Using autoencoders for building recommender systems

Formulation of the problem

Like many recommender systems, we have the products, users, and rates that expose users (explicitly or implicitly) products. Our task to predict the evaluation of products, which have not yet assessed the user and thus to predict the products that can be highly appreciated by the users, or products that may be of interest to users. (What is the function of recommender systems - find products which may be of potential interest to the user.)

It was necessary to develop a recommender system that would:

1. It was optimal in terms of speed after the training model.
2. It would require a minimum cost of processing new data received. Those recommender systems that would not require a complete re-training or additional training after receiving new data, or to the operation of this kind would be minimal (perhaps we have lost as a job, but would not require significant investment to rebuild the model).

Decision

The basic concept of the proposed solutions

As in any classic recommender systems we have: 1) products, 2) users and 3) users ratings of products.

For each product and user, we define the vector in the N-dimensional space. Those. each product and each user is represented by a point in our N-dimensional space. We are using two spaces. One users space, a second - for products. (Vectors users IDs and vectors products IDs).

We also use the data about the user, the product evaluation in the construction of the model and in obtaining results for the users of products ratings.

For example, for MovieLens 1M data set we use:

- such information about you as the user's gender and age (vector size 2 elements);
- or for the film (the product), we use the year of the film (unit vector);
- and for the rating in what day of the week it was exhibited (the unit length of the vector).

(See the source code.) It should also be borne in mind that also the vectors used for users IDs and products (size N).

Such data heuristically allow you to make more accurate rate of the users for the products. This data is converted into a vector:

- Information about the user;
- Information about the product;
- Information about rates;
- Rates itself or multiple rates.

Using autoencoders

For N-dimensional vectors for each product / user, we use the following scheme using autoencoder-s.

Let me remind you that autoencoder trained to compress data filed on its inputs and to provide input data in a compressed form.

We use two autoencoders. The input (and the target value), we provide the list in M (number equal to 5-7) entries, each of which is:

1. For autoencoder, **which is trained in the assessment of user data** , we use the values: 1) vector of the user ID; 2) User data set evaluation this product (the vector of a data of the user); 3) rating that the user put this product (vector of rating data); 4) rates itself (rates vector). For the autoencoder all values are given for a single product (one training cycle), although for different rates (which we have in the system), by a users of this product. (For data MovieLens1M we take M records: 1) a vector of N values (vector of user ID); 2) vector of 2 values (-- data about the user); 3) the vector of values of size one (- day of the week when the use rated the product); 4) vector from one value - the rate. Those, we put M * (N + 2 + 1 + 1) values on the autoencoder inputs in the case of MovieLens1M.)
2. For the autoencoder, **which is trained on the data of products rates**, we use the values: 1) the product identifier vector (vector of N elements); 2) Information about the product, which assessed the user (product data vector); 3) the rates data of the product by the user (vector of rating data); 4) rates itself (rates vector). Moreover, all values are set for one user and for different products, which were evaluated according to this user.

Thus, we get into the process of learning (learning at each step) autoencoders at the inputs (and target values), a list of estimates of M:

1. For a single product - M user evaluations rated the product (in the first autoencoder)
2. For a single user - M product assessments, the estimated user data (second autoencoder)

We also get, **the first autoencoder vector encodes a product ID, and the second vector encodes the user ID.**

In the minimum case, we can set (for the first autoencoder) vector only user IDs and evaluation. And for the second autoencoder -- only vector product identifiers and evaluation. Those, we not use the user data products and assessments. As shown in Fig. 1.

