Model Compression

Rich Caruana
Microsoft Research

joint with Cristi Bucila and Alex Niculescu-Mizil

Breiman's Constant

$$complexity \times error \geq bc$$

- Complexity and loss are linked entities
- As models become more accurate, become more complex
- How do we measure complexity…

Original Motivation for Model Compression

Summary of Large Empirical Study

	Threshold Metrics			Rank/Ordering Metrics			Probability Metrics			
Model	Accuracy	F-Score	Lift	ROC Area	Average Precision	Break Even Point	Squared Error	Cross-Entropy	Calibration	Mean
BST-DT	**0.860**	**0.854**	**0.956**	**0.977**	**0.958**	**0.952**	**0.929**	**0.932**	**0.808**	**0.914**
RND-FOR	**0.866**	0.871	**0.958**	**0.977**	**0.957**	**0.948**	0.892	0.898	0.702	0.897
ANN	**0.817**	**0.875**	**0.947**	0.963	0.926	0.929	0.872	0.878	**0.826**	0.892
SVM	0.823	0.851	0.928	0.961	0.931	0.929	0.882	0.880	0.769	0.884
BAG-DT	0.836	0.849	0.953	0.972	**0.950**	0.928	0.875	0.901	0.637	0.878
KNN	0.759	0.820	0.914	0.937	0.893	0.898	0.786	0.805	0.706	0.835
BST-STMP	0.698	0.760	0.898	0.926	0.871	0.854	0.740	0.783	0.678	0.801
DT	0.611	0.771	0.856	0.871	0.789	0.808	0.586	0.625	0.688	0.734
LOG-REG	0.602	0.623	0.829	0.849	0.732	0.714	0.614	0.620	0.678	0.696
NAÏVE-B	0.536	0.615	0.786	0.833	0.733	0.730	0.539	0.565	0.161	0.611

- After Platt Scaling, boosted trees are best models overall across all metrics
- Neural nets are best models overall if no calibration is applied post-training

Summary of Large Empirical Study

	Threshold Metrics			Rank/Ordering Metrics			Probability Metrics			
Model	**Accuracy**	**F-Score**	**Lift**	**ROC Area**	**Average Precision**	**Break Even Point**	**Squared Error**	**Cross-Entropy**	**Calibration**	**Mean**
BEST	**0.928**	**0.918**	**0.975**	**0.987**	**0.958**	**0.958**	**0.919**	**0.944**	**0.989**	**0.953**
BST-DT	**0.860**	**0.854**	**0.956**	**0.977**	**0.958**	**0.952**	**0.929**	**0.932**	**0.808**	**0.914**
RND-FOR	**0.866**	0.871	**0.958**	**0.977**	**0.957**	**0.948**	0.892	0.898	0.702	0.897
ANN	**0.817**	**0.875**	**0.947**	0.963	0.926	0.929	0.872	0.878	**0.826**	0.892
SVM	0.823	0.851	0.928	0.961	0.931	0.929	0.882	0.880	0.769	0.884
BAG-DT	0.836	0.849	0.953	0.972	**0.950**	0.928	0.875	0.901	0.637	0.878
KNN	0.759	0.820	0.914	0.937	0.893	0.898	0.786	0.805	0.706	0.835
BST-STMP	0.698	0.760	0.898	0.926	0.871	0.854	0.740	0.783	0.678	0.801
DT	0.611	0.771	0.856	0.871	0.789	0.808	0.586	0.625	0.688	0.734
LOG-REG	0.602	0.623	0.829	0.849	0.732	0.714	0.614	0.620	0.678	0.696
NAÏVE-B	0.536	0.615	0.786	0.833	0.733	0.730	0.539	0.565	0.161	0.611

Can We Do Even Better?

	Threshold Metrics			Rank/Ordering Metrics			Probability Metrics			
Model	**Accuracy**	**F-Score**	**Lift**	**ROC Area**	**Average Precision**	**Break Even Point**	**Squared Error**	**Cross-Entropy**	**Calibration**	**Mean**
ES	**0.956**	**0.944**	**0.99**	**0.996**	**0.985**	**0.979**	**0.979**	**0.981**	0.988	**0.977**
BEST	0.928	0.919	0.975	0.988	0.958	0.958	0.919	0.944	**0.989**	0.953
BST-DT	**0.860**	**0.854**	**0.956**	**0.977**	**0.958**	**0.952**	**0.929**	**0.932**	**0.808**	**0.914**
RND-FOR	**0.866**	0.871	**0.958**	**0.977**	**0.957**	**0.948**	0.892	0.898	0.702	0.897
ANN	**0.817**	**0.875**	**0.947**	0.963	0.926	0.929	0.872	0.878	**0.826**	0.892
SVM	0.823	0.851	0.928	0.961	0.931	0.929	0.882	0.880	0.769	0.884
BAG-DT	0.836	0.849	0.953	0.972	**0.950**	0.928	0.875	0.901	0.637	0.878
KNN	0.759	0.820	0.914	0.937	0.893	0.898	0.786	0.805	0.706	0.835
BST-STMP	0.698	0.760	0.898	0.926	0.871	0.854	0.740	0.783	0.678	0.801

Meta Method: Ensemble Selection

- Train *many* different models:
  - different algorithms
  - different parameter settings
  - all trained on same train set
  - all trained to "natural" optimization criterion
- Add *all* models to library:
  - no model selection
  - no validation set
  - some models bad, some models good, a few models excellent
  - ***yields diverse set of models, some of which are good on almost any metric***
- Forward stepwise *model selection* from library:
  - start with empty ensemble
  - try adding each model one-at-a-time to ensemble
  - commit model that maximizes performance on metric on a test set

Basic Ensemble Selection Algorithm

Model Library

Ensemble

Model 1
Model 2
Model 3
Model 4
Model 5
Model 6
Model 7
Model 8
Model 9

Basic Ensemble Selection Algorithm

Model Library	AUC Score on the 1k validation set	Ensemble
Model 1	0.8453	
Model 2	0.8726	
Model 3	0.9164	
Model 4	0.8142	
Model 5	0.8453	
Model 6	0.8745	
Model 7	0.9024	
Model 8	0.7034	
Model 9	0.8342	

Basic Ensemble Selection Algorithm

Model Library	AUC Score on the 1k validation set	Ensemble
Model 1	0.8453	
Model 2	0.8726	
Model 3	0.9164	
Model 4	0.8142	
Model 5	0.8453	
Model 6	0.8745	
Model 7	0.9024	
Model 8	0.7034	
Model 9	0.8342	

Basic Ensemble Selection Algorithm

Model Library

Model 1
Model 2

Model 4
Model 5
Model 6
Model 7
Model 8
Model 9

Ensemble

Model 3 **0.9164**

Basic Ensemble Selection Algorithm

Model Library			AUC Score on the 1k validation set		Ensemble	
Model 1	+ Ensemble	=	0.8327	0.8453	Model 3	0.9164
Model 2			0.8702	0.8726		
Model 4			0.9284	0.8142		
Model 5			0.9047	0.8453		
Model 6			0.8832	0.8745		
Model 7			0.9126	0.9024		
Model 8			0.8245	0.7034		
Model 9			0.9384	0.8342		

Basic Ensemble Selection Algorithm

Model Library		AUC Score on the 1k validation set
Model 1		0.8327
Model 2		0.8702
Model 4		0.9284
Model 5	+ Ensemble =	0.9047
Model 6		0.8832
Model 7		0.9126
Model 8		0.8245
Model 9		0.9384

Ensemble

Model 3 0.9164

Basic Ensemble Selection Algorithm

Model Library

Model 1
Model 2

Model 4
Model 5
Model 6
Model 7
Model 8

Ensemble

Model 3 **0.9164**
Model 9 **0.9384**

Basic Ensemble Selection Algorithm

Model Library			AUC Score on the 1k validation set		Ensemble	
Model 1			0.8502	0.8327	Model 3	0.9164
Model 2			0.9243	0.8702	Model 9	0.9384
Model 4			0.8992	0.9284		
Model 5	+ Ensemble	=	0.8090	0.9047		
Model 6			0.9424	0.8832		
Model 7			0.9045	0.9126		
Model 8			0.9243	0.8245		

Basic Ensemble Selection Algorithm

Model Library		AUC Score on the 1k validation set
Model 1		0.8502
Model 2		0.9243
Model 4		0.8992
Model 5	+ Ensemble =	0.8090
Model 6		0.9424
Model 7		0.9045
Model 8		0.9243

Ensemble

Model 3	0.9164
Model 9	0.9384

Basic Ensemble Selection Algorithm

Model Library

Model 1
Model 2

Model 4
Model 5

Model 7
Model 8

Ensemble

Model 3	**0.9164**
Model 9	**0.9384**
Model 6	**0.9424**

Very accurate models: variations of this approach win many data mining competitions

Surprisingly: haven't hit the supervised learning performance ceiling yet!

But there is a BIG Problem

- Ensembles too big, too slow for many applications
- Best ensemble for one problem/metric has 422 models:
  - 72 boosted trees (28,642 individual decision trees!)
  - 1 random forest (1024 decision trees)
  - 5 bagged trees (100 decision trees in each model)
  - 44 neural nets (2,200 hidden units total, >100,000 weights)
  - 115 knn models (both large and expensive!)
  - 38 SVMs (100's of support vectors in each model)
  - 26 boosted stump models (36,184 stumps total -- could compress)
  - 122 individual decision trees
  - …
- Best ensemble:
  - takes ~1GB to store models
  - takes ~1 second to execute per test case!

This Really is a BIG Problem

- Web search: must execute in real-time, large “test set”
- Image recognition: “test set” too large (video even worse) if you have to sweep model over all patches
- Satellites: small hardened memory, low processor speed
- Mars rover: low power, small memory, slow processor
- Hearing aids: low power, small memory, slow processor
- Smart phones: limited processor memory, cost
- Digital cameras: low power, real-time decision making, sweep over all patches to find faces
- …

This Really is a Problem

- Web search: must execute in real-time, large "test set"
- Image recognitio[illegible]est set" to[illegible]ge (video even worse)
- Satellites: smal[illegible]ow processor speed
- Mars rover: low pow[illegible]emory, slow processor
- Hearing aids: lo[illegible]mory, slow processor
- PDA & cell pho[illegible]nited p[illegible]or memory, cost
- Digital cameras: low power, real-time decision making
- …
- Can't afford big models/ensembles on many problems
- Breiman's constant at work!

