Pricing for online shops

Scenario

The issue of automatic price optimization in e-commerce is increasing in importance. This is in particular due to the fact that significant increases in margins can be achieved using intelligent pricing strategies. In addition to the standard algorithms basically designed for optimizing the prices for each product in an online shop, special algorithms are developed that are used for such matters as the profit-oriented sale of combined products or the rapid selling off of perishable goods. The time has now come for the issue of automatic price optimisation to become the focus of the DMC task. In 2012 this again comprises an offline and an online part that are assessed independently of each other.

Tasks

In the offline part, the first task, the prices and associated sales figures are stated for a specific period and for selected products. With the aid of this data a model is to be developed that describes the dependencies between the data. Based on this model, the sales figures of the subsequent period have to be forecasted in an application phase. The winner of the first task is the team that predicts the sales figures as precisely as possible.

The challenge for the second task is to implement an agent that takes on the pricing role for an online shop. The agents and therefore the shops of the individual participants finally compete against each other in a multi-agent system simulation where they aim to maximize their profits. Therefore, the market demand is stipulated by the prudsys AG. The winner is the team that achieves the highest profit.

Data

Data files are required for the individual tasks. These data files are structured text files containing data sets. Please note the following requirements regarding the files:

1. Each data set is in a row of its own, ending with "CR" ("carriage return", 0xD) or "CR" and "LF" ("carriage return" and "line feed", 0xD and 0xA).
2. The first row has the same structure as the data sets, but contains the names of the respective columns (data fields).
3. The top row and each data set contain several fields separated from each other by the "|" symbol.
4. There is no escape character, quotes are not used.
5. ASCII is the character set used.

The only column headings that can actually appear here are day, itemID, price and quantity. The field values contain character strings. For day, itemID and quantity, the value ranges cover any non-negative integer values, whereas the value range for price permits positive, real numbers with a maximum of two digits after the decimal point.

Cumulative data in the following format is provided for the first task so that the model is able to learn (*"train.txt"*):

Column name	Description
day	Time stamp to indicate a day
itemID	Number of product
price	Price of product
quantity	Sales units of product

An excerpt from a sample file with fictitious data could look like this:

```
3|4|2|5
3|5|9.95|13
…
14|10|57.55|2
…
```

On the third day 5 units of product 4 were sold at a price of 2 and 13 units of product 5 were sold at a price of 9.95 each. On day 14 2 units of product 10 were sold at a price of 57.55 each.

The format of the application data is as follows (*"class.txt"*):

Column name	Description
day	Time stamp to indicate a day
itemID	Number of product
price	Price of product

For example

```
43|10|61
43|11|8.92
…
48|14|13.45
…
```

Submission

Participants can submit their results up until 15 May 2012. The detailed task descriptions below explain how to submit the results.

Task 1: Forecast sales figures

Procedure

The product prices vary in order to manage the sales figures in an online shop. The development of the purchase data is recorded over the course of eight weeks. The aim should be to learn the dependencies between the prices, periodicity over time and the sales figures using a mathematical model. Full data will be provided for the first six weeks. The relevant file is called *"train.txt"*. The sales figures for the seventh and eighth weeks should then be forecast using the given prices. The file with the associated data is called *"class.txt"*.

Submission

For the submission of the solution a file has to be provided in the following format:

Column name	Description
day	Time stamp to indicate a day
itemID	Number of product
price	Price of product
quantity	Forecast sales units of product

Here, the first three columns must completely match the columns in the original *"class.txt"* file. The last column should contain the calculated forecast sales figures in each case.

The results file must be sent to dmc_task1@prudsys.de as a zipped text file email attachment. The name of both the zip file and the included text file must be made up of the team name, the task number *task1* and the type (zip or txt):

“*<Teamname>_task1.zip*“, (z.B. *TU_Chemnitz_1_task1.zip*)

and

“*<Teamname>_task1.txt*“, (z.B. *TU_Chemnitz_1_task1.txt*).

The team name has been sent to the team leader in the entry confirmation.

Evaluation

The solutions received will be graded and compared with the following error function:

$$E = \sqrt{\sum_i \sum_j \left(m_{ij} - \hat{m}_{ij}\right)^2} .$$

Here, the m_{ij} are the actual real sales of product *i* on day *j* and $\hat{m}_{ij}$ are the forecast sales of product *i* on day *j*. The teams whose error functions have the smallest values will win. If the team results are equal the winner will be determined by drawing lots.

