Distributed Machine Learning

A New Era

Yi Wang

Novelty

- Machine Learning
  - modeling exponential-family distributed
  - engineering for speed
- Distributed Machine Learning
  - modeling long-tail distributed
  - engineering for scalability

Stories

- Existing methods and frameworks (2007~2010)
  - Methods: Frequent itemset mining, Collaborative filtering, Spectral clustering, Graph partitioning, Restricted Boltzmann machine, Latent topic modeling
  - Frameworks: MPI, MapReduce, Pregel, GBR
- My own methods and frameworks (2010~2014)
  - MapReduce Lite (C++) for language models
  - Peacock (Go) for latent topic modeling

Lessons

- Internet services relies on machine intelligence.
- Intelligence comes from learning users' behavior.
- Value lies in long tails.
- It is more about *big* than *fast*.
- Good system = good algorithm + good architecture.
- More about engineering than math.
- It is Industrial Revolution!

Lesson: Services and Intelligence

- Internet services intelligent enough to replace human:
  - Search engine: librarians
  - Online advertising: advertising agent companies
  - Recommender systems: markets, bookstores, presses
  - Credit models: bankers, loaners
- Service quality depends on intelligent system quality.

Lesson: Collaborative Intelligence

- Intelligent systems do not “invent” intelligence, they summarize knowledge from user behavior data.
  - Language model: Blog posts, tweets, IM chats
  - Click model: query-click sessions from search engine
  - Semantic model: search queries
  - Recommendation model: likes, rates, transactions
- Internet services collects “big data” and enables computers smarter than any human expert.

Lesson: Value Lies in Long-tail

- Given query “chomsky piraha”, return which ad?
  - Most queries are long-tail, in-frequent, less known.
  - “Chomsky studied Piraha language and founded linguistic science”.
- Value lies in long-tail
  - An ignorant system returns random ads: CTR=1%.
  - Intelligent systems return linguistic textbooks. CTR=50%
  - 50 times revenue boosting!!!

https://ywdxvpn.tencent.com:8443/prx/000/http/10.168.132.140/charlie

Activity Community Persons Company Computer Science Computer Engineer

query:

红酒木瓜汤

Submit

---

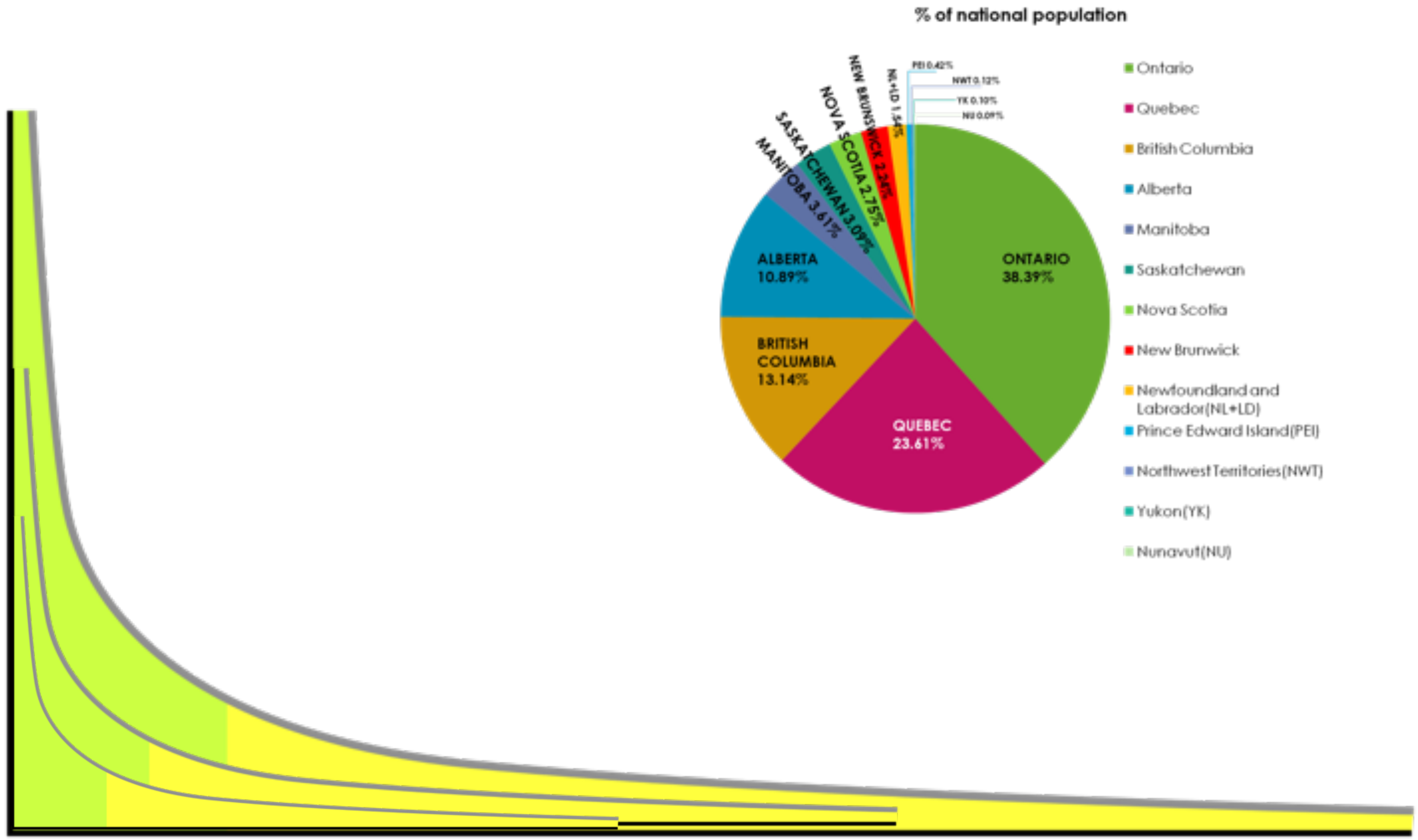
tokens:

