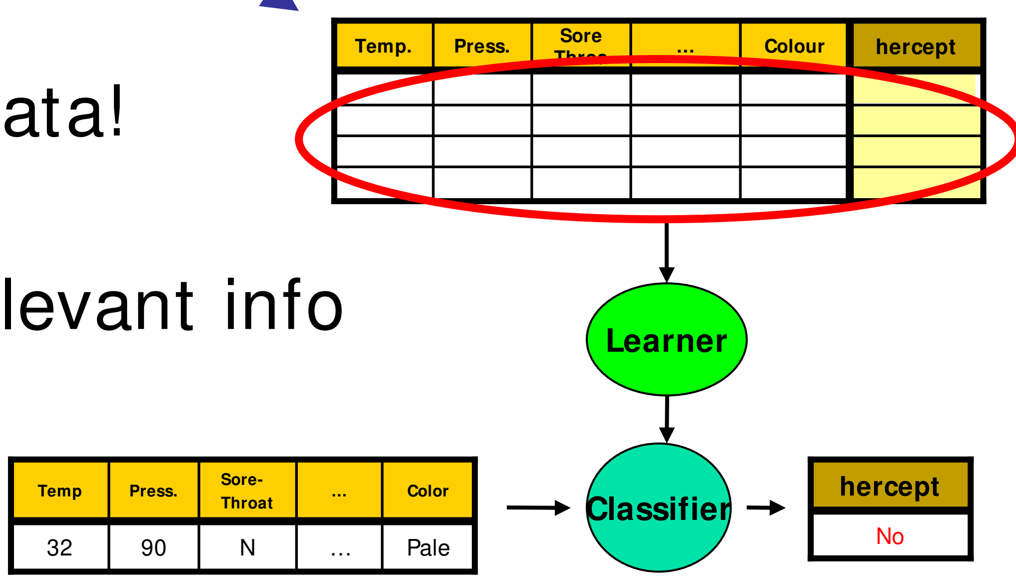
Budgeted Learning of Naïve Bayes Classifiers

Russ Greiner
w/ Omid Madani, Dan Lizotte, Aloak Kapoor
Alberta Ingenuity Centre for Machine Learning
Computing Science Department
University of Alberta

(UAI'03; UAI'04; COLT'04; ECLM'05, UBDM'05)

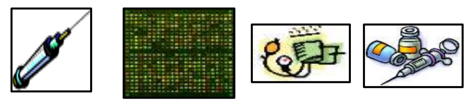
Challenge

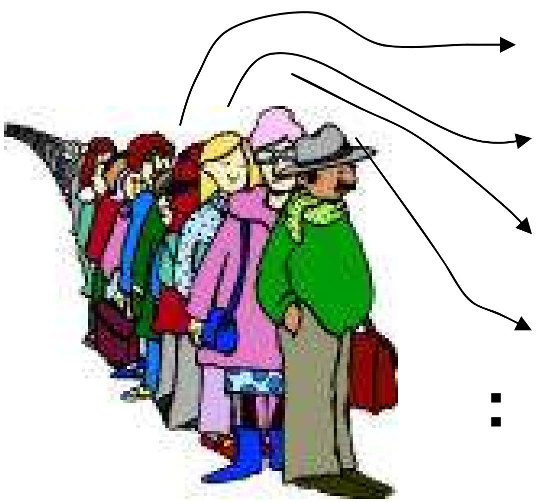
- Machine Learning Challenge
  - Build CLASSIFIER:
    Will patient respond well to Herceptin?
  - based on training data
- But…
  - Start of study… no data!
  - Instead…
    have $$ to gather relevant info



Need Training Data !

- … that learner can use to build good classifier
- Run *Clinical Trials*

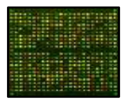


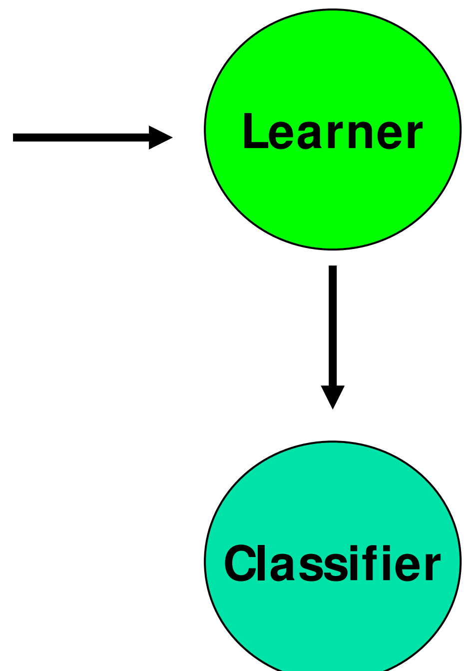


				Response
b	0	5	b	1
b	1	3	a	0
a	1	1	a	0
b	1	1	a	0
a	0	3	a	1

:

Typical Supervised Learning

					*Response*
Person 1	b	0	5	b	1
Person 2	b	1	3	a	0
	a	1	1	a	0
	b	1	1	a	0
	a	0	3	a	1



How to Gather Data?

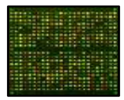
- Why run *EVERY* test on *each* training patient ?
- Unnecessary, if test results are correlated
- Inefficient, as tests are EXPENSIVE!

… especially given **FIXED BUDGET**

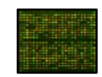
Blood-Factors	Gender	Pulse-Rate	Age	Blood Pressure	Height	Weight	Micro-Array
$5	0.00	0.02	0.01	0.50	0.05	0.05	$95

- General problem
  - Given Costs of tests, Total fixed budget:
  - Decide *which tests* to run on *which patients*
    to obtain info needed to produce effective classifier

Budgeted Learning

					*Response*
Person 1	?	?	?	?	1
Person 2	?	?	?	?	0
	?	?	?	?	0
	?	?	?	?	0
⋮	?	?	?	?	1

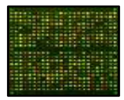
Costs

-  $ 5.00
-  $50.00
-  $ 0.50
-  $19.75

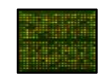
Total Budget: $100

Budgeted Learning

Remaining Budget:
~~$100~~ ~~$95~~ ~~$90~~ … $0

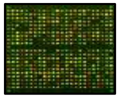
					*Response*
Person 1	b	0	0		1
Person 2	d			a	0
⋮				c	0
					0
					1

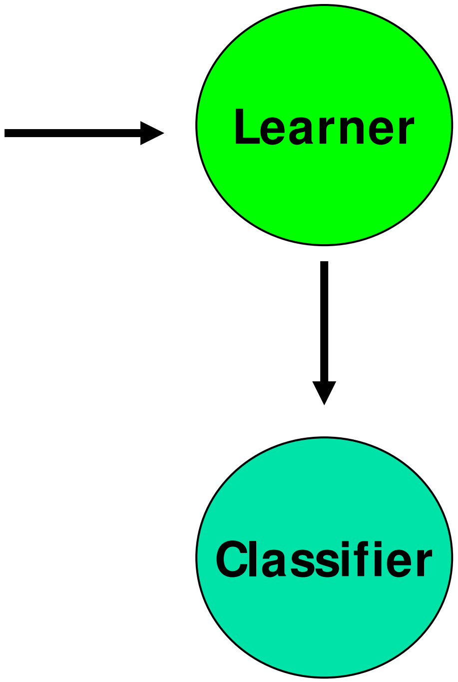
Costs

-  $ 5.00
-  $50.00
-  $ 0.50
-  $19.75

Total Budget:
$100

Budgeted Learning

					*Response*
Person 1	b	0	0		1
Person 2	d			a	0
⋮				c	0
					0
					1



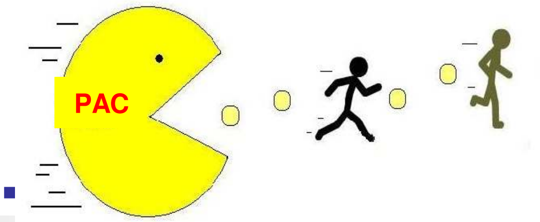
Querying Strategy

- *A Querying Strategy*
  - specifies when to test
    - which feature for
    - which individual

    subject to spending at most budget, *b*
  - *Returns a classifier with*
    highest (posterior) expected accuracy

- Goal: Optimal *Querying Strategy*
  - "typically" identifies classifier with high expected accuracy
  - … minimizes Expected Regret

Related Work: PAC, …



- Computational learning theory:
  - Find m = m(… ε, δ, …), given ε, δ
    - Asymptotic, constants hidden
  - *Full training* instance

5	+	0	a

- Budgeted Learning:
  - Firm budget … m=63
  - *Individual* feature queries

	+		

What Budget Learning isn't…

- Budgeted Learning

  Train **(fixed size)** **Test**

- Standard Learning

  Train **(varying size)** **Test**

- On-line Learning

  Train + Test

- Exper. Design (I)

  Train + Test

Related Work: Active Learning

- ~~Budgeted Learning~~
- Active Learning

f_1	f_2	f_3	f_4	Class
b	0	5	b	?
b	1	3	a	?
a	1	1	a	?
b	1	1	a	?
a	0	3	a	?

BudgetLearning = MDP

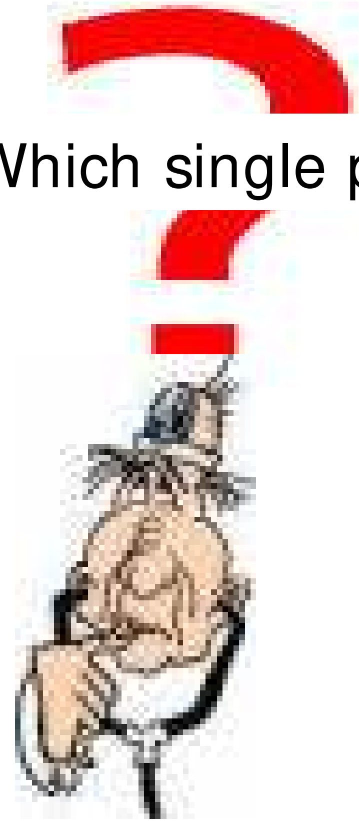
- Budgeted Learning is a
  Depth-limited Markov decision process
  - State = current distribution
  - Action = specific ⟨instance, feature⟩ probe
  - Reward = 0, except final state: quality
- But
  - State space is exponential
  - … ≈ POMDP
- ?? Special purpose algorithm here??

Talk Overview

- Motivation
- Active Model Selection
  (≈multi-armed bandit scenario)
  - Bayesian Framework
  - Hardness
  - Algorithms
  - Empirical comparisons
  - Theoretical Results
- Naïve Bayes models
- Learn & Classify under Hard Constraints
- Conclusions

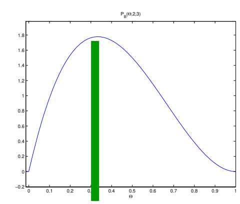
Which treatment works best, ***unconditionally***?

Which single pill?

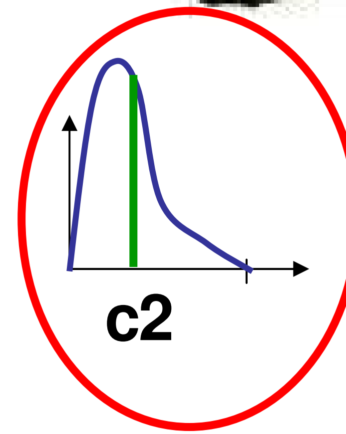


Active Model Selection: Budgeted Coins Problem

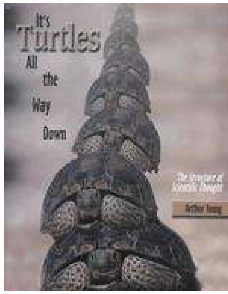
- Input:
  - *n independent* coins
    For each coin:
    - Prior over head probability Θ_i
    - Tossing cost r_i
  - Total budget b
- After several flips (total cost: $\sum_i r_i \leq b$)
  choose <u>a single</u> coin c^* for future tosses
- Measure of coin performance:
  (expected) head probability of c^*
- Measure of strategy: expected regret …

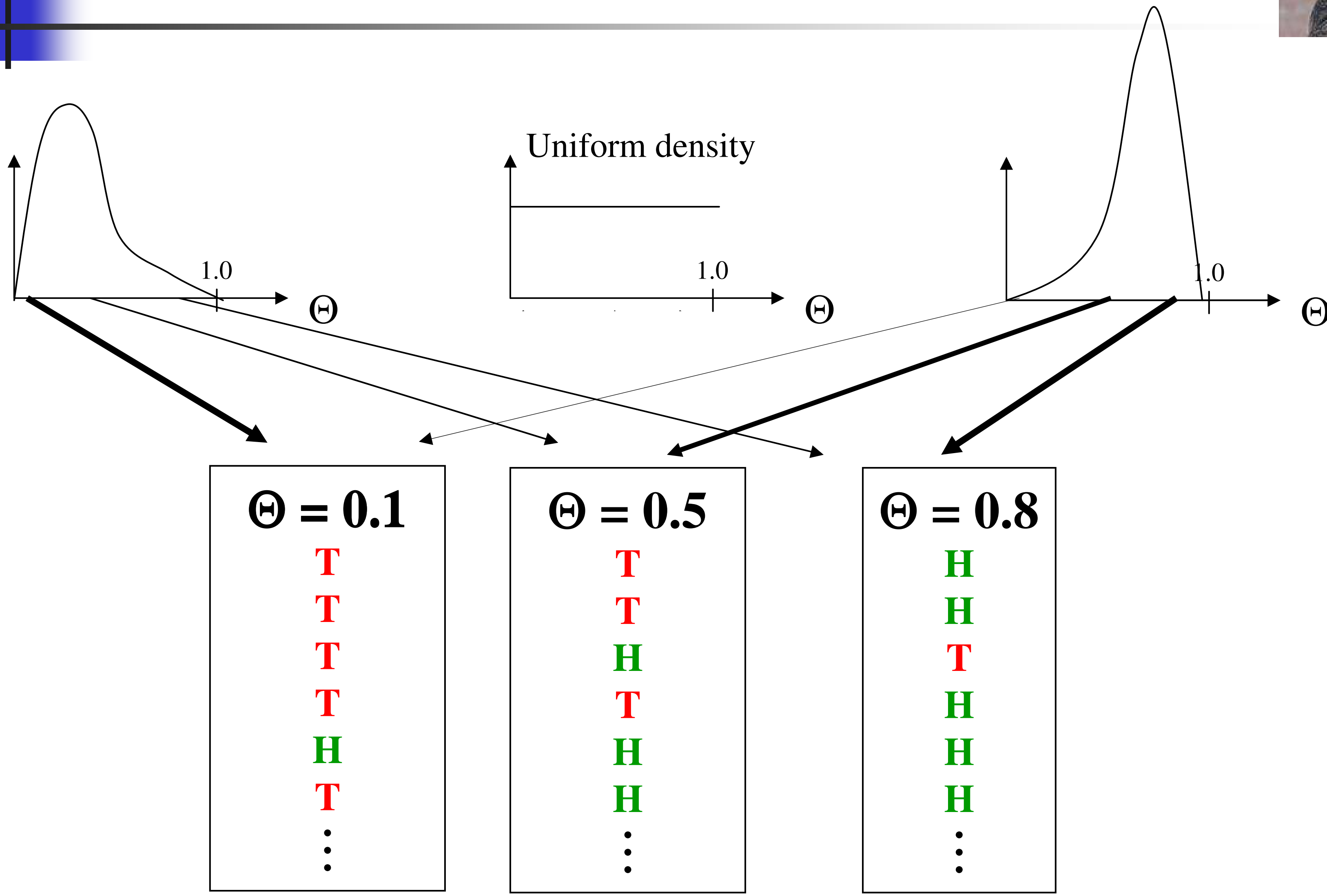


c1



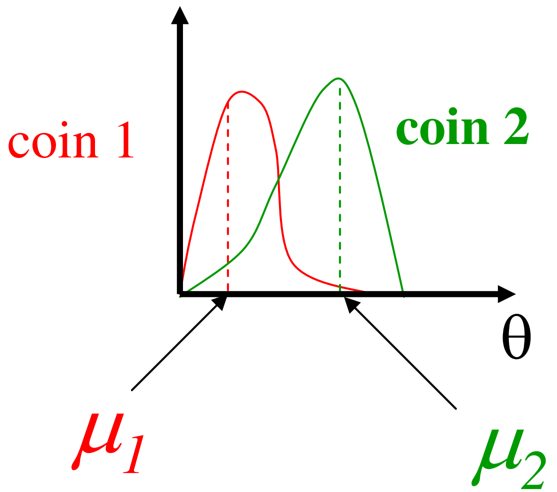
Two (related) Distributions: Parameter, Instances





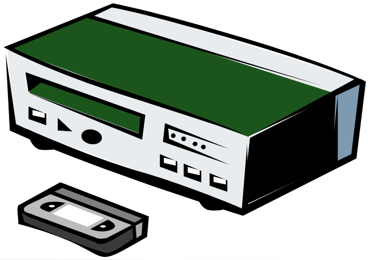
Maximizing Expected Mean

- Two coins, Θ_1 and Θ_2
  each with own distribution



- Which coin should we pick?
- Compute mean, $\mu_i = E(\Theta_i)$
- As $\mu_2 > \mu_1$, we should pick *coin* 2.