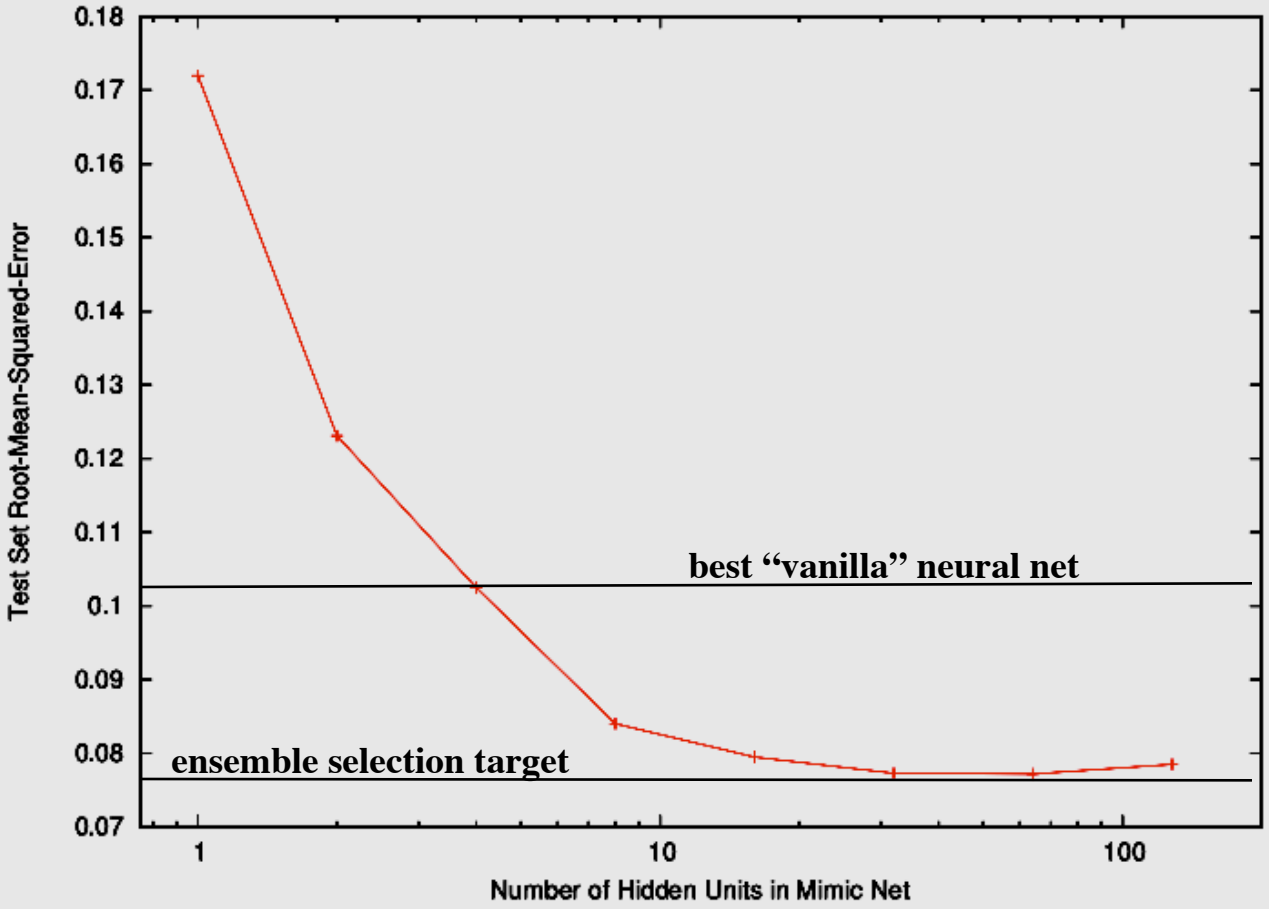
Our Solution: Model Compression

- Train complex model (ensemble) any way you want
- Train simpler model to mimic the complex ensemble

$$f(small) \approx f(\mathbf{BIG})$$

- Pass large unlabeled data (synthetic or real unlabeled data) through BIG model/ensemble and collect predictions
  - 100,000 to 10,000,000 synthetic training points
  - extensional representation of the ensemble model
- Train ***copycat mimic*** model on this large synthetic train set to mimic the high-performance model/ensemble

Preview of Compression Results



Why Mimic with Neural Nets?

- Decision trees?
  - mimic decision trees are enormous (depth > 1000 and > 10^6 nodes) making them expensive to train, store, and execute
  - synthetic data must be very large because of recursive partitioning
  - single tree does not seem to model ensemble accurately enough
- SVMs?
  - number of support vectors increases quickly with complexity
  - support vector pruning?
- KNN?
  - Bayes-optimal with large-enough case-base
  - required case-base for high accuracy usually too big?
  - pruning, clever data structures, good kernel might make feasible

Why Mimic with Neural Nets?

- model complex functions with modest # of hidden units
- compress millions of training cases into 1000's of weights
- models with 1000's of weights fit in processor memory
- execution cost low (just matrix multiplies)

- expensive to train!!!
- lose intelligibility (but ensemble probably wasn't)
- lose modularity
- NNs still too expensive for some applications?

New Problem: Unlabeled Data?

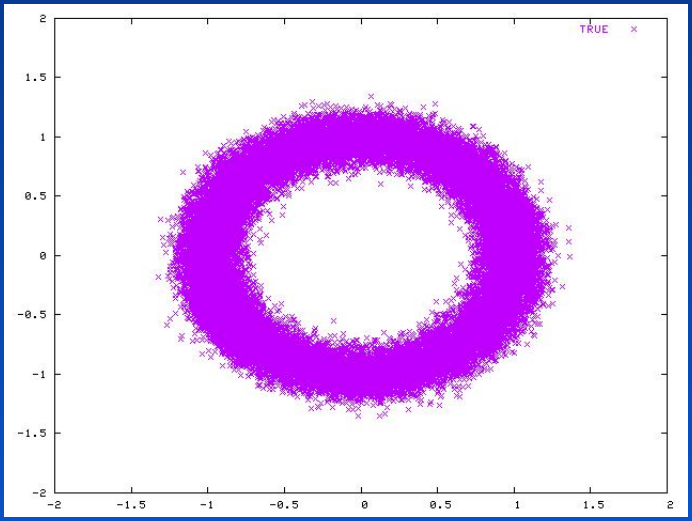
- Original labeled training set probably is small
- But we need a large train set to train mimic model

- Ideally should come from same p(x) as train data:
  - labeling large data with complex model/ensemble is expensive
    - easy solution: parallelize on multiple computers
  - training mimic model with large data is expensive
  - don't waste capacity by modeling irrelevant regions:
    - want to keep the mimic model small
    - don't want to distract mimic model --- focus only on important regions
  - these issues more critical as dimensionality increases

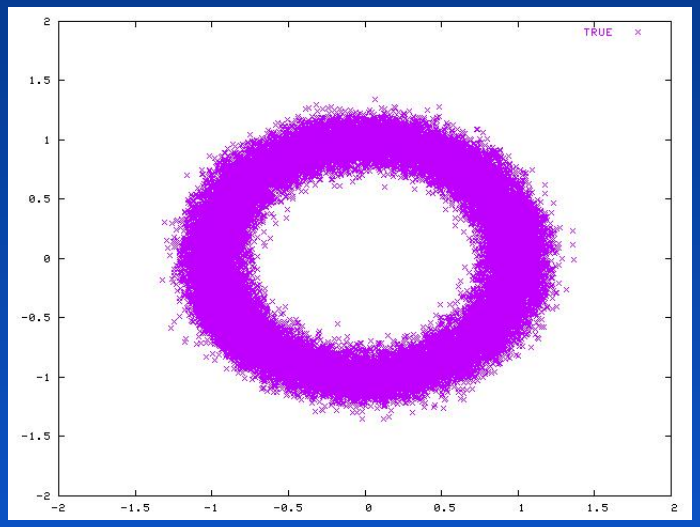
New Problem: Unlabeled Data?

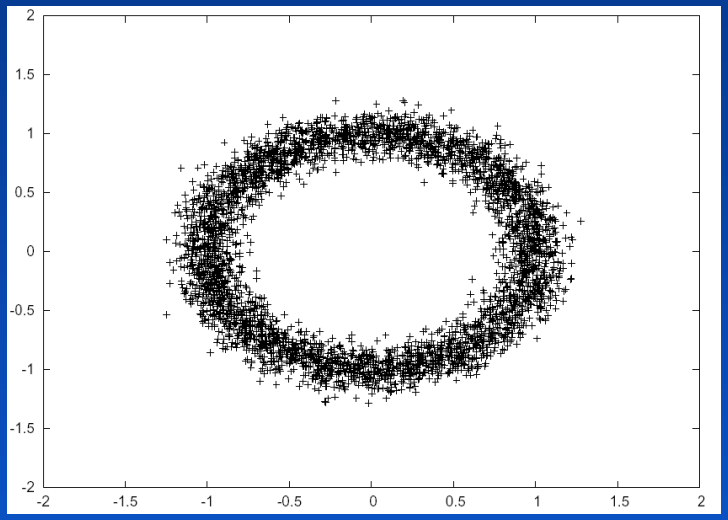
- For some domains copious unlabeled data is available
  - Text, web, images, …
- If not available, need to generate synthetic data
  - Random univariate sampling
  - NBE (Naïve Bayes Estimations)
  - Munging
- Want to capture function with minimum data --- need samples from true p(x) (from original manifold)

Sample From True Distribution p(x)



