랭킹 SVM과 RDBMS의 밀결합 통합
(Tightly Coupled Integration of Ranking SVM and RDBMS)

송 재 환 [†] 오 진 오 [††] 양 은 석 [††] 유 환 조 [†††]
(Jaehwan Song) (Jinoh Oh) (Eunseok Yang) (Hwanjo Yu)

요 약 지난 십 년간 랭킹은 데이터 마이닝 분야의 활발한 연구분야였다. 그러나 랭킹은 다른 데이터 마이닝 기법들과 비슷하게 RDBMS와는 독립적으로 개발되었고, 그로 인해 기존에 널리 사용되고 있는 RDBMS들과의 연동성이 떨어진다는 단점이 존재하게 되었다. 다른 데이터 마이닝 기법들은 소결합이나 밀결합 접근법을 이용하여 RDBMS와 연동하기 위한 연구가 활발하게 진행되어 왔고, 그 결과 실제로 사용 가능한 응용시스템들이 나오게 되었다. 그러나 랭킹에서는 이와 같은 노력들이 잘 이루어지지 않고 있다. 본 논문에서는 랭킹 작업을 RDBMS에 연동하여 효율적으로 수행하기 위하여 MySQL에 Ranking SVM을 통합하는 작업을 진행하였다. 밀결합 접근법을 기반으로 하는 우리의 구현은 MySQL에 랭킹을 위한 새로운 SQL 명령어를 추가하였고 랭킹 작업의 효율성을 확인하기 위해서 소결합 접근법을 기반으로 하는 Ranking SVM과 성능을 비교 평가하여 훈련단계에서 10~40%, 예측단계에서 평균 60%의 성능향상을 확인할 수 있었다.

키워드 : 데이터 마이닝, 랭킹 SVM, 관계형 데이터베이스, 밀결합 통합, DMQL, 랭킹 SQL

Abstract Rank learning and processing have gained much attention in the IR and data mining communities for the last decade. While other data mining techniques such as classification and regression have been actively researched to interoperate with RDBMS by using the tightly coupled or loose coupling approaches, ranking has been researched independently without integrating into RDBMS. This paper proposes a tightly coupled integration of the Ranking SVM into MySQL in order to perform the rank learning task efficiently within the RDBMS. We implemented new SQL commands for learning ranking functions and predicting ranking scores. We evaluated our tightly coupled integration of Ranking SVM by comparing it to a loose coupling implementation. The experiment results show that our approach has a performance improvement of 10~40% in the training phase and 60% in the prediction phase.

Key words : Data mining, Ranking SVM, RDBMS, Tightly coupled integration, DMQL, Ranking SQL

· 본 연구는 2008년 정부의 재원으로 두뇌한국 21 사업과 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2008-331-D00528)
· 이 논문은 제35회 추계학술대회에서 '밀결합 접근법 기반의 Ranking SVM과 RDBMS의 통합'의 제목으로 발표된 논문을 확장한 것임

† 정 회 원 : (주)LG CNS 기술서비스부문
jhsong@lgcns.com
†† 비 회 원 : 포항공과대학교 컴퓨터공학과
kurin@postech.ac.kr
ighesy@postech.ac.kr
††† 정 회 원 : 포항공과대학교 컴퓨터공학과 교수
hwanjoyu@postech.ac.kr

논문접수 : 2008년 12월 19일
심사완료 : 2009년 5월 12일

정보과학회논문지: 데이타베이스 제36권 제4호(2009.8)

1. 서 론

지난 몇 십 년간 데이터 마이닝 분야는 연관규칙(association rule) 마이닝, 분류와 예측(classification and prediction), 군집화(clustering), 그리고 텍스트와 웹 마이닝 등의 연구주제들을 중심으로 데이터를 분석하고 데이터로부터 유용한 정보들을 추출하는 기법들이 제안되고 연구되어 왔다. 이러한 연구주제들 중 랭킹은 지난 몇 년간 가장 활발하게 연구되어 왔던 분야이다. 랭킹을 주제로 하여 Ranking SVM[1], Rank Boost[2], Adarank[3], 그리고 Rank Net[4]과 같은 알고리즘들이 개발되었으며, 그 효율성과 정확성 역시 실제 응용시스템에서 사용되어 사용자에게 효용을 줄 정도의 수준에 이르렀다.

