Democratic Approximation of Lexicographic Preference Models

Fusun Yaman, Thomas J. Walsh,
Michael L. Littman, Marie desJardins

Presented at The 25th International Conference on Machine Learning (ICML 2008), Helsinki, Finland

Preference Learning

- Suppose we know some features of flights and some examples of how a customer chose their flights.
  - Good Food
  - Good Price
  - Few Cancelations
  - Good Service
  - Data Access

0	Chosen over	1
1		1
1		0
0		1
0		1

- Can we determine which of these flights the same customer will choose?

0	1	1	0	1

Versus

1	0	0	1	0

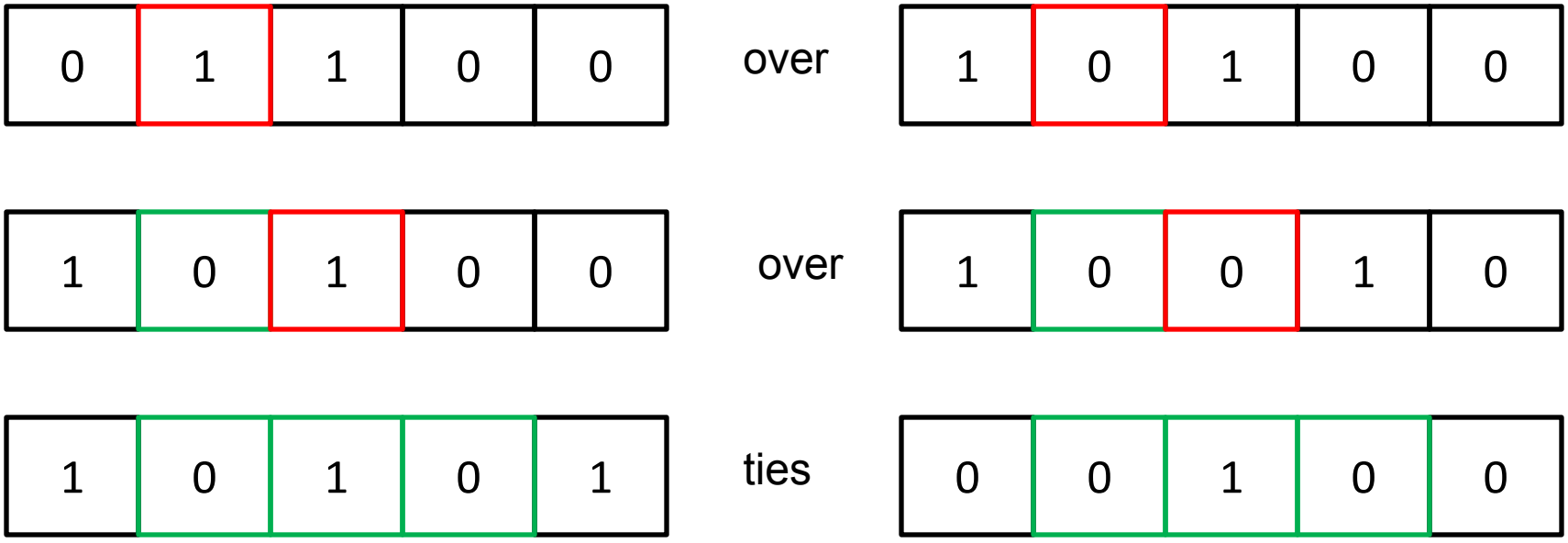
Lexicographic Preference Models

- LPMs are a basic preference model
  - *Few Cancelations* over *Lots of Cancelations* (always)
  - *Good Price* over *Bad Price* (within "Few Cancelations" group)
- Formally, given objects described by n variables $[x_1 \ldots x_n]$, an LPM L on X is a *total order* $\sqsubseteq_L$ on a *subset* $R \subseteq X$
  - R = *relevant variables*, I = X – R = *irrelevant variables*
- Used in many research areas, including Multicriteria Optimization (Bertsekas & Tsitsiklis, 97), Linear Programming, and Game Theory (Quesada, 03)

