Описание проделанной работы на момент 24.02.2016

Описание эксперимента

Для работы были выбраны реальные предпочтения пользователей между 10-ю машинами. Общее количество пользователей = 60. Пользователям предлагалось сделать выбор между двумя автомобилями (так для каждой пары автомобилей).

Задача: по заданным данным попытаться написать алгоритм, который будет предсказывать выбор пользователя.

Более подробное описание данных

Характеристика каждой из машин

Body type: Sedan (1), SUV (2)

Transmission: Manual (1), Automatic (2)

Engine capacity: 2.5L, 3.5L, 4.5L, 5.5L, 6.2L

Fuel consumed: Hybrid (1), Non-Hybrid (2)

Характеристика каждого из пользователей

Education: No response (0), High school (1), Bachelors (2), PhD (3)

Age: No response (0), Below 25 (1), Between 25-30 (2), Between 30-35 (3), Above 40 (4)

Gender: No response (0), Male (1), Female (2)

Region: No response (0), South (1), West (2), North East (3), Mid West (4)

Формат предпочтений

<id пользователя> <номер выбранной машины> <номер невыбранной машины> <КВ>

КВ – контрольный ли вопрос, значение равно либо 1(если контрольный), 0 иначе. В работе пока данная информация никак не учитывалась

Идея решения

Шаг 1. Данные (предпочтения) пользователей были поделены на три части learning questions, test question и check question размера 80%, 10%, 10% от общего размера входных данных о предпочтениях.

Learning questions – обучающая выборка нашего алгоритма.

Check questions – выборка, на которой как раз мы и будет проверять работу нашего алгоритма.

Test questions – поговорим об этих данных в ходе описания работы алгоритма.

Для решения поставленной задачи была использована идея алгоритма ближайших соседей KNN (k nearest neighbours) с некоторыми модификациями(эвристиками). Было реализована два метода выбора ближайших соседей Тривиальный и Основной.

Тривиальный метод поиска K ближайших соседей

Основная идея – представим каждое из наших предпочтений как вектор размерности 12:

1-4 координаты(характеристики пользователя):
{ <Education> <Age> <Gender> <Region> }

5-8 координаты(характеристики 1-ого автомобиля):
{ <Body type <Transmission> <Engine Capacity> <Fuel consumed> }

9-12 координаты(характеристики 2-ого автомобиля):
{ <Body type <Transmission> <Engine Capacity> <Fuel consumed> }

На выборке **learning questions** просто будем добавлять в наши известные данные очередной вектор.

Когда алгоритм получает очередной запрос для выбора предпочтения пользователя, на который он не знает ответа – **check questions** он формирует из запроса вектор (как описано выше). Затем заводит два значения forLeft, forRight, изначально перед обработкой запроса эти значения выставляются в 0. Для значения forLeft он ищет среди уже известных ответов (данные полученные из learning question) **K** ближайших соседей (евклидова метрика с некоторыми изменения в масштабе), после этого алгоритм пробегается по этим соседям и отнимает от значения forLeft расстояние до очередного соседа. После того как алгоритм узнал значение forLeft он делает почти аналогичные действия для forRight, только не на исходном вектора, а на векторе, полученным из исходного заменой 5-8 координат на 9-12 и наобарот (то есть теперь вторая машина считается лучше). После этого, алгоритм сравнивает веса forLeft и forRight. Если перевесил forLeft, то действительно алгоритм обнаружил, что вес (приоритет) первой машины над второй для данного пользователя выше, если же нет, то это означает, что алгоритм распознал вторую машину как более приоритетную для данного пользователя, что не является таковым. Так алгоритм проходит по всей выборке **check questions** и считает, сколько же

неправильных (не соответствующих действительности) предпочтений было получено.
Для поиска оптимального **K** как раз и используется **test questions.** Алгоритм перебирает все возможные значения параметра **K** (пользователей мало) и на каждом из значений запускает описанный выше алгоритм на **learning questions.** После этого запоминает самое оптимальное значение **K** – то, на котором было получено наименьшее количество неправильных предпочтений. И теперь **check questions** уже запускается на конкретном **K**.

Основной метод поиска K ближайших соседей

1-ый шаг: Для каждого опрошенного человека строим граф его предпочтений. Вершины графа каждого человека – предлагаемые машины. У *i-го* пользователя будет ребро *(a,b)* если в learning выборке присутствовало предпочтение *<i> <a> <b>*. Тогда для *i-го* человека машина *a* лучше, чем машина *b*, если в графе присутствует путь от вершины с номером *a* до вершины с номером *b*, причём обратного пути в графе нет.

2-ой шаг: Теперь нужно рассчитать близость между двумя участниками опроса.

(Близость по характеристикам)
Во-первых, у каждого пользователя есть собственные характеристики: возраст, регион, пол, образование. Поэтому первый из критериев близости двух участников – схожесть их характеристик, здесь было опять же использовано евклидово расстояние с некоторым масштабов + также для каждого критерия были введены веса, то есть некоторые целые числа w_1, w_2, w_3, w_4, показывающие на сколько важную роль играет та или иная характеристика.

