Tuning for Software Analytics: is it Really Necessary?

Wei Fu

wfu@ncsu.edu

Roadmap

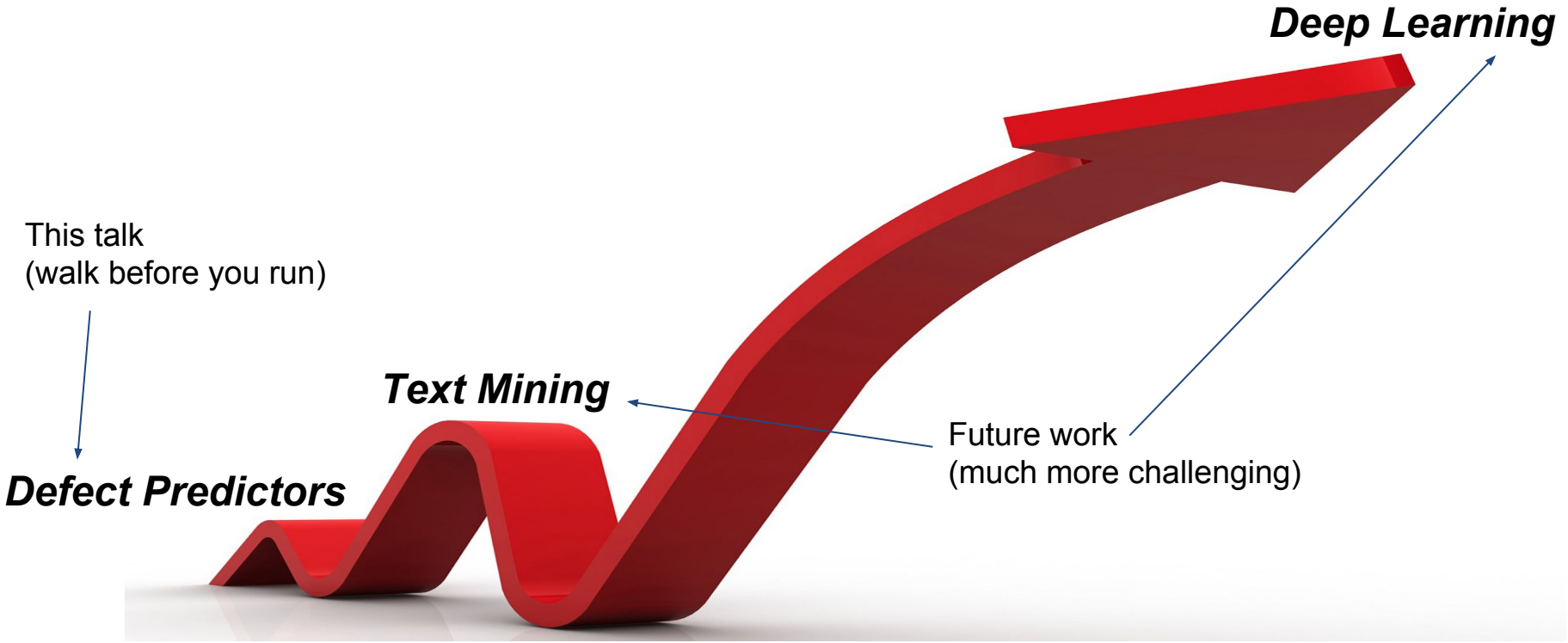
- Expectation management
- Motivation
- Background
- Progress Report
- Future Work & Challenges

What to Tune?

What to Tune?


Deep Learning
This talk
(walk before you run)
Text Mining
Defect Predictors

What to Tune?

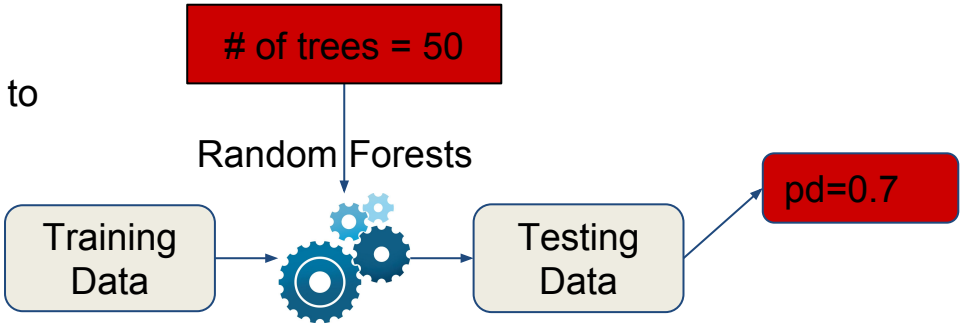

Deep Learning
This talk
(walk before you run)
Text Mining
Future work
(much more challenging)
Defect Predictors

Roadmap

- Expectation management
- Motivation
- Background
- Progress Report
- Future Work & Challenges

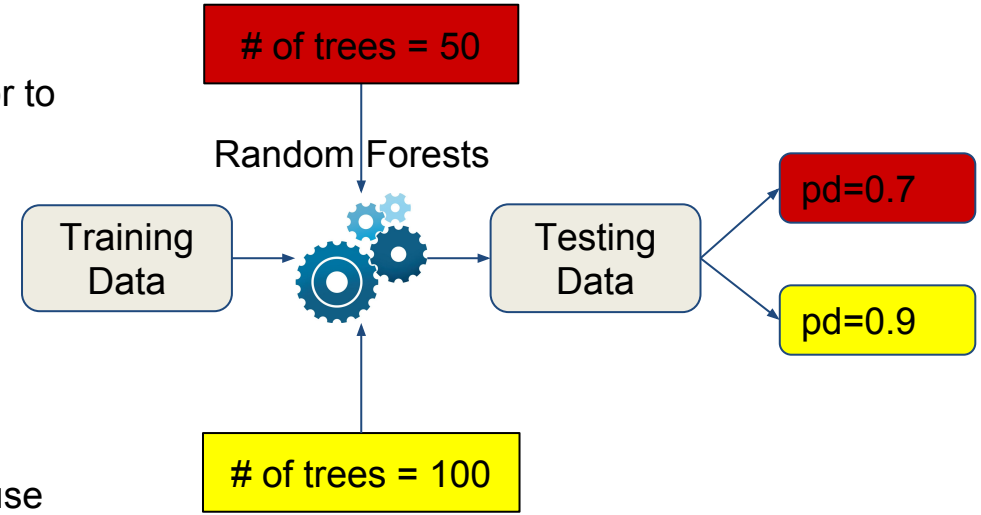
What is Parameter Tuning?

- Data mining algorithms usually have some “magic parameters’’ to control their behavior to explore data.
  - Random Forests:
    - Number of trees
    - Depth of the tree
    - …...
- When tuning a data miner, data miner will use different heuristics and generate different models.



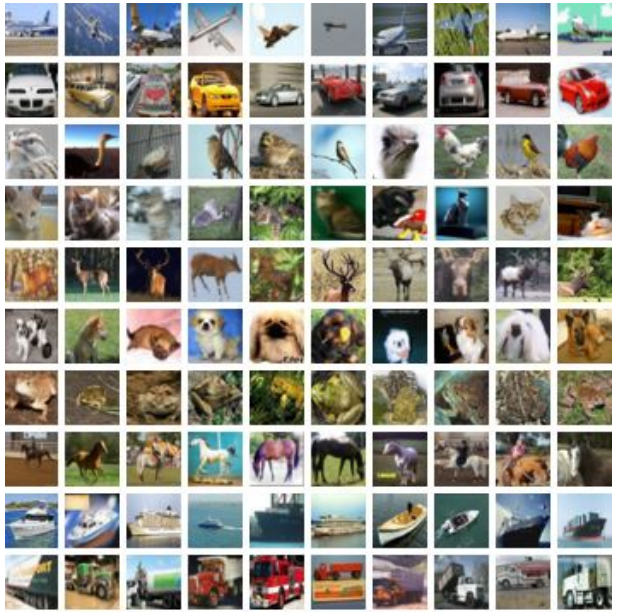
What is Parameter Tuning?

- Data mining algorithms usually have some “magic parameters’’ to control their behavior to explore data.
  - Random Forests:
    - Number of trees
    - Depth of the tree
    - …...
- When tuning a data miner, data miner will use different heuristics and generate different models.



Note: outside of SE data science, other communities endorse tuning

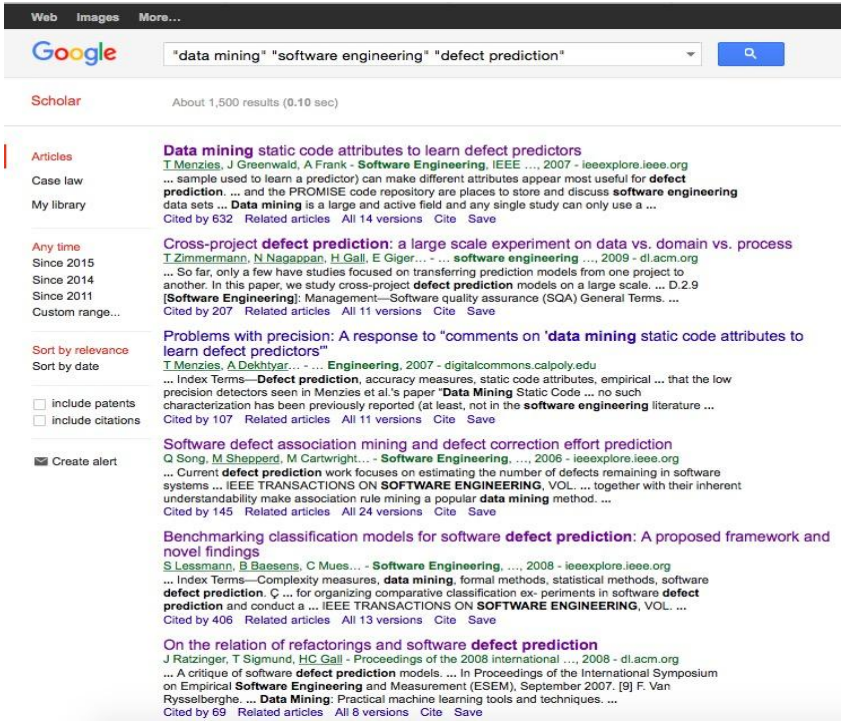
- Other communities
  - Test error improved from 11% to 9.5% on image classification [Snoek' 2012]
  - Random search is able to find good hyper-parameters within a small fraction of the computation time [Bergstra '2012]
  - Analytic parameter selection yields good generalization performance of SVM [Cherkassky '2004]



CIFAR-10 *

* https://www.cs.toronto.edu/~kriz/cifar.html

But for “software defect prediction”…. … mostly ignored



- “Software defect prediction”
  - Guessing where the bugs are, using models built via data mining
  - (see slide20 for more details)
- Recent lit review
  - Only 2/50 acknowledged impact of tunings.
  - Almost all using learners “off-the-shelf”
- Two exceptions:
  - [Tantithamthavorn 2016](Grid Search)
    - 43 HPC nodes with 24 hyperthreads
    - 65% learners: 30 minutes
    - 35% learners: over 30 minutes
  - [Lessmann 2008](Grid Search)
    - tune a small set of their learners
    - no runtime reports

Why ignored?

Cause they are so well explored all already… right?

CPU intensive!

Why ignored?

- Arcuri et al reported their tuning may require weeks, or more, of CPU time[Arcuri' 2011].
- Wang et al. needed 15 years of CPU to explore 9.3 million candidate configurations for software clone detectors[Wang' 2013].

CPU intensive!

Why ignored?

CPU intensive!

Open issues (for SE data science)

Questions	Answers
Does tuning improve learners' performance?	
Does tuning change learners' rank?	
Is tuning very slow?	
Should data miner be used "off-the-shelf"?	

We'll get back to these… much later in the talk

Roadmap

- Expectation management
- Motivation
- Background
- Progress Report
- Future Work & Challenges

How to Tune?

- Mathematical Optimization
- Grid Search
- Evolutionary Algorithms

Mathematical Optimization

$$\begin{aligned} min \quad & f_0(X) \\ s.t. \quad & f_i(X) \leq b_i, i = 1, \ldots, m. \end{aligned}$$

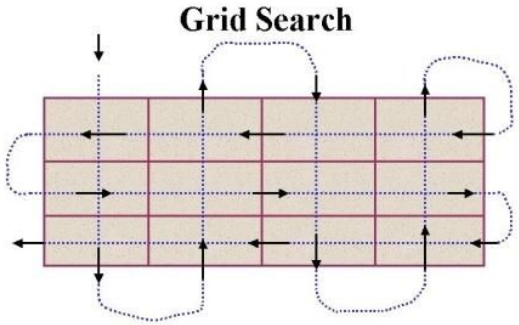
← Objective

← Constraint functions

- Based on the property of objective function and constraint function:
  - linear programing
  - non-linear programing
  - ….
- If differentiable, methods, like gradient descent, are to solve such problem
  - Linear regression and logistic regression
  - SVM, regularization parameter C[Cherkassky'2004]
- Note used much in SE because software not simple differential functions:
  - Also, issue of computational complexity of SE problems[Harman 2001]

Grid Search

- Divide all C configuration options into N values
- Evaluate N^C different combinations
  - Slow
  - Miss important optimization



9 parameter into 7 values
7^9= 40 million evaluations!