LPM Decision Example

- Given 2 Objects, A and B and an LPM L
  - Find the smallest variable, X* in $\sqsubseteq_L$ Such that X* has different values in A and B and choose the one where X*=1, otherwise there is a tie.
- From our flight example, suppose we have

LPM L = X2 < X3 < X4 (price < cancels < service)

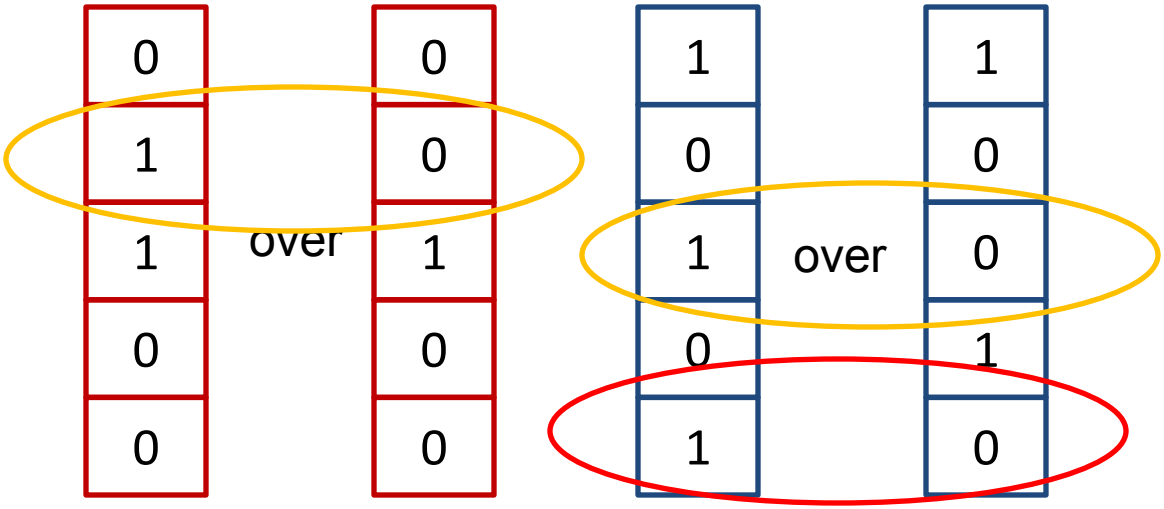


LPM Learning

- Given
  - Set of observations O, where o_i = (A, B), for A= [x_1…x_n] (also B=[x_1...x_n]), where A is preferred over B.
- Find
  - An LPM L that is consistent with O.
    - L implies A is preferred over B or that they are equally preferred.
- Finding a consistent LPM was studied in previous work…

State of The Art

- The Greedy Approach (Schmitt and Martignon, 2006)
  - Iteratively construct L, choose a variable that creates the fewest conflicts with the observed data (break ties randomly).
- Drawback: This algorithm only gives us 1 of *many possible* consistent LPMs.
  - The bias in the random selection will greatly impact predictions performance.



{X2, X3}

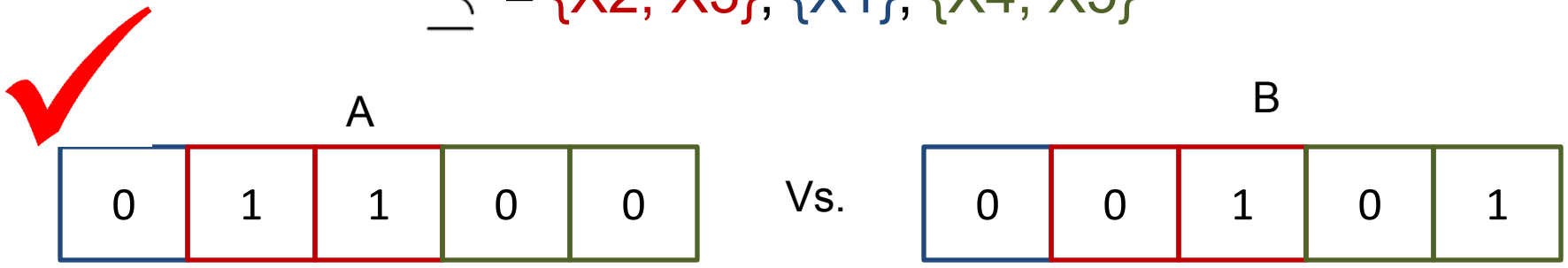
{X2, X5, X3}?