Beta Distributions



- Coin ~ Beta(a,b)

$$\text{Expected head probability} = \frac{a}{a+b}$$

$$\text{Expected tail probability} = \frac{b}{a+b}$$

- Dynamics and updates:

Tossing a coin with

Beta(3, 7)

probability of heads

$h, \quad \frac{3}{3+7}$

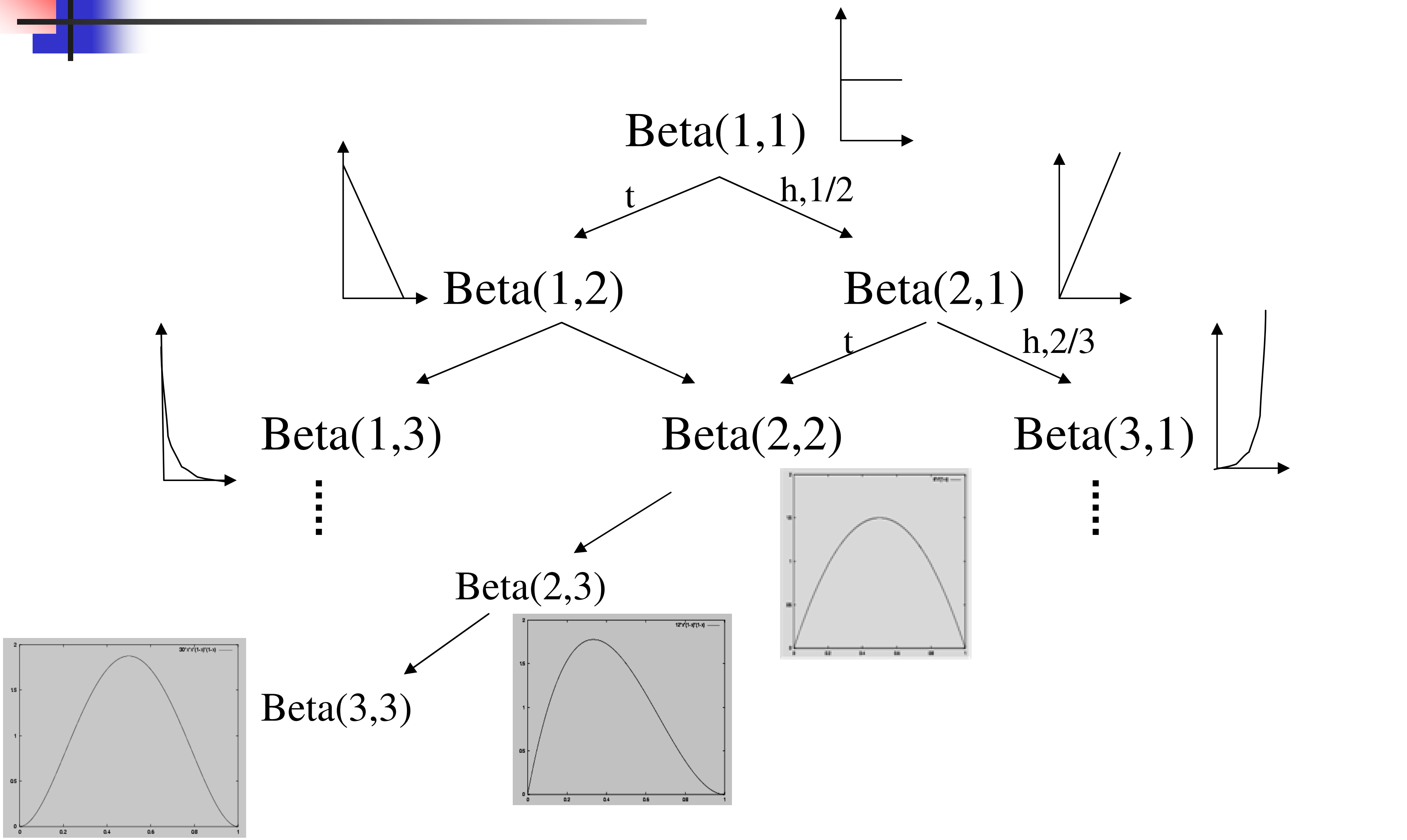
$t, \quad \frac{7}{3+7}$

posterior

Beta(4, 7)

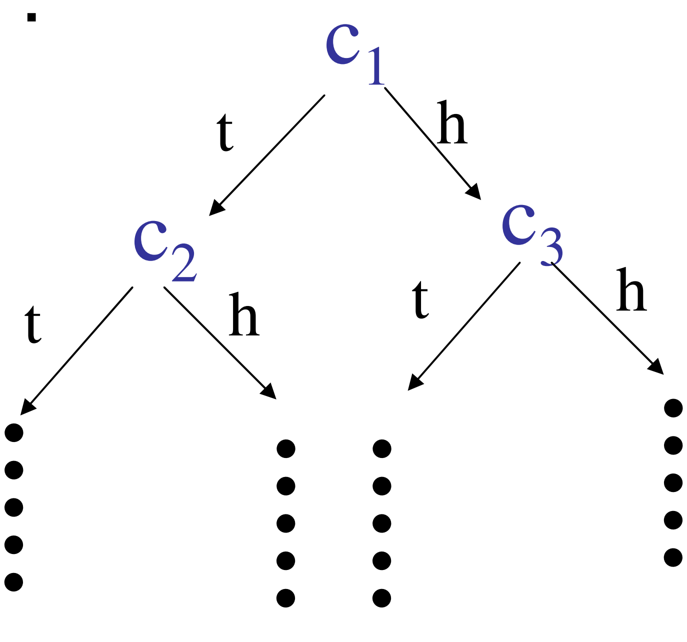
Beta(3, 8)

Example


Beta(1,1)
t
h,1/2
Beta(1,2)
Beta(2,1)
t
h,2/3
Beta(1,3)
Beta(2,2)
Beta(3,1)
Beta(2,3)
Beta(3,3)

Strategies

- Strategy ≡ Prescription of
  - which coin to toss at each time
- *Strategy tree* :

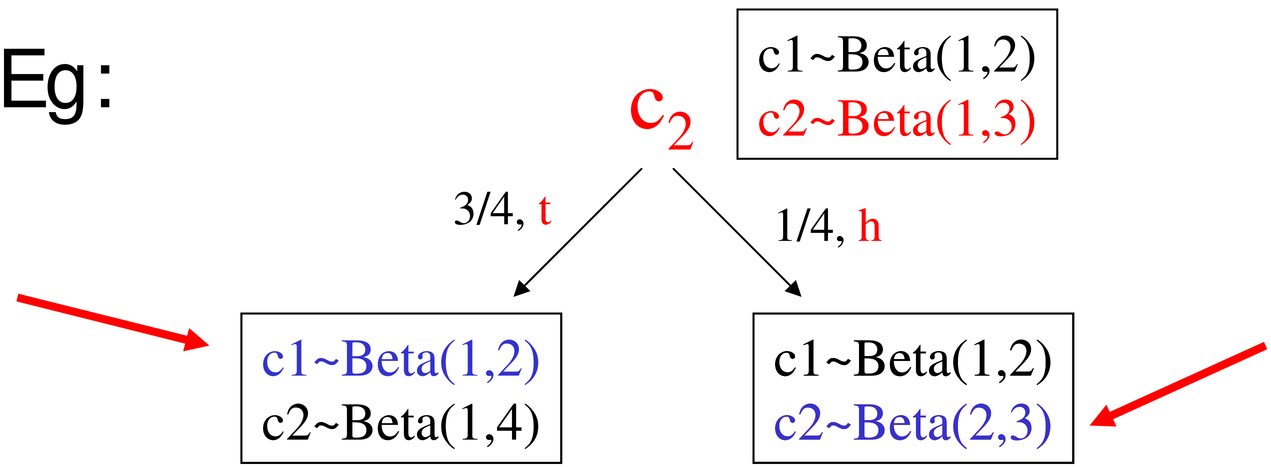


Quality of a Strategy

- *Expected Mean* of a **strategy**:

$$\sum_{\text{leaf } i} \Pr(\text{ reach leaf i }) \times \left(\text{mean returned at leaf i}\right)$$

- Eg:



$$\frac{3}{4}[\quad \frac{1}{3} \quad] + \frac{1}{4}[\quad \frac{2}{5} \quad] = \frac{21}{60}$$

Example Scenario

This is Lookahead of 1

- Two coins: **c1: Beta(1,2)** **c2: Beta(1,3)**
- Budget of 1… which to toss?

c1

c1: Beta(1,3)	c1: Beta(2,2)
c2: Beta(1,3)	c2: Beta(1,3)

Expected Mean

$$=\frac{2}{3}\times\frac{1}{4}+\frac{1}{3}\times\frac{2}{4}=\frac{20}{60}$$

c2

c1: Beta(1,2)	c1: Beta(1,2)
c2: Beta(1,4)	c2: Beta(2,3)

Expected Mean

$$=\frac{3}{4}\times\frac{1}{3}+\frac{1}{4}\times\frac{2}{5}=\frac{21}{60}$$

➡ **Toss c2 !**

Related Work (II): Bandit Problems

- Multi-armed Bandit Problems
  - Berry&Fristedt, *Bandit Problems: Sequential Allocation of Experiments*. 1985
  - On-line
  - Exploitation versus Exploration tradeoff
- AMS:
  - During training: only *Exploration*
  - Reward: function of final state

- (Std) Bandit Problem

  Train + **Test**

- AMS

  Train (fixed size)

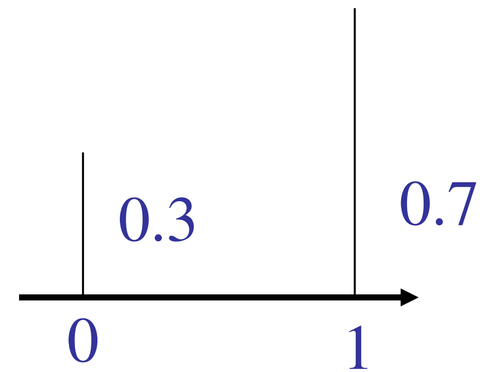
  Test

Talk Overview

- Motivation
- Active Model Selection
  (≈multi-armed bandit scenario)
  - Bayesian Framework
  - Hardness
  - Algorithms
  - Empirical comparisons
  - Theoretical Results
- Naïve Bayes models
  (learning classifiers)
- Learn & Classify under Hard Constraints
- Conclusions

Complexity Results

- Obvious Dynamic Program: $O(\, b^k \,)$
  - If (fixed) k coins: Poly-time !
- AMS is in PSPACE
- AMS is NP-Hard:
  - Under *non-identical* coin costs
  - Proof: Using *bi-modal* coin priors:
    - Knapsack reduces to AMS
    - Maximize profit = Maximize "success" probability



- If costs are *identical* + priors *uni-modal*…

Unknown…

Intuitions

- In general... (identical costs)
  toss coin c_i if this toss has a
  fair chance of improving max'm mean,
  given budget
- Typically, this means ...
  - c_i's mean is *high* *and/or*
  - c_i's *variance is high* (few trials so far)
    $\Rightarrow$ easy to "move distribution"

- But <u>exceptions</u> exist ...

Example Scenario

Even though c1 has
- higher prior
- higher variance !

- Two coins: **c1: Beta(1,2)** **c2: Beta(1,3)**
- Budget of 1… which to toss?

c1

c1: Beta(1,3)
c2: Beta(1,3)

c1: Beta(2,2)
c2: Beta(1,3)

Expected Mean

$$= \frac{2}{3} \times \frac{1}{4} + \frac{1}{3} \times \frac{2}{4} = \frac{20}{60}$$

c2

c1: Beta(1,2)
c2: Beta(1,4)

c1: Beta(1,2)
c2: Beta(2,3)

Expected Mean

$$= \frac{3}{4} \times \frac{1}{3} + \frac{1}{4} \times \frac{2}{5} = \frac{21}{60}$$

⇒ **Toss c2 !**

Algorithms

1. Round-robin
2. Random
3. Greedy
4. Allocational: Single-coin look-ahead
5. Biased-robin
6. Interval Estimation
7. Gittins indices

1. Round-Robin

c1	c2	c3	c4	c5
-	+	+	+	-
+	+	+	-	-

2. Random

c1	c2	c3	c4	c5
-	+	+	+	-
+	+	-		-

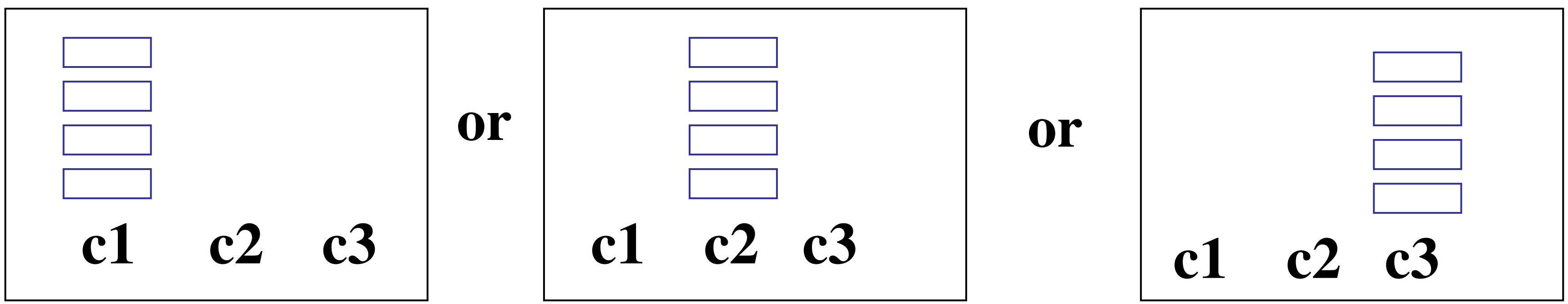
3. Greedy

- True budget b (say b= 10)
- At each time:
  - Find best action $a^{(1)}$ assuming budget is $b_{temp} = 1$
  - Perform $a^{(1)}$
  - Repeat

Lookahead 1

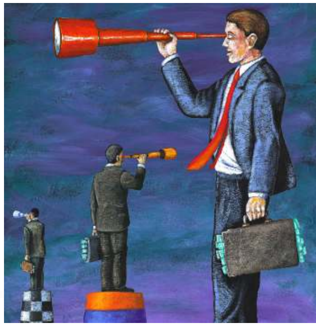
4. Single Coin Full Lookahead

- Remaining budget b=4, #coins=3. toss =
- Options…



- Decide which is best,
  - … flip that coin ONCE
- Perform this comparison at **<u>every time point</u>**!