Evolutionary Algorithms
E.g. GA:

1. Population ← initializePopulation(N)
2. evaluatePopulation(Population)
3. While not stopCondition()
   a. Parents ← selectParents(Population)
   b. Offspring ← crossOverMutate(Parents)
   c. evaluateFitness(offspring)
   d. Replace least-fit population with new offspring
4. Return (Population)

E.g.: MOEA/D, NSGA-III, DE

Other approaches:
particle swarm, ant-colony, bee colony, etc.

Future work!

Differential Evolution

1. Population ← InitializePopulation(N)
2. evaluatePopulation(Population)
3. While not StopCondition()
   a. For parent in Population
      i. Offspring ← CrossOverMutate(parent)
      ii. Score ← evaluateFitness(Offspring)
      iii. If Score > parentScore
         1. Replace(parent, Offspring)
4. Return (Population)

- Pick three (X, Y, Z) from the population
- For i in len(parent)
- New[i] = X[i]+ f * (Y[i]-Z[i]) (e.g., f=0.75)
- If random < CR:(CR=0.3)
  - parent[i] = New[i]

Why DE?

* Competitive with particle swarm optimization and other GAs [Vesterstrom 2004]

* Easy to implement

What is software defect prediction?

Inputs = static code measures

amc	average method complexity	e.g. number of JAVA byte codes
avg_cc	average McCabe	average McCabe's cyclomatic complexity seen in class
ca	afferent couplings	how many other classes use the specific class.
cam	cohesion amongst classes	summation of number of different types of method parameters in every method divided by a multiplication of number of different method parameter types in whole class and number of methods.
cbm	coupling between methods	total number of new/redefined methods to which all the inherited methods are coupled
cbo	coupling between objects	increased when the methods of one class access services of another.
ce	efferent couplings	how many other classes is used by the specific class.
dam	data access	ratio of the number of private (protected) attributes to the total number of attributes
dit	depth of inheritance tree	
ic	inheritance coupling	number of parent classes to which a given class is coupled (includes counts of methods and variables inherited)
lcom	lack of cohesion in methods	number of pairs of methods that do not share a reference to an instance variable.
locm3	another lack of cohesion measure	if m, a are the number of *methods*, *attributes* in a class number and $\mu(a)$ is the number of methods accessing an attribute, then $lcom3 = ((\frac{1}{a}\sum_j^a \mu(a_j)) - m)/(1-m)$.
loc	lines of code	
max_cc	maximum McCabe	maximum McCabe's cyclomatic complexity seen in class
mfa	functional abstraction	number of methods inherited by a class plus number of methods accessible by member methods of the class
moa	aggregation	count of the number of data declarations (class fields) whose types are user defined classes
noc	number of children	
npm	number of public methods	
rfc	response for a class	number of methods invoked in response to a message to the object.
wmc	weighted methods per class	
defect	defect	Boolean: where defects found in post-release bug-tracking systems.

Output = defect predicted?

Defect Prediction

Easy to use:

- Static code attributes can be automatically collected[Nagappan 2005]

Widely used:

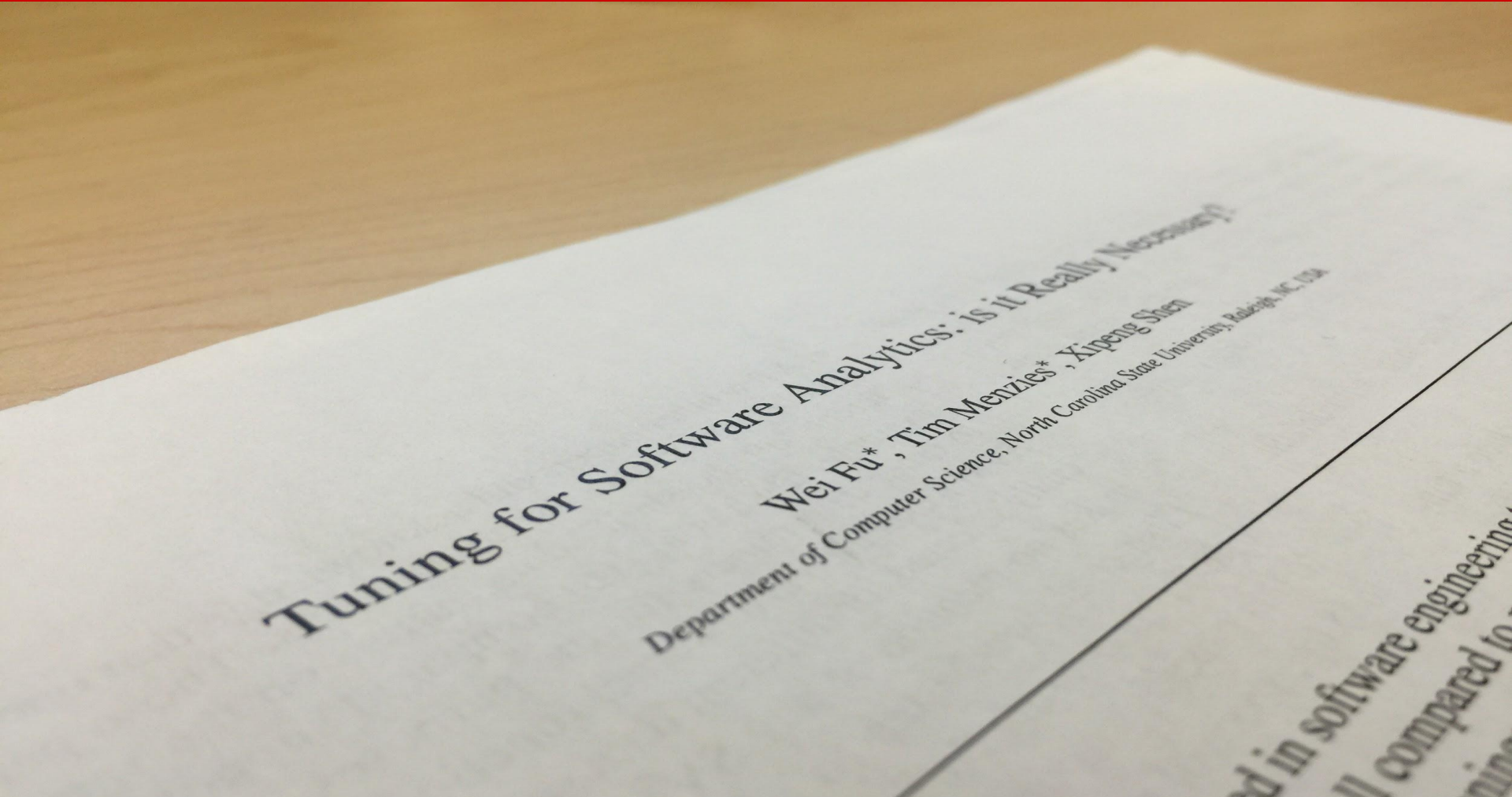
- Defect prediction models have been reported at Google[Lewis 2013]

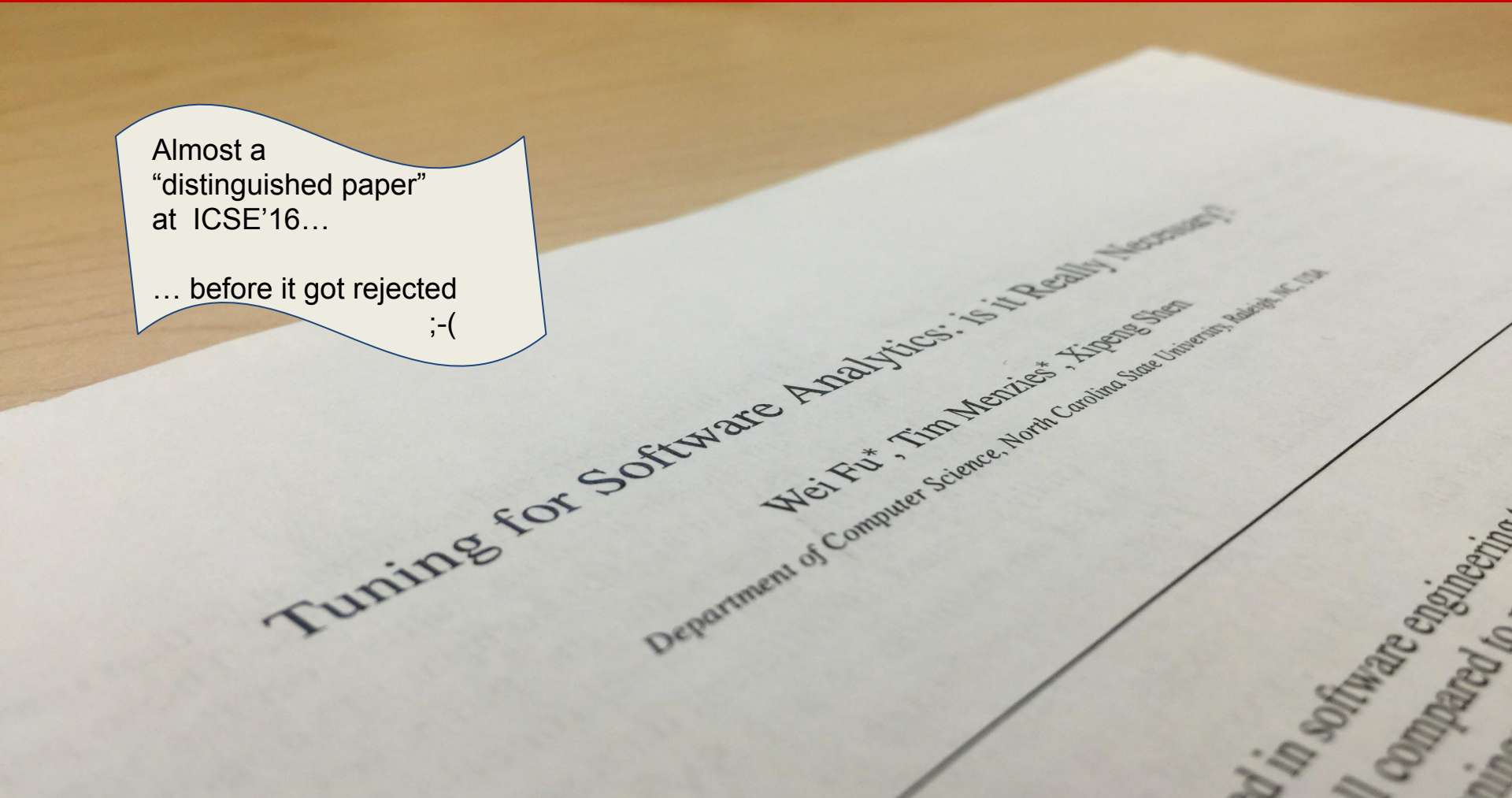
Useful:

- Competitive with supposedly more sophisticated technology [Rahman14]

Roadmap

- Expectation management
- Motivation
- Background
- Progress Report
- Future Work & Challenges


Tuning for Software Analytics: is it Really Necessary?
Wei Fu* , Tim Menzies* , Xipeng Shen
Department of Computer Science, North Carolina State University, Raleigh, NC, USA


Almost a
“distinguished paper”
at ICSE’16…
… before it got rejected
;-(
Tuning for Software Analytics: is it Really Necessary?
Wei Fu* , Tim Menzies* , Xipeng Shen
Department of Computer Science, North Carolina State University, Raleigh, NC, USA

Experiments

Tuning algorithm: Differential Evolution

457 citations since 2008

Defect Learners: From [Lessmann 2008]

- CART
- Random Forests
- WHERE-based learner[Menzies 2013]

WHERE-based learner: fastmap[Floutsos 1995] to cluster data + decision tree

Parameters in Defect Learners

Learner Name	Parameters	Default	Tuning Range	Description
Where-based Learner	threshold	0.5	[0.01,1]	The value to determine defective or not .
	infoPrune	0.33	[0.01,1]	The percentage of features to consider for the best split to build its final decision tree.
	min_sample_split	4	[1,10]	The minimum number of samples required to split an internal node of its final decision tree.
	min_Size	0.5	[0.01,1]	Finds min_samples_leaf in the initial clustering tree using $n\_samples^{min\_Size}$.
	wriggle	0.2	[0.01, 1]	The threshold to determine which branch in the initial clustering tree to be pruned
	depthMin	2	[1,6]	The minimum depth of the initial clustering tree below which no pruning for the clustering tree.
	depthMax	10	[1,20]	The maximum depth of the initial clustering tree.
	wherePrune	False	T/F	Whether or not to prune the initial clustering tree.
	treePrune	True	T/F	Whether or not to prune the final decision tree.
CART	threshold	0.5	[0,1]	The value to determine defective or not.
	max_feature	None	[0.01,1]	The number of features to consider when looking for the best split.
	min_sample_split	2	[2,20]	The minimum number of samples required to split an internal node.
	min_samples_leaf	1	[1,20]	The minimum number of samples required to be at a leaf node.
	max_depth	None	[1, 50]	The maximum depth of the tree.
Random Forests	threshold	0.5	[0.01,1]	The value to determine defective or not.
	max_feature	None	[0.01,1]	The number of features to consider when looking for the best split.
	max_leaf_nodes	None	[1,50]	Grow trees with max_leaf_nodes in best-first fashion.
	min_sample_split	2	[2,20]	The minimum number of samples required to split an internal node.
	min_samples_leaf	1	[1,20]	The minimum number of samples required to be at a leaf node.
	n_estimators	100	[50,150]	The number of trees in the forest.