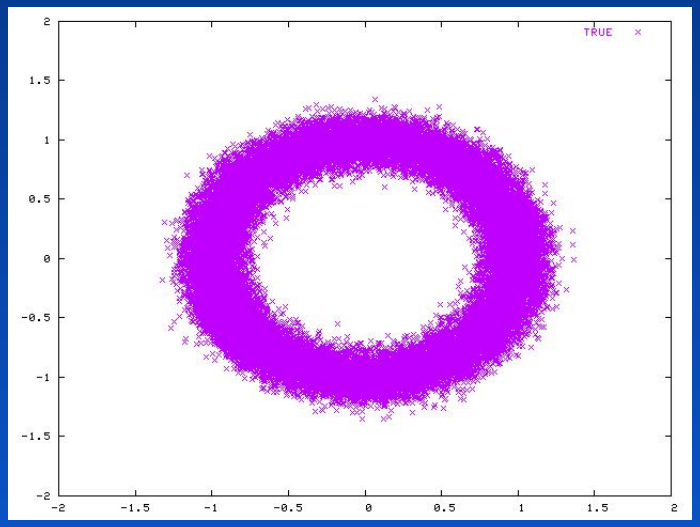
Small Sample from True Distribution

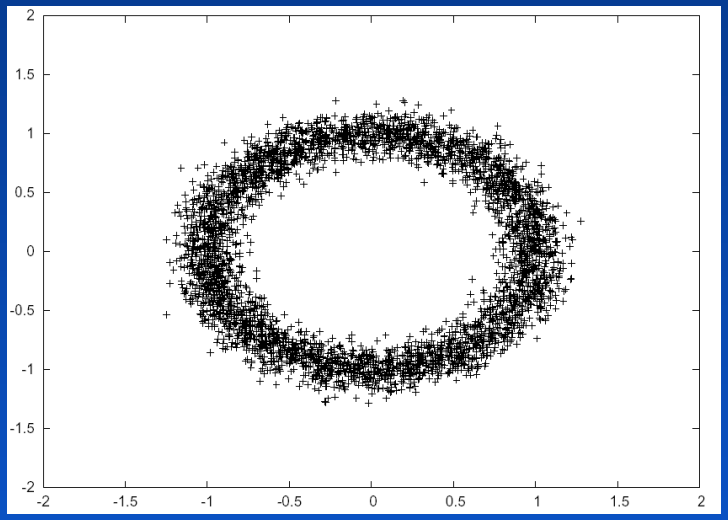




- Want a method that generates synthetic samples that look like this

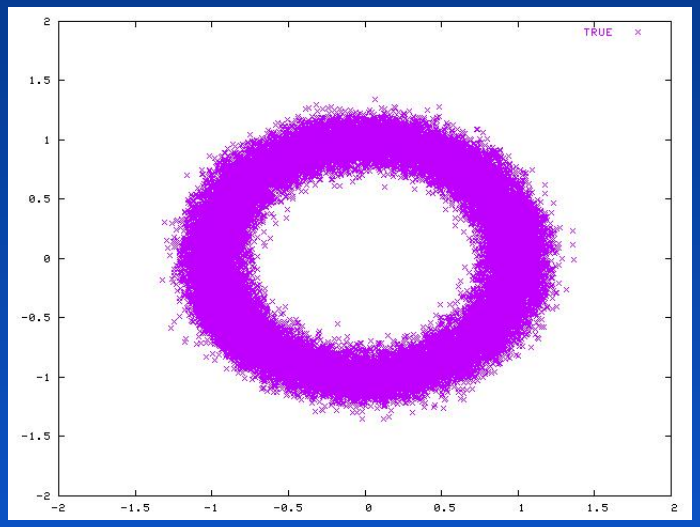
Synthetic Data: Random

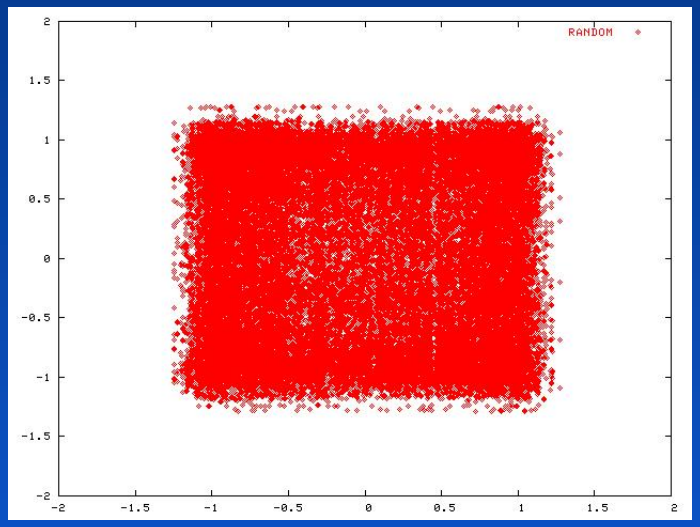




- Values for attributes generated randomly from their univariate distributions
- Model $p(x_1)$ and $p(x_2)$ separately

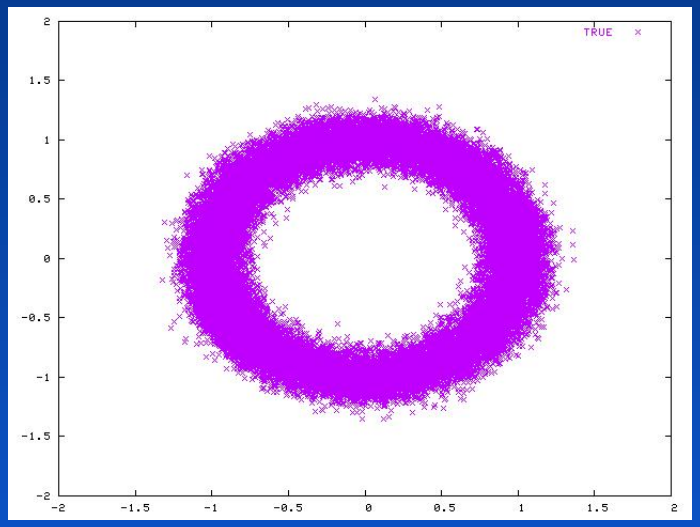
Synthetic Data: Random

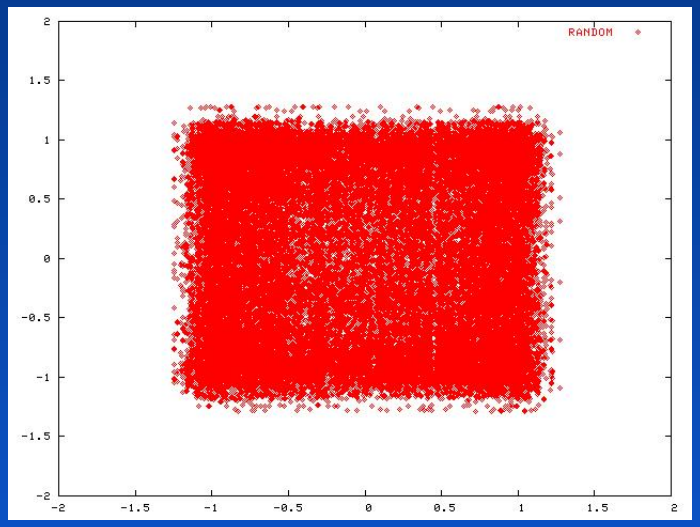




- Values for attributes generated randomly from their univariate distributions
- Model $p(x_1)$ and $p(x_2)$ separately

Synthetic Data: Random



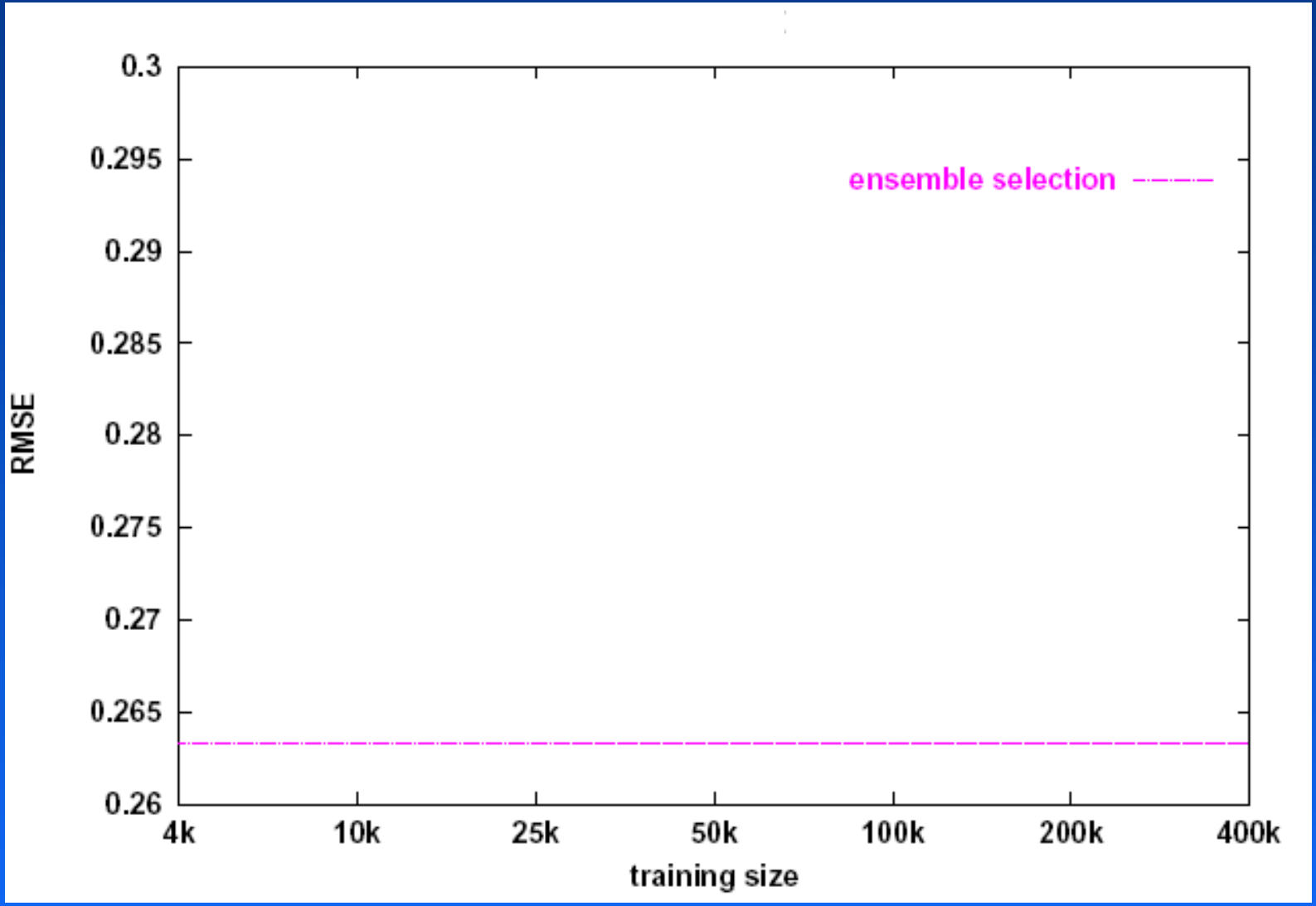


- Conditional structure of the data is lost
- Many generated examples cover uninteresting regions of the space
- Doesn't look promising, but let's try it