4. Single Coin Lookahead



- For each coin i:
  - Imagine spending *entire* remaining budget b on coin# i
  - (Note: b+1 possible outcomes)
  - Calculate expected loss
- Toss coin with lowest single-coin-allocation-loss
  - ***ONCE***
- Repeat (budget now b-1)

5. Biased-Robin

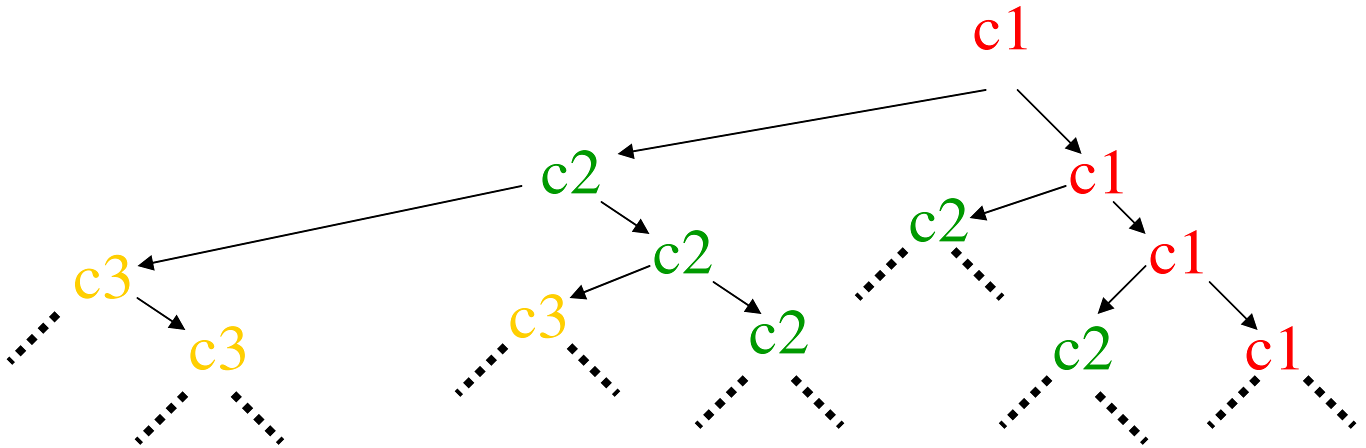
c1	c2	c3	c4	c5
+	+	+	-	+
-	+	+		-
-	+	-		
	-			

- If "+", keep using.
- If "—", go to next.

"Play the winner" … [Robbins, 52]

5. Biased-Robin

- Optimal strategy for identical priors has pattern:



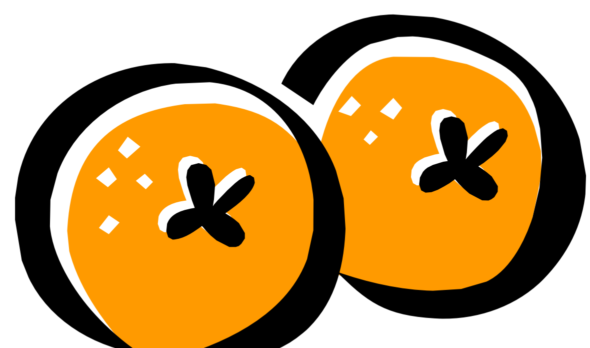
- *Biased-Robin* =

  Continue tossing same coin while it gives heads.
  If tails, go to next coin.

Skip IntEst, Gittins

Comparison of Policies

Policy	Uses data?	Uses budget?
Round Robin Random	No	No
Biased Robin	Yes	No
Greedy	Yes	No
SingleCoinLook	Yes	Yes



Talk Overview

- Motivation
- Active Model Selection
  (≈multi-armed bandit scenario)
  - Bayesian Framework
  - Hardness
  - Algorithms
  - Empirical comparisons
  - Theoretical Results
- Naïve Bayes models
  (learning classifiers)
- Learn & Classify under Hard Constraints
- Conclusions

Comparing Different Situations

- Problem: Each situation has own
  - $\Theta_{max} = max_i \ \Theta_i$
    Random variable corresponding to highest probability
  - Different runs, with different Θ_{max}'s, are *incomparable*
- Regret = $\Theta_{max} - \Theta^*$
  = difference of head prob between *best coin* c_{max} vs *chosen coin* c^*
- Always want Regret = 0

Skip Details

Example of Regret

- Chose c_2 from $\{c_1, c_2\}$
- If $\Theta_2 \geq \Theta_1$,
  - regret = 0
  - Else, regret = $\Theta_1 - \Theta_2$
- As we don't know actual probabilities, need to minimize *expected* regret

Expected Regret

- **Expected regret**, if *coin i* is chosen:

$$E(\ \Theta_{max} - \Theta_i\ ) = E(\Theta_{max}\ ) - E(\Theta_i)$$

where

- $\Theta_{max} = \ max_i\ \Theta_i$

  Random variable corresponding to highest probability

- $\mu_i = E(\Theta_i)$

  Mean of coin i

Minimum Regret = Highest Mean

- To minimize regret, pick *highest mean coin*:

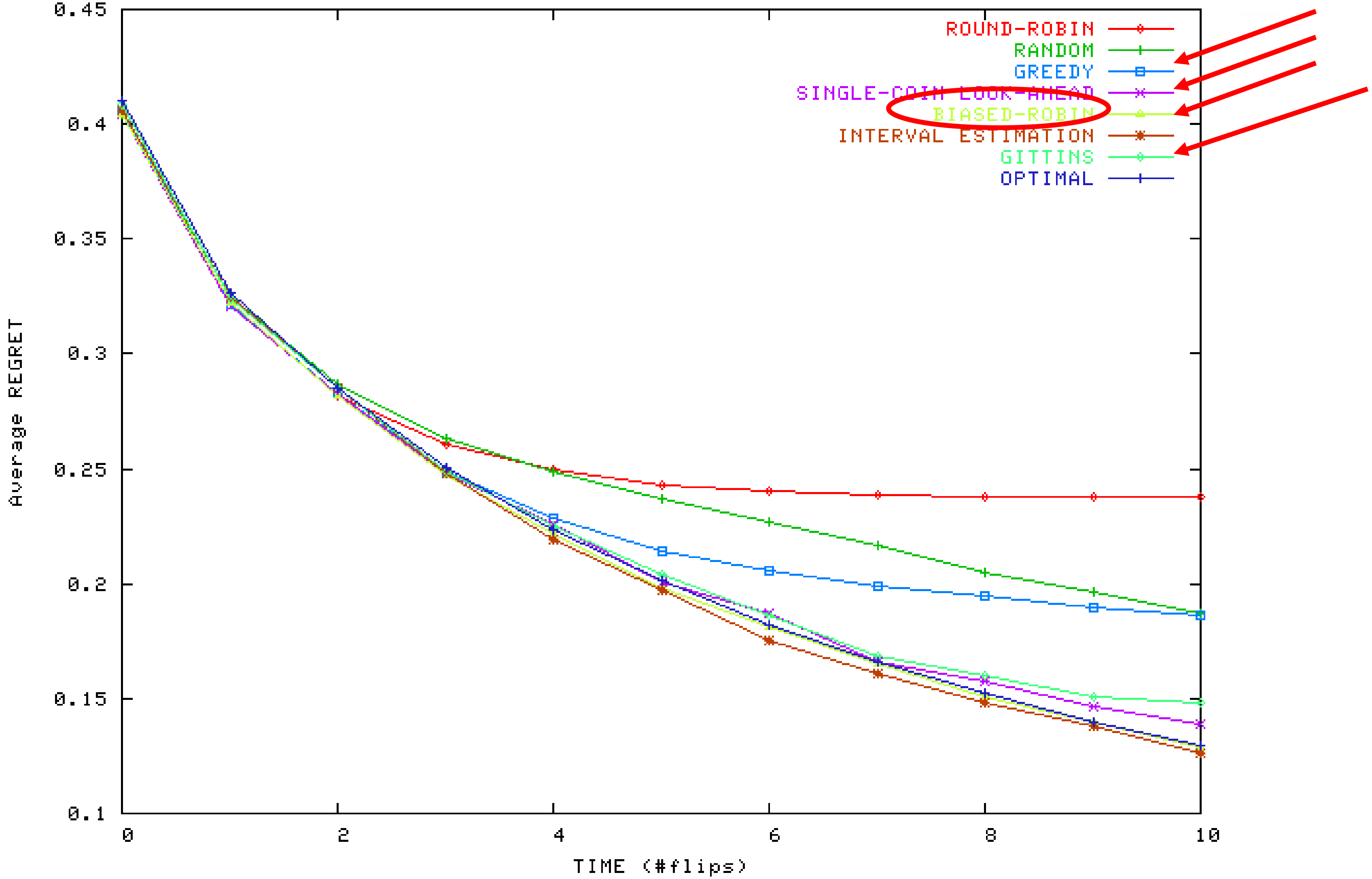
$$
\begin{aligned}
&min_i\, E(\Theta_{max} - \mu_i\,) \\
&\quad = E(\;\Theta_{max}\,) - max_i\, E(\mu_i) \\
&\quad = E(\;\Theta_{max}\,) - \mu_{max}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{E}\,(\,\Theta_{\mathrm{max}}\,) &= \mathrm{E}(\,\mathrm{max}_{\mathrm{i}}\,\Theta_{\mathrm{i}}\,) \\
\mu_{max} &= \mathrm{max}_{\mathrm{i}}\,\mathrm{E}(\,\Theta_{\mathrm{i}}\,)
\end{aligned}
$$

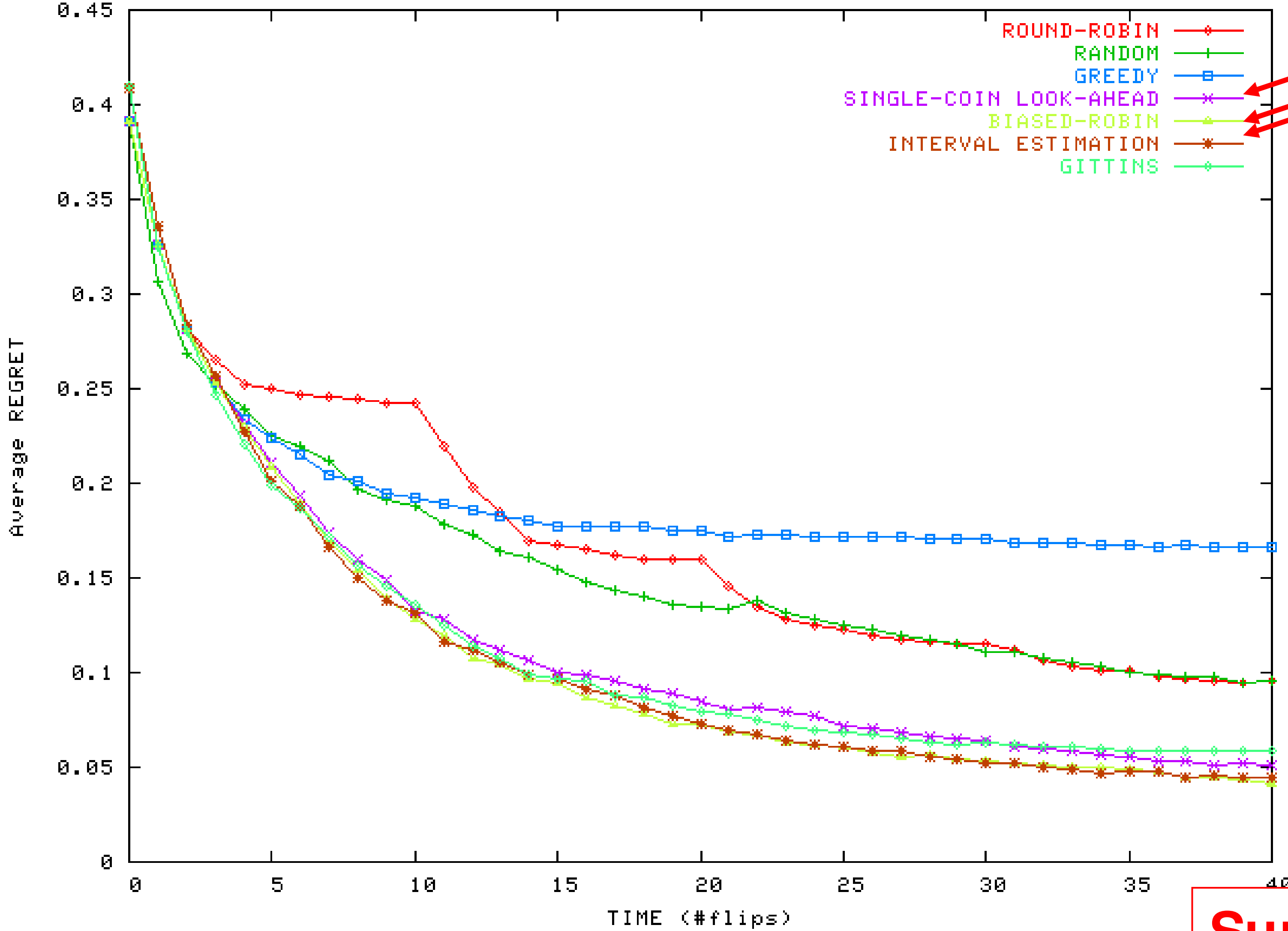
Empirical Results

- Uniform Priors *Beta(1,1)*
  - n=10, b=10 (optimal)
  - n=10, b=40
- Skewed "positive" *Beta(n,1)*
  - Beta(5,1), n=10, b=10
  - Beta(10,1), n=10, b=40
- Skewed "negative" *Beta(1,n)*
  - Beta(1,5), n=10, b=10
  - Beta(1,10), n=10, b=40