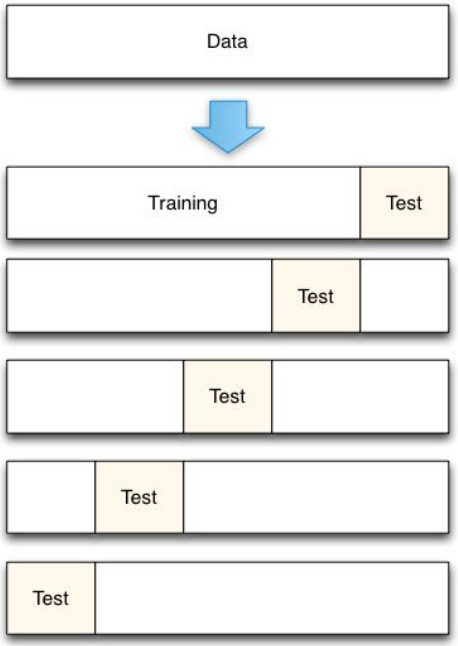
Data Sets

10 open source java projects from PROMISE repo[Menzies 2016]:

- ***ant , camel, ivy, jedit, log4j, lucene.***
- ***poi, synapse, velocity, xerces.***

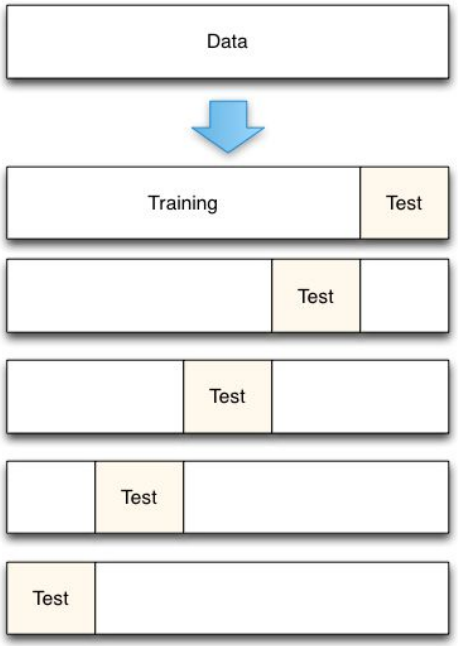
amc	average method complexity	e.g. number of JAVA byte codes
avg_cc	average McCabe	average McCabe's cyclomatic complexity seen in class
ca	afferent couplings	how many other classes use the specific class.
cam	cohesion amongst classes	summation of number of different types of method parameters in every method divided by a multiplication of number of different method parameter types in whole class and number of methods.
cbm	coupling between methods	total number of new/redefined methods to which all the inherited methods are coupled
cbo	coupling between objects	increased when the methods of one class access services of another.
ce	efferent couplings	how many other classes is used by the specific class.
dam	data access	ratio of the number of private (protected) attributes to the total number of attributes
dit	depth of inheritance tree	
ic	inheritance coupling	number of parent classes to which a given class is coupled (includes counts of methods and variables inherited)
lcom	lack of cohesion in methods	number of pairs of methods that do not share a reference to an instance variable.
locm3	another lack of cohesion measure	if m, a are the number of $methods, attributes$ in a class number and $\mu(a)$ is the number of methods accessing an attribute, then $lcom3 = ((\frac{1}{a}\sum_j^a \mu(a_j)) - m)/(1-m)$.
loc	lines of code	
max_cc	maximum McCabe	maximum McCabe's cyclomatic complexity seen in class
mfa	functional abstraction	number of methods inherited by a class plus number of methods accessible by member methods of the class
moa	aggregation	count of the number of data declarations (class fields) whose types are user defined classes
noc	number of children	
npm	number of public methods	
rfc	response for a class	number of methods invoked in response to a message to the object.
wmc	weighted methods per class	
defect	defect	Boolean: where defects found in post-release bug-tracking systems.

How to Design Experiment



Cross-Validation

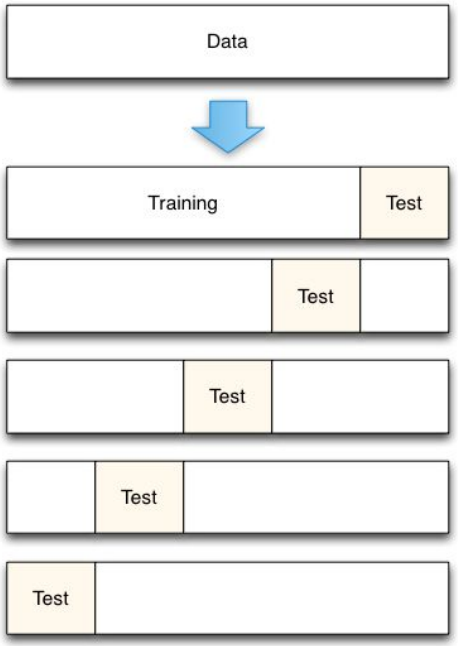
How to Design Experiment



Cross-Validation

- Slow (e.g.leave-one-out)

How to Design Experiment



Cross-Validation

- Slow (e.g.leave-one-out)
- Mix up older data and newer data

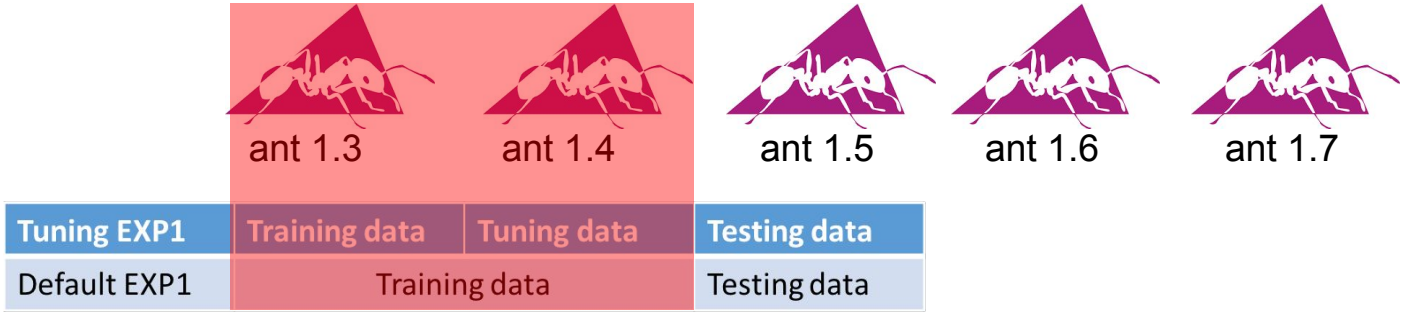
Data from the past might be used to test on future data ☹

Our Design

- Incremental learning approach

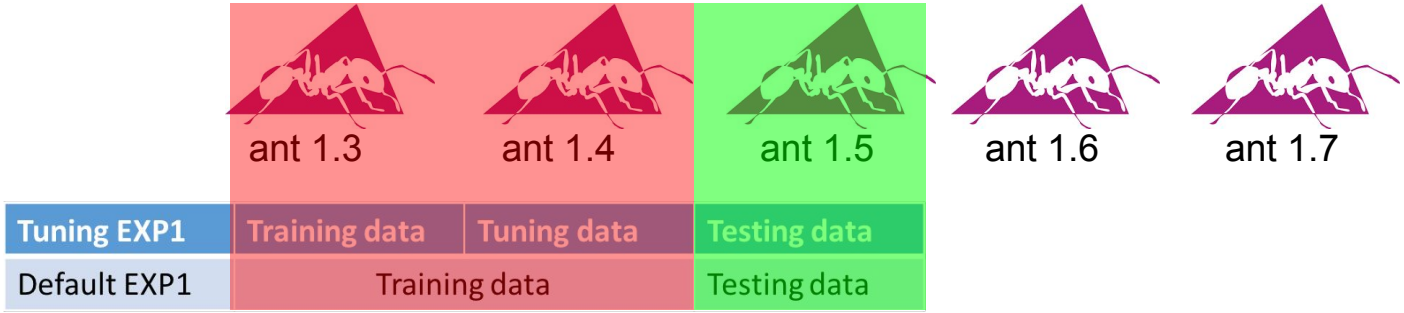
Our Design

- Incremental learning approach



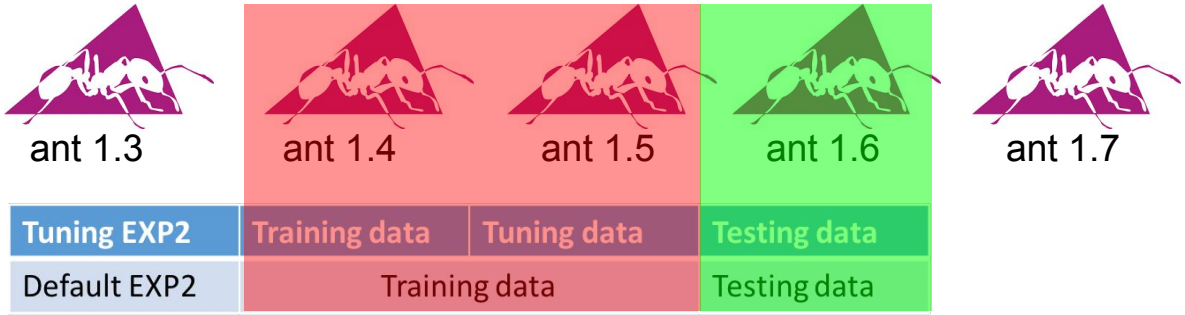
Our Design

- Incremental learning approach



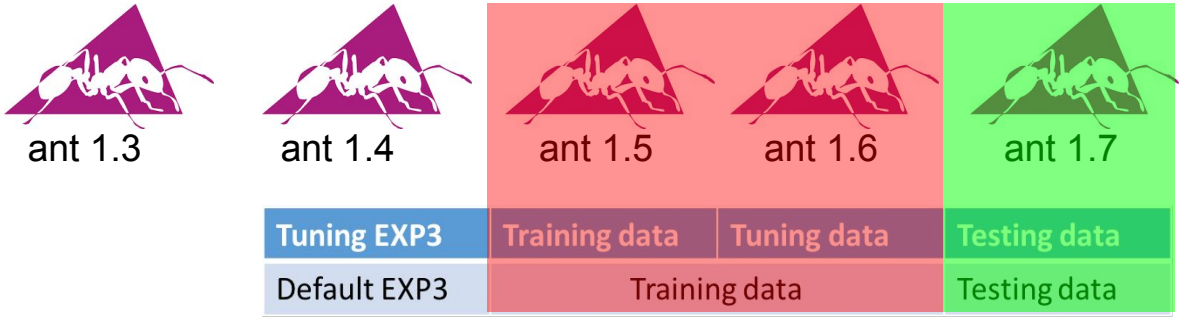
Our Design

- Incremental learning approach



Our Design

- Incremental learning approach



Tuning Goals

		Predicted results	
	Total	Predicted positive	Predicted negative
Actual results	Actual positive	True positive(D)	False negative(B)
	Actual negative	False positive(C)	True negative(A)

Pd = D/(B+D)

Pf = C/(A+C)

Precision = D/(D+C)

F-measure = 2* pd*precision/(pd+precision)

Tuning Goals

		Predicted results	
	Total	Predicted positive	Predicted negative
Actual results	Actual positive	True positive(D)	False negative(B)
	Actual negative	False positive(C)	True negative(A)

