Distributed Bayesian Machine Learning

Biletskyy B.A.
Institute of Cybernetics NAS Ukraine

Image Computer
Kyiv, 10-04-2018
Agenda

• Motivation
• Distributed Computing
• Distributed Machine Learning
• Naïve Bayes
  • Multinomial
  • Bernoulli
  • Categorical
• Efficiency
• Q&A
Motivation

• More complex Machine Learning models require bigger datasets for training

• Amount of data doubles every 20 months (Volume in 3V’s of BigData)*

• Computational power doubles every 18 months (Moore’s Law) **


Moore’s Law Is Over

Andrew Danowitz, Kyle Kelley, James Mao, John P. Stevenson, Mark Horowitz  Communications of the ACM, Vol. 55 No. 4, Pages 55-63
Scalability

Vertical vs. Horizontal
Scalability

• Scalability
  • Horizontal -> more computers
  • Vertical -> more CPU cores, **GPU**

• Parallel Computing
  • Nondeterministic (execution order is not guaranteed)
  • Hard to reason about

• Some parallel computing techniques
  • SOA
  • Actors Model
  • **MapReduce**
MapReduce

• $A, B$ – input and output datatypes (or finite sets $|A| < \infty$, $|B| < \infty$)
• $(a_1, \ldots, a_n)$ – distributed input dataset, $a_i \in A$
• $f : A \times \cdots \times A \mapsto B$ – heavy computation we need to parallelize

MapReduce representation:

$$f(a_1, \ldots, a_n) = m(a_1) \oplus m(a_2) \oplus \cdots \oplus m(a_n)$$
MapReduce

\[ f(a_1, \ldots, a_n) = m(a_1) \oplus m(a_2) \oplus \ldots \oplus m(a_n) \]

- Map \( m: A \mapsto B \)
  - computes partial result \( m(a_i) \) for each \( a_i \)
  - pure function
MapReduce

\[ f(a_1, ..., a_n) = m(a_1) \oplus m(a_2) \oplus ... \oplus m(a_n) \]

Reduce \( \oplus : B \times B \mapsto B \)

- aggregates partial results \( m(a_i) \oplus m(a_j) \)
- satisfies axioms:
  1. \( b_1 \oplus b_2 = b_2 \oplus b_1, \forall b_1, b_2 \in B \) (commutativity)
  2. \( (b_1 \oplus b_2) \oplus b_3 = b_1 \oplus (b_2 \oplus b_3), \forall b_1, b_2, b_3 \in B \) (associativity)
  3. \( \exists e \in B : b \oplus e = b, \forall b \in B \) (zero-element)
MapReduce Word Count Example

\[(a_1, \ldots, a_l) \in A^l \text{ – input: collection of } l \text{ documents over vocabulary } V, |V| < \infty\]
\[(b_1, \ldots, b_{|V|}) \in B \text{ – output: word counts vector } f(a_1, \ldots, a_l) = (b_1, \ldots, b_{|V|})\]

MapReduce representation:
\[f(a_1, \ldots, a_l) = m(a_1) \oplus m(a_2) \oplus \ldots \oplus m(a_l)\]

- Map: \(m: A \mapsto B\) – returns word counts \(m(a_i)\) for document \(a_i\)
- Reduce: \((x_1, \ldots, x_{|V|}) \oplus (y_1, \ldots, y_{|V|}) = (x_1 + y_1, \ldots, x_{|V|} + y_{|V|})\) – aggregates partial word count vectors
  1. \(b_1 \oplus b_2 = b_2 \oplus b_1, \forall b_1, b_2 \in B\) (commutativity)
  2. \((b_1 \oplus b_2) \oplus b_3 = b_1 \oplus (b_2 \oplus b_3), \forall b_1, b_2, b_3 \in B\) (associativity)
  3. \(\exists e \in B: b \oplus e = b, \forall b \in B\) (zero-element \(e = (0, \ldots, 0)\))
How MapReduce Works

\[ f(c) = m(c_1) \oplus m(c_2) \oplus ... \oplus m(c_{n-1}) \oplus m(c_n) \]
Distributed (Supervised) Machine Learning