Beta(1,1); n=10, b=10

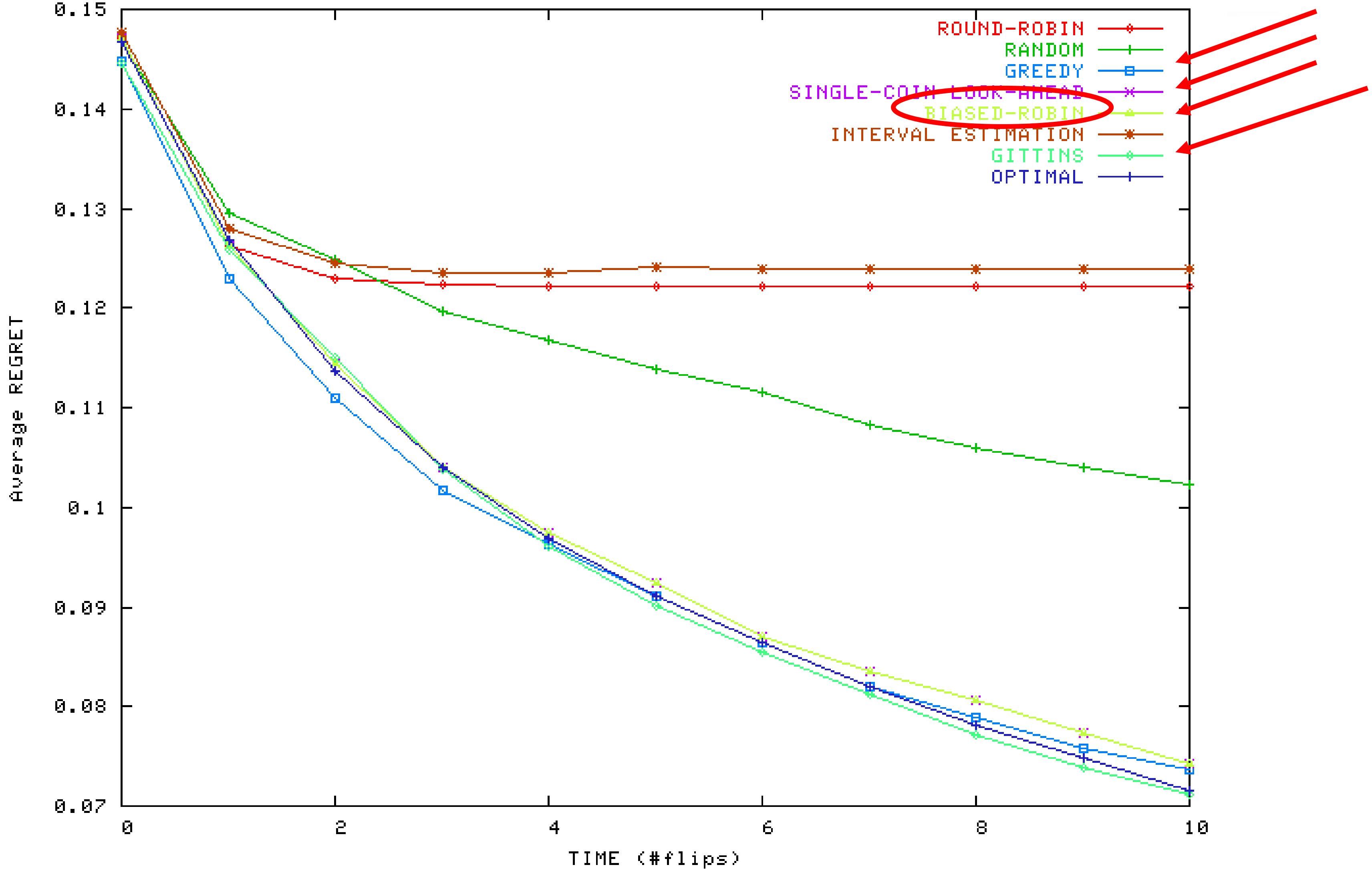


Beta(1,1); n=10, b=40

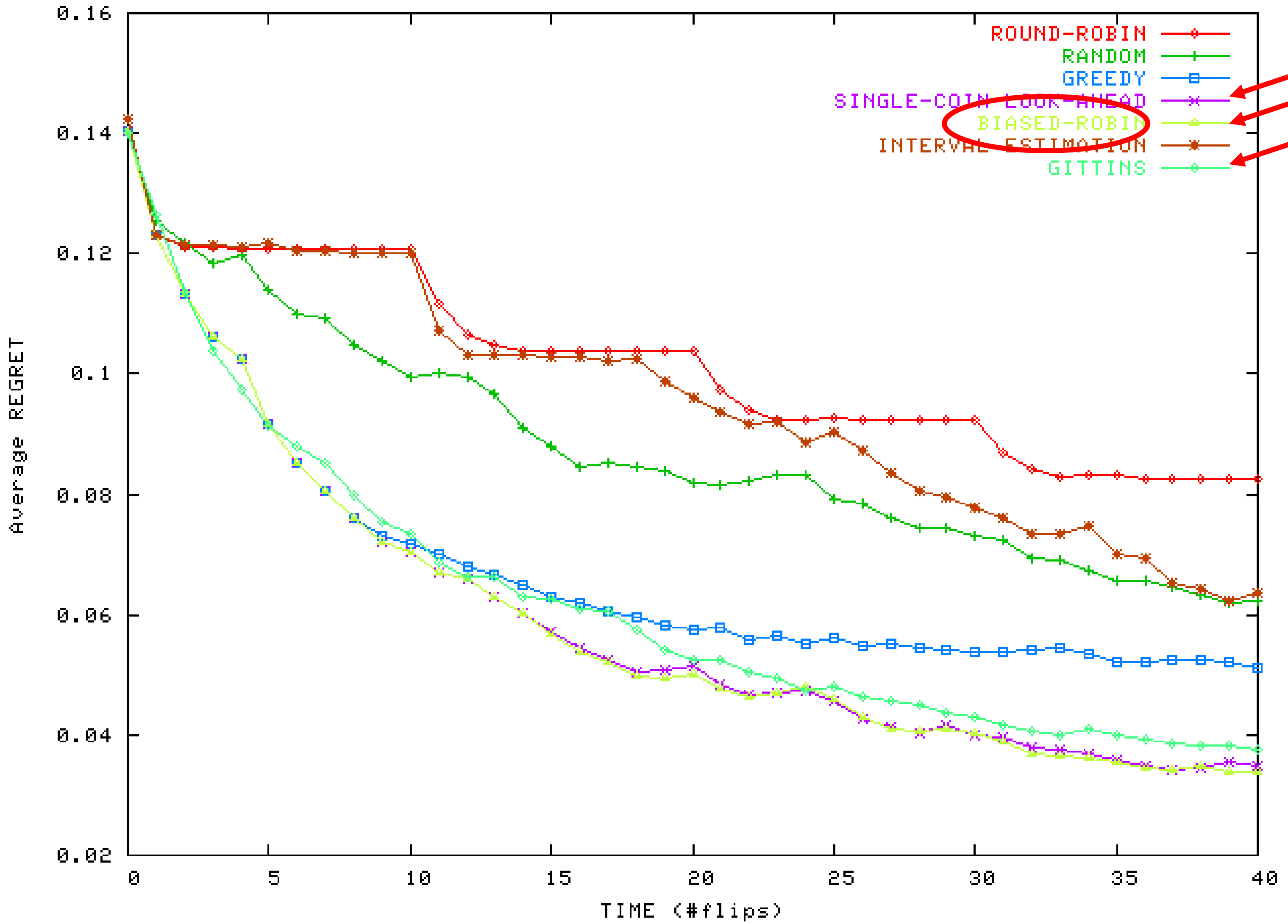


Summary

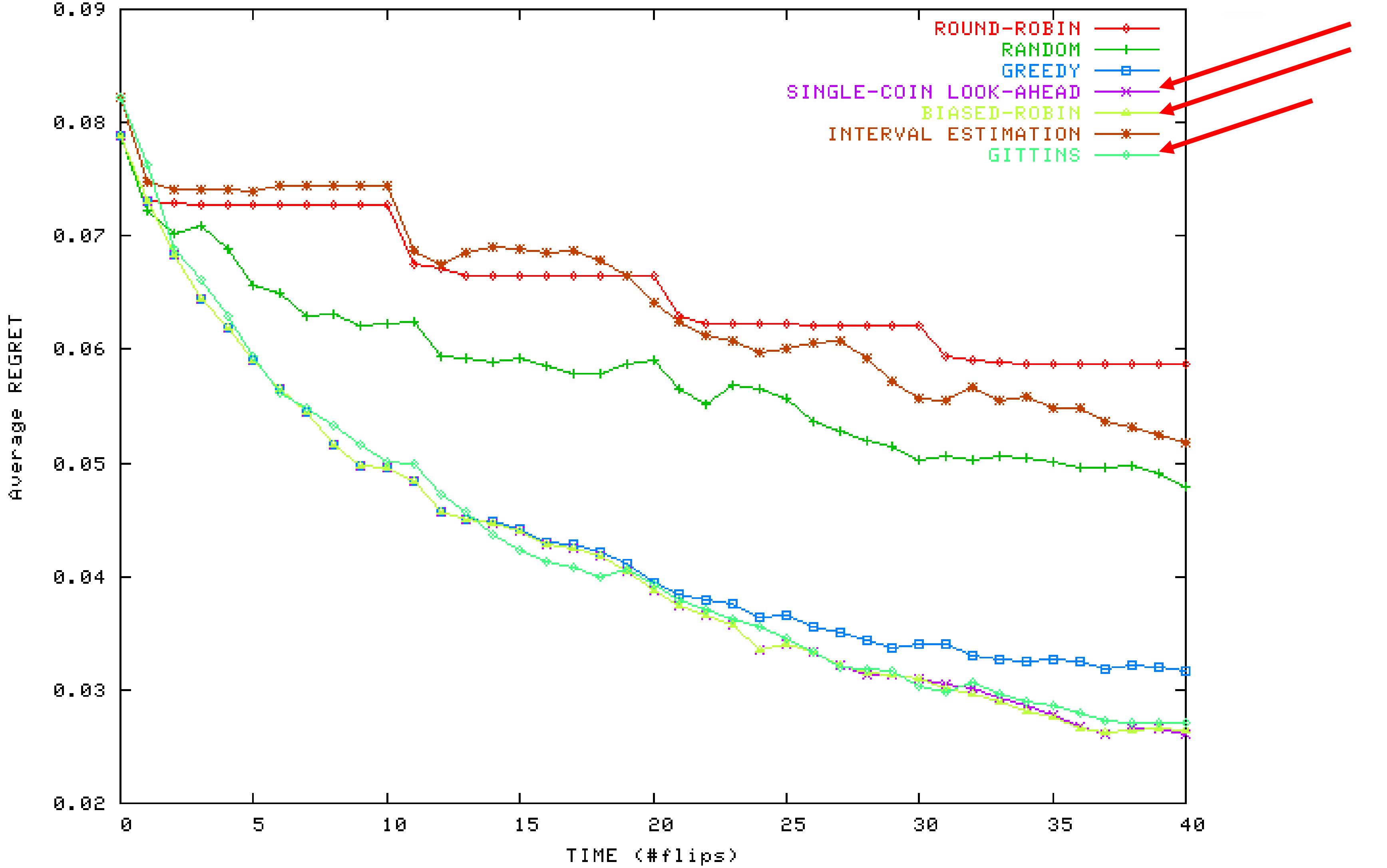
Beta(5,1); n=10, b=10



Beta(5,1); n=10, b=40



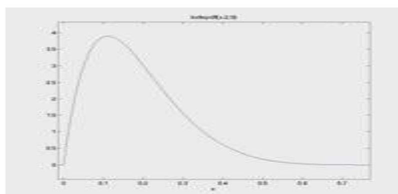
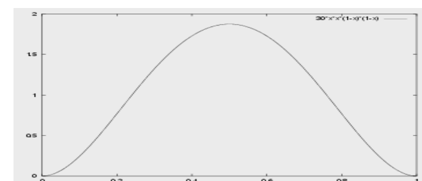
Beta(10,1); n=10, b=40