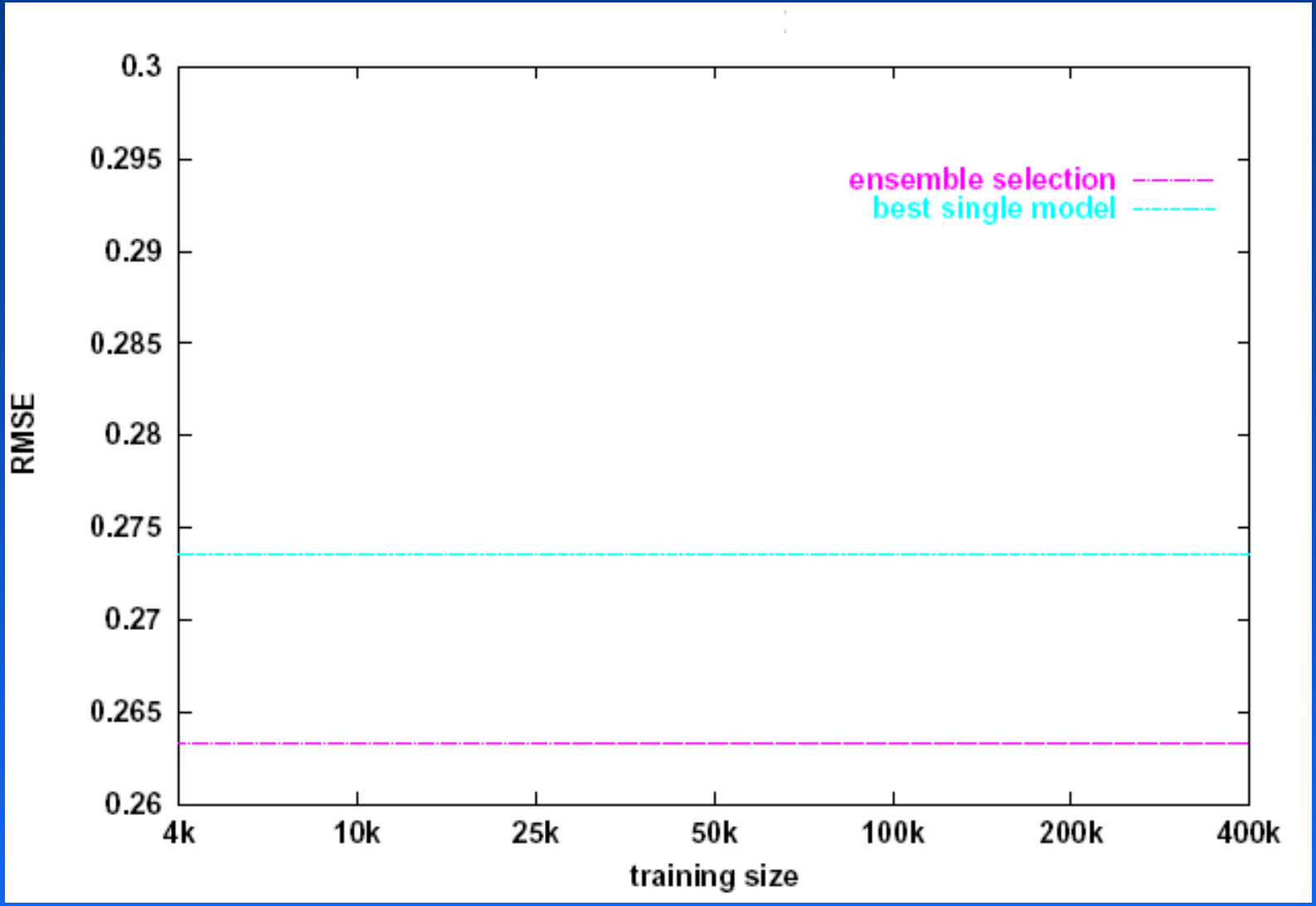
Experimental Setup: Datasets

PROBLEM	#ATTR	%POS	TRAIN SIZE	TEST SIZE
ADULT	14/104	25	4000	35222
COVTYPE	54	36	4000	25000
HS	200	24	4000	4366
LETTER.P1	16	3	4000	14000
LETTER.P2	16	53	4000	14000
MEDIS	63	11	4000	8199
MG	124	17	4000	12807
SLAC	59	50	4000	25000