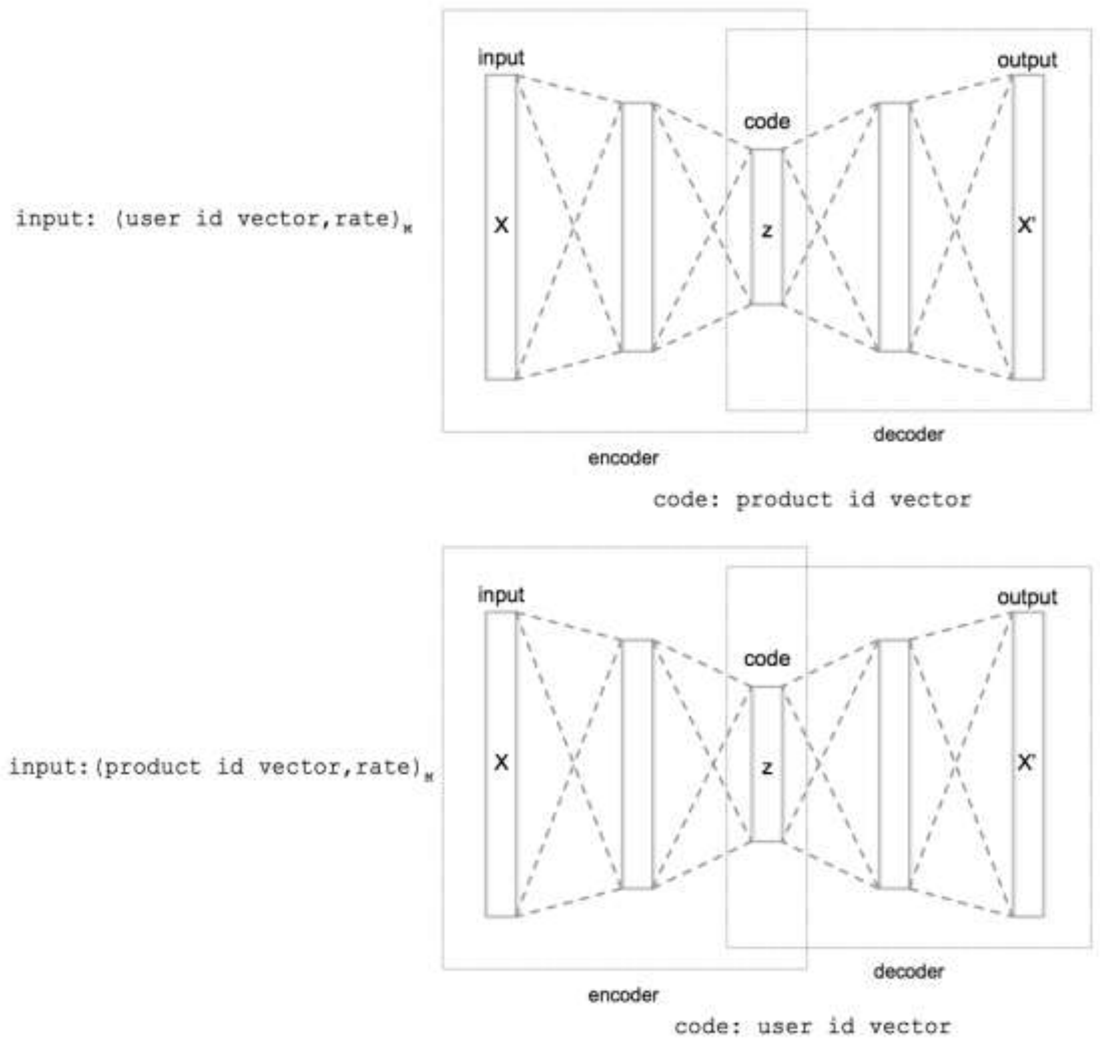

Fig. 1. Driving training autoensoders when using the minimum data for training (only vector of user IDs, product IDs of the vector and evaluation). Also scheme for vectors encoding identifiers when autoencode.

How to receive users ID vectors and products ID vectors

They are obtained as follows.

After several cycles of training of autoencoders (about 100..1000), we put to the second autoencoder value of a single user products ratings (for each user putting K different options (24..64), or the values of users ratings for a single product for the first autroencoder.