Round-Robin vs Biased-Robin

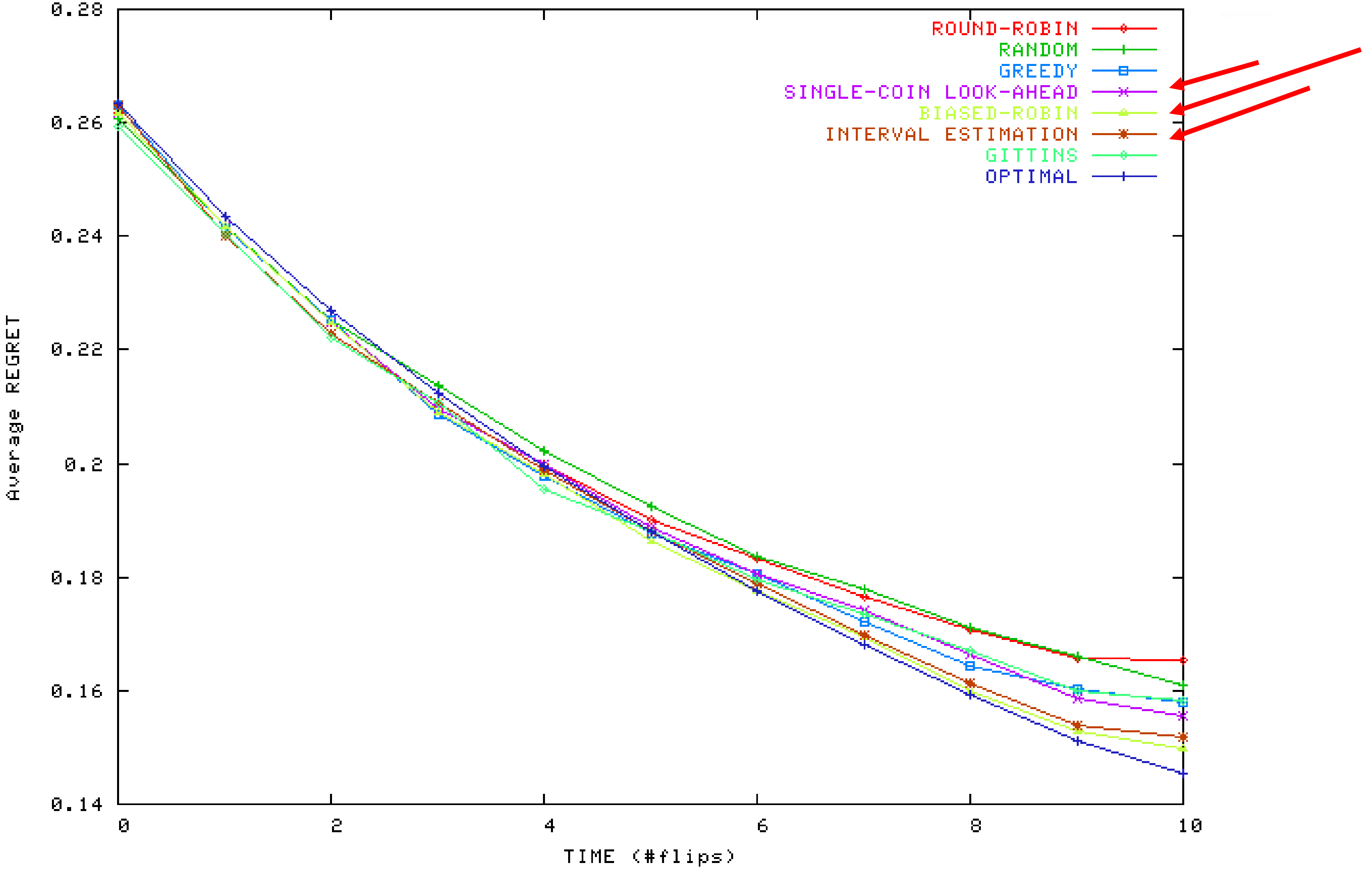
- Quickly (after a few tests),

  see that some coins are NOT "good"…

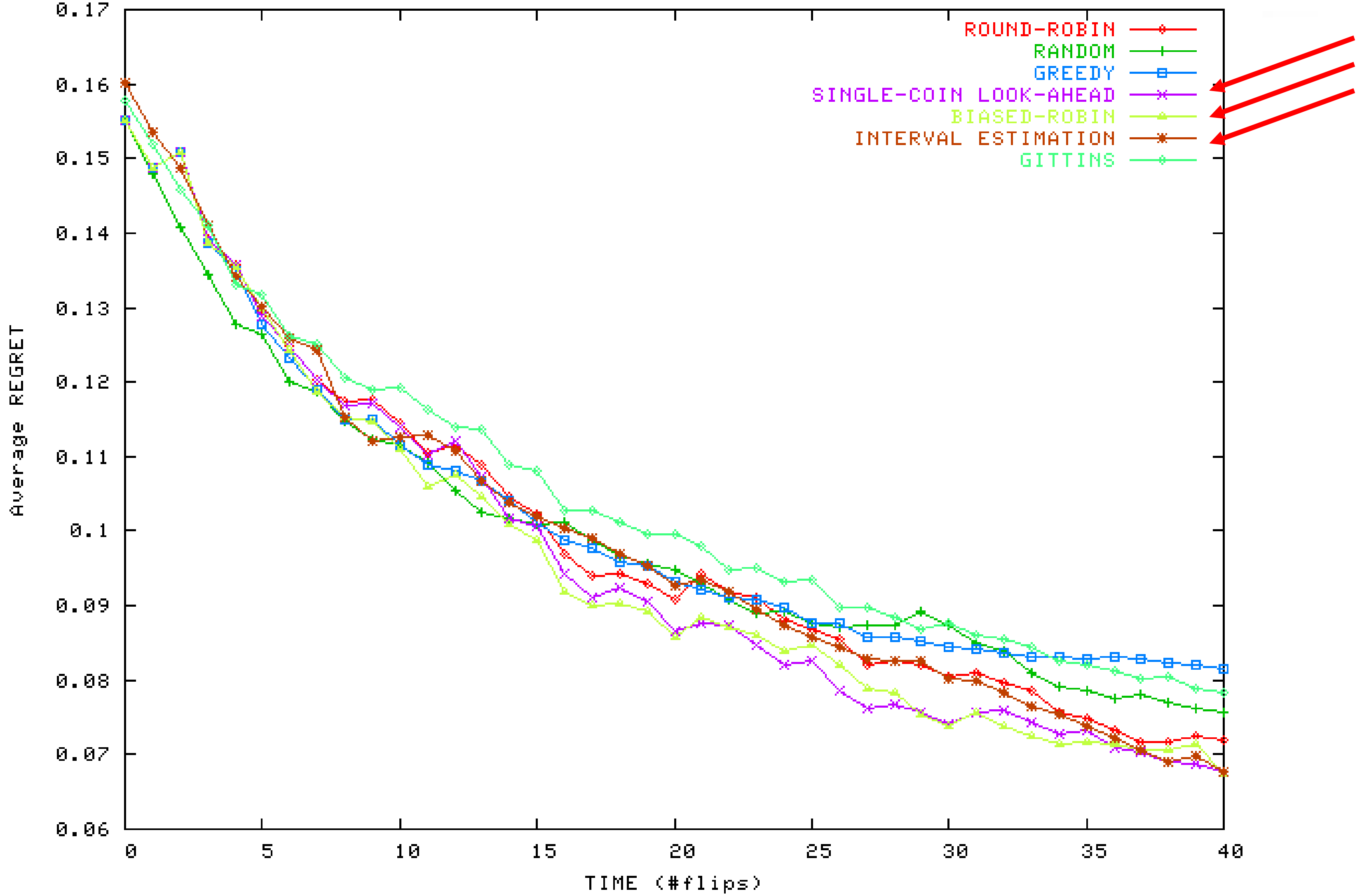
  Beta(1,5)  Beta(3,2) 

- RoundRobin must continue to test each coin
  - including these ineffective ones !
- Biased-Robin can avoid "wasting" tests…

Beta(1,5); n=10, b=10



Beta(1,10); n=10, b=40



Why is RoundRobin ok here?

- $c \sim$ Beta(1,10)

$\Rightarrow$ c typically returns tails

$\Rightarrow$ No real winners here...

$\Rightarrow$ Round-robin as good as anything else...

Comments on Algorithms

Round-Robin, Biased-Robin, …

can skip coin c_i if no chance

- After 9 flips,

  $c_1 \sim$ Beta(1, 3)

  $c_2 \sim$ Beta(6, 1),

  $c_3 \sim$ …

- 1 more flip… c_1 has NO chance!

Talk Overview

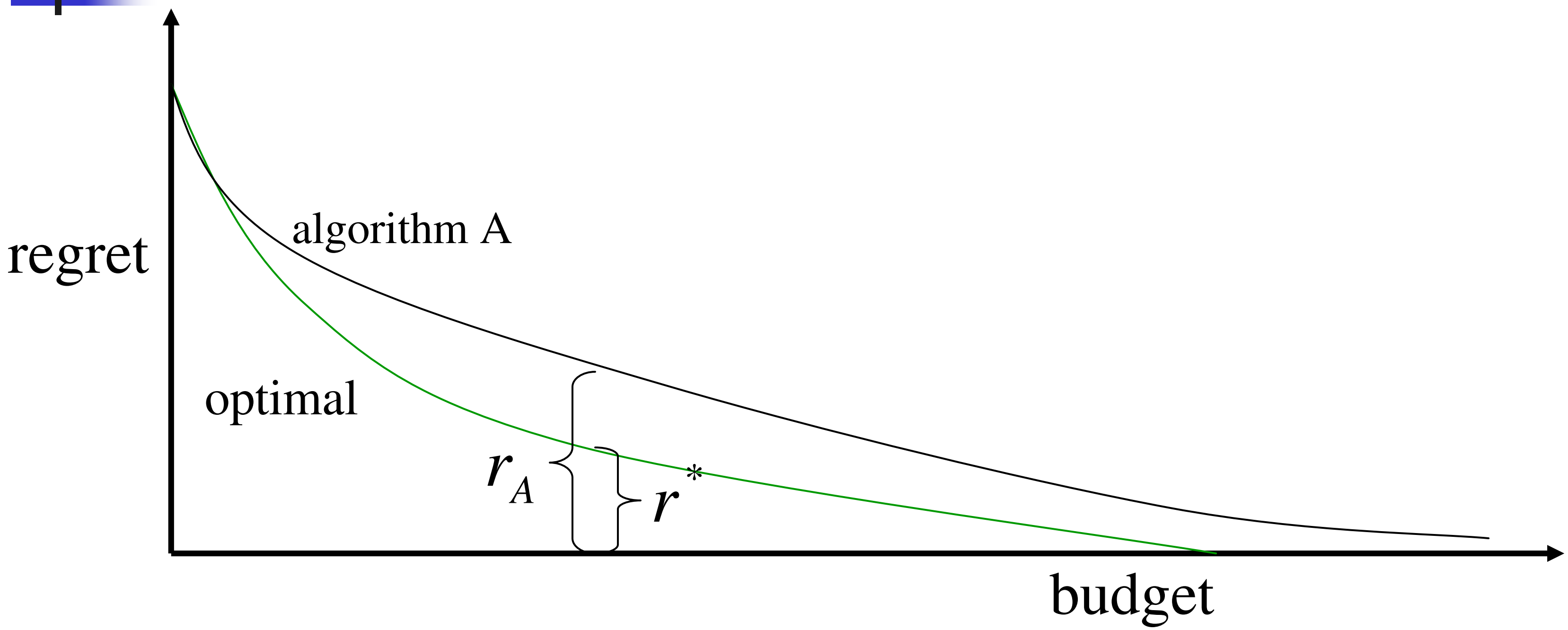
- Motivation
- Active Model Selection
  (≈multi-armed bandit scenario)
  - Bayesian Framework
  - Hardness
  - Algorithms
  - Empirical comparisons
  - Theoretical Results
- Naïve Bayes models
  Learn & Classify under Hard Constraints
- Future Work

Closed Forms

- Uniform priors
  - $E\left(\Theta_{\max}\right) = \dfrac{n}{n+1}$
- Round-robin (RR)
  - n coins
  - budget $b = k \times n$

$$E(\mu_{\max} | RR) = \frac{1}{k+2}\left[k+1-\sum_{i=1}^{n}\left(\frac{i}{k+1}\right)^{n}\right]$$

Approximability



Algorithm A is ***APPROXIMATION Algorithm***

iff

$\frac{r_A}{r^*}$ is bounded by a constant (for any budget, coins, …)

Approximability (con't)

- NOT approximation alg's
  - Round Robin
  - Random
  - Greedy
  - Interval Estimation
  - Biased-robin
- Unknown…
  - ? Single-coin look-ahead
  - ? Gittins

Talk Overview

- Foundations
- Active Model Selection
  (≈multi-armed bandit scenario)
- Learning Naïve Bayes parameters
  (learning classifiers)
  - Framework
  - "Sampling" Algorithms
  - Empirical Comparisons
- Learn & Classify under Hard Constraints
- Conclusions

Initial Situation

	f_1	f_2	f_3	f_4	Class
Instance 1	?	?	?	?	1
Instance 2	?	?	?	?	0
⋮	?	?	?	?	0
	?	?	?	?	0
	?	?	?	?	1

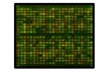
Intermediate Situation

Given current values, we should probe
- **which feature,**
- **of which instance?**

	f_1	f_2	f_3	f_4	Class
Instance 1	a	0	1		1
Instance 2	b				0
					0
					0
					1

Task

Costs

-  $ 5.00
-  $50.00
-  $ 0.50
-  $19.75

Given

- Cost of features

Remaining Budget: $57

For each

- Remaining budget and state

b	0		0	1
d			a	0
	1			0
c				0
				1

Compute

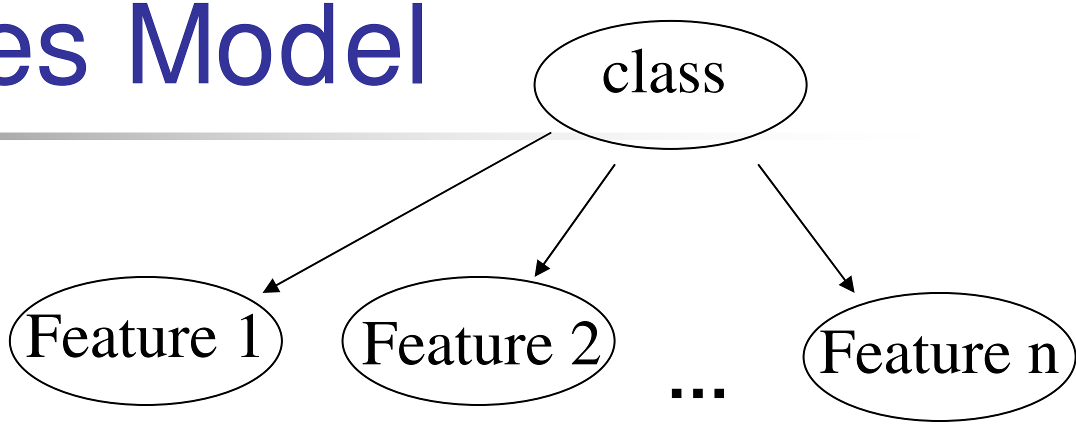
- Which feature of which instance

Coins ⇒ NaïveBayes

- Flipping a coin ⇒ querying a feature
- Twice as many choices:
  For each query, must decide
  - which feature, and
  - what the class label should be

  Action ***act***$_{ij}$ = query from $P(X_i | Y_j)$
- *Two* beta distributions for each X_i,
  - one for Y=1, one for Y=0
- Distributions are updated from counts of X_i = 1 or 0
  - exactly like coins problem

Naïve Bayes Model



- Very simple generative model
  - Features independent, given class
  - Each +class instance "the same", …
- handles missing data
- # of parameters is linear – *O(n)*
  - easy to estimate…

Algorithms

- Round-robin
- Random
- Biased-robin
  - As long as *loss* of single feature is decreasing, keep querying it
- Greedy
- Single-Feature Look-ahead (sfl)
  - Depth d = how far to investigate

- (IntervalEstimate, Gittins)

Policy 1: Round Robin (RR)

- Purchase random, complete instances

Costs

X_1 = 1

X_2 = 1

X_3 = 10

X_4 = 5

X_5 = 3

X_1	X_2	X_3	X_4	X_5	Y
0	1	1	0	0	1
					0
1	1	0	1	0	0
					1
1	0	0	0	0	0
					1

Remaining Budget: ~~60~~

~~40~~

~~20~~

0

Policy 2: Biased Robin (BR)

- More discriminative; plays the winner.

Costs

$X_1 = 1$

$X_2 = 1$

$X_3 = 10$

$X_4 = 5$

$X_5 = 3$

0					1
0					0
					0
	1				1
					0
1					1

Remaining Budget: ~~60~~

~~59~~

~~58~~

~~57~~

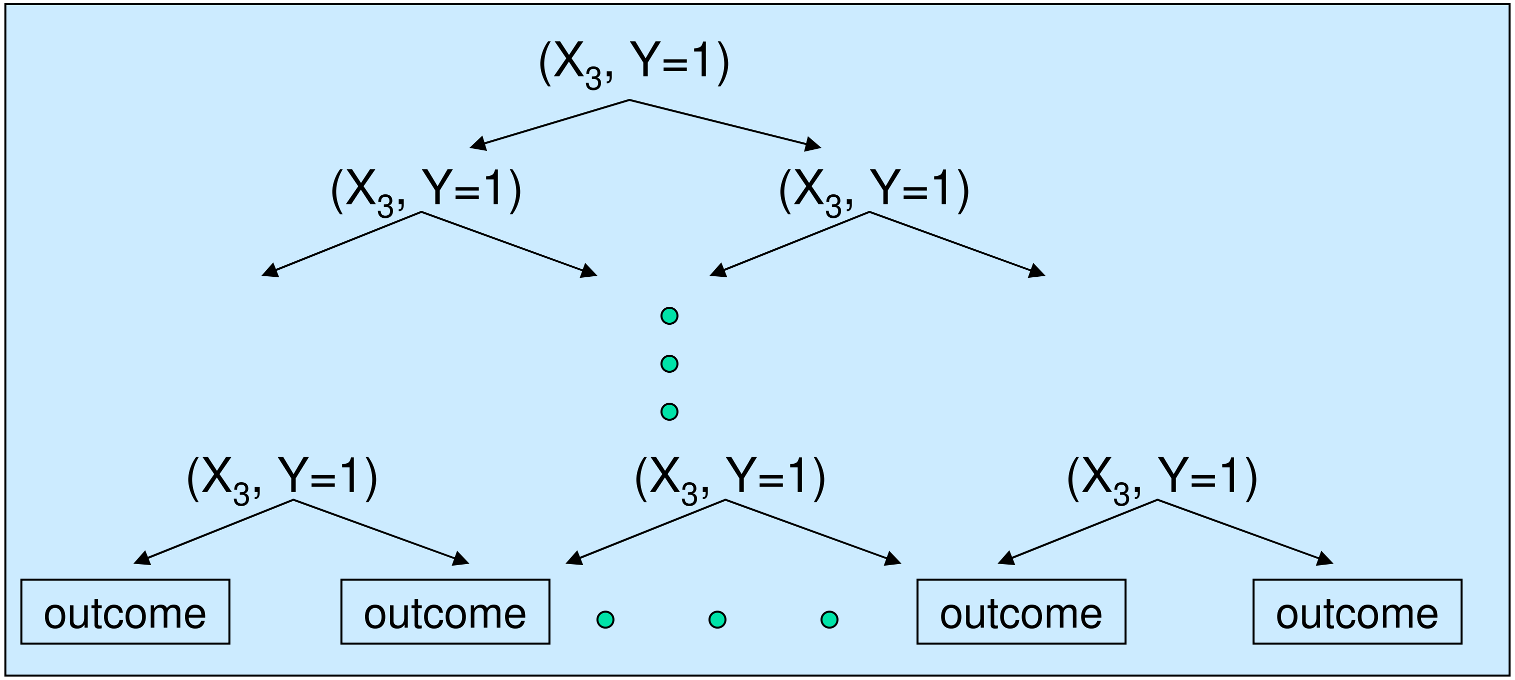
56

Loss:

Policy 4: Single Feature Lookahead

$$SFL(X_i, y) = \sum_{j \in outcomes(d)} P(j)\, Loss(j)$$

- expected loss of spending next **"d" dollars** on a **single** feature-class pair (X_i, y)



- Purchase best (X_i^*, y^*). *once*, and recur.