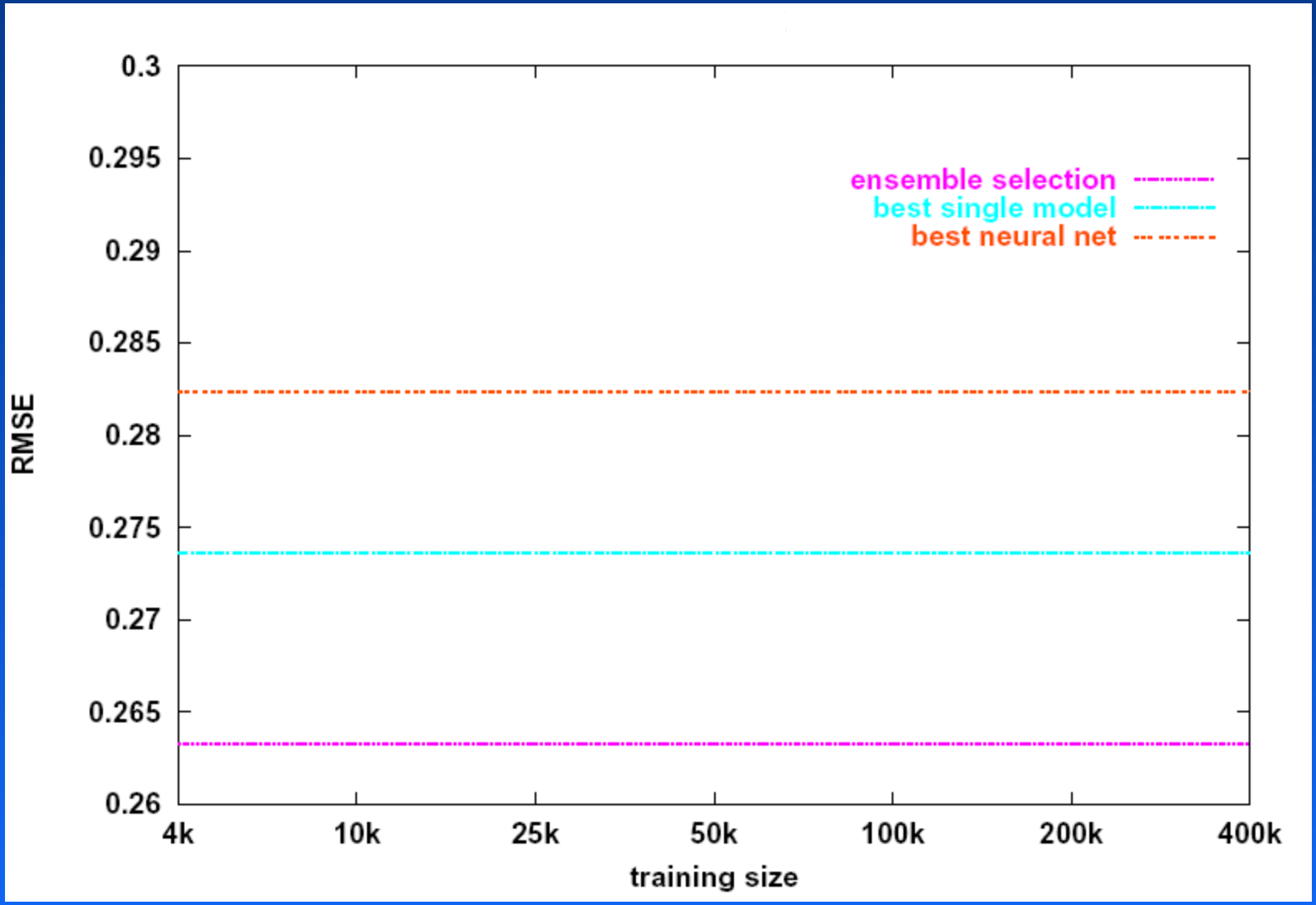
Average Results by Train Size



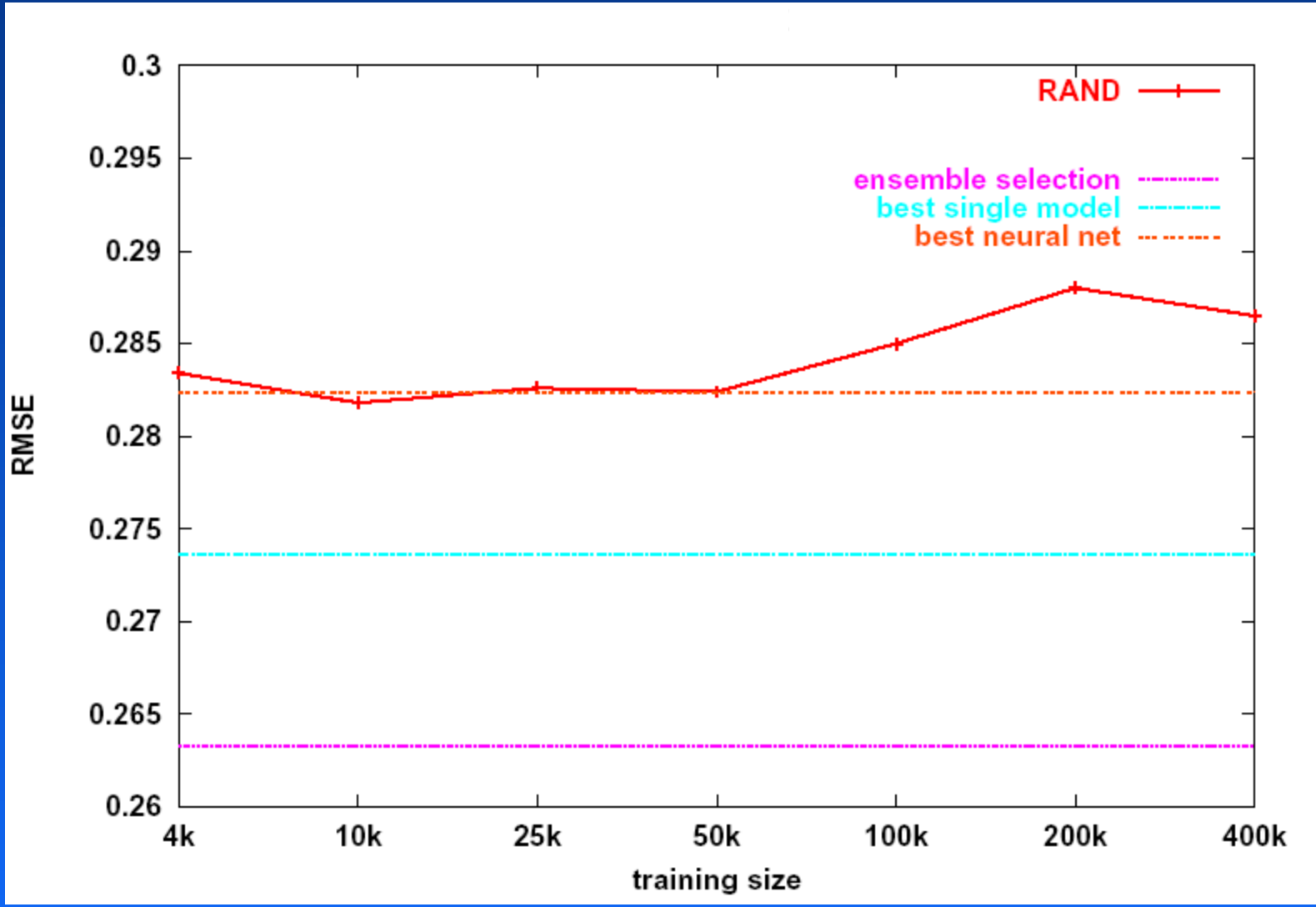
Average Results by Train Size



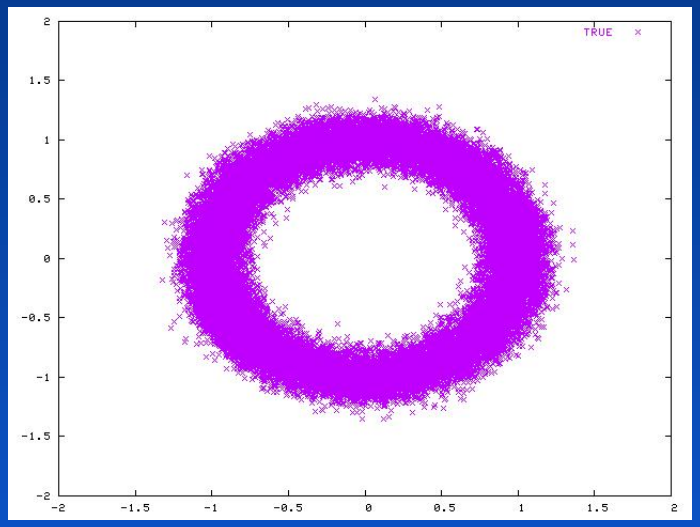
Average Results by Train Size

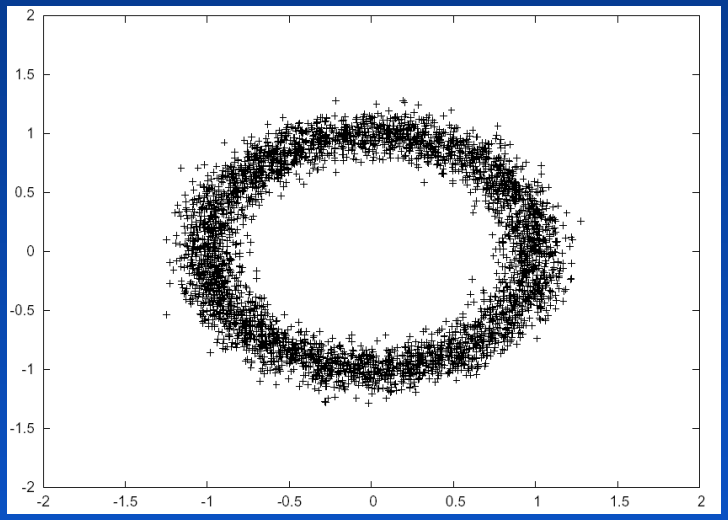


Average Results by Train Size



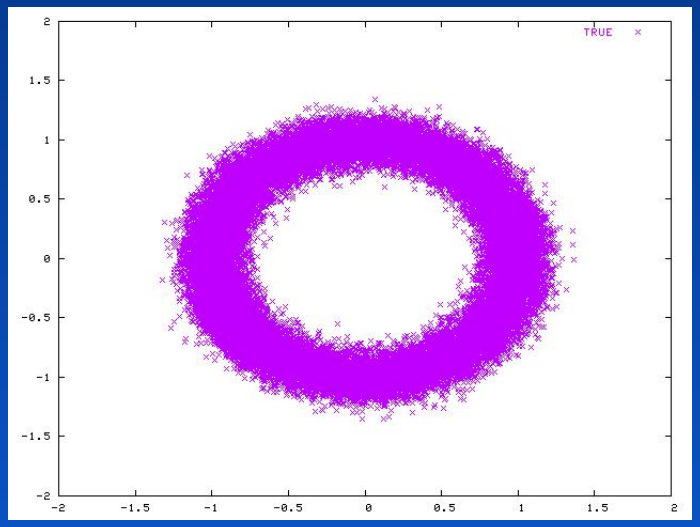
Synthetic Data: ???

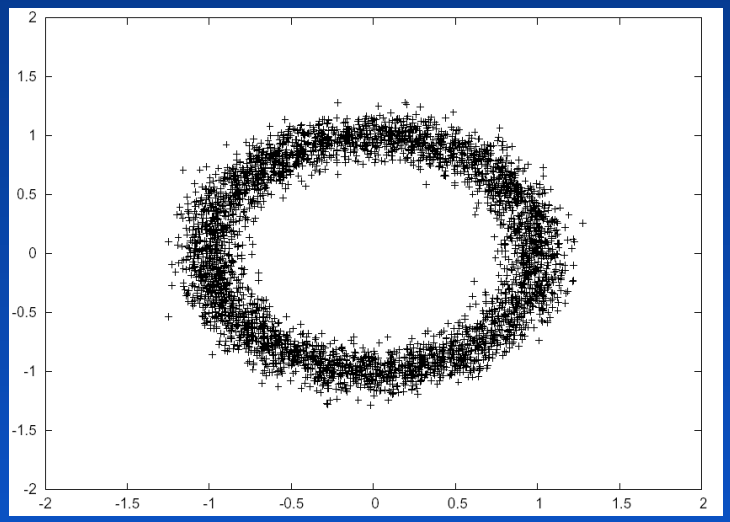




- Random didn't work very well
- Estimate joint distribution from train set?

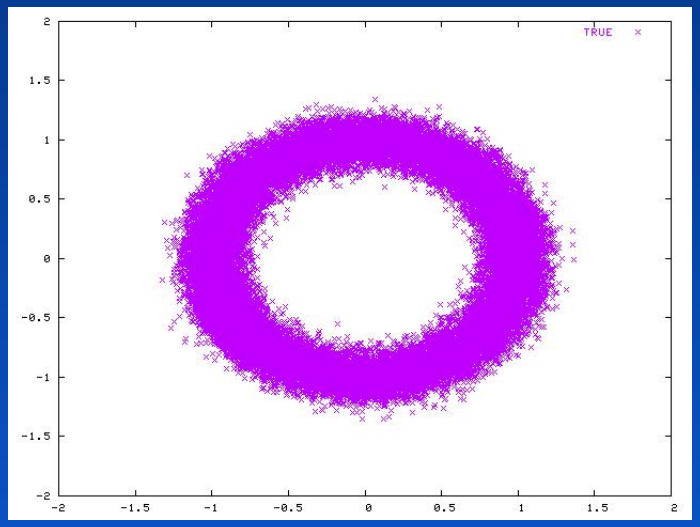
Synthetic Data: NBE

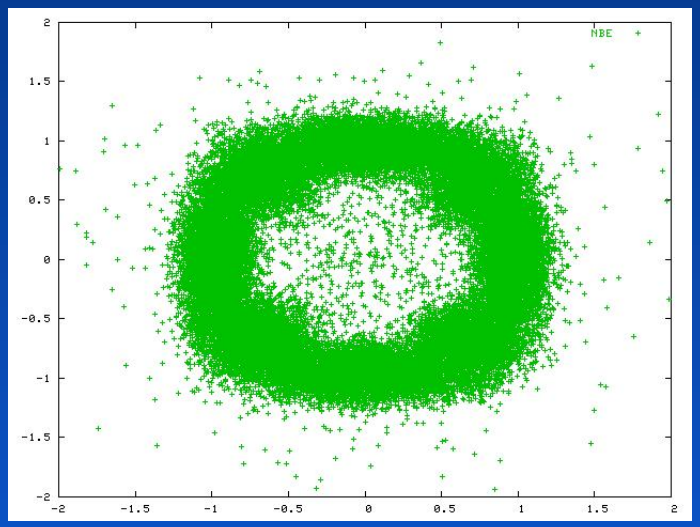




- Estimate joint distribution from train set
  - NBE (Naïve Bayes Estimation) algorithm
    - [Lowd and Domingos, 2005]
    - carve-up space until regions linear enough for NB
  - Code for learning and sampling from p(x)

