**참고문헌**

* Hyperparameter Search Space Pruning – A New Component for Sequential Model-Based Hyperparameter Optimization

**틀 관련 논문 참고(form)**

* High-dimensional Bayesian Optimization with Multi-task Learning for RocksDB
* Ottertune
* Rsottertune 논문
* 주연님 논문
* 한국 정보처리 학회 논문1
* 한국 정보 처리 학회 논문 2

**제목**

파라미터 분류를 통한 redis 파라미터 튜닝 연구

1. **요약**

파라미터 튜닝이란 데이터 베이스가 제공하는 파라미터 값을 수정하여 최적의 성능을 낼 수 있게 조율하는 과정이다. 응용 프로그램을 변경하거나 데이터 베이스 모델을 바꾸는 작업과 보다 데이터 베이스 성능을 향상시키기 위해서 비용이 적은 장점이 있다. 하지만 파라미터 개수가 데이터 베이스 종류에 따라 수십개에서 수백개로 다양하며 각 기능이 모두 다르기 때문에 최적의 조합을 찾기란 어려운 일이다. 선행 연구와 같이 BO를 통해 값을 추출할 수 있지만 파라미터 개수와 비례하여 차원이 늘어나기 때문에 차원이 커지는 문제가 발생한다. 이는 탐색공간이 커져 연산량이 많아지고 시간 역시 오래 걸리는 상황이 발생한다. 본 논문에서는 통계적으로 파라미터를 분류하여 탐색 공간을 줄인 다음 BO를 진행하는 방식을 제안한다.

파라미터 값을 랜덤하게 할당하여 벤치마킹을 한 결과값을 군집화 한다. 그리고 각 군집 별로 파라미터들과의 연관성을 조사해 높은 상관관계를 가진 파라미터를 매칭시킨다.

각 모델별로 비교 결과 디폴트 조합과 전체를 대상으로 한 BO보다 높은 성능이 나타남을 확인하였다.

탐색 공간에 있어서 차원의 개수가 파라미터 개수와 같기 때문에

데이터 베이스는 다양한 파라미터를 제공한다. 파라미터 개수는 데이터베이스 종류에 따라 수십개에서 수백개로 다양하며 각각의 기능은 모두 다르다.

**1\_1. 서론 – 한계점, 본 논문의 제안 내용**

레디스를 포함한 대부분의 데이터 베이스에는 다양한 파라미터가 존재하고 조합에 따라 성능이 달라질 수 있다. 사용자의 경우 목적에 맞는 최적의 파라미터 조합을 찾아야 한다. 하지만 파라미터 값들 중 성능을 최고로 끌어올리 위한 조합을 찾는 것은 어려운 일이다.

개수가 많고 기능이 모두 다르기 때문에 일일이 모든 경우를 고려하는 것이 힘들기 때문이다. 그리고 특정 파라미터 끼리 연관된 경우 종속성까지 고려해줘야 한다.

일부 선행 연구에서는 BO(Bayesian Optimization)을 이용하여 최적의 파라미터 값을 추출한다.

단위 시간당 처리량을 출력으로, 모든 파라미터를 입력으로 하여 최대의 처리량을 출력으로 가지는 파라미터 조합을 추정한다. 하지만 데이터 베이스에 따라 많게는 수백개의 파라미터가 존재하기 때문에 차원이 커지는 문제가 발생한다.

하지만 이와 같은 경우 모든 파라미터별로 연관성을 찾기 때문에 탐색공간이 커진다는 단점이 있다. 그리고 차원이 파라미터 개수만큼 늘어나 차원의 저주 문제에 마딱드릴 수 있게 된다.

본 논문에서는

통계기법들을 통해 파라미터를 분류하여 탐색 공간을 줄인 후 BO를 진행하는 방법을 제안한다.

**2. 관련 연구**

**2.1 파라미터 튜닝(출처 명시!)**

튜닝이란 응용 프로그램, 데이터 베이스 시스템, 그리고 운영체제의 조율을 통하여 시스템의 성능 향상하기 위한 조율 과정으로 볼 수 있다. 데이터 베이스 시스템 튜닝은 정보 시스템의 성능을 향상 시키기 위하여 수행하는 튜닝의 한 종류이며 파라미터 튜닝이 이에 해당된다.

파라미터 튜닝이란 데이터 베이스가 제공하는 파라미터 값을 변경시켜 성능을 향상시키는 방법이다. 파라미터 조합에 따라 성능이 바뀌며 모델을 새로 생성하거나 응용 프로그램 자체를 교체하는 작업 보다 비교적 적은 비용으로 데이터 베이스의 성능을 최적화 시킬 수 있다.