하지만 지난 몇 년간의 연구들은 기존의 데이터베이스 관리 시스템으로 이용하는 RDBMS와 연동하기 힘든 단점을 가지고 있다. 이것은 그 동안의 연구가 기계학습(Machine Learning)이나 정보 검색(Information Retrieval) 등의 분야에서 사용되는 알고리즘들을 기반으로 하여 진행되었기 때문이다. 이로 인해 랭킹 알고리즘들은 기존에 존재하는 RDBMS와는 전혀 별개로 그 개발이 이루어졌으며 그 결과 기존의 MySQL, Oracle, MS-SQL 등과 같은 RDBMS들과는 전혀 연동이 되지 않고 있다[5]. 하지만 실생활에서 사용되는 거의 모든 데이터베이스는 RDBMS를 이용한다는 점에서 랭킹의 적용에 있어서 한계점이 분명히 존재하게 된다. 데이터 마이닝 기법들을 RDBMS에 연동시키려는 노력들은 Rakesh Agrawal이 이러한 문제에 대한 제안[5]을 한 이후 꾸준히 계속되어 왔다. 그리고 이러한 노력들이 결실을 맺어 DMQL[6], MSQL[7]과 같은 데이터 마이닝 기법을 지원하는 SQL 언어들이 만들어지고, 실제로 Oracle이나 OLE-DB를 기반으로 하여 데이터 마이닝 기법을 구현하는데 성공하였다[8]. 하지만 랭킹은 주제의 중요성에도 불구하고 아직까지 이러한 연동시스템의 설계와 구현에 대한 연구가 활발히 진행되지 않고 있다.

본 논문에서는 위와 같은 문제에 집중하여 기존에 제안되었던 랭킹 알고리즘 중 그 성능이 우수하다고 알려진 Ranking SVM[1]과 오픈 소스 데이터베이스 엔진인 MySQL[9]을 통합하여 MySQL 내부에서 랭킹 작업을 수행할 수 있도록 새로운 랭킹 SQL 명령어를 구현하였다. 밀결합 방식에 기반을 둔 우리의 구현 시스템은 소결합 방식 시스템과의 성능 비교에서 훈련 단계에서 10~40%, 예측 단계에서 평균 60%의 질의처리 시간 향상을 확인할 수 있었다.

논문의 나머지 부분은 다음과 같이 구성되어 있다. 2장에서는 데이터 마이닝 기법을 RDBMS에 연동하는 접근법들과 Ranking SVM에 대해 소개하고, 3장에서는 밀결합 접근법 기반의 Ranking SVM과 RDBMS의 통합을 위한 시스템 설계와 구현에 대해 자세하게 설명한다. 4장에서는 실험을 통해서 우리가 구현한 시스템과 기존 시스템의 성능을 비교 평가해보고 마지막으로 5장에서는 결론과 함께 향후 연구되어야 할 과제에 대해서 다루게 된다.

2. 배경 지식

2.1 통합화의 결합(Coupling) 정도

Rakesh Agrawal은 새로 개발되는 데이터 마이닝 시스템들이 기존의 RDBMS와의 연결관계가 약함을 지적하고 이러한 데이터 마이닝 시스템들을 RDBMS에 좀 더 밀접하게 연결시킬 것을 제안하였다[5]. 그는 데이터 마이닝 시스템을 RDBMS에 통합시키는 방식으로 소결합 통합(Loosely Coupled Integration)과 밀결합 통합(Tightly Coupled Integration) 접근법을 제안하였으며 각 접근법의 장단점을 밝혔다.

일반적으로, 소결합(Loose Coupling)은 데이터 마이닝 시스템이 데이터베이스(DB) 혹은 데이터웨어하우스(DW) 시스템의 일부 기능을 사용하는 것을 의미한다. 이 방식은 DB나 DW시스템이 관리하는 데이터 저장소에서 질의처리, 인덱싱 그리고 다른 시스템 기능을 사용하여 데이터의 임의 부분을 추출하고, 데이터 마이닝을 수행한 후에 마이닝 결과를 파일이나 DB 혹은 DW에 저장한다. 이 과정에서 데이터 마이닝 시스템은 DB 또는 DW 시스템이 제공하는 데이터 구조와 질의 최적화 방법을 활용하지 못하므로, 대규모 데이터 집합에서 높은 확장성과 좋은 성능을 얻기는 매우 어려운 단점이 있다[10].