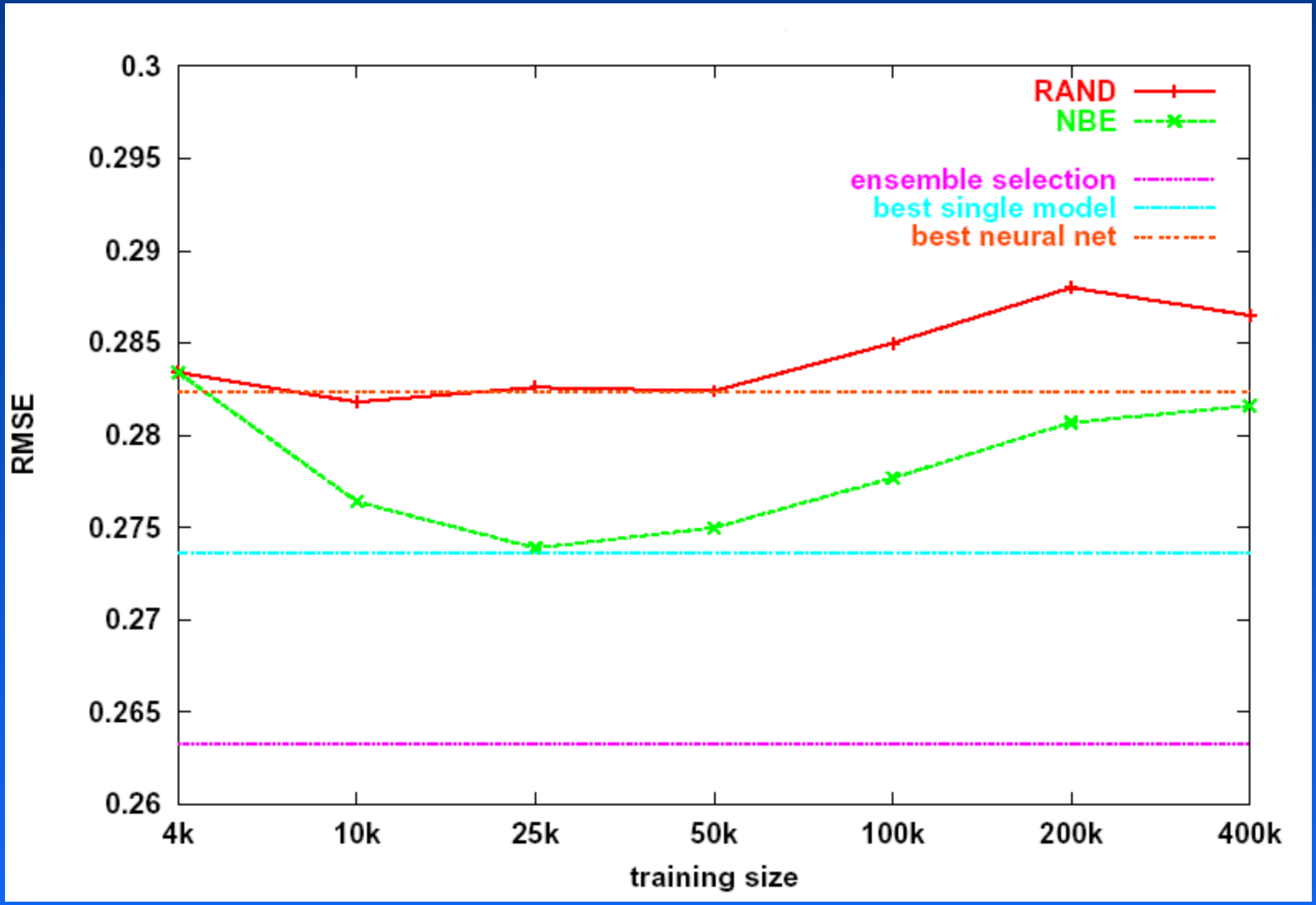
Synthetic Data: NBE





- Artifacts from carving-up space
- Sampling strays outside true p(x)
- Both get worse as dimensionality increases
- But maybe it is good enough for compression?

Average Results by Train Size



Random and NBE do not work well enough.

Developed a new, better algorithm.

Random and NBE do not work well enough.

Developed a new, better algorithm.

Munging

1. To imperfectly transform information.
2. To modify data in a way that cannot be described succinctly.

Algorithm MUNGE:

- Given: training examples *T*, probability param *p*, variance param *s*
- Returns: unlabeled synthetic-training set *D*

1. Loop until D large enough
2. $T' \leftarrow T$
3. for all examples *e* in *T* do
4. $e' \leftarrow$ the closest example of *e* from *T'*
5. for all attributes *a* of example *e* do
6. with probability *p*:
7. if *a* is continuous then
8. $e_a \leftarrow norm(e'_a, sd)$; $e_a \leftarrow norm(e'_a, sd)$, where $sd = |e_a - e'|/s$
9. else
19. swap the values of attribute *a* for examples *e* and *e'*
20. end if continuous
23. end for all attributes
24. end for all examples
25. $D \leftarrow D \cup T'$
26. end loop

Algorithm MUNGE:

- Given: training examples *T*, probability param *p*, variance param *s*
- Returns: unlabeled synthetic-training set *D*

1. Loop until D large enough
2. $T' \leftarrow T$
3. for all examples *e* in *T* do
4. $e' \leftarrow$ the closest example of *e* from *T'*
5. for all attributes *a* of example *e* do
6. with probability *p*:
7. if *a* is continuous then
8. $e_a \leftarrow norm(e'_a, sd)$; $e_a \leftarrow norm(e'_a, sd)$, where $sd = |e_a - e'|/s$
9. else
19. swap the values of attribute *a* for examples *e* and *e'*
20. end if continuous
23. end for all attributes
24. end for all examples
25. $D \leftarrow D \cup T'$
26. end loop

Algorithm MUNGE:

- Given: training examples *T*, probability param *p*, **variance param *s***
- Returns: unlabeled synthetic-training set *D*

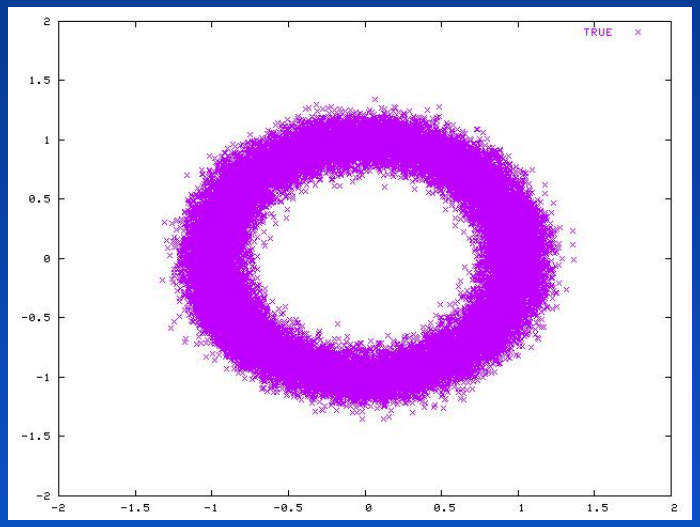
1. Loop until D large enough
2. $T' \leftarrow T$
3. for all examples *e* in *T* do
4. $e' \leftarrow$ the closest example of *e* from *T'*
5. for all attributes *a* of example *e* do
6. with probability *p*:
7. if *a* is continuous then
8. $e_a \leftarrow norm(e'_a, sd)$; $e_a \leftarrow norm(e'_a, sd)$, where $sd = |e_a - e'|/s$
9. else
19. swap the values of attribute *a* for examples *e* and *e'*
20. end if continuous
23. end for all attributes
24. end for all examples
25. $D \leftarrow D \cup T'$
26. end loop

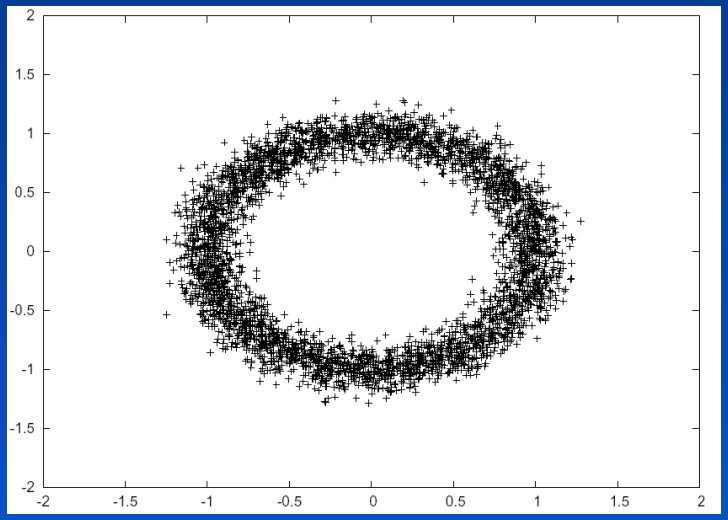
Algorithm MUNGE-Bootstrap:

- Given: training examples *T*, probability param *p*, variance param *s*
- Returns: unlabeled synthetic-training set *D*

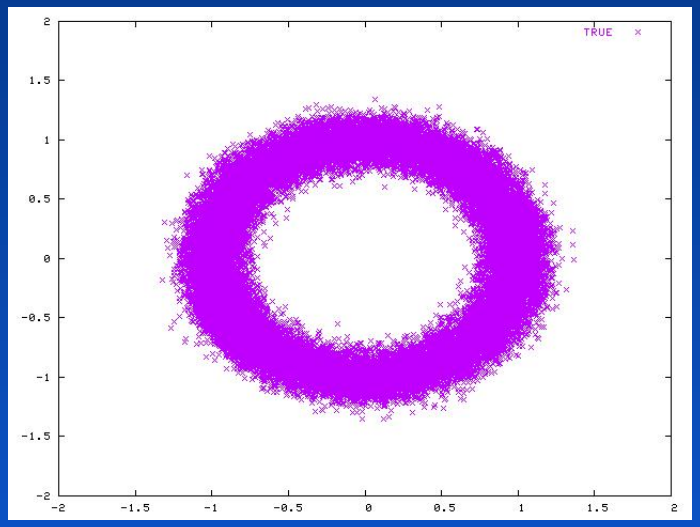
1. Loop until D large enough
2. $T' \leftarrow Bootstrap\_Sample(T)$
3. for all examples *e* in *T* do
4. $e' \leftarrow$ the closest example of *e* from *T'*
5. for all attributes *a* of example *e* do
6. with probability *p*:
7. if *a* is continuous then
8. $e_a \leftarrow norm(e'_a, sd)$; $e_a \leftarrow norm(e'_a, sd)$, where $sd = |e_a - e'|/s$
9. else
19. swap the values of attribute *a* for examples *e* and *e'*
20. end if continuous
23. end for all attributes
24. end for all examples
25. $D \leftarrow D \cup T'$
26. end loop