For example, for a single user, we putted K options (random) product rates (second autoencoder), ie K formed of different inputs for autoencoder. At the same time we got the K different values autoencoder-compressed and received at work.(That is, we have K vectors of partial user IDs.) (It is clear that we must build a autoencoder, which would have a middle layer consisting of N elements, that would be equal to the size of the vector user/product IDs.) We find the average of these K values. This will be the new value of the user ID vector, to which we will move from the current value of the user ID vector. So we are doing for a number of (L1) users and obtain averages.

Similarly, we obtain the average L2 values of autoencoder-compressed values, trained on users opinions (or encoding product identifiers).

If initially we use a random value vector IDs of users and products, we can move from our original values to the received L1 and L2 average compressed (average compressed will be the new target vectors identifiers).

We use heuristics to smooth the movement of the target values of IDs. We are gradually changing the vector of users and products IDs to the target values of the vectors obtained users and products IDs.

In order to obtain satisfactory values for the vectors of users and the products IDs we must consider another aspect of autoencoders - is the "breath" or coded values or swimming group of values within certain limits. At the "breath" of encoded values autoencoders all received values move to a some bias. As a result, if we will use the encoded values without any correction, the vectors will be difficult to converge identifiers. This problem is solved quite easily. As this is done, I will not describe in detail. This point is resolved by finding new secondary displacement vectors obtained of users/products IDs relative to the previous value of the vector users/products IDs. And correction of previous vector values IDs further carried out without taking into account the average displacement. (See source code for more information.)

Preparation products ratings for users

When we have IDs, they are ordered in the N-dimensional space of the users or the products. Related products/users (based on user ratings) are at a short distance of (Euclidean metric). Next, we can use a neural network for the prediction function on: 1) the vector product ID; 2) data on the product; 3) ratings data (not the rates itself, but, for example, day of week / time when the user wants to see / have watched the film); 4) vector user ID; 5) user data. Using these data, we train a neural network to predict the rating(s), which can give the user to the product under given conditions.

Building a model

We reiterate our training of autoencoders and neural network to approximate the function by the user evaluation of the product. There are some nuances of cold start training of autoencoders of random initial values. The proposed heuristics for the solution of these problems can be found in the proposed source code for this article.

After the training, we have a model consisting of usera and products IDs, of neural networks - two autoencoders and neural network for prediction of ratings.

Working with the system after receiving the model

After learning system, we can predict the ratings for products that are not rated by users.

But it is also an important point is the possibility that we can:

1. Add new ratings and correct users and products IDs vectors
2. Adding new products and new users. Those, calculating new vectors for users and products IDs.

To calculate the vectors of the new identity we use autoencoders of the resulting model. In this case, we use the same process as the process of obtaining identifiers of the vectors in the training model, except that we do not teach autoencoders on the new data, but only calculates the identifiers.

The process of calculating the identifiers can be used and when new estimates. In this case, we can adjust identifiers. In the framework of the already trained model, ie, again not retrain / train no further autoencoders.

When calculating/correcting identifiers, we, of course, in order to adjust the processing speed, we do not correct the part of the model given by neural networks. But the new data may change/extend the model, so that such a change/extension can not be expressed only in the calculation of the vectors of users and products IDs, so a good heuristic can be additional training model through a certain period of time (once a day / week / month / quarter). This additional training can be done in parallel within the prediction system rates, while adding of new products, users, new rates.

What makes the proposed approach

In addition to solving problems, putting in the problem, as this approach makes it possible to enter into the machine learning system memory. Here, the memory can be immediately understood the calculated identifiers. The model can represent the objects of the external world, bearing in itself a description of the external world from the point of view of the system (some systems rates of data objects). This IDs can be used to describe system objects.

In addition, this approach makes it possible to set a large number of parameters that define the vectors of users and products IDs, as well as ratings parameters.