밀결합(Tightly Coupled)은 데이터 마이닝 시스템이 자연스럽게 DB/DW 시스템 안에 통합되는 것을 의미한다. 데이터 마이닝 질의와 기능들은 마이닝 DB/DW의 질의 분석, 데이터 구조, 인덱싱 방식 및 질의 처리 기법들을 기반으로 최적화된다. 이 방식은 데이터 마이닝 기능의 효율적인 구현, 높은 시스템 성능, 통합된 정보처리 환경 구현을 용이하게 해주므로 가장 바람직하다. 그러나 RDBMS의 소스코드를 분석하고 수정해야 하기 때문에 구현하기 매우 까다로운 단점이 존재한다[10].

2.2 Ranking SVM

SVM(Support Vector Machine)은 훈련(Training) 데이터를 비선형 매핑을 통해 고차원으로 변환하고 이 새로운 차원에서 최적으로 분리하는 선형분리 초평면(hyperplane)을 찾는다. SVM은 훈련 시간이 매우 느림에도 불구하고 복잡한 비선형 의사결정 영역을 모형화할 수 있기 때문에 매우 정확하여 분류(classification)

```
statement::=rank_svm_learn | rank_prediction
rank_svm_learn::="RANK_SVM_LEARN" svm_model_table train_table svm_parameter_list
rank_prediction::="RANK_PREDICTION" svm_model_table test_table prediction_table
svm_parameter_list::="(" "LINEAR" "," cval ")" |"(" "RBF" "," cval "," gval ")"
svm_model_table::=TABLE IDENTIFIER
train_table::=TABLE IDENTIFIER
test_table::=TABLE IDENTIFIER
prediction_table::=TABLE IDENTIFIER
cval::=NUM
gval::=NUM
```

그림 1 랭킹 SQL 명령어 BNF

에서 널리 사용되고 있다. Ranking SVM[1]은 Classifying SVM[11]을 Ranking에 적합하도록 목적 함수(objective function)를 데이터 쌍 사이의 거리를 기반으로 하도록 수정하여 다음과 같은 목적 함수를 작성하여 문제를 해결한다.

데이터 쌍간의 거리를 최소화하도록 결정되는 w를 이용하여 각 데이터의 점수를 구하여 점수를 기반으로 하는 랭킹 기법을 사용한다. 즉, 모델 훈련에 사용되는 데이터를 기반으로 전체의 데이터에 점수를 매길 수 있는 선호도 함수를 만들어 내고, 이 함수를 기반으로 각 데이터의 점수를 계산하여 랭킹 작업을 진행한다. 그러므로 일반적으로 Ranking SVM은 모델 훈련과 예측(Prediction) 두 단계를 거쳐 진행된다.

본 논문에서는 기존에 구현되어 있는 Ranking SVM을 MySQL에 밀결합 접근법을 기반으로 통합하였다. 통합하기 위한 Ranking SVM 코드로 Joachims 그룹이 구현하여 배포하는 SVM^{light}를 사용하였고[12], 이후의 Ranking SVM은 SVM^{light}를 이용한 것을 의미한다.

3. 통합화 구현

3.1 개 요

우리가 구현한 밀결합 방식의 통합시스템에서는 기존의 RDBMS에 랭킹을 위한 새로운 SQL 명령어를 추가하였고, 이 명령어를 사용하여 모델 훈련과 예측의 정상적인 랭킹 작업이 원활하게 이루어지도록 구현하였다.

데이터 마이닝 기법들의 일반적인 특징 중의 하나는 기존의 RDBMS에 있는 오퍼레이션들과는 달리 훈련을 통하여 모델이라는 중간 산물이 나오는 것이다. 데이터 마이닝 시스템을 RDBMS에 통합하는 과정에서 가장 먼저 부딪히는 문제는 "모델을 어떻게 구현 및 관리하는가"이다. 이러한 문제로 인하여 OLE-DB을 비롯한 다른 데이터 마이닝 기법을 통합한 RDBMS에서는 모델이라는 클래스를 선언하게 된다[8,13,14]. 본 논문의 목표가 범용적인 마이닝 기법을 수용할 수 있는 RDBMS의 프레임워크를 만드는 것이라기 보다는 Ranking SVM이라는 데이터 마이닝 기법을 RDBMS에 통합하는 것이기 때문에 다른 데이터 마이닝 기법에서 생성되는 모델에 대해서는 고려하지 않았다. 이러한 경우에, 우리는 Ranking SVM에서 생성되는 모델 정보를 관리하기 위해서 기존의 RDBMS 테이블을 사용함으로써 위 문제를 해결하였다.