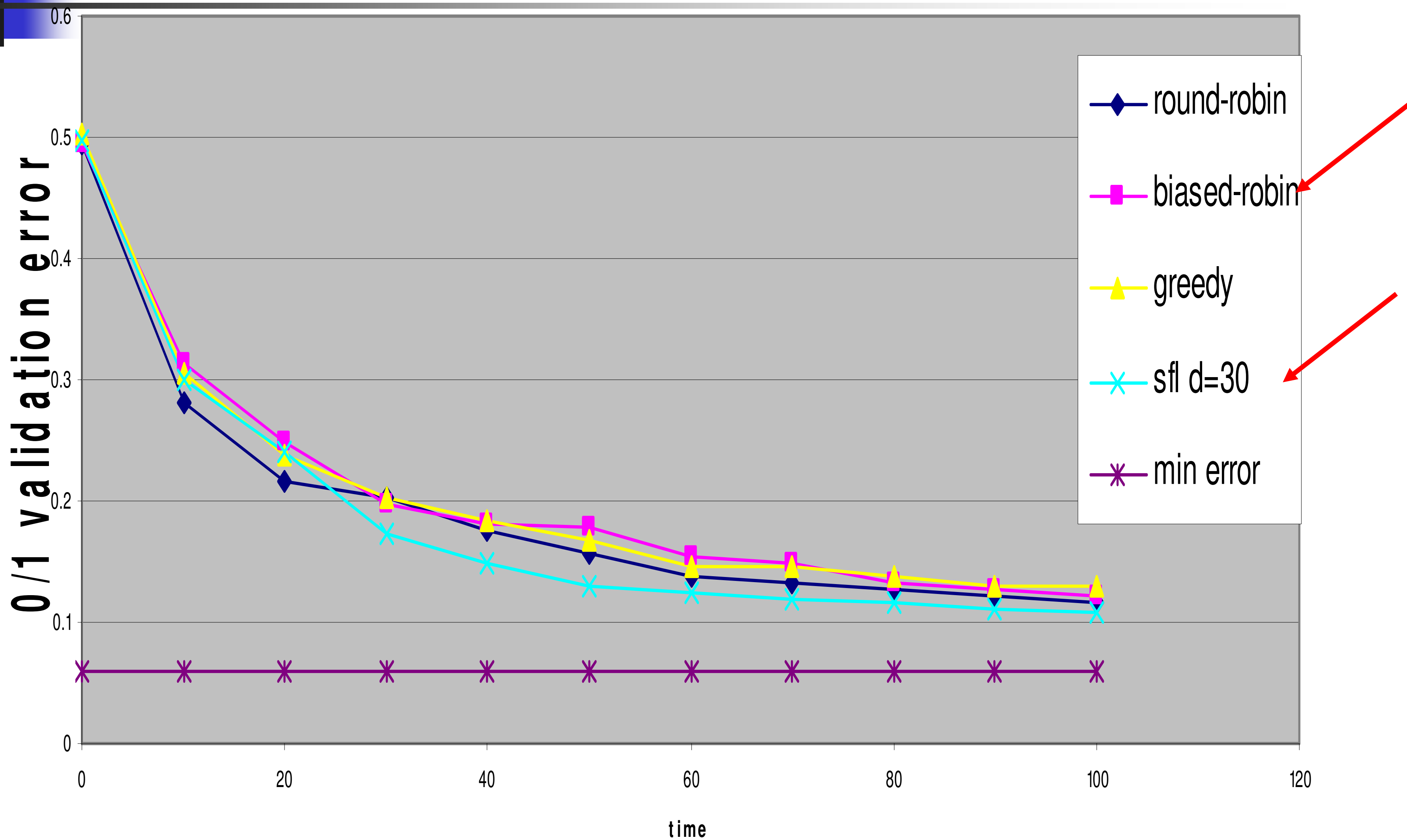
Empirical Studies

- Synthesized data
  - Each parameter $\theta_{+fi|+}$, $\theta_{-fi|-}$ ~ Beta(1,1)
    - … each feature slightly discriminant
  - Single Discriminative Feature
    - P(+f1 |+) =0.9; P(-f1| --) = 0.1
    - … "P(+fi)" independent of class i=2..n
- UCIrvine data

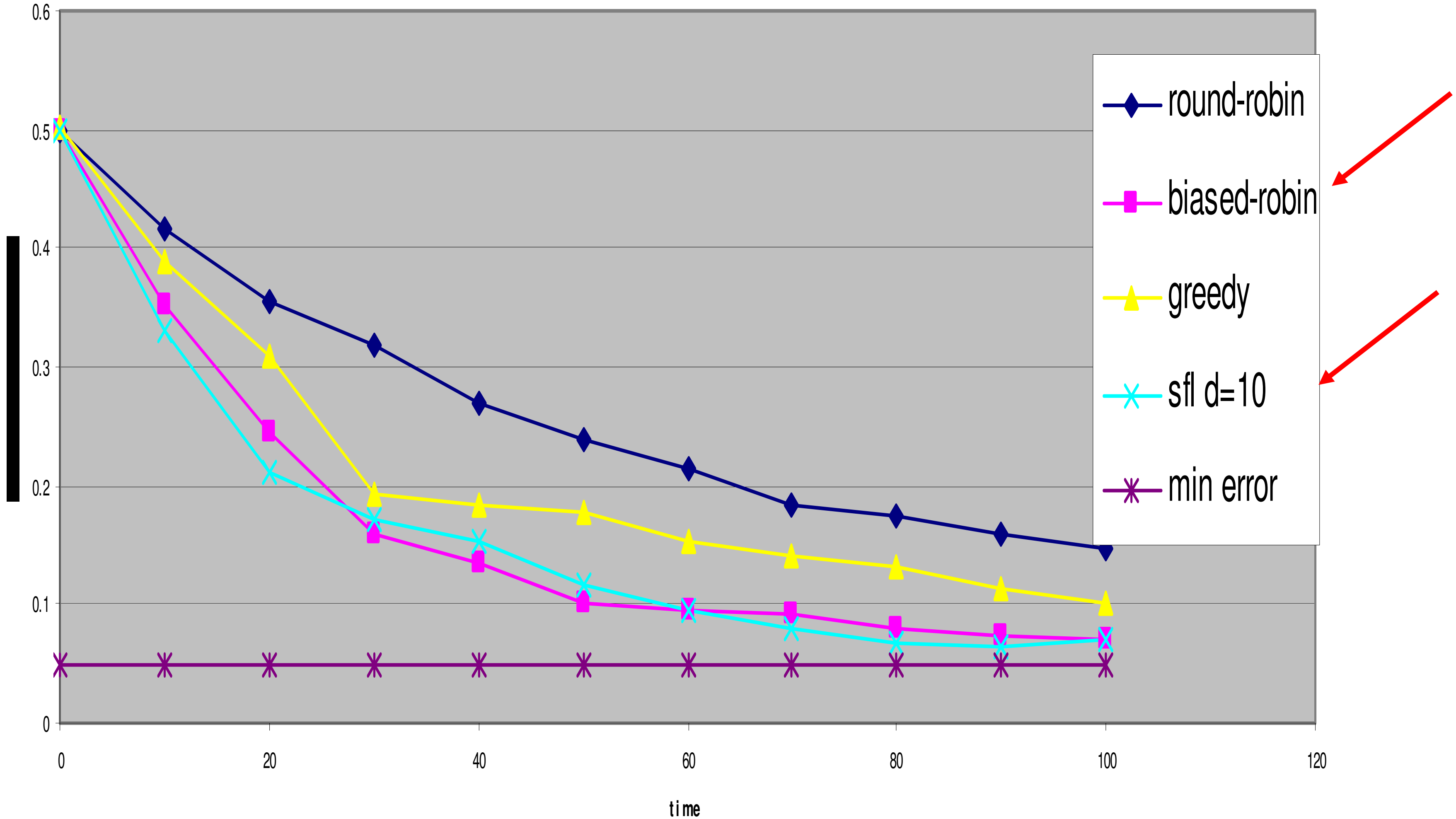
(Each point: average over 50 runs)

Performance on “No Great Feature”

$\theta_{+fi|+}, \theta_{-fi|-} \sim \mathrm{Beta}(1,1)$



Single Discriminative Feature
n=10



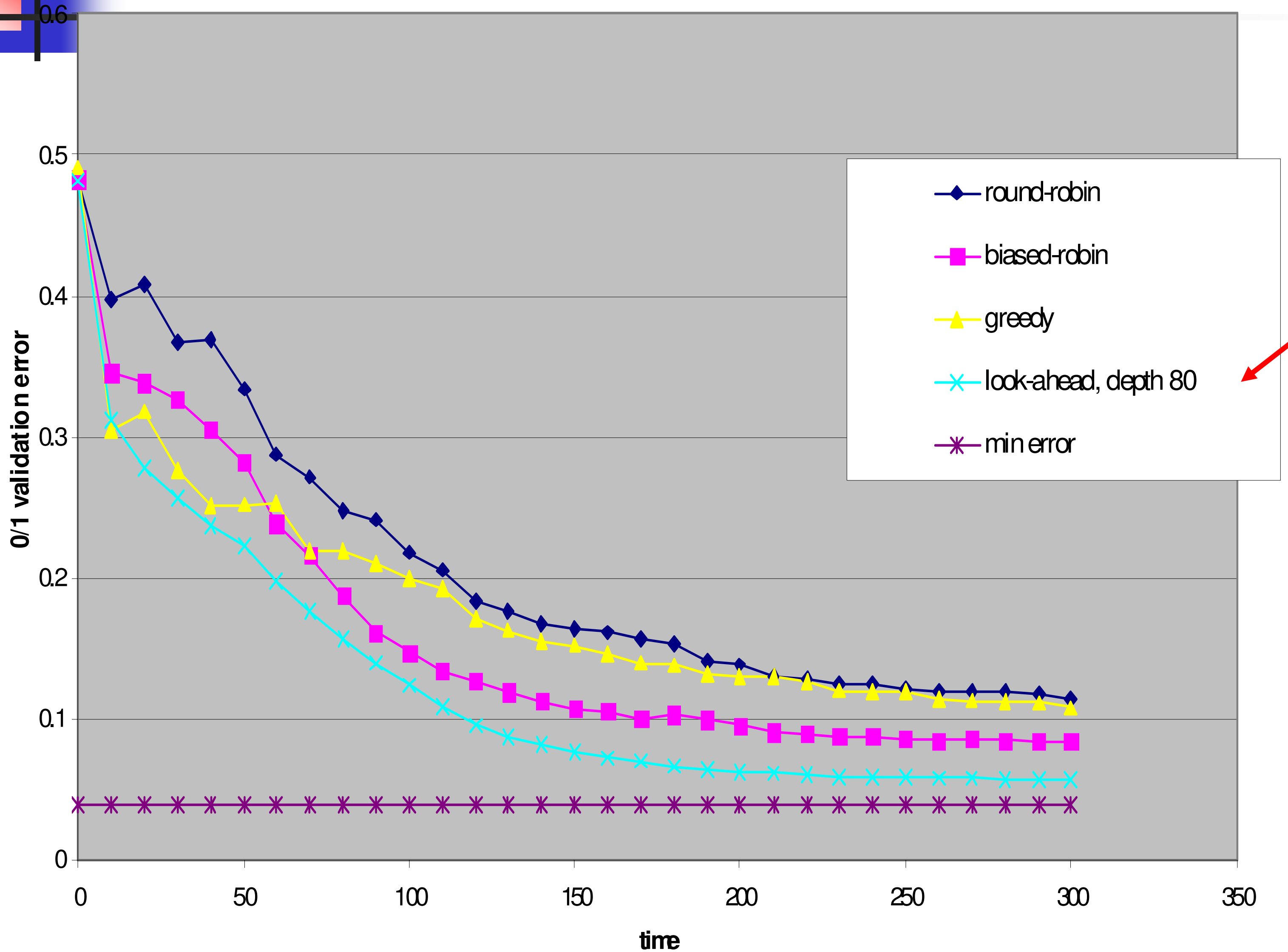
Comments (synthesized data)

- When some feature is discriminant,
  - Biased-Robin, SFL “look” for it…
  - …big advantage!
- If not…
  - all strategies about the same…

Empirical Studies

- Synthesized data
- UCIrvine data
  - Mushroom
    - 8124 instances
    - 23 features (1 very discriminant)
  - House voting
  - … investigate sfl(d) over d…

UCI Mushroom Dataset



Which features were probed?

- 8124 instances x 23 features = <u>186,582 probes</u>
  - … get within 0.01 (0.04 vs 0.03) of optimal in <u>300</u> !