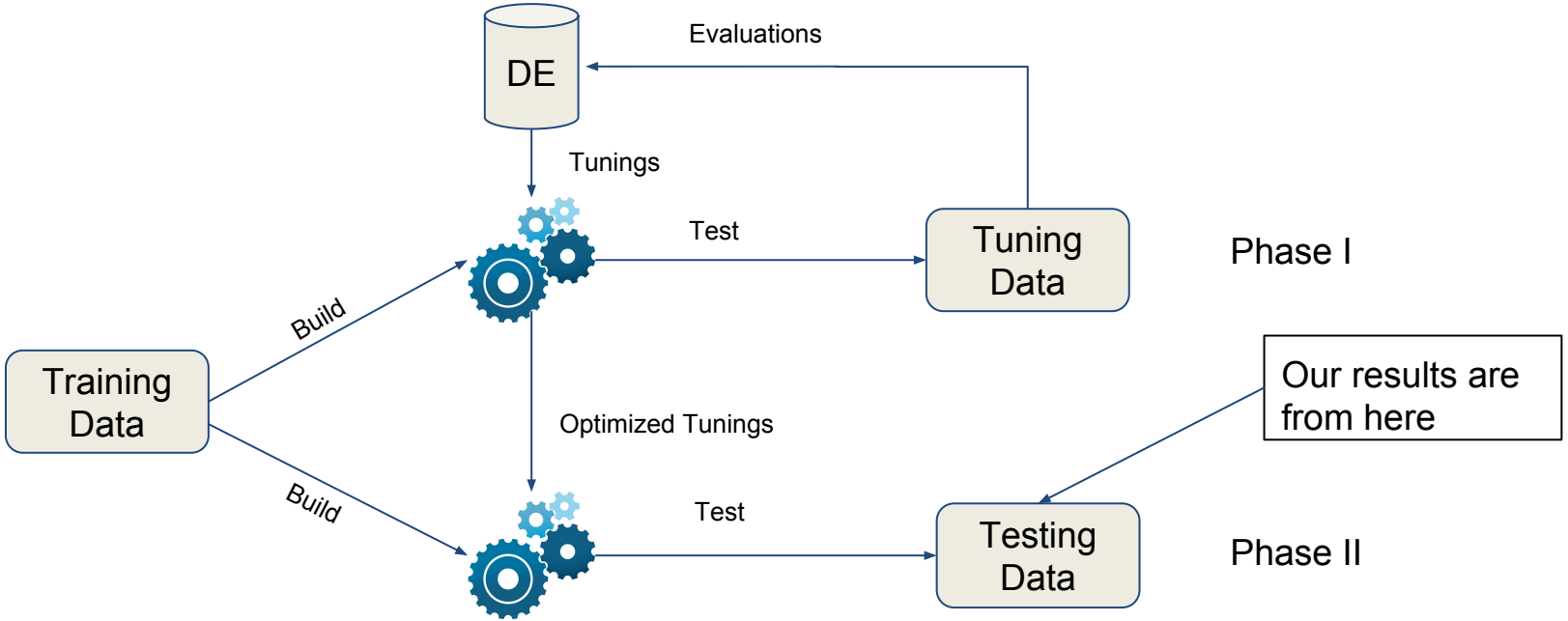
Pd = D/(B+D)

Pf = C/(A+C)

Precision = D/(D+C)

F-measure = 2* pd*precision/(pd+precision)

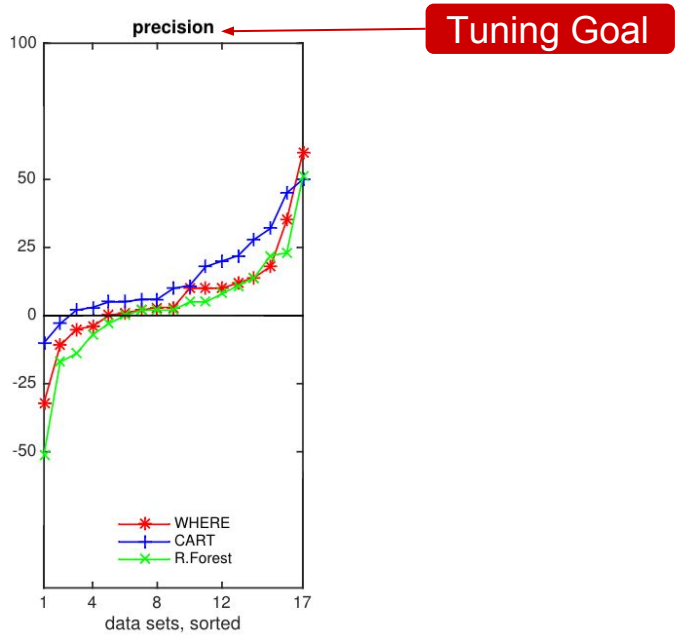
Experiment


DE
Evaluations
Tunings
Test
Tuning Data
Phase I
Training Data
Build
Build
Optimized Tunings
Test
Testing Data
Phase II
Our results are from here

Research Questions

Questions	Answers
Does tuning improve learners' performance?	
Does tuning change learners' rank?	
Is tuning very slow?	
Should data miner be used "off-the-shelf"?	

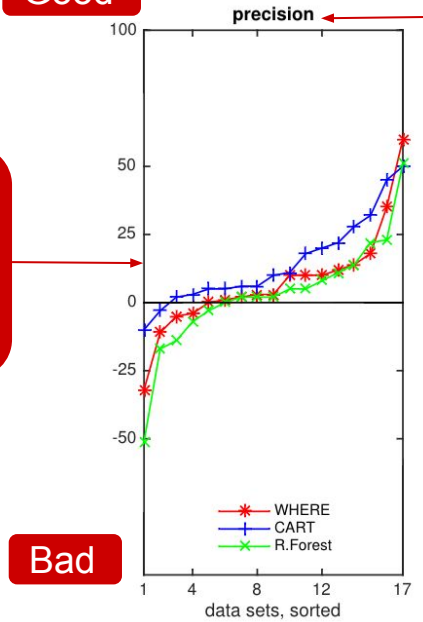
Does Tuning Improve Performance?



Does Tuning Improve Performance?

Good

Tuning Goal



Bad

For each point/data set:

Y = goal(tuned_learner) - goal(untuned_learner)

Y > 0: tuning improves the performance

Y< 0: tuning does worse

Does Tuning Improve Performance?

Good

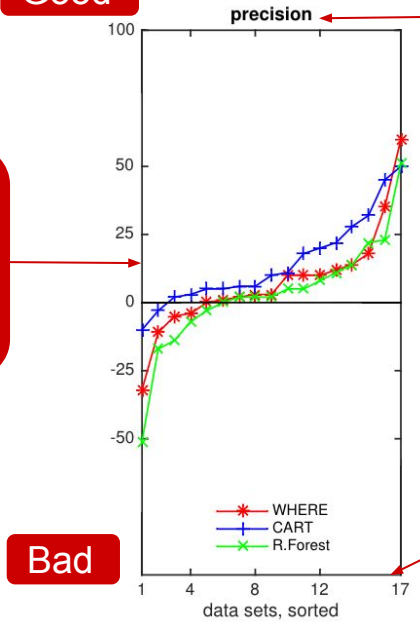
Tuning Goal

For each point/data set:

Y = goal(tuned_learner) - goal(untuned_learner)

Y > 0: tuning improves the performance

Y< 0: tuning does worse



Bad

17 data sets, sorted by improvements(x-axis)

Does Tuning Improve Performance?

Good

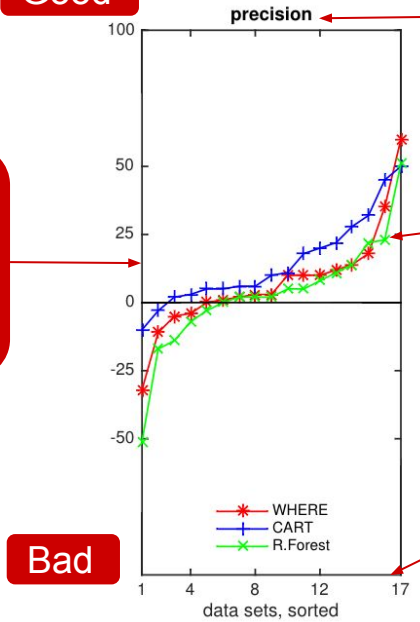
Tuning Goal

For each point/data set:

Y = goal(tuned_learner) - goal(untuned_learner)

Y > 0: tuning improves the performance

Y< 0: tuning does worse

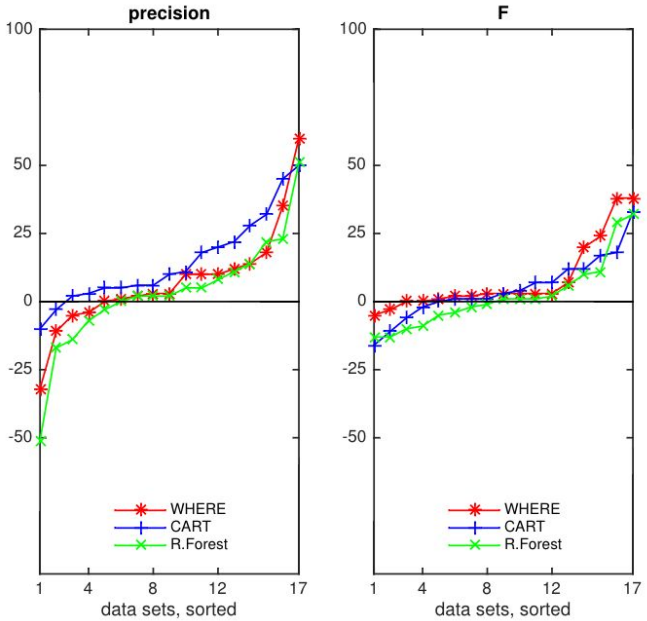


Lines are sorted independently, the right side is always greater than the left sigd

17 data sets, sorted by improvements(x-axis)

Bad

Does Tuning Improve Performance?



Learner	Precision	F-measure
WHERE	13/17	15/17
CART	15/17	13/17
Random Forest	12/17	10/17

Research Questions

Questions	Answers
Does tuning improve learners' performance?	
Does tuning change learners' rank?	
Is tuning very slow?	
Should data miner be used "off-the-shelf"?	

Research Questions

Questions	Answers
Does tuning improve learners' performance?	Yes
Does tuning change learners' rank?	
Is tuning very slow?	
Should data miner be used "off-the-shelf"?	

Does Tuning Change Learners' Rank?

	WHERE		CART		Random Forest	
Data set	default	Tuned	default	Tuned	default	Tuned
antV0	0	35	15	**60**	21	44
antV1	0	60	54	56	**67**	50
antV2	45	55	42	52	56	**67**
camelV0	20	30	30	50	28	**79**
camelV1	27	28	**38**	28	34	27
ivy	25	21	21	**26**	23	20
jeditV0	34	37	56	**78**	52	60
jeditV1	30	42	32	**64**	32	37
jeditV2	4	**22**	6	17	4	6
log4j	96	91	95	98	95	**100**
lucene	61	75	67	70	63	**77**
poiV0	70	70	65	**71**	67	69
poiV1	74	76	72	90	78	**100**
synapse	61	50	50	**100**	60	60
velocity	34	**44**	39	**44**	40	42
xercesV0	14	17	17	14	**28**	14
xercesV1	86	54	72	**100**	78	27

Table 4: Precision results (best results shown in bold).

Tuning Goal

Does Tuning Change Learners' Rank?

	WHERE		CART		Random Forest	
Data set	default	Tuned	default	Tuned	default	Tuned
antV0	0	35	15	**60**	21	44
antV1	0	60	54	56	**67**	50
V2	45	55	42	52	56	**67**
V0	20	30	30	50	28	**79**
camelV1	27	28	**38**	28	34	27
ivy	25	21	21	**26**	23	20
jeditV0	34	37	56	**78**	52	60
jeditV1	30	42	32	**64**	32	37
jeditV2	4	**22**	6	17	4	6
log4j	96	91	95	98	95	**100**
lucene	61	75	67	70	63	**77**
poiV0	70	70	65	**71**	67	69
poiV1	74	76	72	90	78	**100**
synapse	61	50	50	**100**	60	60
velocity	34	**44**	39	**44**	40	42
xercesV0	14	17	17	14	**28**	14
xercesV1	86	54	72	**100**	78	27

Test Data

Table 4: Precision results (best results shown in bold).

Tuning Goal

Does Tuning Change Learners' Rank?

