A Study of Two Sampling Methods for Analyzing Large Datasets With ILP

Ashwin Srinivasan
Oxford University Computing Laboratory,
Oxford,UK

Presented by
Anand Sivaraman (04/23/2003)

Agenda

- Introduction
- An ILP Algorithm
- Two Sampling Methods
  - Subsampling
  - Logical Windowing
- Empirical Evaluation
- Concluding Remarks

Introduction

- Large quantities of data available
- Analyzing data to obtain some sense
- Methods to analyze this data
  - Deal with sheer bulk of data
- Programs developed within framework of Inductive Logic Programming(ILP)

ILP

- ILP – Program attempts to construct a concise theory (set of rules) for the observations presented
- Time taken by an ILP program - Dominated by theorem proving effort to evaluate theory
- Motivates two methods
  - Subsampling
  - Logical Windowing
- Allow for theory construction by sampling fractions of the data

Sampling Hypothesis

- Test utility of methods
- Compare
  - Estimated predictive accuracy
  - Time for theory construction
- Sampling Hypothesis:

  "An ILP system equipped with a sampling method constructs, in lesser time, a theory that is no worse in predictive accuracy to that obtained from the same algorithm without a sampling method"

Agenda

- Introduction
- An ILP Algorithm
- Two Sampling Methods
  - Subsampling
  - Logical Windowing
- Empirical Evaluation
- Concluding Remarks

An ILP Algorithm

- Specifications:
  - B – background knowledge - a set of clauses $=\{C_1, C_2, \ldots\ldots\}$
  - I – optional set of constraints on acceptable hypotheses
  - E – finite set of examples = $E^+ \cup E^-$ where
    - *Positive examples* - $E^+ = \{e_1, e_2, \ldots.\}$ – set of definite clauses;
    - *Negative Examples* - $E^- = \{f_1, f2, \ldots..\}$ – consistent set of Horn clauses
    - *Prior Necessity* - B *"does not entail"* E^+
- $H=\{D_1, D_2, \ldots\}$ – output of the algorithm given B,I,E is an explanation for E

An ILP Algorithm (contd....)

- A hypothesis H is an explanation for $E = E^+ \cup E^-$ if the following conditions are met:
  - *Weak Sufficiency* - Each D_i in H is a definite clause s.t. $B \cup \{ D_i \}$ *"entails"* $e_1 \vee e_2 \vee \ldots.$, where $\{e_1, e_2, \ldots.\} \subset E^+$
  - *Strong Sufficiency* – B U H *"entails"* E^+
  - *Weak Consistency* – B U H *"does not entail"* $\square$
  - *Strong Consistency* – (1) B U H U E^- *"does not entail"* $\square$
    - (2) B U H U I *"does not entail"* $\square$

An ILP Implementation

generalise$(B, I, \mathcal{L}, E)$: Given background knowledge B, hypothesis constraints I, a finite training set $E = E^+ \cup E^-$, returns a hypothesis H in $\mathcal{L}$ such that H explains the E.

1. $i = 0$
2. $E_i^+ = E^+$, $H_i = \emptyset$
3. if $E_i^+ = \emptyset$ return H_i otherwise continue
4. increment i
5. $Train_i = E_{i-1}^+ \cup E^-$
6. Let $e_i^+ \in E_{i-1}^+$, D_i be the "best" definite clause in $\mathcal{L}$ s.t. $B \cup H_{i-1} \cup \{D_i\} \models \{e_i^+\}$, $B \cup H_{i-1} \cup \{D_i\} \not\models \square$, $B \cup H_{i-1} \cup \{D_i\} \cup E^- \not\models \square$, and $B \cup H_{i-1} \cup \{D_i\} \cup I \not\models \square$
7. $H_i = H_{i-1} \cup \{D_i\}$
8. $E_p = \{e_p : e_p \in E_{i-1}^+ \text{ s.t. } B \cup H_i \models \{e_p\}\}$.
9. $E_i^+ = E_{i-1}^+ \backslash E_p$
10. Go to Step 3

An ILP Algorithm

search$(B, H, I, \mathcal{L}, O, E, n, e)$: Given background knowledge B, a set of clauses H, hypothesis constraints I, an "open list" O of potential clauses, a training set $E = E^+ \cup E^-$, an upper limit on the number of clauses searched n, and an example e, returns a clause C in $\mathcal{L}$ such that $B \cup H \cup \{C\} \models e$, and C is consistent with E^- and I.