• $X$ – object data type, $|X| < \infty$
• $Y$ - set of labels, $|Y| < \infty$
• $\tau = ((x_1, y_1), \ldots, (x_l, y_l))$ – training sample, $x_i \in X, y_i \in Y, \tau \in T$

• Problem statement: training set calculate label for new object

• 2 Phases:
  • Training
  • Prediction
Machine Learning: Training

• $\gamma$: $X \mapsto Y$ – decision rule
  • makes predictions $\gamma(x) = y \in Y$
  • $\gamma \in \Gamma$
  • $\Gamma$ – set of all decision rules

• $Q$: $T \mapsto \Gamma$ – training procedure
  • builds decision rule $\gamma = Q(\tau)$ from a training set $\tau \in T$

• Computationally expensive
Machine Learning: Prediction

• $y = \gamma(x)$ – labelling of new objects, given trained classifier $\gamma \in \Gamma$

• Relatively computationally cheap
(Counter) Examples

• Neural Networks
  • Training (hardly horizontal scalable)

• Structural Classification
  • Classification is optimization problem of quadratic complexity

• KNN
  • Training phase is executed on
SparkML

- Spark – high level API over MapReduce + in-memory computing
- SparkML – distributed Machine Learning
  - Neural Networks
  - Decision Trees
  - SVM
  - ...
  - Naïve Bayes*
    - Inspired by NLP use-cases
    - Multinomial
    - Bernoulli

Naive Bayes Classification

- $x = (x_1, ..., x_n) \in X$ – object representation, $Y$ – labels, $\gamma: X \mapsto Y$ – decision rule, $\tau = (\tau_1, ..., \tau_m)$ - training sample
- Conditional probability:
  $$\gamma(x) = \arg\max_{y \in Y} P(y|x)$$
- Apply Bayes Theorem $P(A|B) = P(B|A)P(A)/P(B)$:
  $$\gamma(x) = \arg\max_{y \in Y} \frac{P(x|y) P(y)}{P(x)}$$
- Denominator doesn’t depend on $y$, we can get rid of it:
  $$\gamma_B(x) = \arg\max_{y \in Y} P(x_1, ..., x_n|y)P(y)$$
- What is $P(x_1, ..., x_n|y)$?
- Naïve Bayes assumption: $x_1, ..., x_n$ are pairwise independent given $y$
  $$\gamma_{NB}(x) = \arg\max_{y \in Y} P(y) \prod_{i=1}^n P(x_i|y)$$
- Use estimates $\hat{P}(y), \hat{P}(x_i|y)$ obtained from $\tau$:
  $$\gamma_{NB}(x) = \arg\max_{y \in Y} \log_2 \hat{P}(y) + \sum_{i=1}^n \log_2 \hat{P}(x_i|y)$$
Bernoulli Naïve Bayes

• Use-case: given hotel review, detect sentiment (positive, negative, neutral)
• Document in class $y$ is generated by tossing an $y$-specific set of $|V|$ unfear coins
• $x = (x_1, ..., x_{|V|})$ – document representation, $x_i \in \{0,1\}$ if word $v_i \in V$ is present/absent in the document

\[
P(x_1, ..., x_{|V|} | y) = \prod_{i=1}^{|V|} [P(v_i | y) x_i + (1 - P(v_i | y))(1 - x_i)] \quad \text{– Bernoulli distribution}
\]
• $P(v_i | y)$ – probability of word $v_i$ in class $y$ (the ratio of documents labeled $y$ containing word $v$)

• Bernoulli Naïve Bayes decision rule:

\[
\gamma_{BNB}(x_1, ..., x_{|V|}) = \arg\max_{y \in Y} P(y) \prod_{i=1}^{|V|} [P(v_i | y) x_i + (1 - P(v_i | y))(1 - x_i)]
\]
Multinomial Naïve Bayes Method