	WHERE		CART		Random Forest	
Data set	default	Tuned	default	Tuned	default	Tuned
antV0	0	35	15	**60**	21	44
antV1	0	60	54	56	**67**	50
[illegible]V2	45	55	42	52	56	**67**
[illegible]V0	20	30	30	50	28	**79**
camelV1	27	28	**38**	28	34	27
ivy	25	21	21	**26**	23	20
jeditV0	34	37	56	**78**	52	60
jeditV1	30	42	32	**64**	32	37
jeditV2	4	**22**	6	17	4	6
log4j	96	91	95	98	95	**100**
lucene	61	75	67	70	63	**77**
poiV0	70	70	65	**71**	67	69
poiV1	74	76	72	90	78	**100**
synapse	61	50	50	**100**	60	60
velocity	34	**44**	39	**44**	40	42
xercesV0	14	17	17	14	**28**	14
xercesV1	86	54	72	**100**	78	27

Test Data

Best result in bold

Table 4: Precision results (best results shown in bold).

Tuning Goal

Does Tuning Change Learners' Rank?

	WHERE		CART		Random Forest	
Data set	default	Tuned	default	Tuned	default	Tuned
antV0	0	35	15	**60**	21	44
antV1	0	60	54	56	**67**	50
[illegible]tV2	45	55	42	52	56	**67**
[illegible]lV0	20	30	30	50	28	**79**
camelV1	27	28	**38**	28	34	27
ivy	25	21	21	**26**	23	20
jeditV0	34	37	56	**78**	52	60
jeditV1	30	42	32	**64**	32	37
jeditV2	4	**22**	6	17	4	6
log4j	96	91	95	98	95	**100**
lucene	61	75	67	70	63	**77**
poiV0	70	70	65	**71**	67	69
poiV1	74	76	72	90	78	**100**
synapse	61	50	50	**100**	60	60
velocity	34	**44**	39	**44**	40	42
xercesV0	14	17	17	14	**28**	14
xercesV1	86	54	72	**100**	78	27

Table 4: Precision results (best results shown in bold).

Test Data

Best result in bold

Results of tuning parameters

Tuning Goal

Does Tuning Change Learners' Rank?

	WHERE		CART		Random Forest	
Data set	default	Tuned	default	Tuned	default	Tuned
antV0	0	35	15	**60**	21	44
antV1	0	60	54	56	**67**	50
[illegible]tV2	45	55	42	52	56	**67**
[illegible]lV0	20	30	30	50	28	**79**
camelV1	27	28	**38**	28	34	27
ivy	25	21	21	**26**	23	20
jeditV0	34	37	56	**78**	52	60
jeditV1	30	42	32	**64**	32	37
jeditV2	4	**22**	6	17	4	6
log4j	96	91	95	98	95	**100**
lucene	61	75	67	70	63	**77**
poiV0	70	70	65	**71**	67	69
poiV1	74	76	72	90	78	**100**
synapse	61	50	50	**100**	60	60
velocity	34	**44**	39	**44**	40	42
xercesV0	14	17	17	14	**28**	14
xercesV1	86	54	72	**100**	78	27

Table 4: Precision results (best results shown in bold).

Test Data

Best result in bold

Results of tuning parameters

Results of default parameters

Tuning Goal

Does Tuning Change Learners' Rank?

	WHERE		CART		Random Forest	
Data set	default	Tuned	default	Tuned	default	Tuned
antV0	0	35	15	**60**	21	44
antV1	0	60	54	56	**67**	50
antV2	45	55	42	52	56	**67**
camelV0	20	30	30	50	28	**79**
camelV1	27	28	**38**	28	34	27
ivy	25	21	21	**26**	23	20
jeditV0	34	37	56	**78**	52	60
jeditV1	30	42	32	**64**	32	37
jeditV2	4	**22**	6	17	4	6
log4j	96	91	95	98	95	**100**
lucene	61	75	67	70	63	**77**
poiV0	70	70	65	**71**	67	69
poiV1	74	76	72	90	78	**100**
synapse	61	50	50	**100**	60	60
velocity	34	**44**	39	**44**	40	42
xercesV0	14	17	17	14	**28**	14
xercesV1	86	54	72	**100**	78	27

Table 4: Precision results (best results shown in bold).

Random Forests is better than CART for defect prediction[Lessmann 2008].

Does Tuning Change Learners' Rank?

	WHERE		CART		Random Forest	
Data set	default	Tuned	default	Tuned	default	Tuned
antV0	0	35	15	**60**	21	44
antV1	0	60	54	56	**67**	50
antV2	45	55	42	52	56	**67**
camelV0	20	30	30	50	28	**79**
camelV1	27	28	**38**	28	34	27
ivy	25	21	21	**26**	23	20
jeditV0	34	37	56	**78**	52	60
jeditV1	30	42	32	**64**	32	37
jeditV2	4	**22**	6	17	4	6
log4j	96	91	95	98	95	**100**
lucene	61	75	67	70	63	**77**
poiV0	70	70	65	**71**	67	69
poiV1	74	76	72	90	78	**100**
synapse	61	50	50	**100**	60	60
velocity	34	**44**	39	**44**	40	42
xercesV0	14	17	17	14	**28**	14
xercesV1	86	54	72	**100**	78	27

Table 4: Precision results (best results shown in bold).

Random Forests is better than CART for defect prediction[Lessmann 2008].

Yes RF is better than CART in ***12/17*** data sets

Does Tuning Change Learners' Rank?

	WHERE		CART		Random Forest	
Data set	default	Tuned	default	Tuned	default	Tuned
antV0	0	35	15	**60**	21	44
antV1	0	60	54	56	**67**	50
antV2	45	55	42	52	56	**67**
camelV0	20	30	30	50	28	**79**
camelV1	27	28	**38**	28	34	27
ivy	25	21	21	**26**	23	20
jeditV0	34	37	56	**78**	52	60
jeditV1	30	42	32	**64**	32	37
jeditV2	4	**22**	6	17	4	6
log4j	96	91	95	98	95	**100**
lucene	61	75	67	70	63	**77**
poiV0	70	70	65	**71**	67	69
poiV1	74	76	72	90	78	**100**
synapse	61	50	50	**100**	60	60
velocity	34	**44**	39	**44**	40	42
xercesV0	14	17	17	14	**28**	14
xercesV1	86	54	72	**100**	78	27

Table 4: Precision results (best results shown in bold).

Random Forests is better than CART for defect prediction[Lessmann 2008].

Yes RF is better than CART in ***12/17*** data sets

After Tuning

No! RF is better than CART in ***5/17*** data sets

Does Tuning Change Learners' Rank?

	WHERE		CART		Random Forest	
Data set	default	Tuned	default	Tuned	default	Tuned
antV0	0	20	20	**40**	28	38
antV1	0	38	37	**49**	38	**49**
antV2	47	50	45	49	**57**	56
camelV0	31	28	39	28	**40**	30
camelV1	34	34	38	32	**42**	33
ivy	39	34	28	**40**	35	33
jeditV0	45	47	56	57	**63**	59
jeditV1	43	44	44	47	46	**48**
jeditV2	8	**11**	10	10	8	9
log4j	47	50	53	37	**60**	47
lucene	73	73	65	72	70	**76**
poiV0	50	74	31	64	45	**77**
poiV1	75	**78**	68	69	77	**78**
synapse	49	56	43	**60**	52	53
velocity	51	53	53	51	**56**	51
xercesV0	19	22	19	**26**	34	21
xercesV1	32	70	34	35	42	**71**

Table 5: F-measure results (best results shown in bold).

Random Forests is better than CART for defect prediction[Lessmann 2008].

Yes RF is better than CART in ***16/17*** data sets

After Tuning

No! RF is better than CART in ***9/17*** data sets

SAME pattern found in F-measure results

Research Questions

Questions	Answers
Does tuning improve learners' performance?	Yes
Does tuning change learners' rank?	
Is tuning very slow?	
Should data miner be used "off-the-shelf"?	

Research Questions

Questions	Answers
Does tuning improve learners' performance?	Yes
Does tuning change learners' rank?	Yes
Is tuning very slow?	
Should data miner be used "off-the-shelf"?	

Is Tuning Impractically Slow?

Runtimes in sec

Datasets	Tuned_Where	Naive_Where	Tuned_CART	Naive_CART	Tuned_RanFst	Naive_RanFst
antV0	50 / 95.47	1.65	60 / 5.08	0.08	60 / 9.78	0.20
antV1	60 / 224.67	3.03	50 / 6.52	0.12	60 / 14.13	0.25
antV2	70 / 644.99	8.24	50 / 9.00	0.24	60 / 16.75	0.44
camelV0	70 / 690.62	7.93	70 / 12.68	0.24	110 / 28.49	0.34
camelV1	60 / 1596.77	23.56	60 / 17.13	0.27	70 / 33.96	0.77
ivy	60 / 66.69	0.97	60 / 4.26	0.07	60 / 8.89	0.19
jeditV0	80 / 459.30	5.33	80 / 8.69	0.11	90 / 18.40	0.32
jeditV1	60 / 421.56	6.59	80 / 9.05	0.12	80 / 17.93	0.36
jeditV2	90 / 595.56	6.88	60 / 7.90	0.14	110 / 27.34	0.38
log4j	50 / 76.09	1.33	50 / 2.60	0.06	80 / 9.69	0.15
lucene	80 / 236.45	2.60	70 / 6.07	0.10	60 / 9.77	0.25
poiV0	60 / 263.12	3.92	70 / 7.42	0.09	130 / 25.86	0.28
poiV1	50 / 398.33	6.94	70 / 9.31	0.13	50 / 12.67	0.29
synapse	70 / 144.09	1.85	50 / 3.88	0.07	50 / 8.13	0.19
velocity	60 / 184.10	2.68	50 / 4.27	0.07	100 / 15.18	0.21
xercesV0	60 / 136.87	1.98	80 / 9.17	0.10	70 / 14.17	0.22
xercesV1	80 / 1173.92	12.78	60 / 10.47	0.16	50 / 18.27	0.40

Evaluations/Runtimes for tuned and default learners with DE(in sec), optimizing for F

Is Tuning Impractically Slow?

of evaluations

Datasets	Tuned_Where	Naive_Where	Tuned_CART	Naive_CART	Tuned_RanFst	Naive_RanFst
antV0	50 / 95.47	1.65	60 / 5.08	0.08	60 / 9.78	0.20
antV1	60 / 224.67	3.03	50 / 6.52	0.12	60 / 14.13	0.25
antV2	70 / 644.99	8.24	50 / 9.00	0.24	60 / 16.75	0.44
camelV0	70 / 690.62	7.93	70 / 12.68	0.24	110 / 28.49	0.34
camelV1	60 / 1596.77	23.56	60 / 17.13	0.27	70 / 33.96	0.77
ivy	60 / 66.69	0.97	60 / 4.26	0.07	60 / 8.89	0.19
jeditV0	80 / 459.30	5.33	80 / 8.69	0.11	90 / 18.40	0.32
jeditV1	60 / 421.56	6.59	80 / 9.05	0.12	80 / 17.93	0.36
jeditV2	90 / 595.56	6.88	60 / 7.90	0.14	110 / 27.34	0.38
log4j	50 / 76.09	1.33	50 / 2.60	0.06	80 / 9.69	0.15
lucene	80 / 236.45	2.60	70 / 6.07	0.10	60 / 9.77	0.25
poiV0	60 / 263.12	3.92	70 / 7.42	0.09	130 / 25.86	0.28
poiV1	50 / 398.33	6.94	70 / 9.31	0.13	50 / 12.67	0.29
synapse	70 / 144.09	1.85	50 / 3.88	0.07	50 / 8.13	0.19
velocity	60 / 184.10	2.68	50 / 4.27	0.07	100 / 15.18	0.21
xercesV0	60 / 136.87	1.98	80 / 9.17	0.10	70 / 14.17	0.22
xercesV1	80 / 1173.92	12.78	60 / 10.47	0.16	50 / 18.27	0.40

Evaluations/Runtimes for tuned and default learners with DE(in sec), optimizing for F

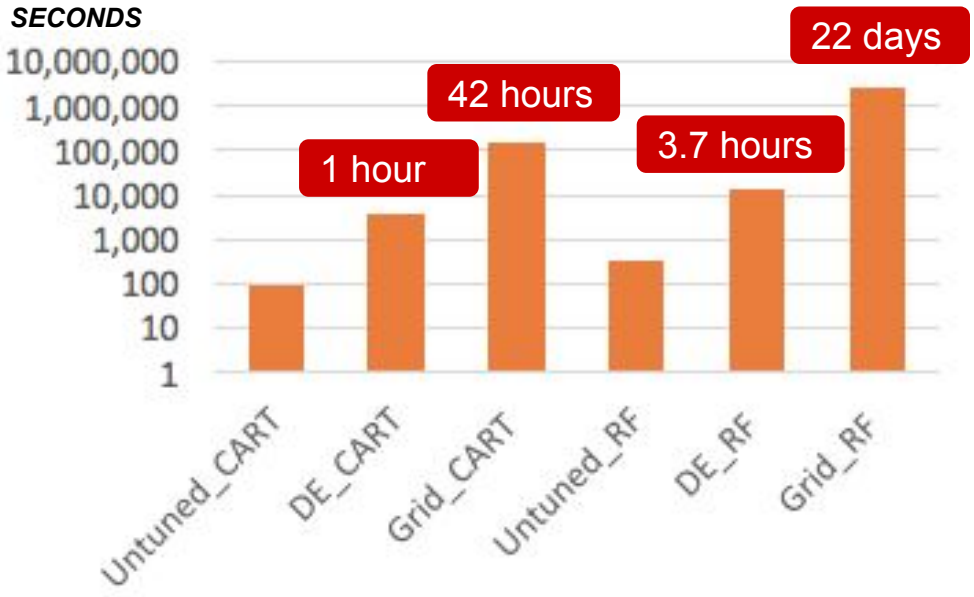
Is Tuning Impractically Slow?