두 번째로 부딪히게 되는 문제는 "SQL 명령어를 어떤 관점으로 설계할 것이냐"라는 문제이다. 이 문제를 위하여 기존의 SQL 문법을 확장하는 방식으로 데이터 마이닝을 위한 SQL 문법을 정하는 연구들이 이루어져 왔다[6,7]. 하지만 이러한 SQL 문법들에는 사용자가 데이터 마이닝에 대한 자세한 정보를 알고 있어야만 사용할 수 있다는 단점이 존재한다. 우리는 이 문제를 해결하기 위하여 사용자 친화적인 새로운 랭킹 SQL 명령어를 구현하였다.

3.2 랭킹을 위한 SQL 명령어

Ranking SVM은 두 단계를 통하여 작업을 수행한다. 첫 번째는 모델을 만들어내기 위한 훈련단계이고 두 번째는 모델을 이용하여 다른 데이터들의 랭킹을 정하는 예측단계이다. 우리는 모델 훈련과 예측을 위해서 RANK_SVM_LEARN과 RANK_PREDICTION이라는 새로운 랭킹 SQL 명령어를 만들었다. 그림 1은 이 명령어들의 BNF(Backus-Naur Form)로 표기한 것이다.

위와 같은 랭킹 SQL 명령어가 적용될 수 있는 예로써, 고객이 온라인 부동산중개 회사로부터 자신이 원하는 집을 구매하고자 할 때, 그 온라인 부동산 중개 회사에서는 판매를 위해 매물로 나온 집들의 정보에서 고객이 선호하는 집에 가까운 집의 순서로 순위를 매겨서 제공하는 경우를 생각해 보자.

이 예의 경우에, 우리가 구현한 랭킹 SQL 명령어를 적용하기 위해서 우리는 커널 타입을 LINEAR 로 하고 커널 파라메터 cval 값을 0으로 하여 다음과 같이 명령어를 실행할 수 있다.

```
SQL> RANK_SVM_LEARN svmmodel prefer_house
                    (LINEAR, 0);
```

우리는 이 모델 훈련 명령어를 MySQL 내에서 다음과 같은 과정으로 처리되도록 구현하였다.

① 먼저, 훈련될 데이터를 *prefer_house* 테이블 에서 가져오는데, 이때 우리는 기존의 SELECT와 같은 SQL 명령어를 이용하지 않고 테이블에 직접 접근하여 데이터를 가져온다. 이렇게 가져온 데이터는 Ranking SVM에서 사용하는 데이터 구조로 변환한다. 우리는 이 과정을 table_to_doc() 함수로 구현하였다. ② 기존의 Ranking SVM에 구현되어 있는 함수인 *svm_learn_ranking()* 함수를 이용하여 모델 훈련을 수행한다. ③ 테이블 생성 코드를 이용하여 훈련된 모델을 저장할 *svmmodel* 테이블을 생성한다. ④ 훈련을 통해 생성된 모델을 *svmmodel* 테이블에 저장하기 위해서 테이블 스키마 구조로 변환시켜야 하는데 우리는 이 과정을 model_to_table() 함수로 구현하였다. 이 때 트랜잭션 동시발생(concurrency) 시에 문제가 없도록 하기 위해서 기존의 mysql_insert() 함수를 이용하였다.

앞의 예에서 만들어진 모델 정보를 이용하여 *candidate_house*의 목록에 순위를 위한 점수를 구해서 *score_house*에 저장하기 위하여 다음과 같이 명령어를 실행할 수 있다.

SQL> RANK_PREDICTION svmmodel
candidate_house score_house ;

우리는 이 예측 명령어를 MySQL 내에서 다음과 같은 과정으로 처리되도록 구현하였다.

① 먼저, *svmmodel* 테이블에 직접 접근해서 훈련된 모델 정보를 가져오고 Ranking SVM에서 사용될 수 있는 모델 구조(models)로 변경한다. 우리는 이 과정을 table_to_model() 함수로 구현하였다. ② 테스트 할 데이터를 *candidate_house* 테이블에 직접 접근해서 가져와서 Ranking SVM에서 사용하는 데이터 구조(doc)로 변환하고 이 doc와 ①에서 구한 models를 기존의 Ranking SVM에 구현되어 있는 classify_example_linear() 함수의 인수들(arguments)로 사용하여 예측을 수행한다. ③ 테이블 생성 코드를 이용하여 예측된 점수를 저장할 *score_house* 테이블을 생성한다. ④ 예측된 점수를 트랜잭션 동시발생(concurrency) 시에 문제가 없도록 하기 위해서 기존의 mysql_insert() 함수를 이용하여 *score_house* 테이블에 저장한다.