红酒(0.589038) 木瓜(0.582175) 汤(0.560452)

---

topics:

id(rank)	weight	topic_words
6147(3672)	0.904523	丰胸(0.170997) 产品(0.080866) 减肥(0.067258) 木瓜(0.048
6338(325)	0.301545	糖尿病(0.081618) 血糖(0.033829) 高血压(0.028768) 孕妇(0.0
8009(3430)	0.301511	奇迹(0.247384) 世界(0.081658) 加点(0.037639) 木瓜(0.037



Long-tail is scale-free.
Mean and median make no sense with long-tail distributions.
Pie-charts make no sense either.

Lesson: Faster or Bigger

- Traditional parallel computing focuses on “faster”.
- Distributed machine learning focuses on “bigger”.
- long-tail data are big: many varieties along the tail,
- Big data requires scalable, fault-recoverable systems.
- Curse of data dependency:

  A job with 100,000 workers running for 5 mins on a “shared cluster”. Zero probability that no worker gets preempted. Recover a failed process requires restarting other processes talked with it. This recursion propagates and the whole job has to be restarted. The restart iterates and the job never end!

Lesson: Engineering or Math

- Modeling long-tail is out of textbook.
  - Dirichlet process, Pitman-Yor process
- Engineering scalable systems requires new knowledge
  - cluster operating systems, Paxos protocol, container.
- Interleaving both.
  - Peacock: new architecture enables learning a million latent topics, large enough that makes asymmetric Dirichlet distribution mimics Dirichlet process.
  - This math property simplifies communication in model learning and makes new architecture scalable.

Lesson: Industrial Revolution

- Distributed machine learning is reshaping the Internet industry.
- It requires new skills on both engineering and math.
- Mastering both enables building “machines”, or
- lack of any is like “handcraft men” being replaced by machines.

Pitfalls

- De-noise data
- Parallelize models in papers and textbooks
- Use standard measures
- Use existing frameworks
- Mix frameworks with cluster operating systems
- Less talking about production
- MPI
- Java or Python

Pitfall: De-noise

- Noise means data points with low frequency.
- Most part of long-tail distributed data are with low frequency.
- De-noising means abandon most part of data.
- Even for exponential-family data, when de-noising works, down-sampling works.
- Long-tail data has no noise!