1. $C = e$, $bestscore = score(e)$, $i = 0$
2. if $O = \emptyset$ or $i > n$ return C otherwise continue
3. Remove the first clause D from O
4. increment i
5. Refine clause D to a set of clauses S_D s.t. for every clause $s \in S_D$ $B \cup H \cup \{s\} \models e$
6. Score each clause in S_D. Let h be the highest score and $BestSet$ be the clauses with score h in S_D
7. If $h > bestscore$ and $\exists Best \in BestSet$ s.t. $B \cup H \cup \{Best\} \not\models \square$, $B \cup H \cup \{Best\} \cup E^- \not\models \square$, and $B \cup H \cup \{Best\} \cup I \not\models \square$ then $bestscore = h$ and $C = Best$
8. Add each clause in S_D to O. Sort O according to descending score of clauses
9. Go to Step 2

Search Complexity

- Select D (Step 3)
  - List is sorted.Hence $O(1)$
- Refine D (Step 5)
  - N literals to be added – Check each for variable-connectivity
  - At most c-1 literals in D, with at most a(c-1) variables
  - Worst Case – check each of at most a input variables in N literals
  - Hence $O(a^2cN)$

- *N – Maximum cardinality of any S_D*
- *a – max arity of any predicate in background knowledge*
- *b – max branching factor in AND – OR proof*
- *c – max number of literals in any clause*
- *d – max proof-depth for a goal*

Search Complexity (Contd...)

- Score Clauses in S_D (Step 6)
  - Check proof for each example in E (AND – OR tree)
  - b^{2d-1} leaves in a tree of depth d
  - At most N elements in S_D
  - Scoring done in $O(b^{2d-1} |E| N)$
- Sort Open List (Step 8)
  - No more than N^{c-1} clauses in open list
  - Simple sorting algorithm takes $O(N^{c-1} \log N^{c-1})$

- *N – Maximum cardinality of any S_D*
- *a – max arity of any predicate in background knowledge*
- *b – max branching factor in AND – OR proof*
- *c – max number of literals in any clause*
- *d – max proof-depth for a goal*

Search Complexity

- Cost for Step 3 is negligible
- Cost can be reduced in Step 8 by using efficient data structures and sorting techniques
- N,a,c are fixed values. Hence, not much scope in Step 5
- Further Savings possible in Step 6
  - Reducing the time for the AND – OR proof
  - Reducing the number of examples
    - Not including all the examples (e.g. "positive-only" learning)
    - Sampling fractions of examples

A Comparison of Time Complexities

Problem	$\lvert E\rvert$	Select	Refine	Score	Sort
Trains [19]	10	10^0	10^1	10^2	10^3
Pharmacophore [27]	28	10^0	10^2	10^3	10^3
Mutagenesis [35]	188	10^0	10^2	10^5	10^4
Mesh [7]	2500	10^0	10^2	10^7	10^4
Tagging [5]	6000	10^0	10^1	10^{12}	10^2
KRK [28]	10000	10^0	10^3	10^6	10^2

Figure 3. A comparison of time-complexities for search steps. The tabulations are based on the following values. Trains: N = 10, a = 1, b = 4, c = 4, d = 1; Pharmacophore: N = 10, a = 2, b = 7, c = 4, d = 1; Mutagenesis: N = 25, a = 2, b = 25, c = 4, d = 1; Mesh: N = 30, a = 1, b = 200, c = 4, d = 1; Tagging: N = 5, a = 1, b = 7, c = 5, d = 5; and KRK: N = 12, a = 6, b = 22, c = 3, d = 1.

