Reinforcement learning for intensional data management

Pierre Senellart

22 Feb. 2018, Télécom ParisTech, Data Science Seminar
Uncertain data is everywhere

Numerous sources of uncertain data:

- Measurement errors
- Data integration from contradicting sources
- Imprecise mappings between heterogeneous schemas
- Imprecise automatic processes (information extraction, natural language processing, etc.)
- Imperfect human judgment
- Lies, opinions, rumors
Uncertain data is everywhere

Numerous sources of uncertain data:

- Measurement errors
- Data integration from contradicting sources
- Imprecise mappings between heterogeneous schemas
- Imprecise automatic processes (information extraction, natural language processing, etc.)
- Imperfect human judgment
- Lies, opinions, rumors
Structured data is everywhere

Data is **structured**, not flat:

- Variety of **representation formats** of data in the wild:
  - relational tables
  - trees, semi-structured documents
  - graphs, e.g., social networks or semantic graphs
  - data streams
  - complex views aggregating individual information

- **Heterogeneous schemas**

- Additional **structural constraints**: keys, inclusion dependencies
Intensional data is everywhere

Lots of data sources can be seen as intensional: accessing all the data in the source (in extension) is impossible or very costly, but it is possible to access the data through views, with some access constraints, associated with some access cost.

- Indexes over regular data sources
- Deep Web sources: Web forms, Web services
- The Web or social networks as partial graphs that can be expanded by crawling
- Outcome of complex automated processes: information extraction, natural language analysis, machine learning, ontology matching
- Crowd data: (very) partial views of the world
- Logical consequences of facts, costly to compute
Interactions between uncertainty, structure, intensionality

- If the data has complex structure, uncertain models should represent possible worlds over these structures (e.g., probability distributions over graph completions of a known subgraph in Web crawling).

- If the data is intensional, we can use uncertainty to represent prior distributions about what may happen if we access the data. Sometimes good enough to reach a decision without having to make the access!

- If the data is a RDF graph accessed by semantic Web services, each intensional data access will not give a single data point, but a complex subgraph.
Intensional Data Management

- Jointly deal with Uncertainty, Structure, and the fact that access to data is limited and has a cost, to solve a user’s knowledge need
- Lazy evaluation whenever possible
- Evolving probabilistic, structured view of the current knowledge of the world
- Solve at each step the problem: What is the best access to do next given my current knowledge of the world and the knowledge need
- Knowledge acquisition plan (recursive, dynamic, adaptive) that minimizes access cost, and provides probabilistic guarantees
formulation

Knowledge need
Knowledge need

formulation

Structured source profiles

modeling

Knowledge need
Knowledge need

Knowledge acquisition plan

Uncertain access result

Intensional access

Knoledge update

Optimization

Modeling

Priors

Structured source profiles

Modeling

Current knowledge of the world

Formulation
Knowledge need

Optimization

Current knowledge of the world

Priors

Uncertain access result

Structured source profiles

Modeling

Knowledge acquisition plan
Current knowledge of the world

Structured source profiles

Knowledge acquisition plan

Knowledge need

Uncertain access result

answer & explanation

optimization

knowledge update

formulation

intensional access

modeling

priors

Kno\wledge update
What this talk is about

- A personal perspective on how to approach intensional data management
- Various applications of intensional data management and how we solved them (own research and my students’)
- Main tool used: reinforcement learning (bandits, Markov decision processes)
- Focus on one such application: database tuning
Outline

Intensional data management

Reinforcement learning

Applications

Focus: Database Tuning

Conclusion
Reinforcement learning

- Deals with agents learning to interact with a partially unknown environment
- Agents can perform actions, resulting in rewards (or penalties) and in a possible state change
- Agents learn about the world by interacting with it
- **Goal:** find the right sequence of actions that maximizes overall reward, or minimizes overall penalty
- **Classic tradeoff:** exploration vs exploitation
Multi-armed bandits

- **Stateless model**
- **$k > 1$** different actions, with **unknown rewards**
- Often assumed that the rewards are from a parametrized probabilistic distribution (Bernoulli, Gaussian, Exponential, etc.)
Markov decision process (MDP)