{X1, X5, X2}?

New Approach 1: Variable Voting

- Given a partial order $\preceq$ on X, vote among the choices for "deciding" variable.
  - D = variables that differ between A and B
  - D* = variables in D with smallest rank according to $\preceq$
  - Each variable in D* has one vote.

$\preceq$ = {X2, X3}, {X1}, {X4, X5}



D = X2, X5

D* = X2

X2=1 for A, so A is preferred over B.

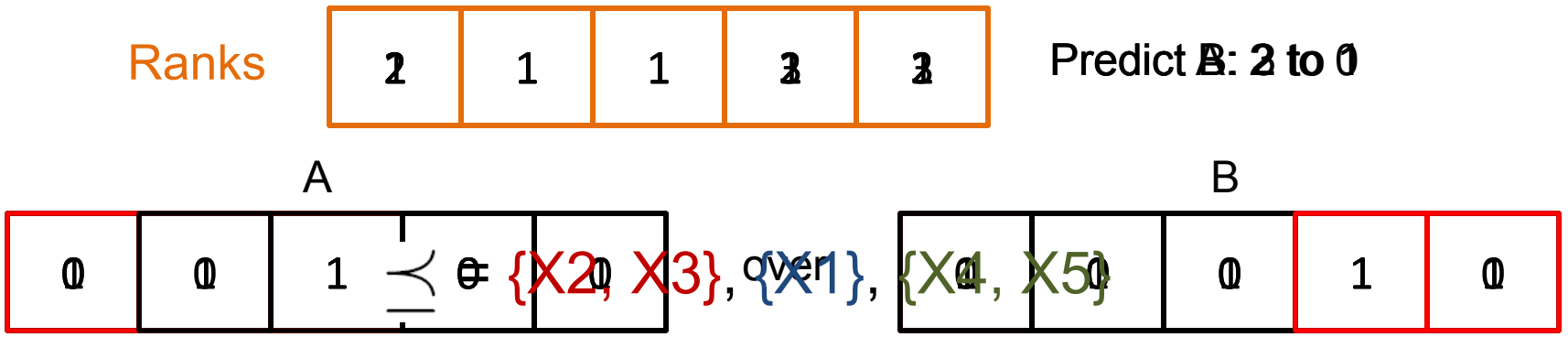
Learning Variable Ranks

- Algorithm *LearnVariableRank* (*LVR*)

  Initially all variables have rank 1 *(most important)*

  While the ranks can change based on the observations

  - **For each** observation, predict a preferred object.
    - **If** the prediction is wrong, increment the ranks of all the variables that predicted incorrectly.



Variable Vote Properties

- Correctness
  - All $\sqsubseteq$ that are a topological sort of $\succ$ from *LVR* are consistent with O. *LVR* never increments the ranks of relevant variables beyond their actual rank.
- Convergence
  - Mistake bound of $O(n^2)$. If all variables are relevant, *LVR* converges to the true LPM. With irrelevant variables, it converges to a correct ordering of the relevant variables.
- Time Complexity
  - $O(n^3m)$ (compared to $O(n^2m)$ for Greedy)

New Approach 2: Model Voting

- A Bayesian Approach – look at a sample of consistent LPMs and let each one vote (equally weighted)
- For a set of LPMs S (consistent with O), choose A over B if

$$\sum P(L|S) * V_{A>B}^{L} \; > \; \sum P(L|S) * V_{B>A}^{L}$$

- Aggregate LPMs (X3, X2, *) allow us to compactly represent sets of models.