하지만 파라미터 종류가 다양하고 기능이 모두 다르기 때문에 직접 값을 설정하는 것은 어려운 일이다. 이와 같은 어려움을 해결하기 위해 기계학습을 통해 선형, 비선형적으로 값을 추천해주는 일부 선행 연구가 존재한다. 본 논문에서는 선행 연구와 같이 파라미터 개수가 수십개인 Redis를 대상으로 값을 기계학습을 통해 값을 추천하는 방식을 제안한다.

**2.2 파라미터 선별을 통한 차원 축소**

기계학습 모델을 사용하여 파라미터 값을 추출할 때 파라미터 종류가 많아지면 입력값에 대한 차원이 커지는 문제점이 있다. 그 결과 과적합이 발생할 수 있으며 이에 따른 필요한 학습 데이터 양이 늘어난다. 그리고 컴퓨터의 연산량이 많아지고 시간이 오래걸리는 문제가 추가로 발생할 수 있다. [1] 논문에서는 RocksDB를 대상으로 10개의 파라미터를 선별하여 차원을 축소하는 방법을 제안한다. 내부(internal) metrics와 연구자의 경험을 토대로 rocks db의 구조를 3부분으로 나눈 후 가장 영향력 높은 10개의 파라미터를 각 부분에 매칭시켜 BO를 통해 값을 추출한다.

[1]의 연구의 경우 RocksDB의 내부 구조를 나누는 과정과 파라미터를 선별하고 각 구조에 매칭시키는 과정에서 연구자의 경험이 포함되기 때문에 객관적이지 못하다고 볼 수 있다. 본 논문에서는 정해진 알고리즘대로 통계적인 방법을 사용하여 파라미터를 분류하는 방법을 제시한다.

**2.4 Bayesian Optimiazation**

목적함수(objective-function, blackbox function)를 최적화하기 위한 기법 으로 Bayesian 이론 기반으로 사전 (prior) 데이터를 반영하는 방법이다. 현재까지 탐색된 데이터를 토대로 목적함수를 추정하는 surrogate model과 추정된 모델을 바탕으로 그 다음에 입력할 최적의 데이터를 추천하는 함수인 Acquisition function로 구성된다. Random search, grid search와 같이 하이파라미터 튜닝 기법 중 하나로 분류되며 머신러닝, 딥러닝 모델을 optimization 할 때 사용된다. 일부 선행연구에서는 데이터베이스를 대상으로 성능을 최적화하기 위한 목적으로 사용되기도 한다.

초기에 랜덤하게 값을 할당한 후 목적함수를 통해 결과값을 구한다. 그리고 이 값들을 반영하여 새로운 입력값을 생성한다. 또다시 목적함수를 통해 새로운 집합에 대해서 결과를 계산하여 집합에 추가한 후 새로운 집합을 계산한다.

**3.본격적인 제안 내용**

**3\_0. 요약**

본 논문에서는 파라미터를 분류하고 BO를 진행하기 위해 사용될 학습 데이터를 먼저 구축한다. 그리고 내부 metrics 를 대상으로 요인 분석과 kmeans clustering을 사용하여 군집화한다. 그 다음 파라미터별로 군집과의 연관성을 구해 각 군집에 매칭시켜 군집 별로 파라미터를 분류한다. 마지막으로 분류된 파라미터 별로 Bo를 순차적으로 진행하여 파라미터를 추출한다.

**3.1 학습 데이터 구축**튜닝할 파라미터들에 대해서 랜덤하게 값을 할당하여 Redis configuration 파일을 생성한다. 그리고 각 파일을 통해 Memtier-Benchmark를 실행하여 내부(internal) metrics와 외부(external) metric를 생성한다. 내부 metrics는 군집화 되어 파라미터를 매칭시키는 데에 사용되고 외부 metrics는 BO의 목적함수 내부 모델을 학습시킬 때 사용된다. 이때 목적함수의 모델은 랜덤하게 생성된 configurateion 파일들을 피처 데이터(Feature Data)로 외부(external) metrics를 타겟 데이터(Target Data)로 사용한다.

**3.2 내부(internal) metrics 클러스터링**

내부(internal) metrics을 대상으로 kmeans clustering을 진행한다. 그 전에 먼저 요인 분석을 통해 내부(internal) metrics간의 상관관계를 고려하여 잠재된 요인(Factor)를 추출한다. 그리고 그림과 같이 요인 분석 결과를 바탕으로 kmeans clutering을 실시하여 내부(internal) metrics를 분류한다. 이때 요인 수의 경우 1이상의 고유값(eigenvalue)을 갖는 요인으로 결정한다. 그리고 kmeans clustering에서 k 값은 엘보우 기법으로 k값이 늘어날때 오류율이 급속히 줄어들다가 거의 감소하지 않는 지점의 k 값을 설정한다.