- Finite set of states
- In each state, actions lead:
  - to a state change
  - to a reward
- Depending on cases:
  - state changes may be deterministic or probabilistic
  - state changes may be known or unknown (to be learned)
  - rewards may be known or unknown (to be learned)
  - current state may even be unknown! (poMDP)
Outline

Intensional data management

Reinforcement learning

Applications

Focus: Database Tuning

Conclusion
Adaptive focused crawling (stateless)
[Gouriten et al., 2014]

- **Problem:** Efficiently crawl nodes in a graph such that total score is high
- **Challenge:** The score of a node is unknown till it is crawled
- **Methodology:** Use various predictors of node scores, and adaptively select the best one so far with multi-armed bandits
Adaptive focused crawling (stateless)  
[Gouriten et al., 2014]

- **Problem:** Efficiently crawl nodes in a graph such that total score is high
- **Challenge:** The score of a node is unknown till it is crawled
- **Methodology:** Use various predictors of node scores, and adaptively select the best one so far with multi-armed bandits
Adaptive focused crawling (stateless)  
[Gouriten et al., 2014]

- **Problem**: Efficiently crawl nodes in a graph such that total score is high
- **Challenge**: The score of a node is unknown till it is crawled
- **Methodology**: Use various predictors of node scores, and adaptively select the best one so far with multi-armed bandits
Adaptive focused crawling (stateful)

- **Problem**: Efficiently crawl nodes in a graph such that total score is high, taking into account currently crawled graph
- **Challenge**: Huge state space
- **Methodology**: MDP, clustering of state space, together with linear approximation to value functions
Optimizing crowd queries for data mining
[Amsterdamer et al., 2013a,b]

- **Problem**: To find patterns in the crowd, what is the best question to ask the crowd next?
- **Challenge**: No a priori information on crowd data
- **Methodology**: Model all possible questions as actions in a multi-armed bandit setting, and find a trade-off between exploration and exploitation
Online influence maximization

[Lei et al., 2015]

- **Problem:** Run influence campaigns in social networks, optimizing the amount of influenced nodes
- **Challenge:** Influence probabilities are unknown
- **Methodology:** Build a model of influence probabilities and focus on influential nodes, with an exploration/exploitation trade-off
Routing of Autonomous Taxis

[Han et al., 2016]

- **Problem**: Route a taxi to maximize its profit
- **Challenge**: Real-world data, no a priori model of the world
- **Methodology**: MDP, with standard Q-learning and customized exploration/exploitation strategy
Cost-Model-Free Database Tuning
[Basu et al., 2015, 2016]

- **Problem:** Automatically find which indexes to create in a database for optimal performance
- **Challenge:** The workload and cost model are unknown
- **Methodology:** Model database tuning as a Markov decision process and use reinforcement learning techniques to iteratively learn a cost model and workload characteristics
Outline

Intensional data management

Reinforcement learning

Applications

Focus: Database Tuning

Motivation
Problem Formulation
Adaptive Tuning Algorithm
COREIL: Index Tuner
Performance Evaluation
Outline

Intensional data management

Reinforcement learning

Applications

**Focus: Database Tuning**

Motivation

Problem Formulation

Adaptive Tuning Algorithm

COREIL: Index Tuner

Performance Evaluation
Motivation

- Current query optimizers depend on pre-determined cost models
- But cost models can be highly erroneous

...the cardinality model. In my experience, the cost model may introduce errors of at most 30% for a given cardinality, but the cardinality model can quite easily introduce errors of many orders of magnitude! I’ll give a real-world example in a moment. With such errors, the wonder isn’t “Why did the optimizer pick a bad plan?” Rather, the wonder is “Why would the optimizer ever pick a decent plan?”