Generating the Models

- Algorith *SampleModels*

  Candidates = {variables Y | Y $\notin$ rulePrefix, $\forall$ (A, B) $\in$ O, A(Y) = 1 or A(Y) = B(Y)

  While *Candidates* $\neq \emptyset$

    If *O* = $\emptyset$ then **return** (***rulePrefix***, *)

    Randomly choose variable ***Z*** from ***Candidates***

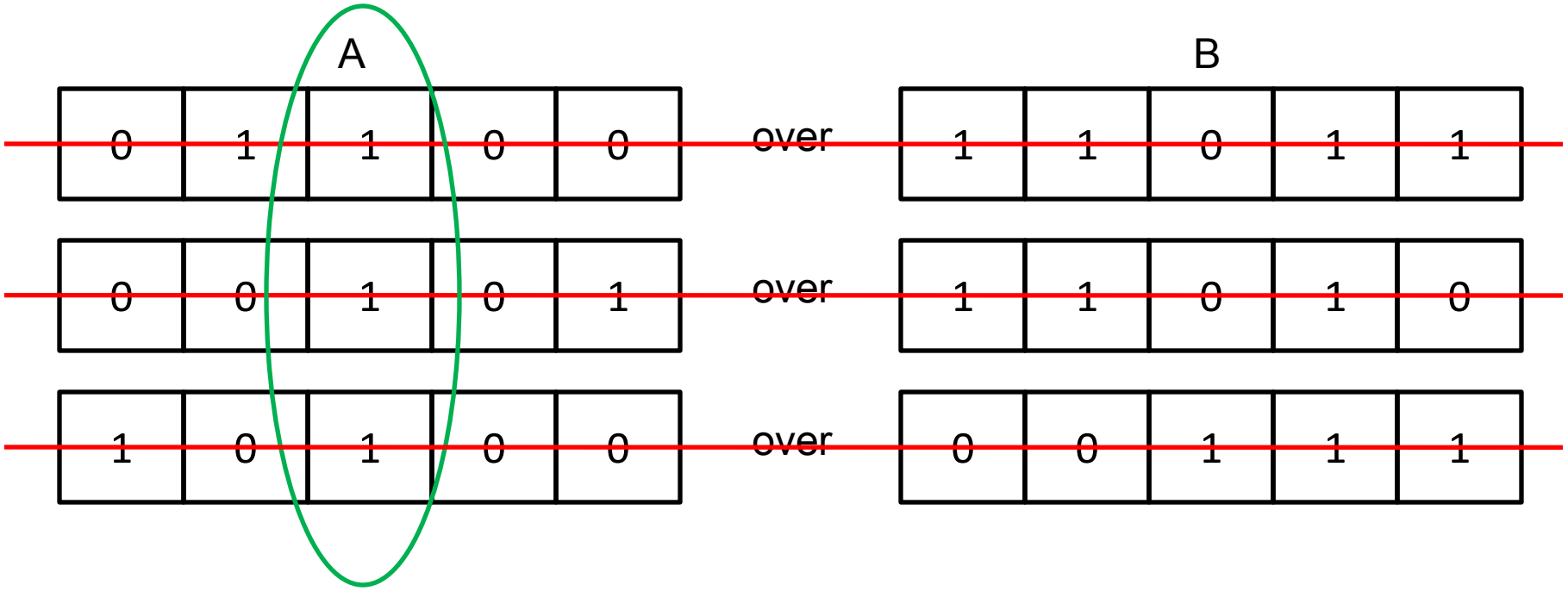
    Remove any ***(C, D)*** $\in$ ***O*** where ***C(Z)*** $\neq$ ***D(Z)***

    rulePrefix = (***rulePrefix***, ***Z***)

    Recompute ***Candidates***

  return *rulePrefix*

Example Model Generation



Candidates = {X3, X2}

model = (X3, X2, X1, *)

Model Vote

Given a set of samples S and objects A and B

SampSize = total number of models represented in S
(computation is linear in n and |S|)