**3.3 knob 매칭**

모든 파라미터 마다 군집과의 상관관계를 바탕으로 연관성이 가장 높은 군집에 매칭시킨다.

군집과의 연관성을 파악하기 위해 군집에 속한 내부 metrics와 파라미터간의 상관계수를 구한다. 그리고 절대값의 평균값을 계산하여 군집별로 연관성을 판단한다. 값이 가장 높은 군집에 파라미터를 매칭 시킨다.

**3.4 군집별 BO 진행**

군집별로 매칭된 파라미터를 대상으로 Baysian Optimization을 진행한다. Throughput 값을 최대로 갖을 수 있도록 파라미터 분류들에 대하여 순차적으로 진행한다. 이때 앞서 진행된 결과값들은 다음 진행 시 적용된다. 그림과 같이 디폴트 파라미터 값이 할당된 리스트에서 첫번째 분류에 대하여 먼저 BO를 진행하여 값을 추출한다. 이때 분류에 속하지 않은 나머지값들은 디폴트 값을 그대로 가진채 목적함수로 입력된다. 그리고 나머지 분류에 대해서 Bo를 진행하여 값을 추출한다. 이때 첫번째 분류에 속했던 파라미터들을 앞서 진행된 Bo를 통해 값이 update된 채로 목적함수로 입력된다.

**4. 실험 및 결과 분석**

**4.1 실험 환경**

본 실험은 분류된 파라미터 별로 BO를 진행한 경우와 파라미터 전체를 대상으로 Bo를 진행한 경우 그리고 default 값 조합 각 3가지 별로 단위 시간당 처리량을 구하여 비교하였다. 워크로드는 Write-Only와 Read-Write (1:1) 로 데이터를 적재만 하는 경우와, 데이터 적재와 조회 비율이 동일한 경우 두가지를 사용하였다. 그리고 Redis의 지속성 방법인 RDB, AOF 방식을 구분하여 진행하였다.

**4.2 결과 분석**

본 논문에서 제안된 분류별로 파라미터를 튜닝한 경우와 전체 파라미터 대상으로 튜닝한 경우 그리고 튜닝을 하지 않은 default 값을 가진 경우를 비교하는 실험을 진행하였다. Write-only, read-write(1:1) 워크로드 별로 RDB, AOF 방식을 구분하였으며 Decision Tree, Gradient Boosting, Lasso, LightGBM, Linear Regression, RandomForest, Ridge, XGBoost 8가지 회귀 모델(regressor model)을 통해 성능을 평가하였다.

그림 1과 2는 Read:Write(1:1) 워크로드를 통해서 RDB와 AOF 작동 방식을 구분하여. 실험한 결과이다. RDB 경우 모든 모델이 전체 파라미터를 대상으로 진행한 것보다

파라미터를 분류한 후 진행한 것이 더 높게 나타났다. 두 가지 모두 모든 모델에서default 결과 값보다 높은 결과값을 확인했다. 특히Lasso 모델에서 차이가 ~ 만큼 가장 크게 나타났다. AOF의 경우 DecisionTree과 GradientBoosting 모델을 제외하고 나머지 6가지 모델에서 분류된 파라미터를 대상으로 진행한 결과값이 더 크게 나타났다. Default 결과값을 넘는 경우는 분류된 파라미터의 경우가 4가지로 확인 되었다.

그림 3, 4는 Write-Only 워크로드를 통해서 RDB, AOF 방식 별로 구분하여 실험한 결과이다. RDB에서는 Lasso를 제외한 7가지 모델이 파라미터를 분류한 후의 결과가 높게 나타났다. 그리고 7가지 모델 모두 default 결과값 보다 높은 결과값을 가졌다. AOF에서는 Lasso와 RandomForest를 제외한 나머지 6가지 모델에서 분류 별로 진행한 결과값이 높게 나타났다. 그리고 모두 default 결과값을 넘었다.

**5. 결론**

전체 파라미터

분류별 파라미터

본 논문에서는 BO를 통해 파라미터 튜닝 작업 시 파라미터를 분류하여 탐색 공간을 줄이는 연구를 수행하였다. 파라미터 전체를 탐색 공간으로 BO를 진행하는 선행연구와는 달리 분류별로 BO를 진행하여 더 높은 처리량을 보이는 값을 추천하는 방식을 제안하였다. 연구 결과 대다수의 모델이 파라미터를 분류한 후 BO를 진행하여 전체 파라미터를 대상으로 진행한 것보다 더 높은 처리량을 보였다.