Guy Lohman, IBM Research, ACM SIGMOD Blog 2014
Proposed Solution

- We propose and validate a tuning strategy to do without such a pre-defined model.
- The process of database tuning is modeled as a Markov decision process (MDP).
- A reinforcement learning based algorithm is developed to learn the cost function.
- COREIL replaces the need of pre-defined knowledge of cost in index tuning.
Outline

Intensional data management

Reinforcement learning

Applications

**Focus: Database Tuning**

Motivation

Problem Formulation

Adaptive Tuning Algorithm

COREIL: Index Tuner

Performance Evaluation
Problem

Database Schema: \( R \)

Customer 1 \rightarrow\text{Warehouse} \rightarrow Customer 2 \rightarrow Customer 3

Queries

1) New order
2) Delivery
3) Stock

Tables

1) History
2) Stock
3) New orders
4) Stocks

Set of all Database Configurations: \( S = \{s\} \)

Schedule of queries and updates: \( Q \)
Transition

Per-stage cost \( C(s_{t-1}, s_t, q_t) = \delta(s_{t-1}, s_t) + \text{cost}(s_t, q_t) \)
Mapping to MDP

States

Per-stage cost

\[ C(s_{t-1}, s_t, q_t) = \delta(s_{t-1}, s_t) + \text{cost}(s_t, q_t) \]

Penalty function

Query execution

Configuration update

Action

Mapping to MDP

Penalty function

Per-stage cost

\[ C(s_{t-1}, s_t, q_t) = \delta(s_{t-1}, s_t) + \text{cost}(s_t, q_t) \]
MDP Formulation

- **State**: Database configurations $s \in S$
- **Action**: Configuration changes $s_{t-1} \rightarrow s_t$ along with query $q_t$ execution
- **Penalty function**: Per-stage cost of the action $C(s_{t-1}, s_t, \hat{q}_t)$
- **Transition function**: Transition from one state to another on an action are deterministic
- **Policy**: A sequence of configuration changes depending on the incoming queries
Problem Statement

- For a policy $\pi$ and discount factor $0 < \gamma < 1$ the cumulative penalty function or the cost-to-go function can be defined as,

\[
V^\pi(s) \triangleq \mathbb{E} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} C(s_{t-1}, s_t, \hat{q}_t) \right] \text{ satisfying } \begin{cases} s_0 = s \\ s_t = \pi(s_{t-1}, \hat{q}_t), & t \geq 1 \end{cases}
\]

- **Goal**: Find out an optimal policy $\pi^*$ that minimizes the cumulative penalty or the cost-to-go function
Features of The Model

- The schedule is sequential
- The issue of concurrency control is orthogonal
- Query $q_t$ is a random variable generated from an unknown stochastic process
- It is always cheaper to do a direct configuration change
- There is no free configuration change
Outline

Intensional data management

Reinforcement learning

Applications

Focus: Database Tuning

Motivation
Problem Formulation
Adaptive Tuning Algorithm
COREIL: Index Tuner
Performance Evaluation
Policy Iteration

A *dynamic programming* approach to solve MDP

- Begin with an initial policy \( \pi_0 \) and initial configuration \( s_0 \)
- Find an estimate \( \overline{V}^{\pi_0}(s_0) \) of the cost-to-go function
- Incrementally improve the policy using the current estimate of the cost-to-go function. Mathematically,

\[
\overline{V}^{\pi_t}(s) = \min_{s' \in S} \left( \delta(s, s') + \mathbb{E} [cost(s', q)] + \gamma \overline{V}^{\pi_{t-1}}(s') \right)
\]

- Carry on the improvement till there is no (or \( \epsilon \)) change in policy
Problems with Policy Iteration

- **Problem 1**: The curse of dimensionality makes direct computation of $\bar{V}$ hard
- **Problem 2**: There may be no proper model available beforehand for the cost function $\text{cost}(s, q)$
- **Problem 3**: The probability distribution of queries being unknown, it is impossible to compute the expected cost of query execution
Solution: Reducing the Search Space

Proposition

Let $s$ be any configuration and $\hat{q}$ be any observed query. Let $\pi^*$ be an optimal policy. If $\pi^*(s, \hat{q}) = s'$, then

$$\text{cost}(s, \hat{q}) - \text{cost}(s', \hat{q}) \geq 0.$$  

Furthermore, if $\delta(s, s') > 0$, i.e., if the configurations certainly change, then

$$\text{cost}(s, \hat{q}) - \text{cost}(s', \hat{q}) > 0.$$  

Thus, the reduced subspace of interest

$$S_{s, \hat{q}} = \{s' \in S \mid \text{cost}(s, \hat{q}) > \text{cost}(s', \hat{q})\}$$
Solution: Learning the Cost Model