Evaluations: 50~100

Datasets	Tuned_Where	Naive_Where	Tuned_CART	Naive_CART	Tuned_RanFst	Naive_RanFst
antV0	50 / 95.47	1.65	60 / 5.08	0.08	60 / 9.78	0.20
antV1	60 / 224.67	3.03	50 / 6.52	0.12	60 / 14.13	0.25
antV2	70 / 644.99	8.24	50 / 9.00	0.24	60 / 16.75	0.44
camelV0	70 / 690.62	7.93	70 / 12.68	0.24	110 / 28.49	0.34
camelV1	60 / 1596.77	23.56	60 / 17.13	0.27	70 / 33.96	0.77
ivy	60 / 66.69	0.97	60 / 4.26	0.07	60 / 8.89	0.19
jeditV0	80 / 459.30	5.33	80 / 8.69	0.11	90 / 18.40	0.32
jeditV1	60 / 421.56	6.59	80 / 9.05	0.12	80 / 17.93	0.36
jeditV2	90 / 595.56	6.88	60 / 7.90	0.14	110 / 27.34	0.38
log4j	50 / 76.09	1.33	50 / 2.60	0.06	80 / 9.69	0.15
lucene	80 / 236.45	2.60	70 / 6.07	0.10	60 / 9.77	0.25
poiV0	60 / 263.12	3.92	70 / 7.42	0.09	130 / 25.86	0.28
poiV1	50 / 398.33	6.94	70 / 9.31	0.13	50 / 12.67	0.29
synapse	70 / 144.09	1.85	50 / 3.88	0.07	50 / 8.13	0.19
velocity	60 / 184.10	2.68	50 / 4.27	0.07	100 / 15.18	0.21
xercesV0	60 / 136.87	1.98	80 / 9.17	0.10	70 / 14.17	0.22
xercesV1	80 / 1173.92	12.78	60 / 10.47	0.16	50 / 18.27	0.40

Evaluations/Runtimes for tuned and default learners with DE(in sec), optimizing for F

Is Tuning Impractically Slow?



Runtimes for tuned and default learners with DE and GridSearch(in sec), optimizing for F & precision

Research Questions

Questions	Answers
Does tuning improve learners' performance?	Yes
Does tuning change learners' rank?	Yes
Is tuning very slow?	
Should data miner be used "off-the-shelf"?	

Research Questions

Questions	Answers
Does tuning improve learners' performance?	Yes
Does tuning change learners' rank?	Yes
Is tuning very slow?	No!
Should data miner be used "off-the-shelf"?	

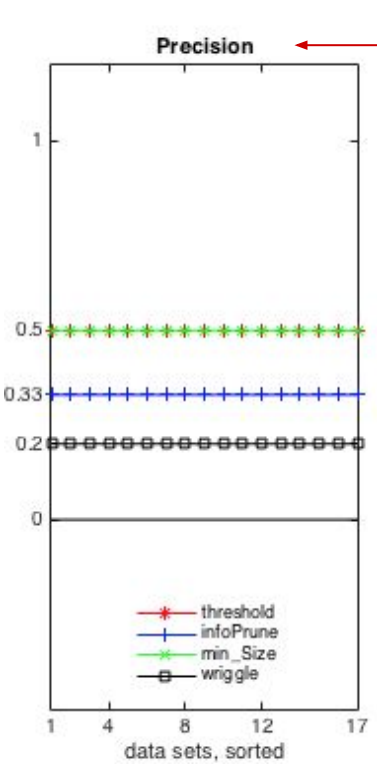
Should We Use “off-the-shelf” Tunings?

Learner Name	Parameters	Default	Tuning Range	Description
Where-based Learner	threshold	0.5	[0.01,1]	The value to determine defective or not .
	infoPrune	0.33	[0.01,1]	The percentage of features to consider for the best split to build its final decision tree.
	min_sample_split	4	[1,10]	The minimum number of samples required to split an internal node of its final decision tree.
	min_Size	0.5	[0.01,1]	Finds min_samples_leaf in the initial clustering tree using $n\_samples^{min\_Size}$.
	wriggle	0.2	[0.01, 1]	The threshold to determine which branch in the initial clustering tree to be pruned
	depthMin	2	[1,6]	The minimum depth of the initial clustering tree below which no pruning for the clustering tree.
	depthMax	10	[1,20]	The maximum depth of the initial clustering tree.
	wherePrune	False	T/F	Whether or not to prune the initial clustering tree.
	treePrune	True	T/F	Whether or not to prune the final decision tree.

Select four representatives

- threshold: 0.5
- infoPrune: 0.33
- min_Size : 0.5
- wriggle: 0.2

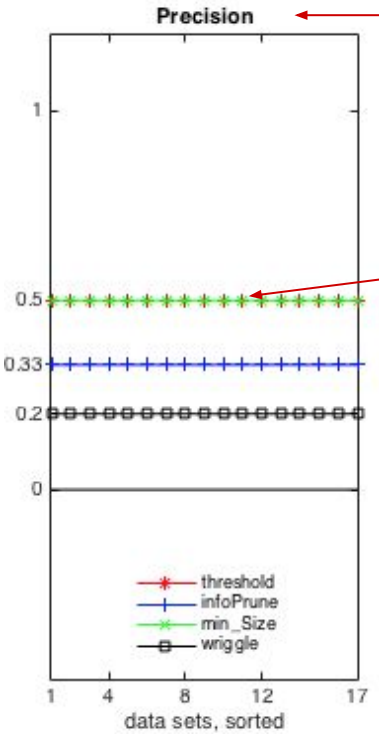
Should We Use “off-the-shelf” Tunings?



Default settings

- threshold: 0.5
- infoPrune: 0.33
- min_Size : 0.5
- wriggle: 0.2

Should We Use “off-the-shelf” Tunings?



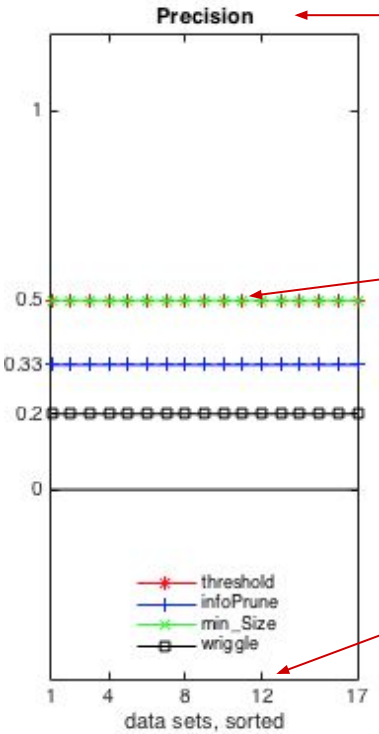
Tuning Goal

y axis is the values of this parameter for each test data

Default settings

- threshold: 0.5
- infoPrune: 0.33
- min_Size : 0.5
- wriggle: 0.2

Should We Use “off-the-shelf” Tunings?



Tuning Goal

y axis is the values of this parameter for each test data

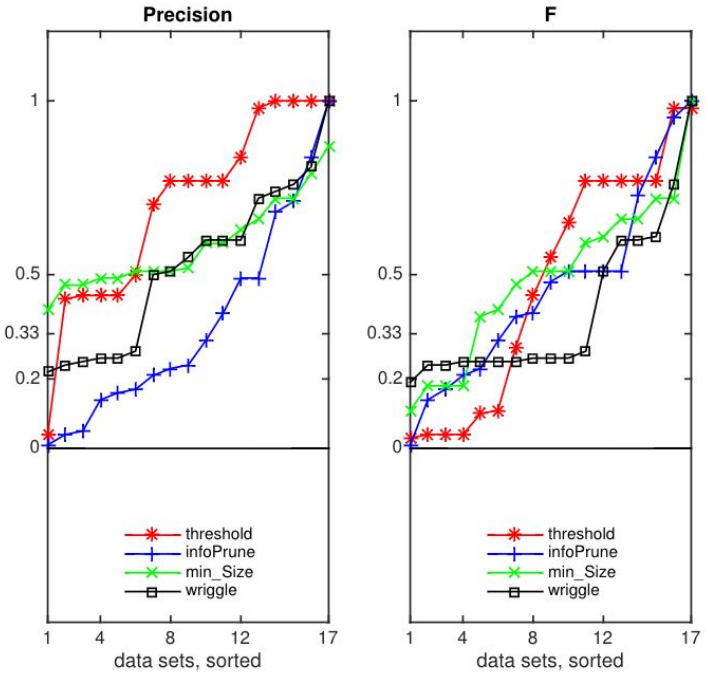
17 data sets, sorted by y values for each parameter.

Default settings

- threshold: 0.5
- infoPrune: 0.33
- min_Size : 0.5
- wriggle: 0.2

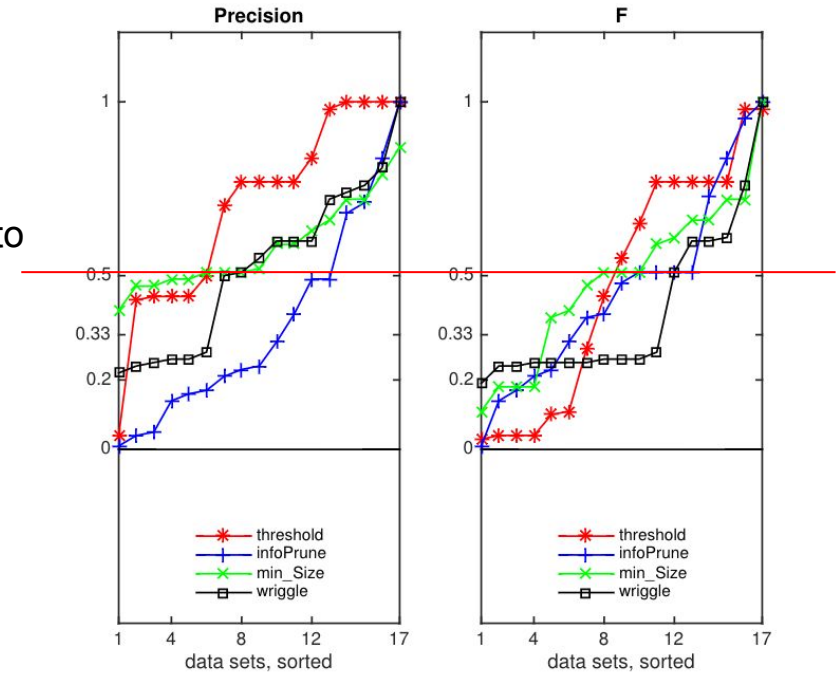
Should We Use “off-the-shelf” Tunings?

- Tunings learned were different
  - in different data sets
  - for different goals.
- Tunings learned by DE were often very different to the default.
  - **threshold: 0.5**
  - **min_Size : 0.5**
  - **infoPrune: 0.33**
  - wriggle: 0.2



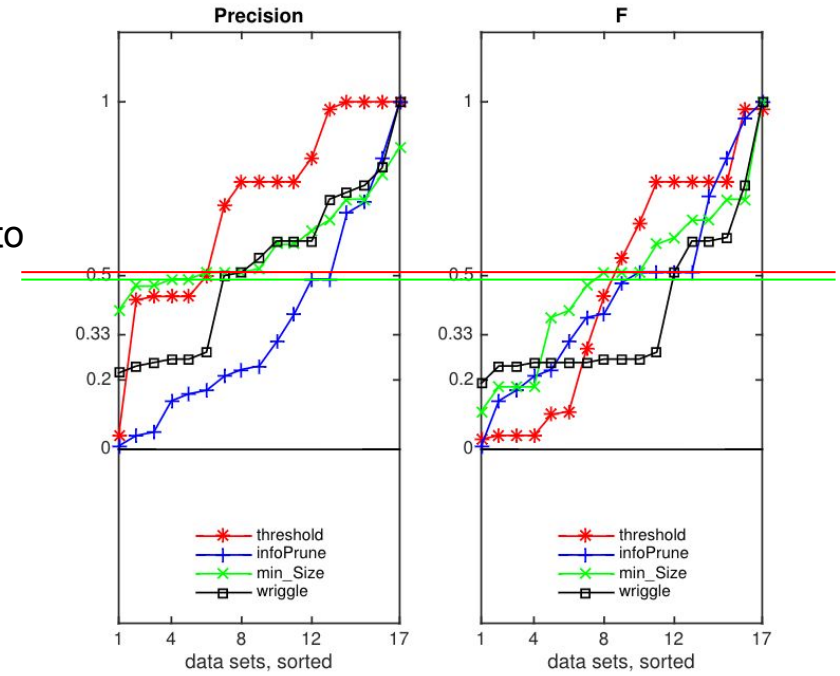
Should We Use “off-the-shelf” Tunings?