Task 2: Pricing Online Market

Detailed scenario

The participants should take on the price decisions for an online shop using an agent they have developed themselves and thus compete in a simulated online market. Therefore, the users can periodically modify the prices in their shops. There is precisely one product to make matters simple. The period of time is 60 days. The periods correspond with days such that new prices can be set for each day. Further, the product's purchase price (p_{EK}) is 10, the recommended retail price is (p_{UVP}) 15. The shops set their prices at the start of a period. The permitted interval for the price is $[0.5 \cdot p_{UVP}, 1.5 \cdot p_{UVP}]$. At the end of a period the shop is told how many products it sold and the prices of the competitors. This information can be used to calculate the profit and develop an intelligent pricing strategy.

The sales function is stipulated by the prudsys AG and has the following characteristics and others:

1. The lower the price of the product in a shop, the higher the sales.
2. The following statement applies: If each shop uses the same price the total revenue (sum of sales made by all shops) is independent of the number of shops.
3. The competitors influence both their own sales and those of other participants using the selected prices.
4. The prices of the past periods affect the current sales.

The final winner is the team whose agent produces the highest profit after 60 periods.

Parameters for implementation

The implementation must take the form of a Java class. It must be able to run on a PC with 1024 MB guaranteed heap space and a 1.6 GHz Intel Core Duo Processor. The "sandbox" principle applies, i.e. the application must not connect to external resources (e.g. http connections). Java libraries may be used, but these must be under free licence. The application must not create files, incur expenditures or emit signals.

The implementation must also support the following simple interface:

```
import java.util.List;

public Interface ShopAgentInterface {

  /**
   * Get the price for the current period.
   *
   * @return price
   * @throws Exception
   */
  public double getPrice() throws Exception;

  /**
```

```
   * Parses the number of sales for the current period.
   *
   * @param sales number of sales
   * @throws Exception
   */
  public void parseSales(int sales) throws Exception;

  /**
   * Parses the prices of the competitors.
   *
   * @param prices prices of the competitors
   * @throws Exception
   */
  public void parsePrices(List<Double> prices) throws Exception;
}
```

The class must be initialized via the class's default constructor (no input parameter). The `getPrice()` method queries the current price. This must be returned with precisely two decimal places. The `parseSales(int sales)` method provides the sales thus achieved. The `parsePrices(List<Double> prices)` method provides the prices of the competitors in an order that remains the same for all periods. The response times of all methods called up, in total over all 60 periods, may not exceed 30 seconds. If this number is exceeded during the run the price is kept constant for the remaining periods. The *SimpleAgent.java* file contains a sample implementation. It uses the `getPrice()` method to calculate the average of the competitor prices and selects the result as the price for the product.

Submission

Result files must be zipped and sent as one file by email to dmc_task2@prudsys.de. The zip file must not be larger than 5 MB. The file name must be made up of the team name, the task number *task2* and the type (.zip):

"<Teamname>_task2.zip", (z.B. *TU_Chemnitz_1_task2.zip*).

The team name has been sent to the team leader in the entry confirmation.

Further the full name of the class implementing the interface must be stated in the email. Any external Java libraries used must also be submitted. The implementation can be submitted as source code or fully compiled byte code. If source code is submitted, it must be possible to compile it with a current jdk1.6 version and an Ant build file must be included. If byte code is submitted, it must be executable with a current jre6 version. A short text file outlining the details of the logic of the implemented algorithm would be helpful.

Evaluation

A simulation software from the prudsys AG will query the prices of the participating agents using `getPrice()`. All prices are handed over jointly to the sales function. The calculated sales of the participating agents are then transferred to them using `parseSales(…)`.

Before the next period starts the `parsePrices(…)` method is used to hand over the prices of the competitors to the participants. The process is repeated for each period.

In order to finally determine which shop has made the highest profit the profit of each agent is calculated using the following formula:

$$G=\sum_{k=1}^{60} m_k(p_k-p_{EK}) ,$$

whereby m_k is the number of products sold in the k-th period (sales), p_k the price of the product chosen in period *k* and p_{EK} is the product's purchase price. The participants' aim is to maximize the sum *G*. This should be achieved by intelligent pricing. Several independent runs will be carried out with a maximum of 20 participant agents each in order to determine the winner. The team who achieves the greatest profit on average wins. In case of a tie, the winner will be found by drawing lots.