(Answer Intersection method)
Во-вторых, для каждой пары пользователей можно узнать, на сколько похожи их предыдущие ответы. Для этого для каждого человека мы заведём множество его ответов – пары вида *<a, b>*, и при расчёте близости людей алгоритм принимает во внимание мощность объединения данных множеств, с некоторыми эвристиками: пусть мы рассматриваем пару людей A, B. Для расчёта их близости у алгоритма есть две переменных – ***same*** и ***dif***. Теперь, пусть мы знаем, что для ***A*** верно, что машина под номеров a предпочтительнее, чем ***b***. Тогда, если данное утверждение верно и для ***B***, то алгоритм увеличивает переменную близости ***same*** на некоторую величину. Если же для пользователя верно, что машина ***b*** лучше, чем ***a***, то увеличивается ***dif*** на некоторую величину. Тогда близостью двух пользователей по их предпочтениям назовём величину равную ***dif – same***. Чем данная величина меньше, тем ответы двух пользователей более похожи друг на друга. После этого для нахождения точной величины близости между двумя людьми алгоритм складывает две величины

близости, с некоторыми коэффициентам, полученные при расчёте близости по характеристикам и близости по ответам. Если данная величина хотя бы для одной пары отрицательна, то мы находим минимум расстояния и добиваем этот минимум до 0, также отнимая этот же минимум от всех остальных пар расстояния. Теперь все расстояния – это неотрицательные числа.

(KNN для машин)
Помимо этого, можно также рассчитать близость между автомобилями по их собственным характеристикам: тип автомобиля, трансмиссия, размер двигателя, потребление топлива, и при рассмотрении очередного запроса предпочтений брать данную информацию про близость автомобилей также во внимание (как именно будет описано ниже).

3-ий шаг: Теперь о работе самого алгоритма на конкретном запросе. Пусть у нас есть запрос <*User*> <*first*> <*second*> - то есть нам нужно определить для человека с номером *User*, какую машину он скорее всего выберет – *first* или *second*. Для User мы находим ближайших K соседей, по метрике близости, которая была описана выше. Пусть расстояние между K ближайшими людей до User по метрике – это d_1, d_2 … d_K. Алгоритм заводит две переменных **vote_for_first, vote_for_second**. Рассматривая очередного ближайшего соседа (пусть его номер = X) для *User*, алгоритм на основе **learning questions** может узнать, правда ли, что для X верно, что *a* лучше, чем *b* (напоминаю, что лучше в нашем плане означает существование пути от *a* до *b* и не существования от *b* до *a*). Если ответ положительный, то алгоритм прибавляет к величине ***vote_for_first*** величину равную Normal(d), где d расстояние между *User* и *X*, а Normal(d) – плотность нормального распределения с некоторым средним и дисперсией в точке = *d*. Если же ответ отрицательный, то мы меняем в запросе местами *a и b* и проделываем точно такие же действия, но теперь, при положительном ответе прибавляет величину Normal(d) уже к ***vote_for_second***. После этого используется идея KNN для самих машин. Пусть у нас в запросе присутствуют две машины a и b. Алгоритм находит у каждой из данных машин K' ближайших по характеристикам машин. Пусть для машины a – это a_1, a_2 … $a_{K'}$, для машины – b_1, b2 … $b_{K'}$. Затем строятся всевозможные пары вида (a, b_j) и (a_j, b), и алгоритм пытается для данного User ответ на запрос <User> <a> <b_j> и <User> <a_j> <b>. В зависимости от ответа на данные запросы, мы опять же меняется либо vote_for_first либо vote_for_second на Normal(d_j), где d_j – расстояние между a и b_j, или же между a_j и b.

Осталось только использовать ***test_questions*** для того, чтобы найти оптимальные K, K', {w_1, w_2, w_3, w_4}, {w'_1, w'_2, w'_3, w'_4}. Всё это делается при помощи функции ***Find_Best_Parametrs*** **–** оптимальные K, K' ищется перебором, так как кол-во опрошенных людей и кол-во машин достаточно мало. Для поиска оптимальных векторов весов алгоритм фиксирует некоторые два множества размера 4 и проверяет всевозможные перестановки данного множества. Для каждого очередного набора {K, K', {w1, w_2, w_3, w_4}, {w'_1, w'_2, w'_3, w'_4}} алгоритм

вызывается на **test_*questions*** и считает кол-во неправильных ответов. Оптимальный набор – набор на котором кол-во неправильных ответов на ***test_questions*** – минимально.

Результаты

Тривиальное решение

Противоречивые ответы	Кол-во неверно обработанных запросов	Общее кол-во вопросов	Доля неверных ответов
+	93	335	27.76
+	83	292	28.42
-	85	283	30.04
-	97	314	30.89
-	93	301	30.9
+	98	306	32.03
-	110	340	32.35
-	105	308	34.09
-	99	290	34.14
+	104	303	34.32
+	104	302	34.44
+	112	322	34.78
-	110	311	35.37
-	111	302	36.75

Основное решение

Явное вычисление на графе	Answers set Intersection Method	Противоречивые ответы	Кол-во неверно обработанных запросов	Общее кол-во вопросов	Доля неверных ответов
-	+	-	35	308	11.36
-	+	-	38	314	12.1
-	+	-	41	302	13.58
-	+	-	46	283	16.25
-	+	-	57	290	19.66
+	-	+	65	302	21.52
-	-	-	67	311	21.54
-	-	+	69	292	23.63
-	-	-	83	340	24.41
-	-	+	83	335	24.78
-	-	-	76	301	25.25
-	-	+	78	306	25.49
-	-	+	89	322	27.64
-	-	+	90	303	29.7