우리가 구현한 랭킹 SQL 명령어는 MySQL 테이블에 직접 접근하여 데이터를 가져와서 랭킹 작업을 수행하므로 Ranking SVM에서 사용하기 위해 DB에서 데이터 추출 후 파일형태로 변환하는 과정이 필요한 기존의 소결합 접근 방식과 비교하여 매우 효율적이라고 볼 수 있다. 또한 데이터 마이닝을 위해서 MySQL 환경과 Ranking SVM 환경을 모두 구성할 필요 없이 통합된 환경만 구성하므로 정보처리 환경 구성 측면에서도 용이하다고 볼 수 있다.

3.3 테이블 스키마

랭킹 SQL 명령어를 사용하기 위해 필요한 데이터 테이블과 랭킹 SQL 명령어 수행을 통해 얻어진 모델 및 예측 테이블의 스키마는 그림 2와 같다.

훈련용 데이터와 테스트용 데이터가 저장되는 데이터 테이블은 질의 식별자와 문서 식별자 해당하는 queryid, docid와 데이터를 벡터화하여 저장해 놓은 fvector, 그리고 데이터의 클래스를 의미하는 clabel 필드를 가지고 있다.

훈련될 모델이 저장될 svm_model_table은 Ranking SVM에서 사용되는 kerneltype, 커널의 각종 파라메터 값을 저장하기 위한 dval, gval, coef_lin, coef_const, 그리고 Ranking SVM의 훈련 결과로 생성되는 모델을 나타내는데 사용되는 서포트 벡터 필드들인 alphaval, svector로 구성되어 있다.

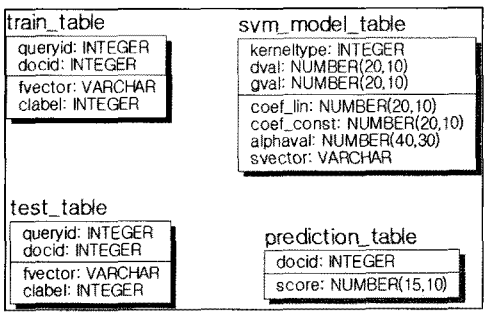

그림 2 테이블 스키마

예측 정보가 저장될 prediction_table은 테스트할 문서 식별자인 docid와 예측 결과로 얻어지는 점수에 해당하는 score 필드를 가지고 있다.

3.4 랭킹 SQL 명령어 실행 결과

그림 3은 우리가 구현한 랭킹 SQL 명령어를 실행한 결과이다. RANK_SVM_LEARN 명령어를 실행하여 훈련 후 훈련된 모델 정보가 저장될 model100 테이블이 생성되었고 모델 정보가 정상적으로 저장되었음을 보여준다. 또한 RANK_PREDICTION 명령어를 실행하여 예측 후 예측 정보가 저장될 prediction100 테이블이 생성되었고 예측 정보가 정상적으로 저장되었음을 보여준다.

그림 4는 소결합 방식의 Raking SVM(좌)과 우리가 구현한 밀결합 방식 통합 시스템(우)의 예측 단계에서 생성된 랭킹 점수를 나타낸다. 밀결합 방식으로 생성된 랭킹 점수를 소결합 방식의 랭킹 점수 자릿수에 반올림하여 맞추면 정확히 일치함을 확인할 수 있다. 이로써, 우리가 구현한 통합 시스템의 랭킹 정확도(accuracy)는 기존 Ranking SVM과 동일하게 유지함을 알 수 있다.

```
mysql> rank_svm_learn model100 train100 [LINEAR, 0];
Query OK, 1 row affected (0.44 sec)

mysql> desc model100;
+------------+----------------+------+-----+---------+-------+
| Field      | Type           | Null | Key | Default | Extra |
+------------+----------------+------+-----+---------+-------+
| kerneltype | int(1)         | YES  |     | NULL    |       |
| dval       | double(20,10)  | YES  |     | NULL    |       |
| qval       | double(20,10)  | YES  |     | NULL    |       |
| coef_lin   | double(20,10)  | YES  |     | NULL    |       |
| coef_const | double(20,10)  | YES  |     | NULL    |       |
| alphaval   | double(40,30)  | YES  |     | NULL    |       |
| svector    | varchar(60000) | YES  |     | NULL    |       |
+------------+----------------+------+-----+---------+-------+
7 rows in set (0.00 sec)

mysql> select count(*) from model100;
+----------+
| count(*) |
+----------+
|       99 |
+----------+
1 row in set (0.00 sec)

mysql> rank_prediction model100 test100 prediction100;
Query OK, 0 rows affected (0.03 sec)

mysql> desc prediction100;
+-------+---------------+------+-----+---------+-------+
| Field | Type          | Null | Key | Default | Extra |
+-------+---------------+------+-----+---------+-------+
| docid | int(10)       | YES  |     | NULL    |       |
| score | double(15,10) | YES  |     | NULL    |       |
+-------+---------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> select count(*) from prediction100;
+----------+
| count(*) |
+----------+
|      100 |
+----------+
1 row in set (0.00 sec)
```