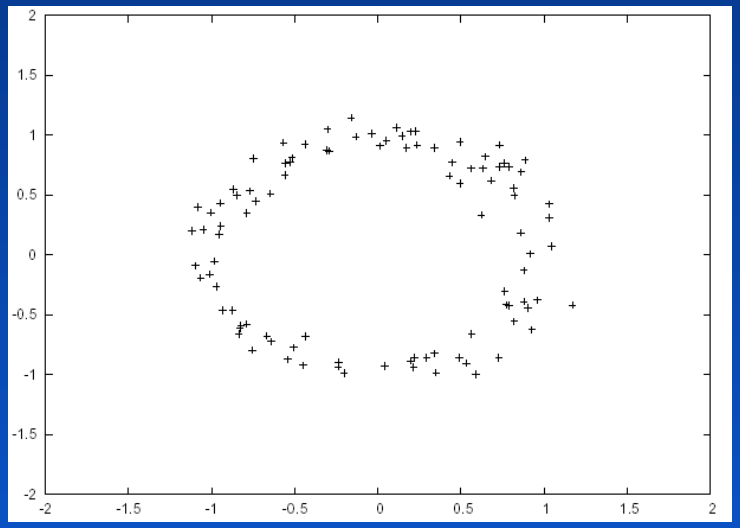
Munging


TRUE
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

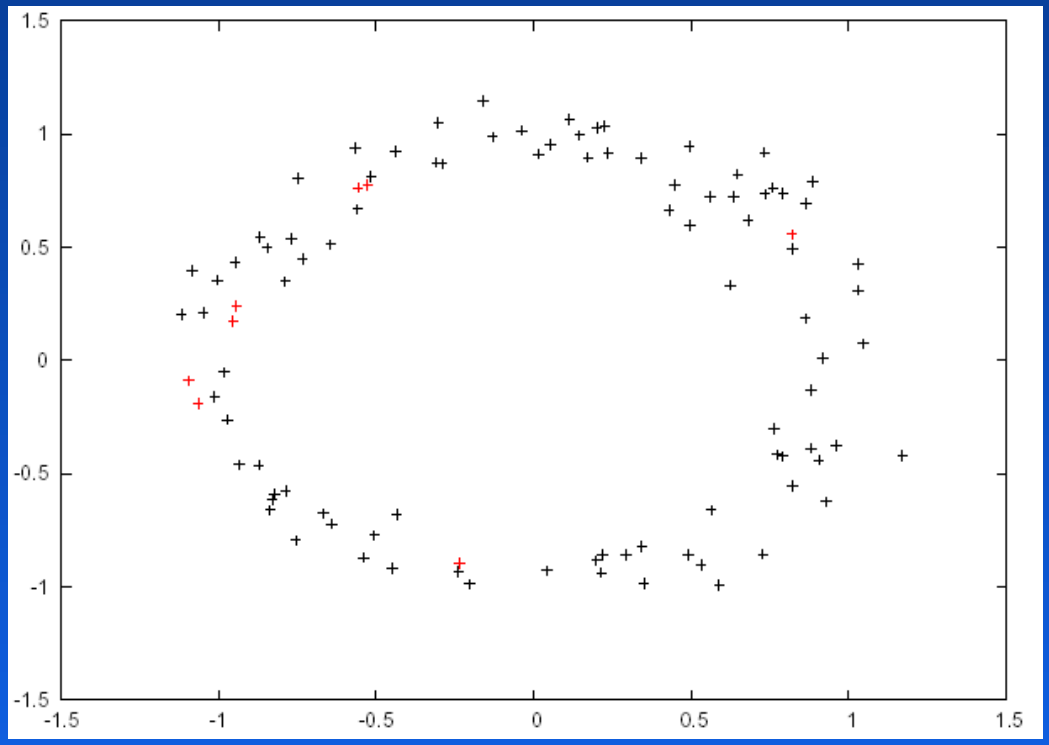
Munging


TRUE
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

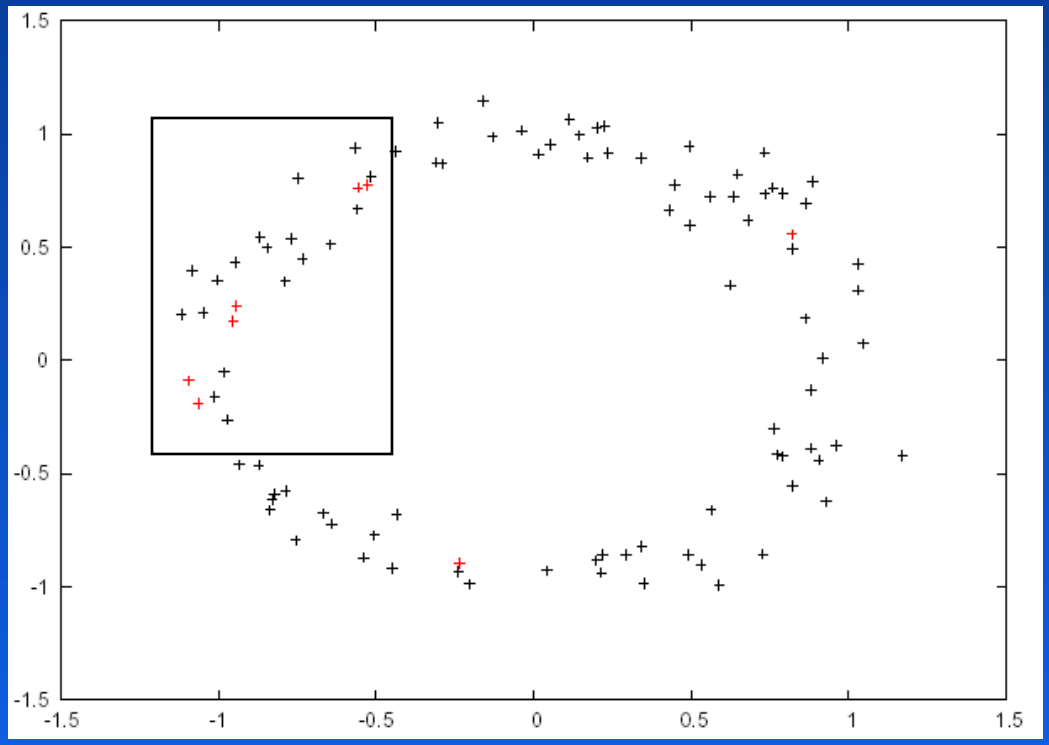
Munging

	x	y
1	-1.09	-0.09
2	-0.94	0.24
3	-0.56	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.06	-0.19
…		
61	-0.53	0.77
…		
89	-0.95	0.17



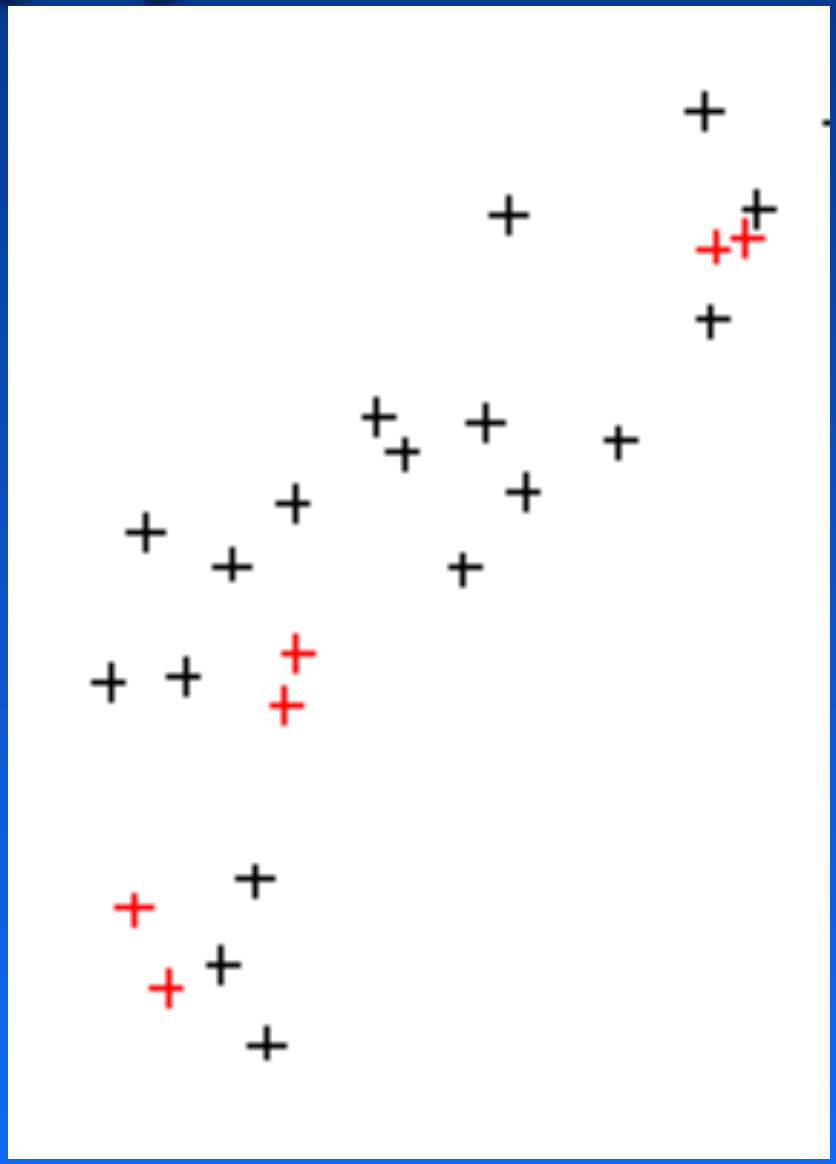
Munging

	x	y
1	-1.09	-0.09
2	-0.94	0.24
3	-0.56	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.06	-0.19
…		
61	-0.53	0.77
…		
89	-0.95	0.17