Agenda

- Introduction
- An ILP Algorithm
- Two Sampling Methods
  - Subsampling
  - Logical Windowing
- Empirical Evaluation
- Concluding Remarks

Subsampling

- Randomly select subsample from dataset
- Score each clause on this subsample
- Complexity results that referred earlier to |E| will now reduce to (m) – size of the subsample
- Choice of (m) - User determined (default – 20000)

Constructing Theories - Subsampling

$ss(B, I, \mathcal{L}, E, m)$: Given background knowledge B, hypothesis constraints I, a finite training set $E = E^+ \cup E^-$, and an upper limit on subsample size m, returns a hypothesis H in $\mathcal{L}$ such that H explains the E.

1. $i = 0$
2. $E_i^+ = E^+$, $H_i = \emptyset$
3. if $E_i^+ = \emptyset$ return H_i otherwise continue
4. increment i
5. $Train_i = E_{i-1}^+ \cup E^-$
6. Let $D_i = bestclause(B, H_{i-1}, I, \mathcal{L}, Train_i, m)$ be the best clause obtained by scoring clauses on a random sample of at most m examples selected from $Train_i$ and implies at least one example in E_{i-1}^+
7. $H_i = H_{i-1} \cup \{D_i\}$
8. $E_p = \{e_p : e_p \in E_{i-1}^+ \text{ s.t. } B \cup H_i \models \{e_p\}\}$.
9. $E_i^+ = E_{i-1}^+ \setminus E_p$
10. Go to Step 3

Remark:choice of M and Coverage Estimates

- Assume clause C had a coverage N =$|E_p|+|E_n|$ on E
- Let p = N/|E|
- Let C have a coverage r=$|e_p|+|e_n|$ on m (sample drawn)
- Let p'= r/m – unbiased estimate of p
- We can at least be (1-α)100% confident that the error in estimating p with p′, will be less than a specified amount ε when the sample size m = $m_{\alpha,\varepsilon} = z^2_{\alpha/2}/4\,\varepsilon^2$,where

$z_{\alpha/2}$ – value of std. Normal dist. leaving area of α/2 to the right

Logical Windowing

- Constructs theories incrementally by sampling from a large pool of data
- Theory revision –
  - generalize a set of clauses using sample data
  - Specialize them to maintain consistency with large data pool
  - Revision – addition / deletion of clauses
- Possible to determine, and eliminate those clauses responsible for errors
- Hence, only partial reconstruction needed

Logical Windowing

- Reconstruction results in repeated effort due to:
  - Repeated re-derivation of unacceptable clauses
  - Repeated search for clauses to explain *incompressible* examples
- Remedies
  - Update constraints on acceptable hypothesis to exclude unacceptable clauses
  - Use a "conditional" variant that prevents search for an explanation for an *incompressible* example

  ✓ *Incompressible : If, given an example e, the search procedure returns e, we say e is "incompressible" with the input provided to the search procedure*

Procedure for Logical Windowing

window$(B, I, \mathcal{L}, E, m)$: Given background knowledge B, hypothesis constraints I, a finite training set $E = E^+ \cup E^-$, and a bound m on the number of errors, returns a hypothesis H in $\mathcal{L}$ such that H explains the E