그림 3 랭킹 SQL 명령어 실행 결과

docid:90386069	-0.6209881	90386069	-0.6209881129
docid:90278894	1.3788581	90278894	1.3788580953
docid:90286638	-0.6201809	90286638	-0.6201800930
docid:90379605	-0.6205758	90379605	-0.6205758696
docid:91016083	-0.9636698	91016083	-0.9636698595
docid:91016084	-0.6206225	91016084	-0.6206225920
docid:91016129	0.3794182	91016129	0.3794182223
docid:91022922	-0.6201750	91022922	-0.6201750940
docid:91032288	0.3794164	91032288	0.3794164571
docid:91035850	-0.6201156	91035850	-0.6201156872
docid:91055012	-0.6203633	91055012	-0.6203633338
docid:91057187	-0.6201406	91057187	-0.6201406322
docid:91073036	-0.6200781	91073036	-0.6200781537
docid:91076147	-0.6200437	91076147	-0.6200437609
docid:91103230	-0.6202334	91103230	-0.6202334495
docid:91110206	0.3794511	91110206	0.3794511559
docid:91120637	1.3799463	91120637	1.3799463350
docid:91123318	0.3793328	91123318	0.3793328539

그림 4 소결합 방식의 점수(좌), 밀결합 방식의 점수(우)

4. 실 험

실험에서는 우리가 구현한 밀결합 방식과 소결합 방식의 성능을 비교평가 하였다. 우리는 소결합 방식 실험을 위해서 기존의 Ranking SVM에 DB 테이블에서 훈련 데이터를 추출해서 파일구조로 생성하는 부분과 모델 훈련 후 파일구조로 생성된 모델 정보를 테이블구조에 저장하는 부분을 연계하는 스크립트를 작성하였다. 예측단계에서도 모델 정보와 테스트 데이터를 DB 테이블에서 조회하여 파일구조로 생성하는 부분과 예측 후 파일구조로 저장된 예측 결과를 테이블구조에 저장하는 부분을 연계하는 스크립트를 작성하였다.

각 실험은 훈련과 예측 작업 각각에 대하여 밀결합 방식과 소결합 방식의 영향을 알아보고자 훈련 부분과 예측 부분을 분리하여 진행하였고, 실험할 때마다 측정 시간이 다소 변동될 수 있는 점을 감안하여 100번 반복하여 진행하였다. 또한 데이터 개수가 증가함에 따라 질의처리 시간에 어떤 영향을 미치는지 확인하기 위하여 20~140개의 개수를 가지는 데이터 셋을 생성하여 데이터 셋 개수를 변화시켜 가면서 실험하였다.

4.1 실험 환경

실험에서는 합성(synthetic) 데이터 셋을 사용하였다. 합성 데이터 셋은 정상 분포를 따르는 랜덤함수를 이용하여 100개의 특성(feature)을 만들어 내고, 임의의 랜덤 스코어 함수를 만들어내 이 스코어 함수에 각 데이터 들을 적용한 결과값을 바탕으로 전체 데이터 셋을 0~4의 5개의 부분(partial) 랭킹으로 나누었다. 이러한 합성 데이터 셋을 여러 개의 데이터 개수를 가지도록 생성하여 실험에 사용하였다.

실험은 Intel Quadcore 2개와 RAM 40G, HDD 4.5TB 사양을 갖춘 DELL 서버의 Linux Kernel 2.6.18, MySQL 5.0.51a에서 수행하였다.