For each L ∈ S

If L is non-aggregate

Vote(winner) += 1

Else If L is aggregate and one of the prefix variables determines a winner

Vote(winner) += number of extensions

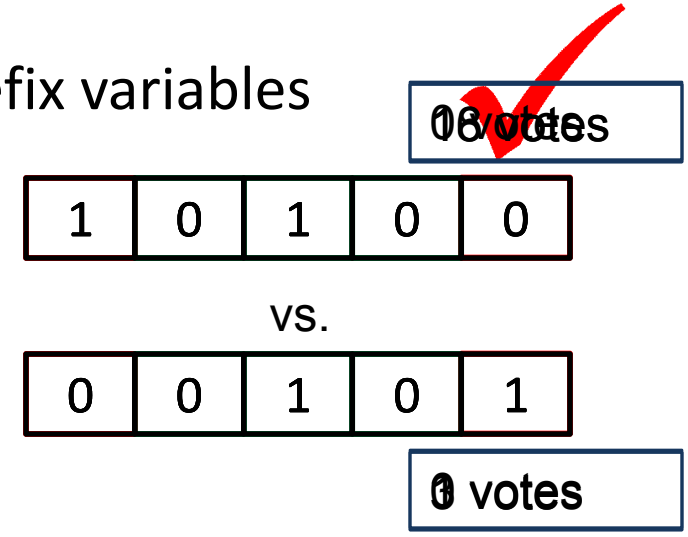
Else (L is aggregate, split decision)

Vote(A) += $N_{A>B}^{L}$ (linear time)

Vote(B) += $N_{B>A}^{L}$ (linear time)

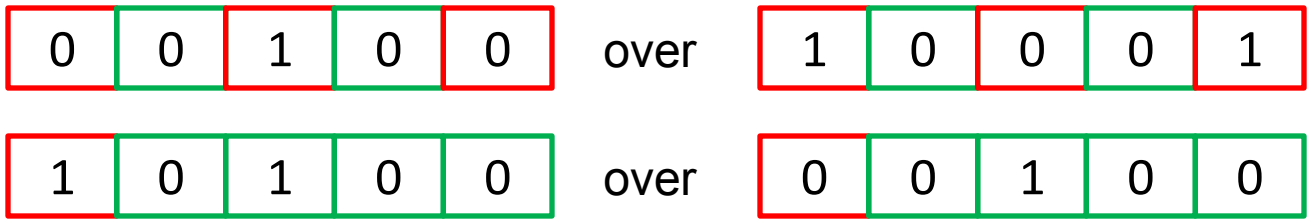
Return the object with the most votes

model = (X3, X2, X5, X1, X4)



Model Vote Properties

- Complexity
  - *SampleModels*: $O(n^2m)$
  - With s calls to *SampleModels*, *ModelVote* takes O(sn) time.
    - All the numerical computations are linear with proper caching.
- Comparison to *Variable Voting*
  - *Model Vote* considers LPMs *Variable Vote* cannot, such as (X3, X1, *) to explain the following.



    - VV never considers X1 < X2 because X2 is never wrong.
    - It seems like this would be helpful with irrelevant variables…

Introducing Bias

- We can limit the number of models considered by using a *bias* based on *domain knowledge.*
- When buying a used car, people consider
  - {*mileage, year, make*} before
  - {*color, #doors, body type*} before
  - {*interior color, cup holders*}
- A learning bias B is a total order on a *partition* of the variables.

  {X1, X2, X3} < {X4, X5} < {X6, X7, X8, X9, X10}

- 9 variables → 905,970 LPMs
  - 3 partitions of 3 variables → 646 LPMs

Voting Algorithms and Bias

- Variable Vote can easily be extended to use bias by initializing the variables based on the bias.

{X1, X2, X3} < {X4, X5}

1	1	1	4	4

- Model Vote can use a bias by
  - Change *SampleModels* to only produce bias-consistent models.
  - For aggregate LPMs, compute the number of *bias-consistent* extensions that vote for each object.
    - This can still be computed efficiently (details in the paper).