- Tunings learned were different
  - in different data sets
  - for different goals.
- Tunings learned by DE were often very different to the default.
  - **threshold: 0.5**
  - **min_Size : 0.5**
  - **infoPrune: 0.33**
  - wriggle: 0.2



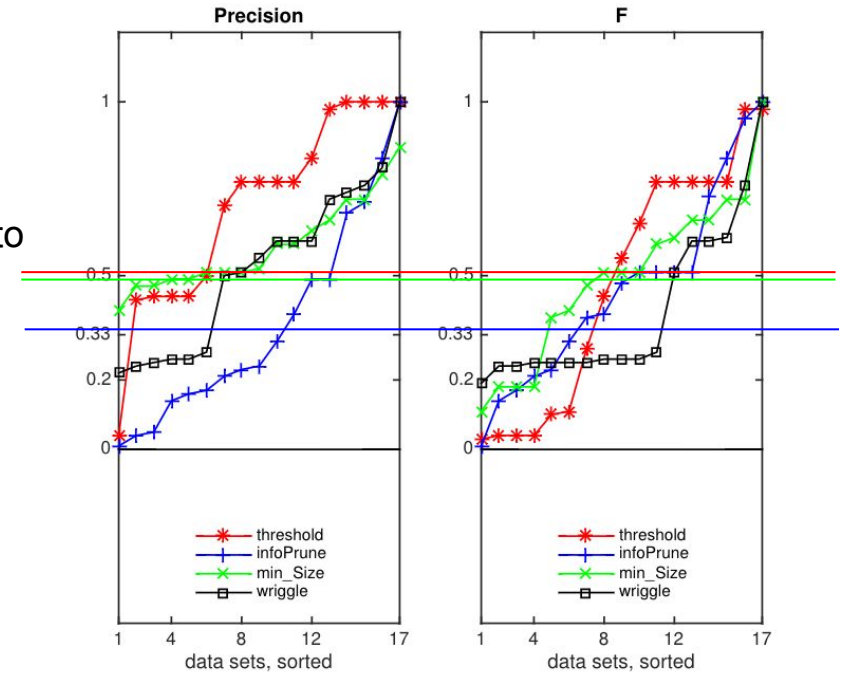
Should We Use “off-the-shelf” Tunings?

- Tunings learned were different
  - in different data sets
  - for different goals.
- Tunings learned by DE were often very different to the default.
  - **threshold: 0.5**
  - **min_Size : 0.5**
  - **infoPrune: 0.33**
  - wriggle: 0.2



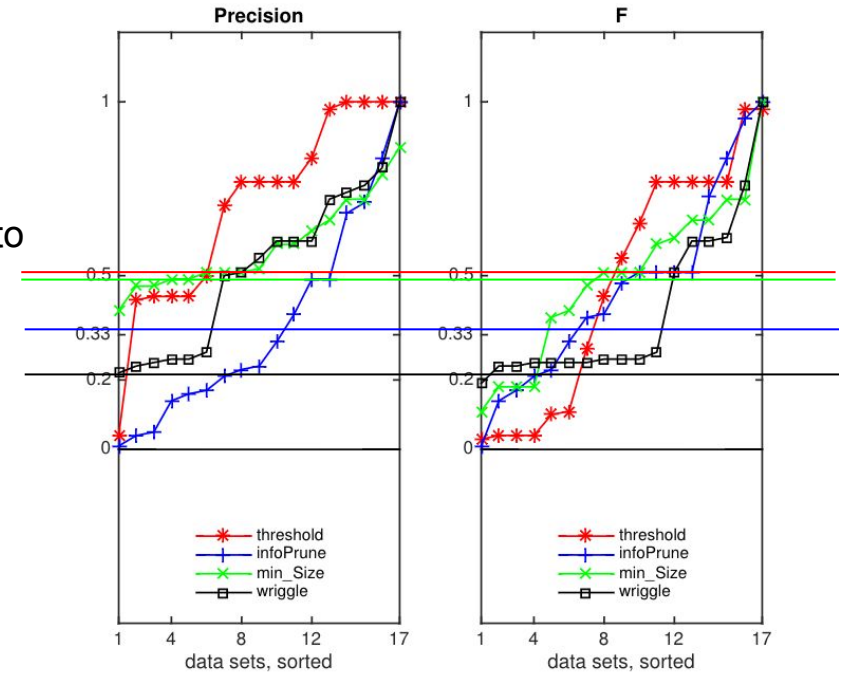
Should We Use “off-the-shelf” Tunings?

- Tunings learned were different
  - in different data sets
  - for different goals.
- Tunings learned by DE were often very different to the default.
  - **threshold: 0.5**
  - **min_Size : 0.5**
  - **infoPrune: 0.33**
  - wriggle: 0.2



Should We Use “off-the-shelf” Tunings?

- Tunings learned were different
  - in different data sets
  - for different goals.
- Tunings learned by DE were often very different to the default.
  - **threshold: 0.5**
  - **min_Size : 0.5**
  - **infoPrune: 0.33**
  - wriggle: 0.2



Research Questions

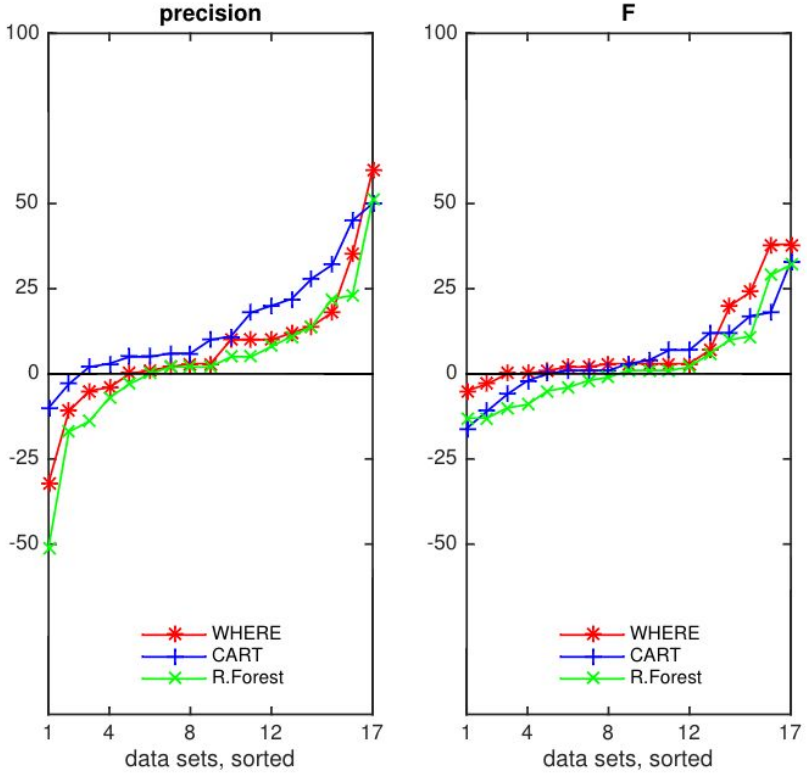
Questions	Answers
Does tuning improve learners' performance?	Yes
Does tuning change learners' rank?	Yes
Is tuning very slow?	No!
Should data miner be used "off-the-shelf"?	

Research Questions

Questions	Answers
Does tuning improve learners' performance?	Yes
Does tuning change learners' rank?	Yes
Is tuning very slow?	No!
Should data miner be used "off-the-shelf"?	No!

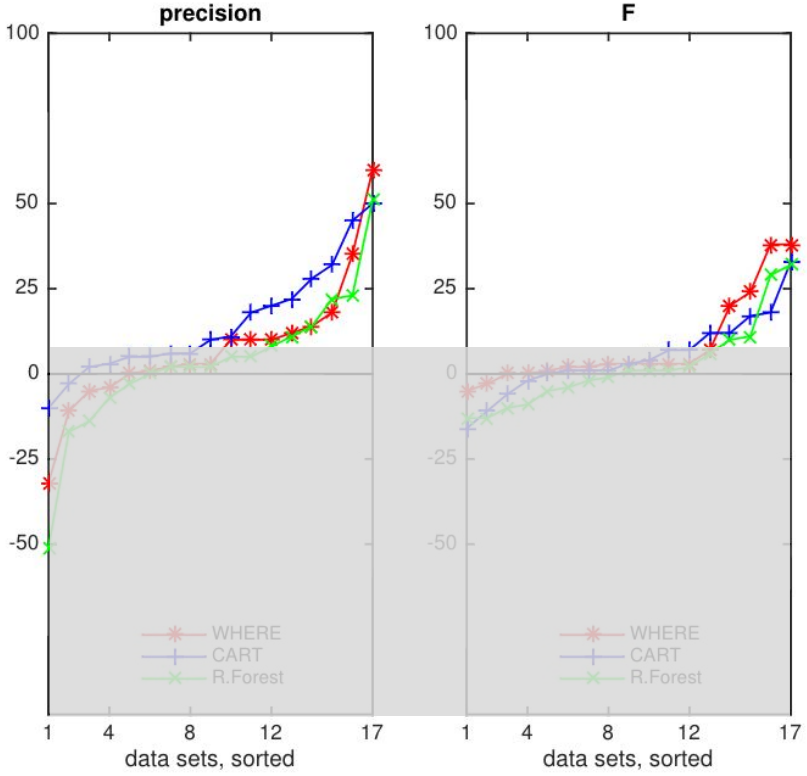

Really?
I have a question!

Review

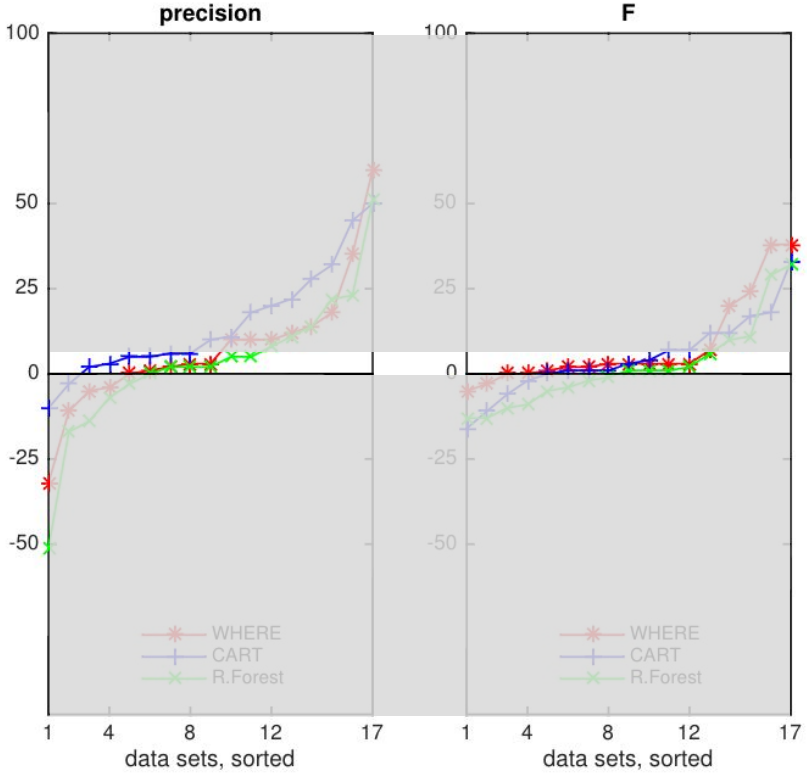


Improvements of tuned learners over untuned learners.