4.2 실험 결과

그림 5는 데이터 셋 개수의 변화에 따른 훈련단계 질의처리 시간의 변화 추이를 보여주고 있다. 소결합 방식과 밀결합 방식 중 밀결합 방식이 더 짧은 질의처리 시간을 가지는 것을 볼 수 있다. 하지만 훈련의 경우 Ranking SVM의 훈련 복잡도(complexity)는 데이터 셋 개수에 따라서 평방(quadratic) 시간으로 증가함을 확인할 수 있다. 이처럼 제곱에 비례하여 증가하는 훈련 시간 때문에 밀결합의 장점을 쉽게 알아 볼 수 없다.

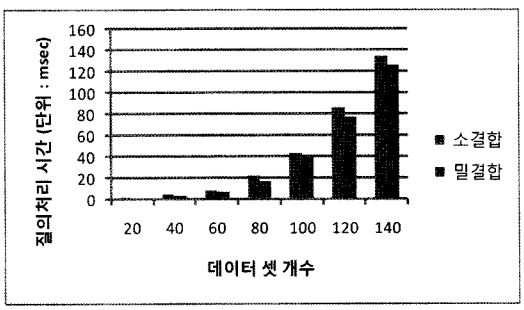

그림 5 훈련단계 질의처리 시간

그런데, 훈련단계의 정규화된 질의처리 시간을 보여주는 그림 6을 살펴보면, 밀결합 방식이 소결합 방식에 비해서 질의처리 시간에서의 성능 향상이 얼마나 있었는지 명확히 알 수 있다. 데이터 셋의 개수가 20일 경우 40%가 넘는 질의처리 시간이 단축되었고, 나머지 경우에도

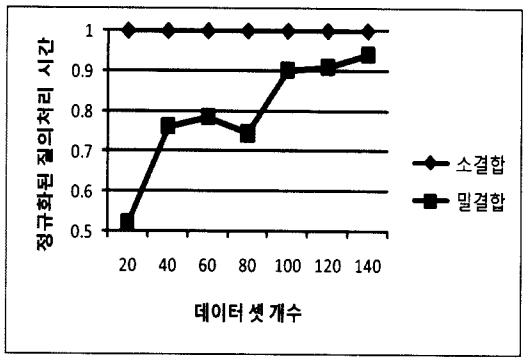

그림 6 훈련단계 정규화된 질의처리 시간

10~20%의 질의처리 시간이 단축되었음을 알 수 있다.

예측단계의 경우는 우리가 구현한 밀결합 방식이 소결합 방식에 비해서 월등한 성능 향상이 있음을 확인할 수 있다. 예측단계의 질의처리 시간이 데이터 셋 개수에 따라서 선형적으로 비례하기 때문에 밀결합 방식의 장점이 그림 7을 통해서도 명확하게 드러난다. 예측단계의 정규화된 질의처리 시간을 나타내고 있는 그림 8을 보면 거의 평균적으로 60% 가량 질의처리 시간이 단축되었음을 알 수 있다.

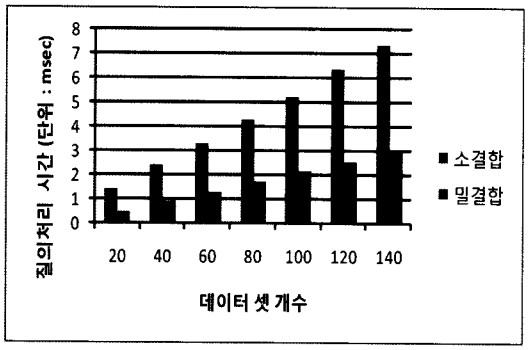

그림 7 예측단계 질의처리 시간

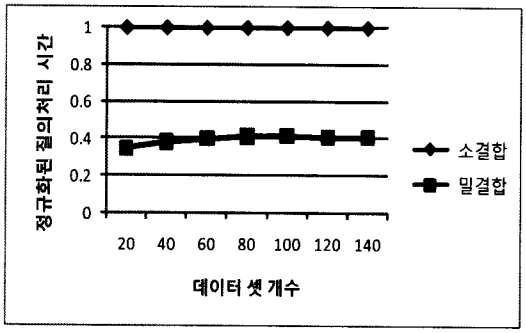

그림 8 예측단계 정규화된 질의처리 시간

5. 결론 및 향후 연구과제

본 논문에서는 데이터 마이닝의 활발한 연구분야인 랭킹이 연구 주제로서의 중요성에도 불구하고 RDBMS와의 연동에 대한 연구가 미진함을 인식하고, 랭킹 알고리즘 중에 가장 효과적이라고 알려진 Ranking SVM을 MySQL에 밀결합 접근법을 기반으로 통합 하였고 기존의 소결합 접근법으로 구현되었던 시스템과 비교하여 월등히 빠른 성능을 보여줌을 실험을 통해서 확인하였다. 이는 앞으로 기존의 RDBMS의 방대한 데이터들을 효율적으로 랭킹 작업에 활용할 수 있음을 의미한다.