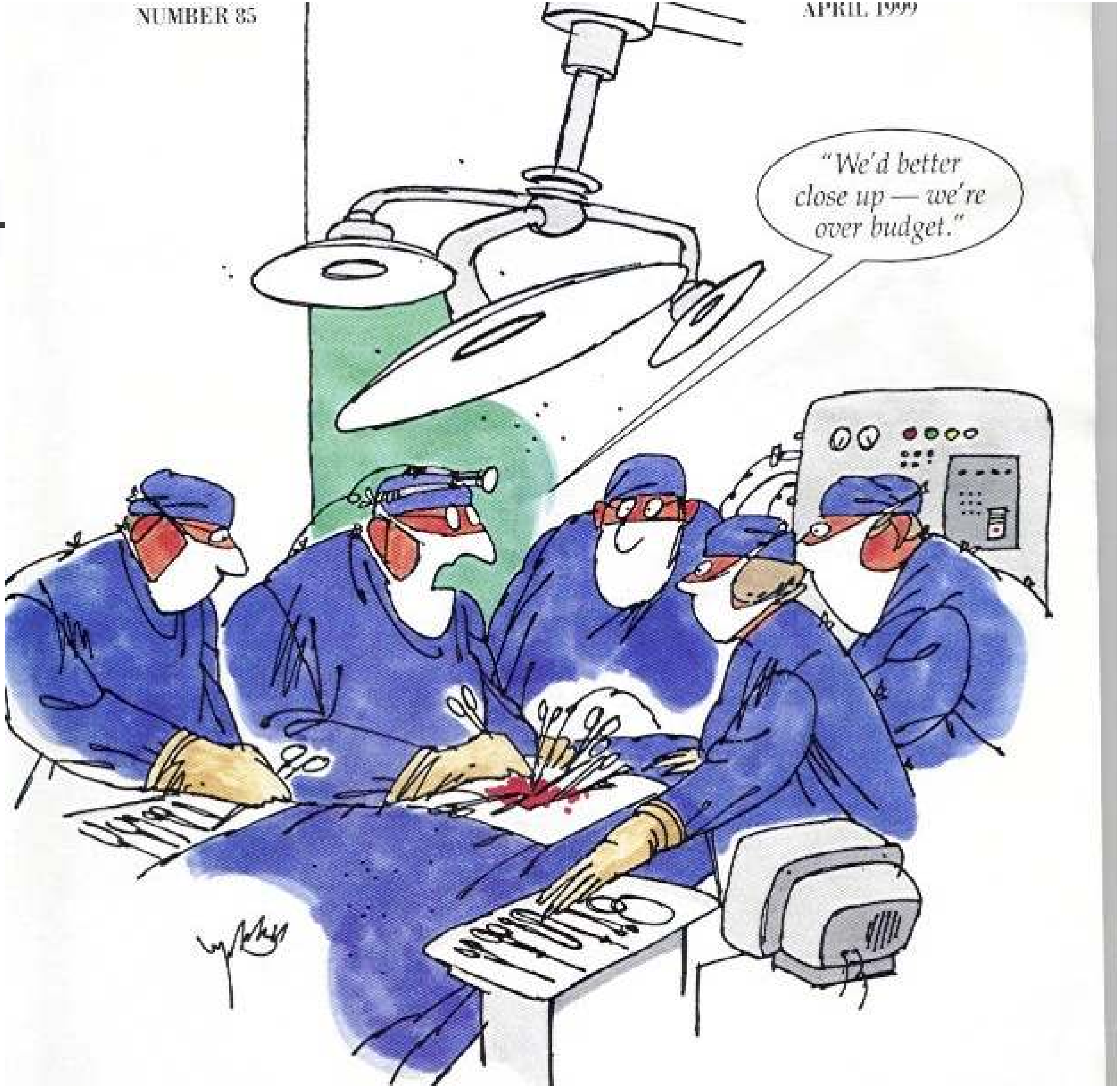
- RoundRobin:
  - Each of 23 features probed ≈ 300/23 ≈ 13 times
- SFL, BiasedRobin:
  - discriminant features (like F# 5): ≈70-110 times
  - other features: ≈1 time
- … SFL, BR did MUCH better than RR

Patterns...

- SFL = (one of) best, in general
  - MUSHROOM, VOTE
  - \+ CAR, DIABETES, CHESS, BREAST
  - ... depth *d* does matter ...
- Biased-Robin best of budget-insensitive
- Run times:
  - RR, BR really fast
  - Greedy ok
  - SFL slowest (≈ minutes/experiment)

Talk Overview

- Foundations
- Active Model Selection
- Learning Naïve Bayes parameters
- Learn & Classify under Hard Constraints
  - Framework
  - Algorithms
  - Empirical Comparisons
- Conclusions


NUMBER 85
APRIL 1999
"We'd better close up — we're over budget."

So far …

- So far…
  - LEARNER must pay for features
  - But CLASSIFIER gets ALL features to *for free !*
- What if CLASSIFIER also pays for features?
- Budgets:
  - Learner budget: b_L
  - Classifier budget (per patient): b_C
- Eg… spend $b_L = \$1000$ to learn a classifier, that can spend only $b_C = \$30$ /patient…
- How???

The Problem

Inputs

Training Pool:

X_1	X_2	…	X_r	Y
?	?	…	?	1
?	?	…	?	0
?	?	…	?	0
		⋮		
?	?	…	?	1

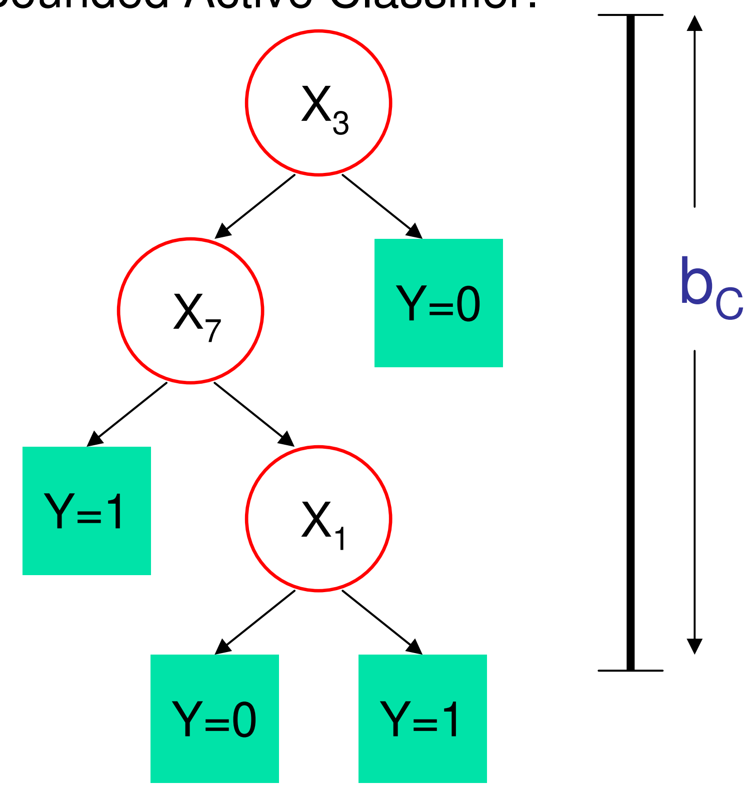
Learner purchases b_L feature-values

Learning budget: b_L
Classification budget: b_C

Feature Cost: $C(X_1), \ldots, C(X_r)$

Output

Bounded Active Classifier:



$$C(X_3) + C(X_7) + C(X_1) \le b_C$$

Optimal Bounded Active Classifier

$$BAC^* = \underset{B \in \{\text{cost } b_c \text{ active classifiers}\}}{\arg\min} \sum_{\mathbf{x},y} P(\mathbf{x}, y)\, L(B(\mathbf{x}), y)$$

Good News:

BAC* can be produced via a dynamic program, given

(1) $P(\, Y = y \mid \mathbf{X} = \mathbf{x} \,)$

(2) $P(\, X_i = x_i \mid \mathbf{X}/X_i = \mathbf{x}' \,)$

where $\mathbf{x}$ is any size $\approx b_C$ feature vector

Bad News:

Only limited learning budget b_L for estimating (1) & (2)

Skip

Double Dynamic Program !!

- After b_L purchases,
  remaining LEARNING budget … = 0,
  Produce optimal depth-b_C d… …e;
  Compute "score"
- Back up:
  - Aft… …es, remaining b_L' = 1,
    … possible "purchase",
    …ing to b_L' = 0 … with score.
    Score is BEST of these
  - … when remaining b_L' = 2,
    consider each possible "purchase", …
    b_L' = 1 situation …

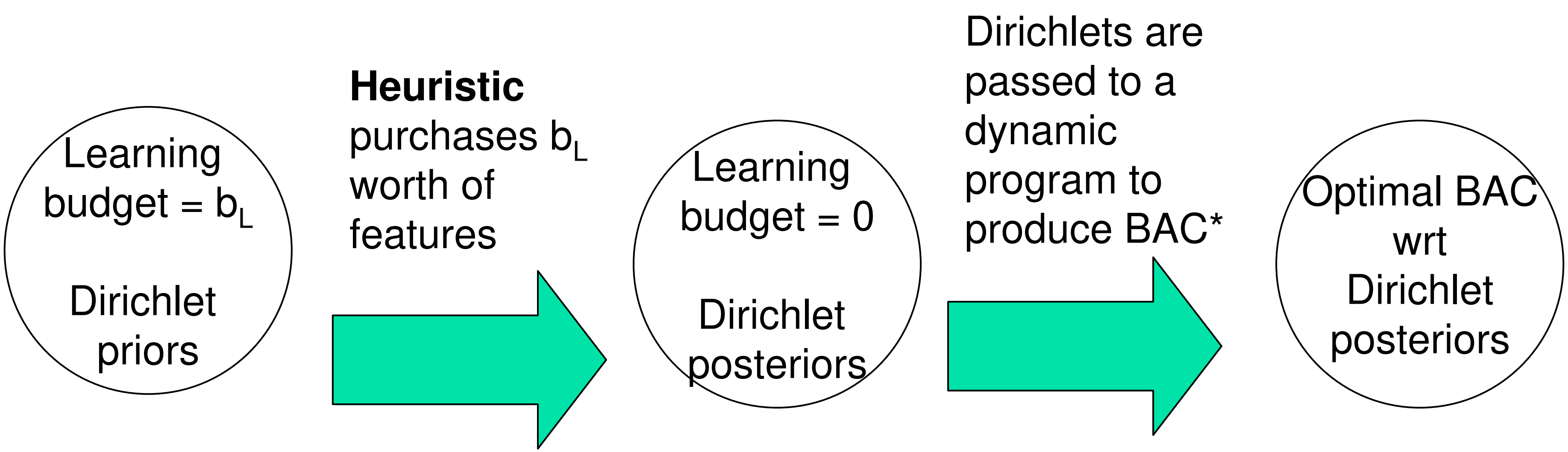
Dynamic Program I

Dynamic Program II

Way too SLOW !!!

Alternative: Heuristic Learning Policies

- ∃? tractable purchasing policy that performs well ?
- … consider 5 different heuristic policies…



Heuristic Policies

1. Round Robin

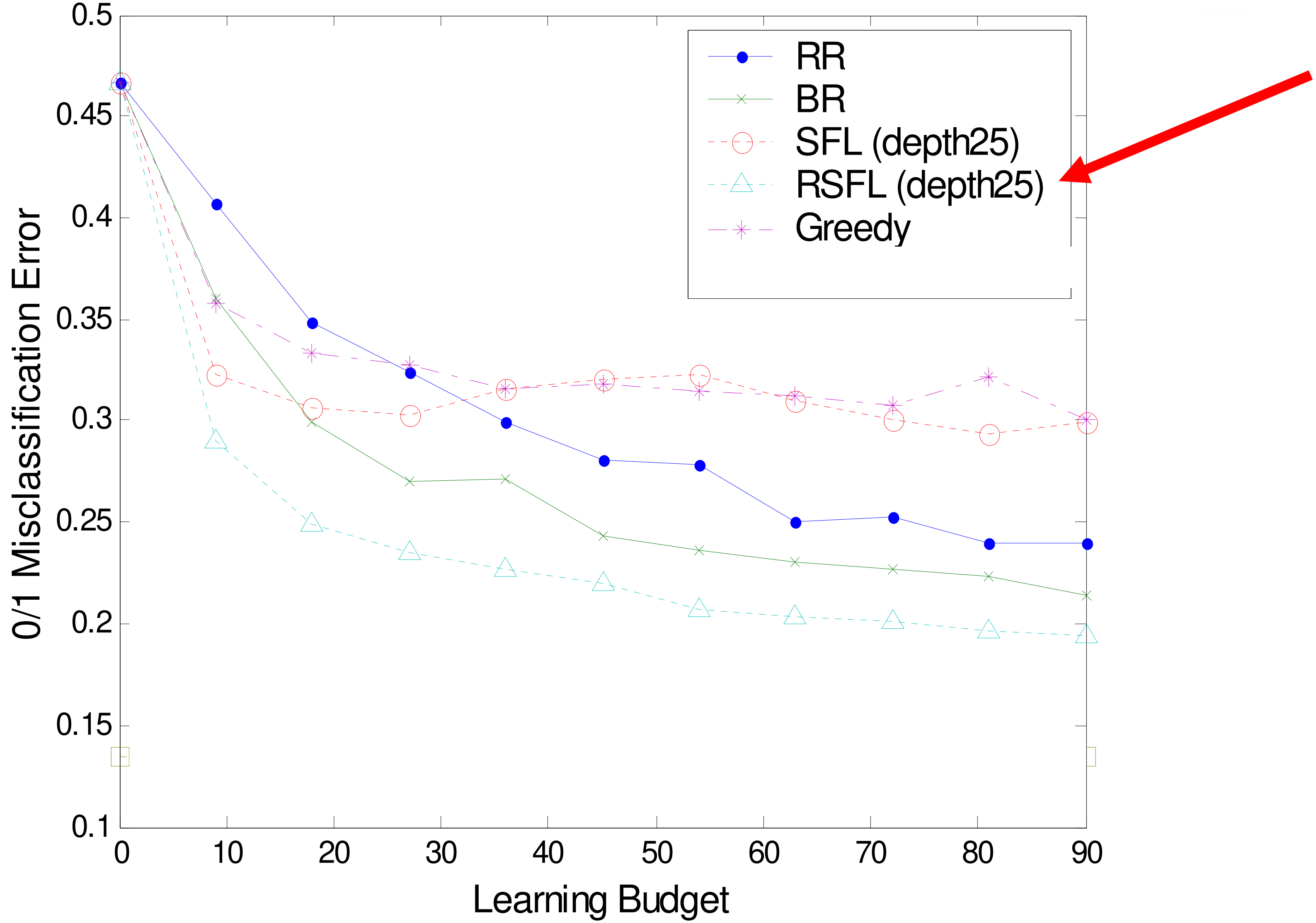
2. Biased Robin

3. Greedy
4. Single Feature Look-ahead (SFL)
5. Randomized SFL

Skip

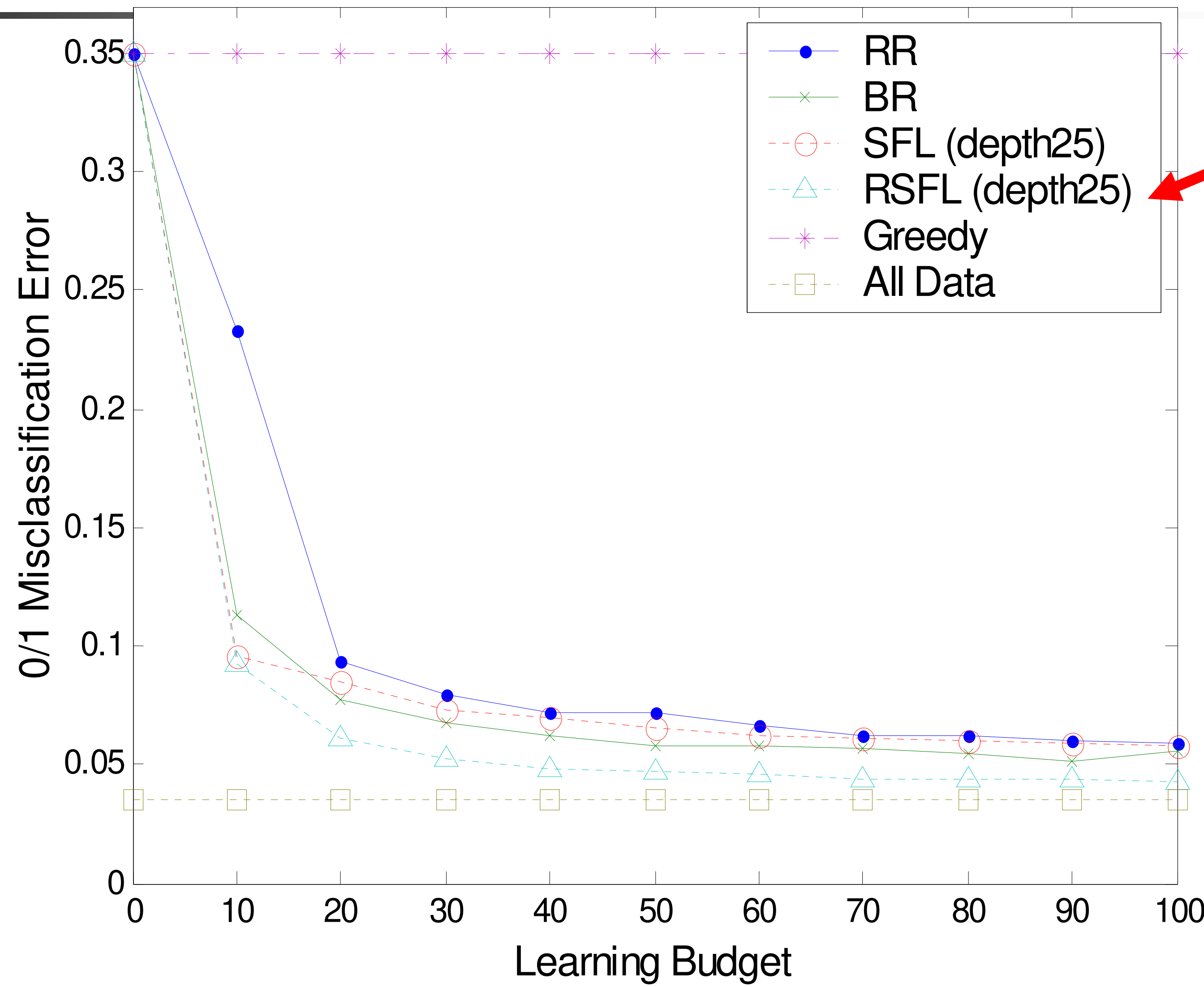
Glass

(Identical Feature Costs)