Experimental Setup

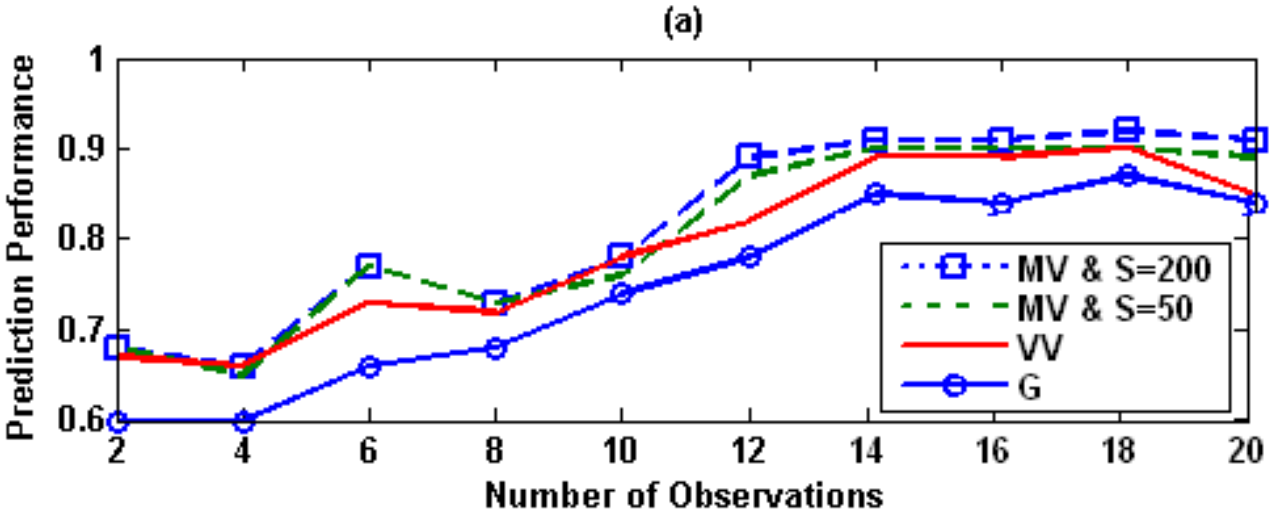
- Prediction Performance on Test Set *T*

Performance(T) = (Correct(T) + .5 * Tie(T)) / |T|

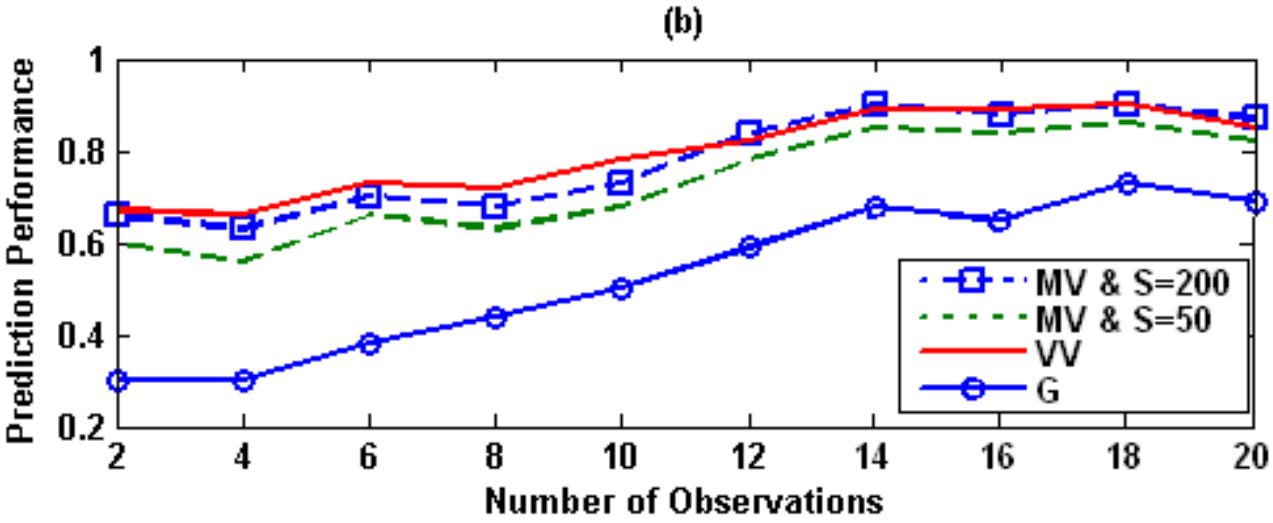
- Concerned with Average and worst results for each algorithm on the test sets.
- Mitigating randomness
  - Ran *Greedy* 200 times for every (O, T) pair
  - Ran *MV* 10 times for each (O, T) pair, varied number of calls to sample models (S=50, S=200)