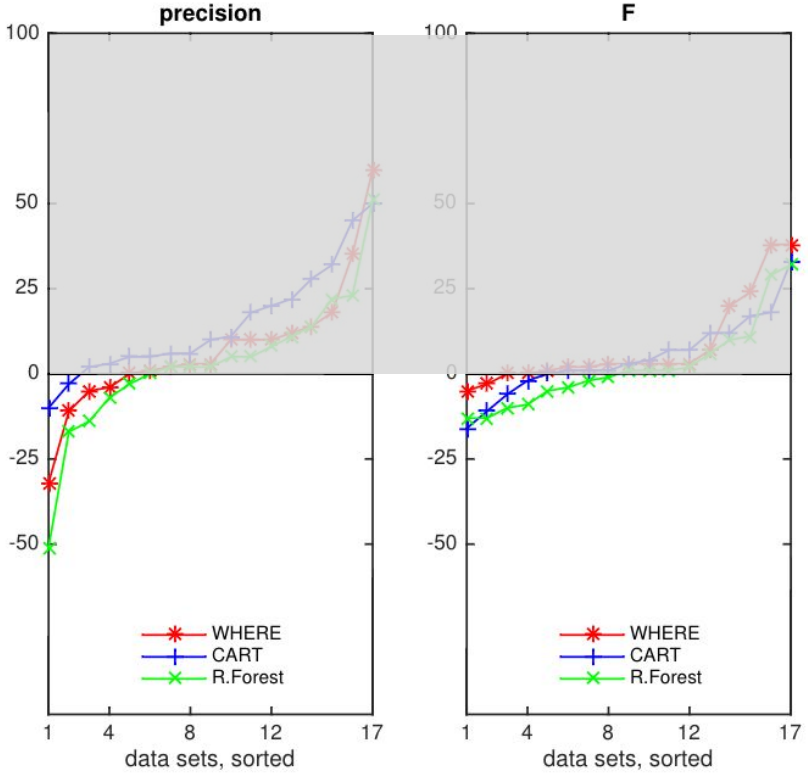
Review


precision
F
100
50
25
0
-25
-50
1
4
8
12
17
data sets, sorted
WHERE
CART
R.Forest

Review


precision
F
100
50
25
0
-25
-50
WHERE
CART
R.Forest
1
4
8
12
17
data sets, sorted

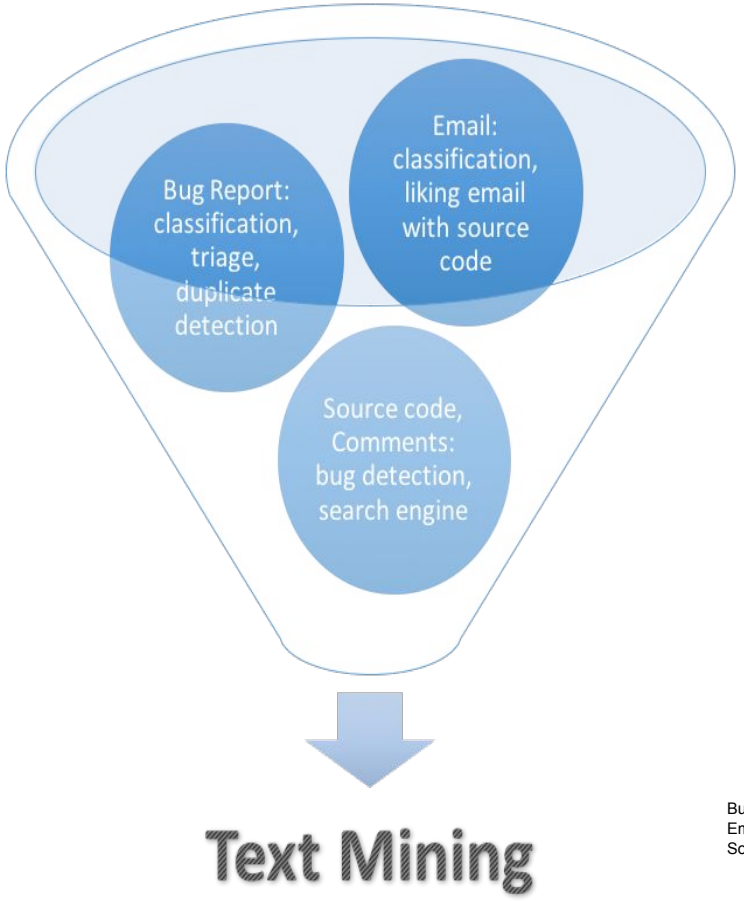
Review


precision
F
100
50
25
0
-25
-50
WHERE
CART
R.Forest
1
4
8
12
17
data sets, sorted

Roadmap

- Expectation management
- Motivation
- Background
- Progress Report
- Future Work & Challenges

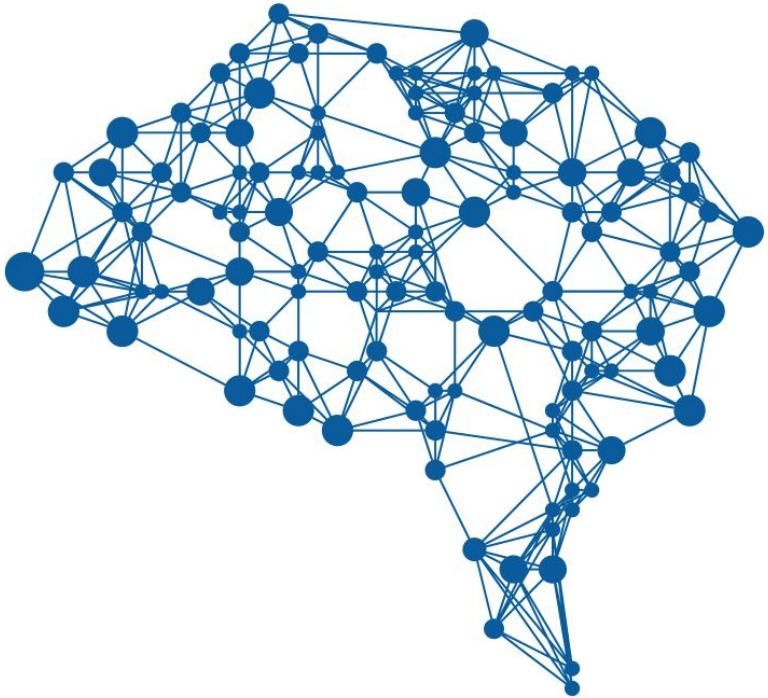
Harder and Harder Problems



- What to tune?
  - Data Mining algorithms
  - Feature processing
    - Tfidf
    - L2norm
    - Hashing trick
    - …..
- Work with Zhe, Rahul, Di, etc.…LN people!

Bug report:[Wang' 2008, Anvik' 2006, Guo' 2010]
Email:[Bacchelli 2011, Dredze 2006]
Source code:[Tan' 2012, Yang' 2012, Shridihara'2011]

Deep Learning



- Code suggestion[White 2015]
- Buggy files localization[Lam 2015]
- What to tune?
  - number of hidden layer,
  - learning rate,
  - amount of regularization
  - ...
- Work with Prof.Tien Nguyen (ISU)

Improve tuning performance

- Work with JC Nam
  - Cluster data to transfer knowledge
  - Improve heterogeneous defect prediction.
- Relevancy-based optimization?
  - Cluster tuning/training data
  - Select clusters that are nearest neighbors to testing as tuning/training data.
  - Apply those tunings in testing.

Questions	Answers
Does tuning improve learners' performance?	Yes
Does tuning change learners' rank?	Yes
Is tuning very slow?	No!
Should data miner be used "off-the-shelf"?	No!

Reference

1. [Snoek' 2012]Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. "Practical bayesian optimization of machine learning algorithms." *Advances in neural information processing systems*. 2012.
2. [Bergstra'2012]Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." *The Journal of Machine Learning Research* 13.1 (2012): 281-305.
3. [Cherkassky'2004]Cherkassky, Vladimir, and Yunqian Ma. "Practical selection of SVM parameters and noise estimation for SVM regression." *Neural networks* 17.1 (2004): 113-126.
4. [Panichella'2013]Panichella, Annibale, et al. "How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms." *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.
5. [Arcuri' 2011]Arcuri, Andrea, and Gordon Fraser. "On parameter tuning in search based software engineering." *Search Based Software Engineering*. Springer Berlin Heidelberg, 2011. 33-47.
6. [Corazza'2010]Corazza, Anna, et al. "How effective is tabu search to configure support vector regression for effort estimation?." *Proceedings of the 6th international conference on predictive models in software engineering*. ACM, 2010.
7. [Wang' 2013]Wang, Tiantian, et al. "Searching for better configurations: a rigorous approach to clone evaluation." *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*. ACM, 2013.
8. [Lessmann '2008]Lessmann, Stefan, et al. "Benchmarking classification models for software defect prediction: A proposed framework and novel findings." *Software Engineering, IEEE Transactions on* 34.4 (2008): 485-496.
9. [Tantithamthavorn '2016]Tantithamthavorn, Chakkrit, et al. "Automated Parameter Optimization of Classification Techniques for Defect Prediction Models." *The International Conference on Software Engineering (ICSE), page To appear*. 2016.
10. [White 2015]M. White, C. Vendome, M. Linares-Vásquez, and D. Poshyvanyk, "Toward deep learning software repositories," in Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on, pp. 334–345,IEEE, 2015.
11. [Lam 2015]A. N. Lam, A. T. Nguyen, H. A. Nguyen, and T. N. Nguyen, "Combining deep learning with information retrieval to localize buggy files for bug reports (n)," in Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on, pp. 476–481, IEEE, 2015.

Reference

12. [Rahman 2013]Rahman, Foyzur, and Premkumar Devanbu. "How, and why, process metrics are better." *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.
13. [Moser 2008]Moser, Raimund, Witold Pedrycz, and Giancarlo Succi. "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction." *Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on*. IEEE, 2008.
14. [Bell 2013]Bell, Robert M., Thomas J. Ostrand, and Elaine J. Weyuker. "The limited impact of individual developer data on software defect prediction." *Empirical Software Engineering* 18.3 (2013): 478-505.
15. [Menzies 2013]Menzies, Tim, et al. "Local versus global lessons for defect prediction and effort estimation." *Software Engineering, IEEE Transactions on* 39.6 (2013): 822-834.
16. [Menzies 2016]Menzies, T., Krishna, R., Pryor, D. (2016). The Promise Repository of Empirical Software Engineering Data; http://openscience.us/repo. North Carolina State University, Department of Computer Science
17. [Nagappan 2005]Nagappan, Nachiappan, and Thomas Ball. "Static analysis tools as early indicators of pre-release defect density." *Proceedings of the 27th international conference on Software engineering*. ACM, 2005.
18. [Lewis 2013]Lewis, Chris, et al. "Does bug prediction support human developers? findings from a google case study." *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.
19. [Menzies 2007]Menzies, Tim, Jeremy Greenwald, and Art Frank. "Data mining static code attributes to learn defect predictors." *Software Engineering, IEEE Transactions on* 33.1 (2007): 2-13.
20. [Harman 2001]Harman, Mark, and Bryan F. Jones. "Search-based software engineering."*Information and software Technology* 43.14 (2001): 833-839.
21. [Wang 2008]Wang, Xiaoyin, et al. "An approach to detecting duplicate bug reports using natural language and execution information." *Proceedings of the 30th international conference on Software engineering*. ACM, 2008.
22. [Anvik 2006]Anvik, John, Lyndon Hiew, and Gail C. Murphy. "Who should fix this bug?."*Proceedings of the 28th international conference on Software engineering*. ACM, 2006.
23. [Guo 2010]Guo, Philip J., et al. "Characterizing and predicting which bugs get fixed: an empirical study of Microsoft Windows." *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*. Vol. 1. IEEE, 2010.

Reference

24. [Tan 2012]Tan, Shin Hwei, et al. "@ tcomment: Testing javadoc comments to detect comment-code inconsistencies." *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*. IEEE, 2012.
25. [Yang 2012]Yang, Jinqiu, and Lin Tan. "Inferring semantically related words from software context." *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*. IEEE Press, 2012.
26. [Sridhara 2011]Sridhara, Giriprasad, Lori Pollock, and K. Vijay-Shanker. "Automatically detecting and describing high level actions within methods." *Software Engineering (ICSE), 2011 33rd International Conference on*. IEEE, 2011.
27. [Bacchelli 2011]Bacchelli, Alberto, Michele Lanza, and Romain Robbes. "Linking e-mails and source code artifacts." *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. ACM, 2010.
28. [Drehze 2006]Dredze, Mark, Tessa Lau, and Nicholas Kushmerick. "Automatically classifying emails into activities." *Proceedings of the 11th international conference on Intelligent user interfaces*. ACM, 2006.
29. [Vesterstrom 2004]Vesterstrøm, Jakob, and Rene Thomsen. "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems." *Evolutionary Computation, 2004. CEC2004. Congress on*. Vol. 2. IEEE, 2004.
30. [Faloutsos 1995]Faloutsos, Christos, and King-Ip Lin. *FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets*. Vol. 24. No. 2. ACM, 1995.