본 논문에서는 합성 데이터 셋을 사용하여 랭킹 작업을 수행하였다. Ranking SVM은 수치(numerical) 데이터 벡터를 사용하여 훈련하므로 범주형(categorical) 속성(예. 직업, 색깔 등)이나 텍스트 데이터(예. 주소, 주석 등)들은 사전처리(preprocessing) 되어야 한다. 수 많은 데이터베이스에 존재하는 이러한 비수치적인 데이터들을 실 시간으로 사전처리 하는 것은 쉽지 않은 작업이다. 랭킹 SQL 명령어를 수행하는 동안에 SQL 내부에서 이러한 비수치적인 데이터들이 사전처리 되어 랭킹이 수행되도록 하는 작업을 향후 연구과제로 남겨둔다.

참 고 문 헌

[1] Ralf Herbrich, Thore Graepel, and Klaus Obermayer, "Large margin rank boundaries for ordinal regression," In Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, 2000.

[2] Yoav Freund, Raj Iyer, Robert E, Schapire, Yoram Singer, "An Efficient Boosting Algorithm For Combining Preference," *Journal of Machine Learning Research*, 2003.

[3] Jin Xu, Hang Li, "Adarank: A Boosting Algorithm for Information Retrieval," SIGIR, Annual ACM Conference on Research and Development in Information Retrieval, 2007.

[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, Greg Hullender, "Learning to Rank using Gradient Descent," *ACM International Conference Proceeding Series*, 2005.

[5] Rakesh Agrawal, Kyoseok Shim, "Developing Tightly-Coupled Data Mining Applications on a Relational Database System," *Proc. Knowledge Discovery and Data Mining*, 1996.

[6] Jiawei Han, Yongjian Fu, Wei Wang, Krzysztof Koperski, Osmar Zaiane, "DMQL: A data mining query language for relational databases," *Proc. SIGMOD*, 1996.

[7] Tomasz Imielinski, Aashu Virmani, "MSQL: A Query Language for Database Mining," Data Mining and Knowledge Discovery, 1999.

[8] Boriana L. Milenova, Joseph S. Yarmus, Marcos

M. Campos, "SVM in Oracle Database 10g: Removing the Barriers to Widespread Adoption of Support Vector Machines," VLDB 2005.
[9] http://www.mysql.com
[10] Jiawei Han, Micheline Kamber, "Data Mining: Concepts and Techniques," Second Edition, Morgan Kaufmann, 2006.
[11] Vapnik, "The Nature of Statistical Learning Theory," Springer, 1995.
[12] http://svmlight.joachim.org/
[13] Amir Netz, Surajit Chaudhuri, Usama Fayyad, Jeff Bernhardt, "Integrating Data Mining with SQL Databases: OLE DB for Data Mining," icde, p. 0379, 17th International Conference on Data Engineering (ICDE'01), 2001.
[14] Zhaohui Tang, Jamie Maclennan, Peter Pyungchul Kim, "Building data mining solutions with OLE DB for DM and XML for analysis," ACM SIGMOD Record, 2005.

송 재 환
1997년 전북대학교 물리학과 학사. 2009년 포항공과대학교 정보통신학과 석사 1997년~현재 LG CNS 기술서비스부문 관심분야는 Database Systems, Data Warehouse, Data Mining, Storage Systems

오 진 오
2008년 포항공과대학교 컴퓨터공학과 학사. 2008년~현재 포항공과대학교 컴퓨터공학과 석·박사 통합과정. 관심분야는 Data Mining, Information Retrieval, Ranking

양 은 석
2007년 포항공과대학교 컴퓨터공학과 학사. 2008년~현재 포항공과대학교 컴퓨터공학과 석사과정. 관심분야는 Data Mining, Clustering, Text Mining

유 환 조
1997년 중앙대학교 컴퓨터공학과 학사 2004년 University of Illinois at Urbana-Champaign 컴퓨터공학 박사. 2004년~2008년 University of Iowa 조교수 2008년~현재 포항공과대학교 컴퓨터공학과 조교수. 관심분야는 Data Mining, Information Retrieval, Databases, Machine Learning