Empirical Comparison

Average Case



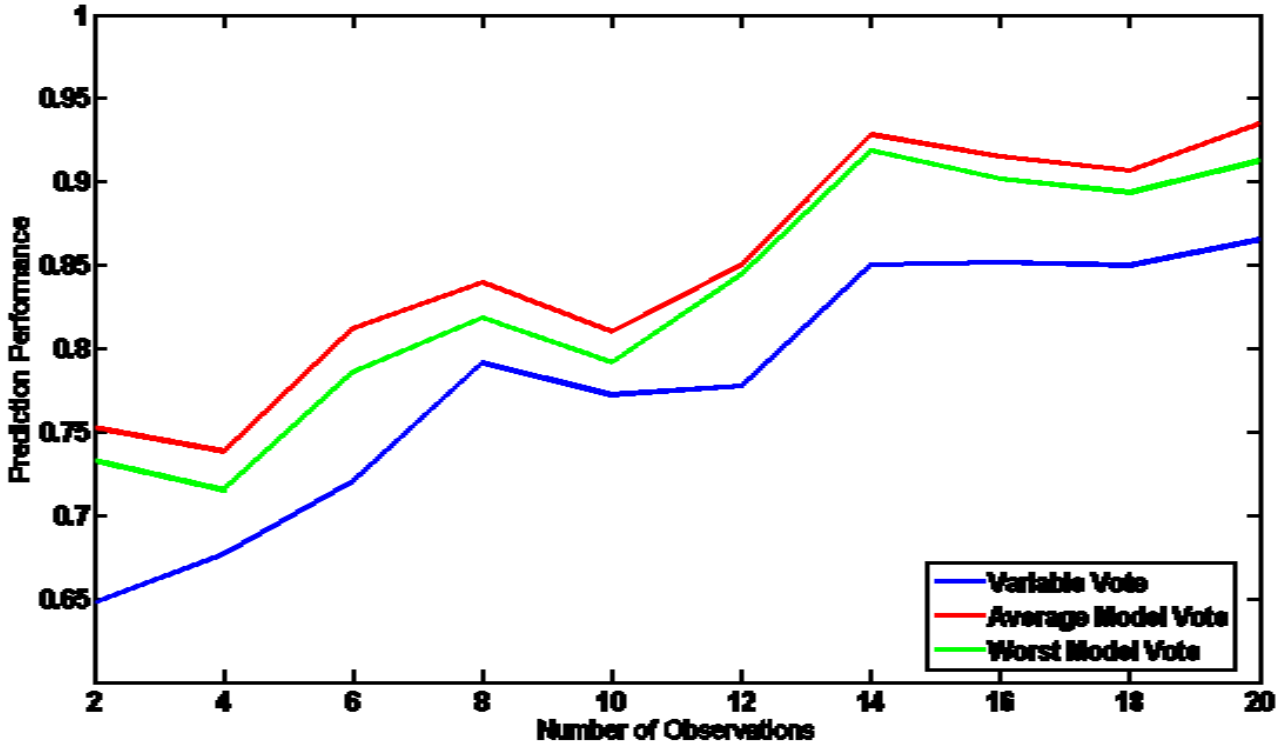
Worst Case



|R| = 15, |I| = 0, |T| = 20, data points average over 20 sets

Model Voting and Irrelevant Variables

- As expected, MV outperforms VV in the presence of irrelevant variables



|R| = 5, |I| = 10, |T| = 20, data points average over 20 sets

May want to mention G worst case

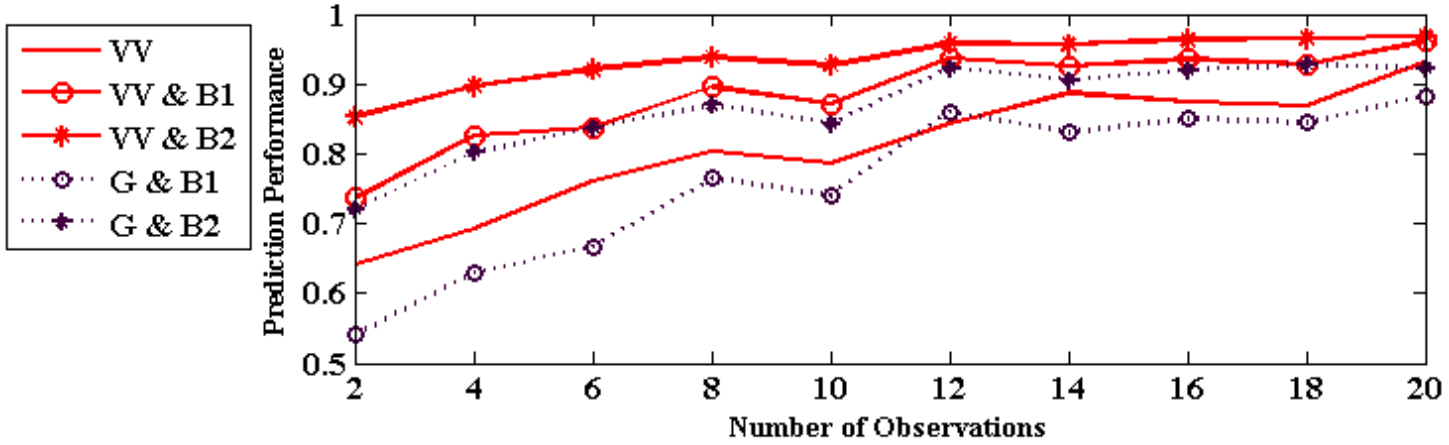
Bias Experiments

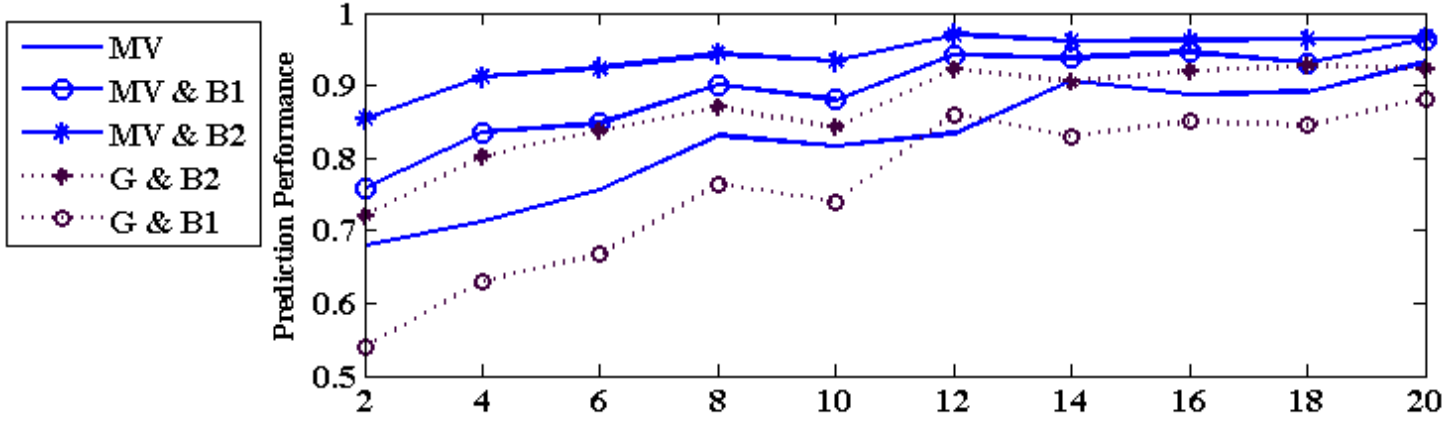
B1 = {X1, X2, X3, X4, X5} < {X6, X7, X8, X9, X10}

B2 = {X1, X2, X3} < {X4, X5} < {X6, X7, X8, X9, X10}

|R| = 15, |I|

|T| = 20





Conclusions & Future Work

- We developed algorithms that learn preference models, but do not commit to a single model.
  - Instead they predict based on a majority of the voting models
  - Our methods are theoretically sound and computationally efficient.
  - Out voting methods outperform the state of the art Greedy method in empirical studies with modest increases in computation.
- Future directions
  - Learning preferred values of variables.
  - Democratic approximation techniques for other types of preference models.