- Changing the configuration from $s$ to $s'$ can be considered as executing a special query $q(s, s')$
- Then the cost model can be approximated as

$$\delta(s, s') = \text{cost}(s, q(s, s')) \approx \zeta^T \eta(s, q(s, s'))$$

- This approximation can be improved recursively using Recursive Least Square Estimation (RLSE) algorithm
- Similar linear projection $\phi(s)$ can be used to approximate the cost-to-go function $V^{\pi_t}(s)$
Outline

Intensional data management

Reinforcement learning

Applications

Focus: Database Tuning

Motivation
Problem Formulation
Adaptive Tuning Algorithm
COREIL: Index Tuner
Performance Evaluation

Conclusion
What is COREIL?

COREIL is an index tuner, that

- instantiates our reinforcement learning framework
- tunes the configurations differing in their secondary indexes
- handles the configuration changes corresponding to the creation and deletion of indexes
- inherently learns the cost model and solves an MDP for optimal index tuning
COREIL: Reducing the State Space

- Let $I$ be the set of all possible indexes.
- Each configuration $s \in S$ is an element of the power set $2^{|I|}$.
- $r(\hat{q})$ be the set of recommended indexes for a query $\hat{q}$.
- $d(\hat{q})$ be the set of indexes being modified (update, insertion or deletion) by $\hat{q}$.
- The reduced search space is:

$$S_{s, \hat{q}} = \{s' \in S \mid (s - d(\hat{q})) \subseteq s' \subseteq (s \cup r(\hat{q}))\}$$

- For $B^+$ trees, prefix closure $\langle r(\hat{q}) \rangle$ replaces $r(\hat{q})$ for better approximation.
COREIL: Feature Mapping Cost-to-go Function

- We can define

\[ \phi_{s'}(s) \triangleq \begin{cases} 
1, & \text{if } s' \subseteq s \\
-1, & \text{otherwise.} 
\end{cases} \quad \forall s, s' \in S \]

**Theorem**

*There exists a unique \( \theta = (\theta_{s'})_{s' \in S} \) which approximates the value function as*

\[ V(s) = \sum_{s' \in S} \theta_{s'} \phi_{s'}(s) = \theta^T \phi(s) \]
COREIL: Feature Mapping Per-stage Cost

\( \beta(s, \hat{q}) \) captures the difference between the index set recommended by the database system and that of the current configuration.

\( \alpha(s, \hat{q}) \) takes values either 1 or 0 whether a query modifies any index in the current configuration.

We define the feature mapping

\[ \eta = (\beta^T, \alpha^T)^T \]

to approximate the functions \( \delta \) and \( cost \)
Outline

Intensional data management

Reinforcement learning

Applications

Focus: Database Tuning

Motivation

Problem Formulation

Adaptive Tuning Algorithm

COREIL: Index Tuner

Performance Evaluation
Dataset and Workload

- The dataset and workload conform to the TPC-C specification
- They are generated by the OLTP-Bench tool
- Each of the 5 transactions are associated with 3 ~ 5 SQL statements (query/update)
- Response time of processing corresponding SQL statement is measured using IBM DB2
- The scale factor (SF) used here is 2
Efficiency

![Graph showing Efficiency over Query #](image)

**COREIL**

**WFIT**
Box-plot Analysis

![Box-plot chart comparing COREIL and WFIT time in milliseconds]

- COREIL
- WFIT

Time (ms)

200
400
600
800

200
400
600
800

Focus: Database Tuning

Applications

Reinforcement learning

Intensional data management
Overhead Cost Analysis

- **COREIL**
- **WFIT**

Query # vs. Time (ms)
Effectiveness

![Graph showing time (ms) vs query # with COREIL and WFIT lines]

- Time (ms)
- Query #
- COREIL
- WFIT
Outline

Intensional data management

Reinforcement learning

Applications

Focus: Database Tuning

Conclusion
In brief

- Intensional data management arises in a large variety of settings, whenever **there is a cost to accessing data**
- **Reinforcement learning** (bandits, MDPs) is a key tool in dealing with such data
- Various **complications** in data management settings: huge state space, no a priori model for rewards/penalties, delayed rewards...
- **Rich** field of applications for RL research!
Merci.
Bibliography I


