter according to the first loss term and the weighted second loss term.

16. The computing system according to claim 15, wherein the weighting the second loss term by using the information gain corresponding to the unlabeled node
10 includes:

normalizing the information gain of each unlabeled node by using a first quantity of information gains corresponding to the first quantity of unlabeled nodes, to obtain a corresponding weighting coefficient; and

performing weighting processing by using the weighting coefficient.
15

17. A non-transitory computer readable storage medium that stores computer executable instructions, which when executed by a processor, configure the processor to perform actions comprising:

performing multiple rounds of updating on a graph neural network based on a
20 relational graph, a round of the multiple rounds including:

processing the relational graph by using a current graph neural network, to obtain multiple classification prediction vectors corresponding to multiple nodes in the relational graph;

allocating a corresponding pseudo classification label to a first quantity
25 of unlabeled nodes in the multiple nodes based on the multiple classification prediction vectors;

determining, for each of the first quantity of unlabeled nodes, an information gain generated by training the current graph neural network by using the unlabeled node; and

30 updating a model parameter in the current graph neural network according to a classification prediction vector and a real classification label that are corresponding to each labeled node in the multiple nodes, and a classification prediction vector, a pseudo classification label, and an information gain that are corresponding to each unlabeled node.

18. The storage medium according to claim 16, wherein the multiple nodes comprise a second quantity of unlabeled nodes, and classification prediction vectors comprise multiple prediction probabilities corresponding to multiple categories;

5 wherein the allocating the corresponding pseudo classification label to the first quantity of unlabeled nodes in the multiple nodes based on the multiple classification prediction vectors includes:

 for each node in the second quantity of unlabeled nodes, in response to that a maximum prediction probability included in a classification prediction vector
10 corresponding to the node reaches a threshold, classifying the node into the first quantity of unlabeled nodes, and determining a category corresponding to the maximum prediction probability as a pseudo classification label of the node.

19. The storage medium according to claim 17, wherein the determining, for
15 each of the first quantity of unlabeled nodes, the information gain generated by training the current graph neural network by using the unlabeled node includes:

 for a first unlabeled node of the first quantity of unlabeled nodes,

 training the current graph neural network by using a first classification
 prediction vector and a pseudo classification label that are corresponding to the first
20 unlabeled node, and determining a second classification prediction vector of the first unlabeled node based on a trained first graph neural network;

 determining first information entropy according to the first classification
 prediction vector;

 determining second information entropy according to the second
25 classification prediction vector; and

 obtaining the information gain based on a difference between the second
 information entropy and the first information entropy.

20. The storage medium according to claim 19, wherein the trained first graph
30 neural network comprises multiple aggregation layers and an output layer; and the determining the second classification prediction vector of the first unlabeled node based on the trained first graph neural network includes:

 performing, at an aggregation layer in the multiple aggregation layers, random
 zeroing processing on vector elements in multiple aggregation vectors for the multiple

nodes that are output by an upper aggregation layer, and determining, based on the multiple aggregation vectors after the random zeroing processing, multiple aggregation vectors that are output by the aggregation layer for the multiple nodes; and

- 5 processing, at the output layer, an aggregation vector output by a last aggregation layer for the first unlabeled node, to obtain the second classification prediction vector.