1. $i = 0$
2. $FP_i = E^-$, $FN_i = E^+$, $H_i = \emptyset$, $I_i = I$, $Train_i = \emptyset$, $Ignore_i = \emptyset$
3. if $FN_i \cup FP_i = \emptyset$ return H_i otherwise continue
4. increment i
5. Let $Errors_i = sample(m, FP_{i-1}, FN_{i-1})$ be a sample of the errors
6. $Train_i = H_{i-1} \cup Errors_i$
7. Let $H'_i = generalise(B, I_{i-1}, \mathcal{L}, Train_i)$ (alternatively, let $H'_i = generalise'(B, I_{i-1}, \mathcal{L}, Train_i, Ignore_{i-1})$)
8. Let C_i be the overgeneral clauses in H'_i
9. $E_i = \{e_i : e_i \in Train_i \text{ s.t. } B \cup (H'_i \backslash C_i) \not\models \{e_i\}\}$
10. $H_i = (H'_i \backslash C_i) \cup E_i$
11. $Ignore_i = H'_i \cap Train_i$
12. $I_i = I_{i-1} \cup \{\Box \leftarrow c_1, \ldots, \Box \leftarrow c_k\}$, where $c_1, \ldots, c_k \in C_i$
13. $FP_i = \{fp_i : fp_i \in E^- \text{ s.t. } B \cup C_i \cup \{fp_i\} \models \Box\}$
14. $FN_i = \{fn_i : fn_i \in E^+ \text{ s.t. } B \cup H_i \not\models \{fn_i\}\}$
15. Go to Step 3

Bound on Iterations

- Each iteration of loop (Steps 3 – 15) will decrease the false negatives by at least m
- Once false negatives = 0, either
  - There are no false positives or
  - Each further iteration decreases them by m
- Hence, window procedure will never iterate more than $|E^+|/m + |E^-|/m = |E|/m$ times

Bounds on Examples Processed

- Assume sampling procedure returns no more than m errors each of false positives and false negatives
- Best case: First sample of 2m errors is sufficient to construct an explanation for $E = E^+ \cup E^-$
- Worst case: All examples needed to construct adequate explanation
  - Clauses constructed in every iteration save the last is over general
  - Function generalize called with 2m,4m,..examples
  - Assume without loss of generality $|E^-| \geq |E^+|$ and $n = |E^-| / |E^+|$
  - Then,total examples processed is composed of

  Sequence 2m,4m,...2 $|E^+|$ (at the end of which there are no false negatives)

  And 2 $|E^+|$+m,...$|E^+|$+$|E^-|$ (at the end of which there are no false positives)
- Sum of examples M =$|E|/2m\ ((4n-2n^2-1)\ |E|+m)$
- Hence, number of examples N processed satisfies $2m \leq N \leq M$

Choice of M

- No prescriptions for selecting optimal window size
- Let $E = E^+ \cup E^-$,and $\beta = max\,(|E^+|/|E|, |E^-|/|E|)$
- Two extreme values of m can be identified:
  - (1) the best scenario for windowing, where $m = m_1 = 1$ false positive and false negative
  - (2) the worst case scenario for windowing, where all explanations requires all examples. In this case, a choice m = m2 = $\beta|E|$ will ensure that no more than $|E|$ examples are processed.
- If windowing is to be useful, m should be closer to m1
- An average value is provided by the geometric mean,
  - $m = \sqrt{m_1 m_2} = \sqrt{\beta|E|}$
- Used in our experiments

Agenda

- Introduction
- An ILP Algorithm
- Two Sampling Methods
  - Subsampling
  - Logical Windowing
- Empirical Evaluation
- Concluding Remarks

Empirical Evaluation

- Evaluated on two tasks
  - Large datasets
- Classifying illegal positions in a KRK chess end game
- Classifying word tokens (tagging) in sentences taken from Wall Street Journal

Data Characteristics

Problem	Training Set			Test Set		
	Total	Pos	Neg	Total	Pos	Neg
KRK	10000	33%	67%	10000	33%	67%
Tag	1000	53%	47%	10000	52%	48%

Method for Tests

- For each task (KRK and Tag), repeat 10 times:
  - Randomly divide the data into training and test sets
    - Test size (N_{test}) = 10000 examples
    - Training set (N_{train}) for KRK =10000 and for Tag = 1000 examples
  - Construct a theory for the training set using Cprogol
  - Construct a theory for the training set using Cprogol+SS
    - $m = m_{0.05,0.05}$
  - Construct theories for the training set using Cprogol+LW
    - $M = \sqrt{\beta|N_{train}|}$
  - For each process record time taken to construct theory and accuracy on test set
  - Analyze for significant differences in accuracies and time for theories obtained