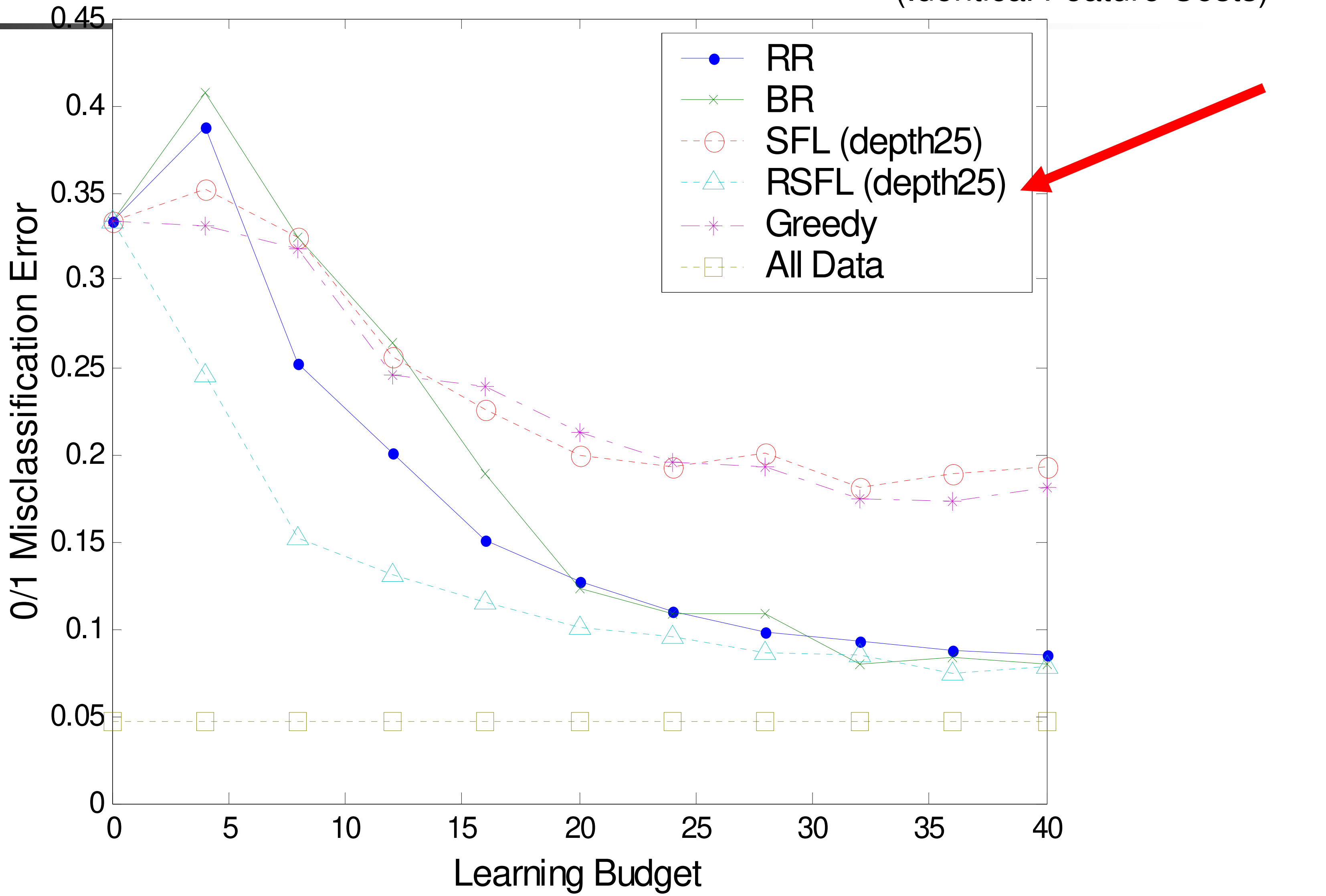
Breast Cancer

(Identical Feature Costs)



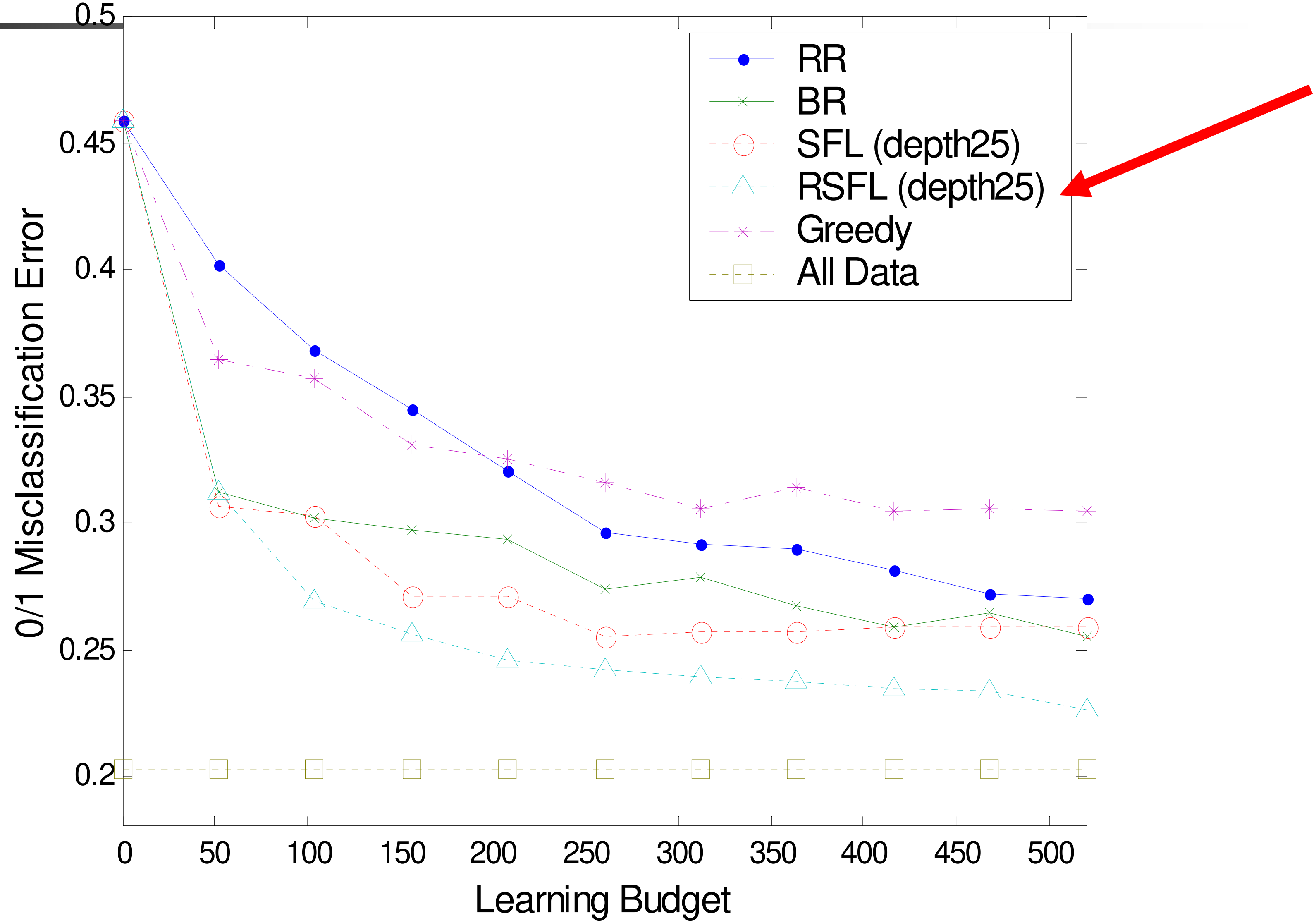
Iris

(Identical Feature Costs)



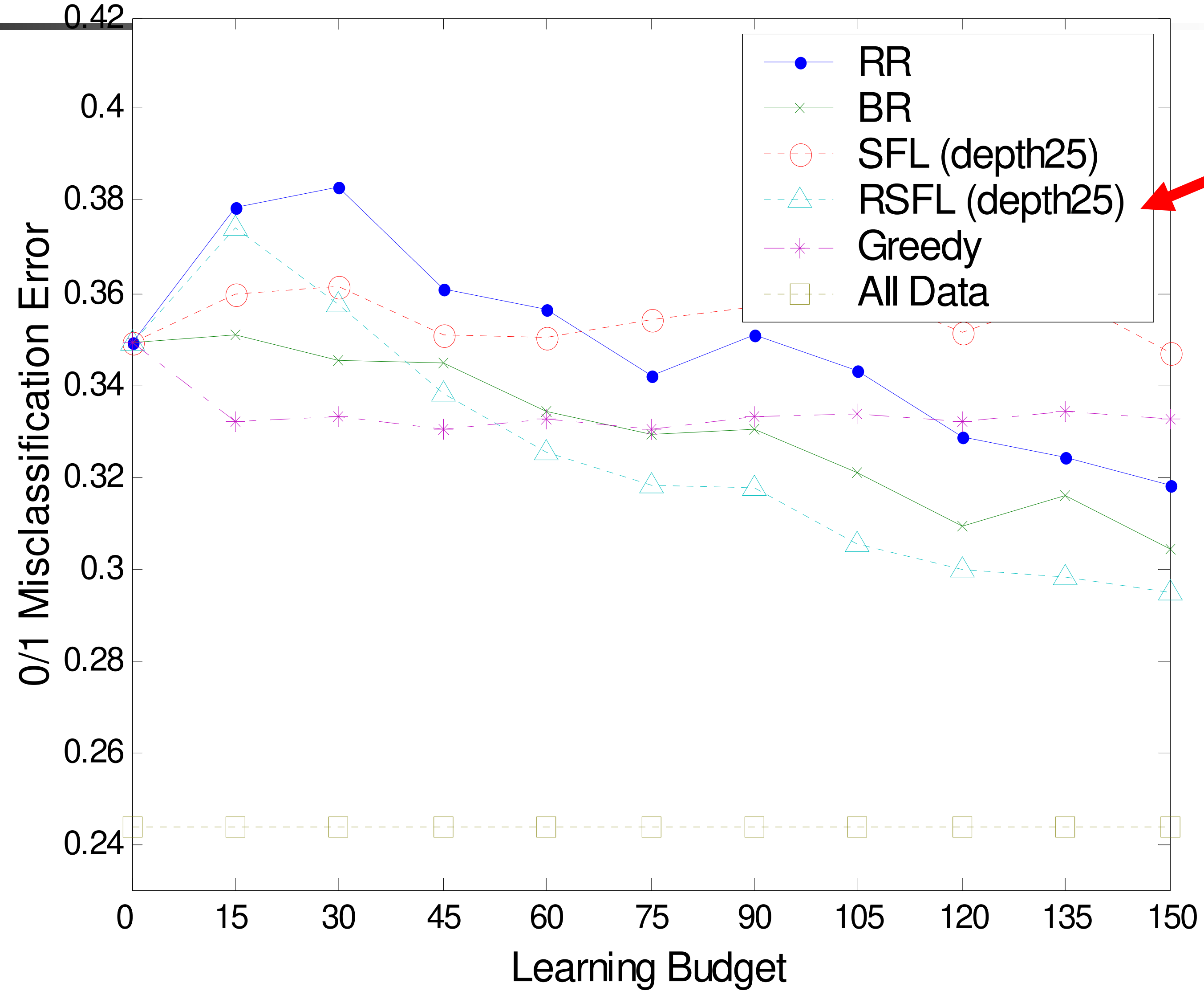
Heart Disease

(Different Feature Costs)



Pima Indians

(Different Feature Costs)



Summary of Results

- Don’t use Round Robin
- Do use
  - Randomized Single Feature Lookahead (RSFL)

Talk Overview

- Foundations
- Active Model Selection
- Learning Naïve Bayes parameters
- Learn & Classify under Hard Constraints
- Conclusions
  - Future Work
  - Contributions

Future Work, Ia (framework)

	f_1	f_2	f_3	f_4	Class
Instance 1	?	?	?	?	?
Instance 2	?	?	?	?	?
	?	?	?	?	?
	?	?	?	?	?
	?	?	?	?	?

Future Work, Ib (framework)

- *Complex cost model*
  - ***non-uniform*** misclassification costs.
  - ***Bundling*** tests
  - ***Decision-theoretic***: optimize *f(budget, regret)*
    - budget + $\tau \times$ regret
- Allow learner to perform more powerful probes
  - purchase X_3 in instance where $X_7 = 0$ and $Y = 1$

Future Work, II: Algorithms

- Other algorithms
  - … from MDP literature ?
    - We tried TD(λ) on coins… linear combination, tiling, …
    - No luck…
- Address current open problems
  - ? NP-hard for uniform cost, uni-modal distr'n
  - Finding *optimal allocation*?
    Bound on effectiveness of best *allocation* strategy?
  - Develop policies with ***guarantees*** on learning performance

Summary

- Defined framework
  - Ability to purchase individual feature values
  - Fixed LEARNING Budget
  - Fixed CLASSIFICATION Budget
- Results show …
  - *Avoid Round Robin*
  - Try clever algorithm
    - Biased Robin
    - Randomized Single Feature Lookahead

Thanks

- Joint work with
  - Omid Madani
  - Dan Lizotte
  - Aloak Kapoor
- All (OM, DL, RG) thank
  - NSERC
  - AICML
  - U of Alberta Computing Science
- OM thanks Alberta Ingenuity
- AK thanks iCORE