• Use-case: given news article – predict topic (sports, health, politics)
• Document of length $l$ in class $y$ is generated by rolling $y$-specific unfear $|V|$-sided die $l$ times
• $x = (x_1, ..., x_{|V|})$ – document representation, $x_i \in \mathbb{Z}$ – number of occurrences of word $v_i \in V$

$$P(x_1, ..., x_{|V|} | y) = K(x) \prod_{i=1}^{|V|} P(v_i | y)^{x_i}$$ – multinomial distribution given $y$
  • $K(x) = \frac{(\sum_{i=1}^{|V|} x_i)!}{\prod_{i=1}^{|V|} x_i!}$ – multinomial coefficient
  • $P(v | y)$ – probability of word $v$ in class $y$ (the ratio of positions containing word $v$ in documents labeled $y$)

• Multinomial Naïve Bayes decision rule:

$$\gamma_{MNB}(x_1, ..., x_{|V|}) = \arg\max_{y \in Y} P(y) \prod_{i=1}^{|V|} P(v_i | y)^{x_i}$$
Categorical Naïve Bayes

• Object in class $y$ is generated by rolling an $y$-specific set of $n$ unfear $|X_1|,..., |X_n|$-sided dice
• $x = (x_1, ..., x_n)$ – object representation, $x_i \in X_i$, $X_i$ – $i$-th attribute space, $|X_i| < \infty$
• $P(x_1, ..., x_n | y) = \prod_{i=1}^{n} P_i(x_i | y)$
  • $P_i(x | y)$ - the ratio of objects in class $y \in Y$ with $i$-th attribute taking value $x \in X_i$

• Categorical Naïve Bayes decision rule:
  $$\gamma_{CNB}(x_1, ..., x_n) = \arg\max_{y \in Y} P(y) \prod_{i=1}^{n} P_i(x_i | y)$$

• SparkML implementation
  • https://github.com/bbiletskyy/categorical-bayes
  • test data: http://archive.ics.uci.edu/ml/index.php
Model Complexity

• Multinomial and Bernoulli Naïve Bayes
  • $\hat{P}(i|y), \hat{P}(y)$
  • $Q(\tau) = \gamma_{\pi, \theta}(x_1, \ldots, x_n)$
    • $\pi = (\hat{P}(y_1), \ldots, \hat{P}(y_{|Y|}))$ — $|Y|$-dimensional vector of class prior probability estimates
    • $\theta = (\theta_{ij}) \in \mathbb{R}^{n \times |Y|}$ — matrix of conditional probability estimates, $\theta_{y,i} = \hat{P}(i|y)$

• Categorical Naïve Bayes
  • $\hat{P}_1(x_1|y), \ldots, \hat{P}_n(x_n|y), \hat{P}(y)$
  • $Q(\tau) = \gamma_{\pi, \theta_1, \ldots, \theta_n}(x_1, \ldots, x_n)$
    • $\pi = (\hat{P}(y_1), \ldots, \hat{P}(y_{|Y|}))$ — $|Y|$-dimensional vector of prior probability estimates
    • $\theta_k = (\theta_{ij}) \in \mathbb{R}^{n \times |Y|}$ — conditional probability estimates matrix, $\theta_{y,i} = \hat{P}(i|y), k = 1, n$
Naïve Bayes Efficiency

• Claim: Naïve Bayes is optimal ML technique for objects with independent attributes

• Machine Learning efficiency measure — average error as function of training sample size
  • Class of problems:
    • \( x = (x_1, \ldots, x_n) \in X^n \) - object representation, \(|X|, |Y|\)
    • \( l = \sum_{y \in Y} l_y \) - total size of training sample, \( l_y \) - number of \( y \)-labeled examples in training sample
  • Statistical learning theory (VC-theory)
    • Upper bounds*: \( v < \sqrt{\frac{|\lg |\Gamma|+\lg \eta}{l}} \) with probability \( 1 - \eta \)
  • Stochastic optimization learning theory
    • Upper bound**: \( v < c \sqrt{\frac{|X|\min\{l_y\} + |Y|}{l}} \)
    • Lower Bound***: \( v > c' \sqrt{\frac{|X|\min\{l_y\} + |Y|}{l}} \)

Q&A