セカンドライフ ゲーム secondlife	セカンドライフ and ゲーム are Japanese. セカンドライフ means secondlife. ゲーム means game. And secondlife is a game (http://secondlife.com/).	14
映像 映画 PV CM 動画コンテンツ 動画投稿 動画 ウェブ	These are all Japaness Kanji and Chinese words means "image"	13
христианство православие orthodox	христианство and православие are Russian. христианство means Christianity. православие means Orthodox. All these three words relate to religious.	8
正妹 taiwan album beauty photo	正妹 is tranditional Chinese in Taiwan. 正妹 means beauty. Lots of people search 正妹 for beauties' photos.	7
whorf piraha chomsky anthropology linguistics	Whorf and Chomsky are all experts in anthropology and linguistics, and they did research in a tribe named Piraha.	6

"Noise" but Interesting Patters from Del.icio.us Tags ▶

Pitfall: Models from Textbook

- Most probabilistic models assume exponential-family distributed data:
  - e.g. LDA: Dirichlet prior and multinomial likelihood.
- Most linear algebraic methods assume exponential-family:
  - e.g., SVD results make sense iff Gaussian distribution.
- Other methods bias towards “thick head” instead of “long tail”:
  - e.g., logistic regression fits frequent instances more than long-tails ones.

Pitfall: Standard Measures

- KDD Cup 2012 Task-2: Ads CTR prediction.
  - Instance distribution over feature space is long-tail.
- For competitors and evaluators:
  - AUC works, but measures only correct ranking order.
  - MAE/MSE fooled by "returning the average CTR".
- Industrial solution:
  - Online experiment systems.
  - Use business measures like ROI, CTR, CPM.

Pitfalls: Use existing frameworks

- MPI ⟷ Pregel, GraphLab, Spark ⟷ MapReduce
  - MPI: flexible; no fault-tolerance no scalability.
  - MapReduce: rigid; scalable.
  - Pregel etc.: flexibility and scalability under conditions
- Pregel: fault-recovery by message-buffering & checkpoint
  - PageRank/Graph-clustering: works well! ▶
  - LDA: message-buffering causes "not enough memory".

Pitfalls: Heavy Frameworks

- A Comparison:
  - MapReduce in Hadoop 1.x: 100,000 lines Java code.
  - MapReduce Lite: 1,000 lines C++ code.
- Reasons:
  - Java-fashion: over-engineering, over-configurability.
  - Sophisticated features: backup worker.
  - No idea of using cluster operating systems.
- Solution: full-stack distributed computing technology.

Distributed Computing Technology Stack

application	PageRank	Indexing	pCTR	DNN
framework	Pregel	MapReduce	SETI	DistBelief
middleware	Chubby (Zookeeper, etcd), Bigtable (HBase), memcachg (memcached)			
cluster OS	Borg (Mesos, YARN, Kubernetes)			
filesystem	GFS (HDFS)			

Pitfalls: Not Product Oriented

- Example: guess what's behind Google Ad system:
  - Multiple venues: AdWords, AdSense, Youtube Ads.
  - Real-time response to user behavior: Re-targeting.
- System is much more than math/models:
  - Real-time and fault-tolerable data collection pipelines;
  - One online learning system behind all venues;
  - Learning system as prediction system.

Pitfalls: Languages no matter

- It does matter to choose programming languages! ▶
- To program distributed systems, we need:
  - light-weighted threads for concurrency
  - channels/blocking-queues for remove callbacks
  - language syntax (select) that wraps OS kernel calls (e.g., epoll and kqueue) for synchronization.
- For C++, Java, Python programmers:
  - Times more lines of code than Go or Lisp.
  - Times more chance to have bugs untraceable.

```
var resp *Response
select {
  case b := <- rpc.Call("B"):
    resp = extract(b)
  case c := <- rpc.Call("C"):
    resp = extract(c)
  case e := <- rpc.Call("E"):
    resp = extract(e)
  case <- time.Timeout(1*second):
    resp = nil
}
// use resp here.
```

A Go program that calls three RPC servers and continue with any one of them returns response, or stops with timeout.

```
var mutex = NewMutex();
var returns = 0;
var timer = setTimeout(timeout,
     1*second);

function rpcResp(resp) {
  mutex.Lock();
  if (retures == 0) {
    clearTimeout(timer);
    use(resp);
  }
  returns++;
  mutex.Unlock();
}
```

```
function timeout() {
  mutex.Lock();
  returns++;
  mutex.Unlock();
}

rpc.Call("B", rpcResp);
rpc.Call("C", rpcResp);
rpc.Call("D", rpcResp);
```

A Javascript program doing the same work has more code and twisted logic. Not mentioning Java or Python or C++.

Release Your Power

- Business tuning or killing-tech dev?
  - Separated: IBM/Microsoft Research
  - Combined: Google, Facebook
- Standalone businesses:
  - ML software: MATLAB, R, liblinear, libsvm
  - ML frameworks: Spark, GraphLab
  - MLaaP: Google Prediction API
  - MLaaS: Scaled Inference