Results

Algorithm	KRK		Tag	
	Acc. (%)	Time (s)	Acc. (%)	Time (s)
CProgol	99.67	3634	68.35	41979
CProgol+SS	99.67	995	69.80	22011
CProgol+LW	99.67	284	68.60	25229

Discussion on Results

- Performance satisfactory but……!!!!!
  - Subsampling generates a large number of low-generality clauses
  - Only half of which have a positive compression score
  - Use a cautious/pessimistic estimate of clause coverage

✓ *Compression score:Number of positive examples covered minus the number of negative examples covered minus the number of literals in the body of a clause*

Results With Pessimistic Estimate

Algorithm	KRK		Tag	
	Acc. (%)	Time (s)	Acc. (%)	Time (s)
CProgol	99.68	3634	68.35	41978
CProgol+SS	99.68	1218	69.80	22010
CProgol+PSS	99.68	1747	68.49	29270

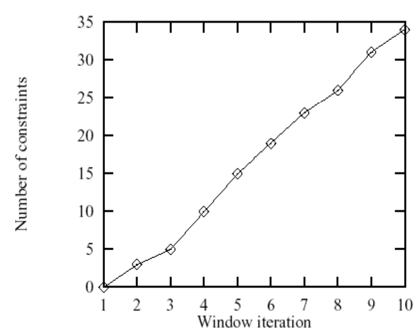
Results of Subsampling

- Investigate time taken for,
  - (1) Search for best clause
    - Expected to remain unchanged
  - (2) Removal of examples covered by best clause
    - Expected to increases with training set
- (2) Increases – as expected
- (1) Also increases !!!! – surprising ?
  - Possible reasons maybe:
    - Time for shuffling data – not implemented
    - Implementation details – need to be taken care of

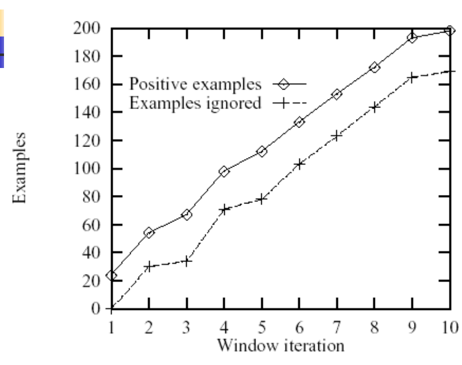
Results of Logical Windowing

- Efficiency considerations for windowing
  - Prevent re-derivation of over-general clauses
  - Ignore positive incompressible examples

Increase in Constraints



Positive Examples Ignored



Discussion

- Conditional generalization plays important role in windowing (at least for these cases !)
- Trade off between
  - No. of constraints (additive cost as constraints added)
  - No. of times unacceptable clause is re-obtained
- Depends on window size

Pre-dominant Drawback

- Lack of directions for selecting a sample size best suited for problem
- Methods adopted here domain independent

Conclusions

- Potential utility of sampling methods
- Shortcomings
  - Results on only two domains
  - KRK too simple
  - Variants and alternatives to windowing not explored
  - Noise-tolerance of windowing not studied
  - No theory proposed for windowing
  - Compared only with large unstructured datasets

References

- A study of two sampling methods for analyzing large datasets for ILP

  Ashwin Srinivasan, Oxford University Computing Laboratory, Oxford, UK

A study of two sampling methods for analyzing large datasets with **ILP**

By: Ashwin Srinivasan

Presented by: Anand Sivaraman

Comments by: Sachin Sharma

Noise Problem in Windowing

Basic Idea :

Good Rule→ misclassifies noisy examples
→noisy examples added to window
→noise in learning window >*DATA*

Example….

- total examples = 11000 *with 10% noise*
- "Correct theory" derived using 1000 examples
- *next :* around 1000 examples will be misclassified with this theory
- Next window will have > 50% noise

Solution…..

```
Noise-Tolerant(Examples, InitSize, MaxIncSize)
Train = RandomSample(Examples, InitSize)
Theory = Φ
Repeat
   NewTheory = findNewTheory(Train)
   for Rule in NewTheory
        EvaluateRule(Examples)
        NewTr = Train
        NewEx = Examples
```

```
        Candidates = Φ
        if(Significant(Rule, Examples)
                Theory = Theory + Rule
                NewTr = NewTr – Cover(Rule,Train)
                NewEx = NewEx – Cover(Rule,Examples)
        else
               Candidates = Candidates + Cover(Rule,Examples)
     for each Positive "example"  not 'Covered' by NewTheory
        Candidates = Candidates + "example"
   Examples = NewEx + Candidates
   Train = NewTr + RandomSample(Example,MaxIncSize)
Until  Candidates =  Φ
```

A study of two sampling methods for analysing large datasets with ILP

By

ASWIN SRINIVASAN
oxford university computing laboratory

Covering Procedure

generalise$(B, I, \mathcal{L}, E)$: Given background knowledge B, hypothesis constraints I, a finite training set $E = E^+ \cup E^-$, returns a hypothesis H in $\mathcal{L}$ such that H explains the E.

1. $i = 0$
2. $E_i^+ = E^+$, $H_i = \emptyset$
3. if $E_i^+ = \emptyset$ return H_i otherwise continue
4. increment i
5. $Train_i = E_{i-1}^+ \cup E^-$
6. Let $e_i{}^+ \in E_{i-1}^+$, D_i be the "best" definite clause in $\mathcal{L}$ s.t. $B \cup H_{i-1} \cup \{D_i\} \models \{e_i^+\}$, $B \cup H_{i-1} \cup \{D_i\} \not\models \Box$, $B \cup H_{i-1} \cup \{D_i\} \cup E^- \not\models \Box$, and $B \cup H_{i-1} \cup \{D_i\} \cup I \not\models \Box$
7. $H_i = H_{i-1} \cup \{D_i\}$
8. $E_p = \{e_p : e_p \in E_{i-1}^+ \text{ s.t. } B \cup H_i \models \{e_p\}\}$.
9. $E_i^+ = E_{i-1}^+ \setminus E_p$
10. Go to Step 3

Partial Correctness

Show an explanation H that has sufficient properties

1) for each D_i in H, $B \cup \{D_i\}$ $e_1 \vee e_2 \vee e_3 \ldots$, where $\{e_1, e_2, \ldots\} \subseteq E+$

2) $B \cup H$ entails E^+

• Prove property 1:

From step 6 the above property is satisfied by all clauses

• Prove property 2: By invariant method

C_1 - before commencing step 1

C_2 - before going around the loop

C_3 - on termination

• Assertions for each check point

A_1 - input B, I, L, E is legal

A_2 - $B \cup H_i$ entails $E^+ \setminus FN_i$

A_3 - $B \cup H_i$ entails E^+

Each time path $(Ci) \rightarrow (Cj)$ traversed

we have to show if Ai is true Aj is true

Partial Correctness

• $(C1) \rightarrow (C2)$

$H_i = H_0 = \phi$ and $E_i^+ = E_0^+ = E^+$ thus $E^+ \setminus E_i^+ = \phi$

A_2 trivially holds

• $(C2) \rightarrow (C3)$

if $A_2 : B \cup H_i$ entails $E^+ \setminus E_i^+$ is true and $E_i^+ = \phi$ then

$A_3 : B \cup H_i$ entails E^+

• $(C2) \rightarrow (C2)$

On iteration i if $A2 = A_2^i : B \cup H_i$ entails $E^+ \setminus E_i^+$ is true

we have to show on iteration $i+1$ $A_2^{i+1} : B \cup H_i$ entails $E^+ \setminus E_{i+1}^+$ also holds