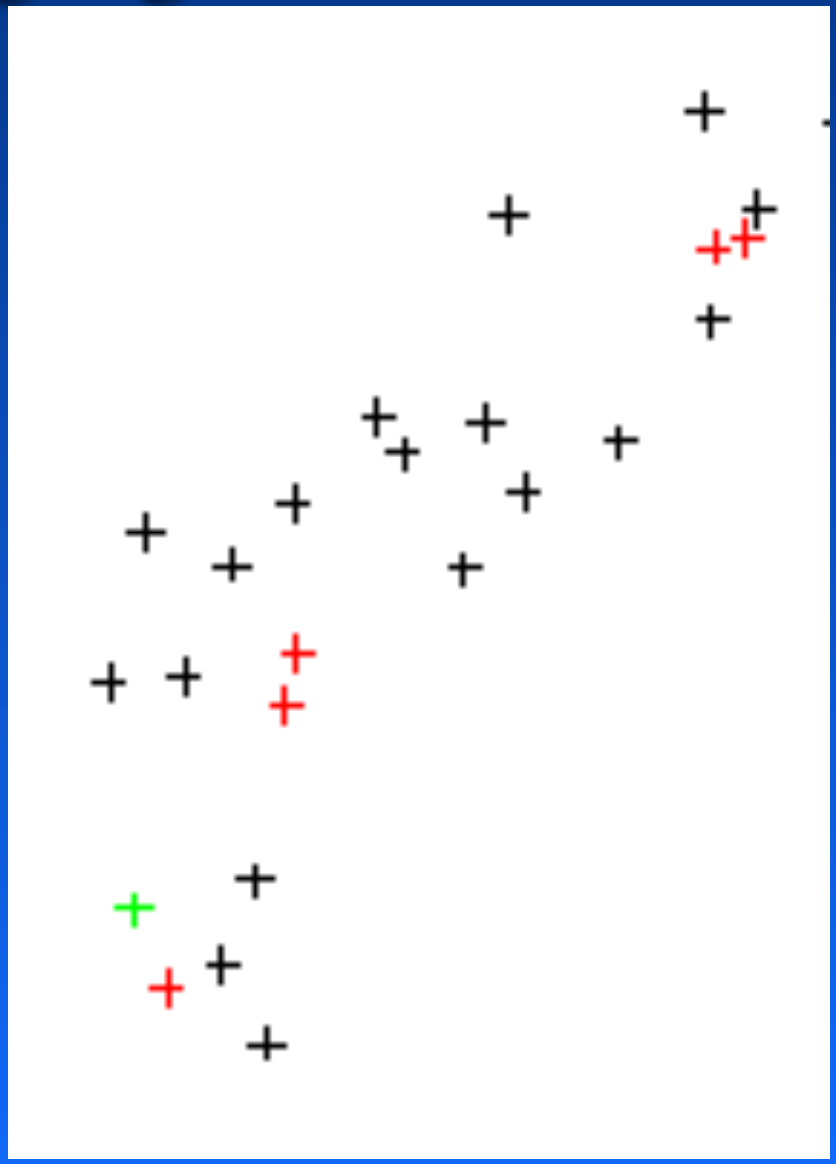
Munging

	x	y
1	-1.09	-0.09
2	-0.94	0.24
3	-0.56	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.06	-0.19
…		
61	-0.53	0.77
…		
89	-0.95	0.17



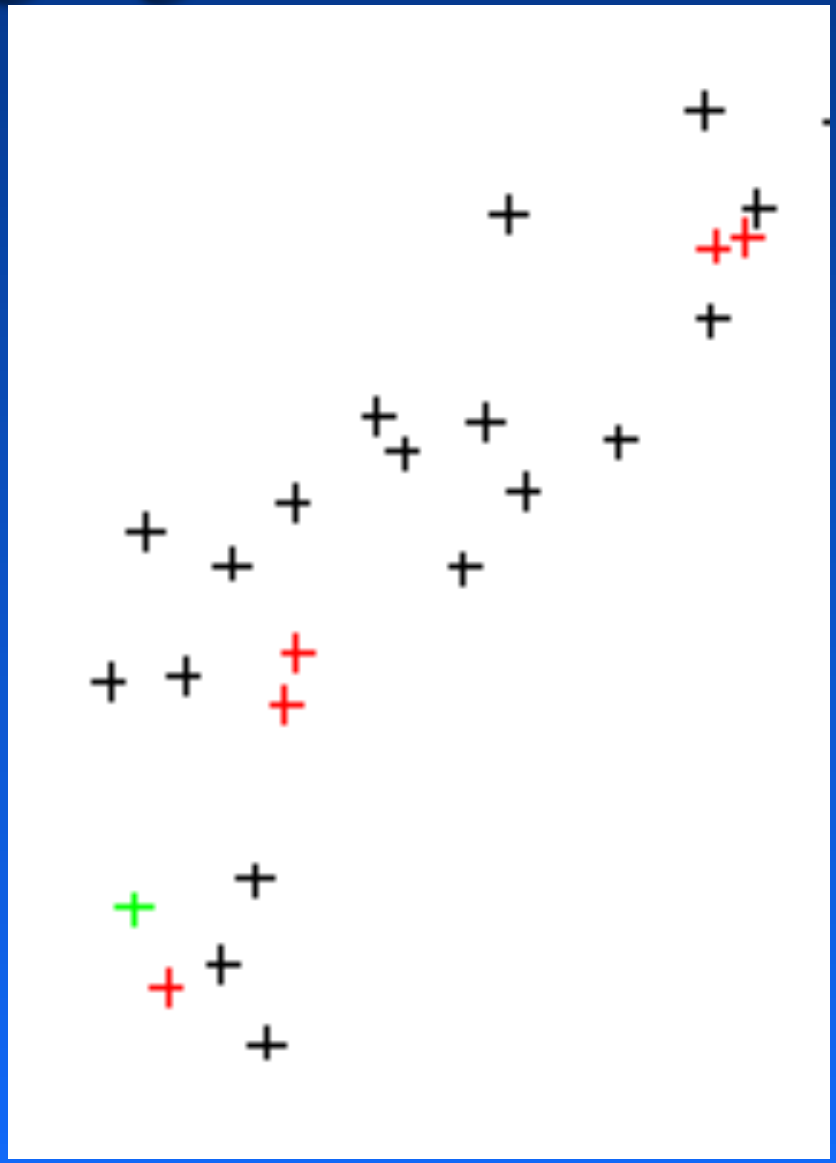
Munging

	x	y
1	-1.09	-0.09
2	-0.94	0.24
3	-0.56	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.06	-0.19
…		
61	-0.53	0.77
…		
89	-0.95	0.17



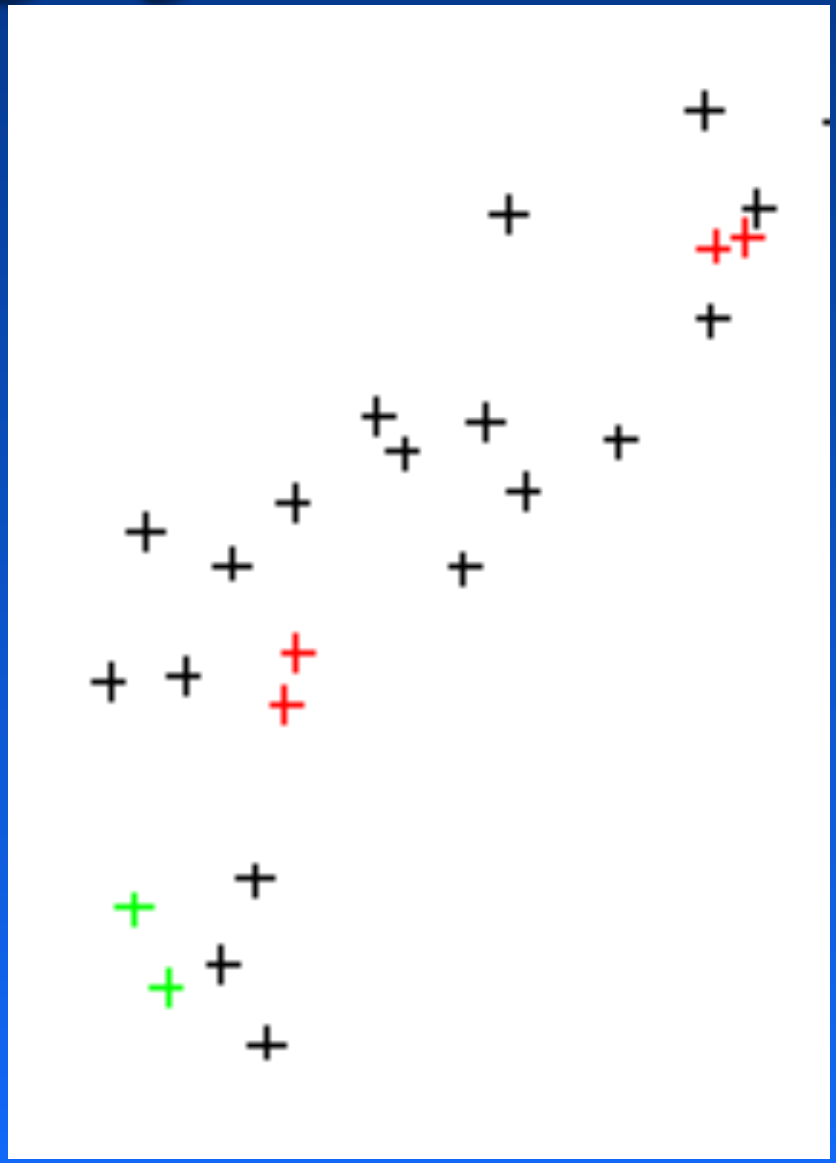
Munging

	x	y	d
1	-1.09	-0.09	-
2	-0.94	0.24	0.35
3	-0.56	0.76	0.99
4	0.82	0.56	2.02
5	-0.23	-0.90	1.18
…			
19	-1.06	-0.19	0.11
…			
61	-0.53	0.77	1.02
…			
89	-0.95	0.17	0.28



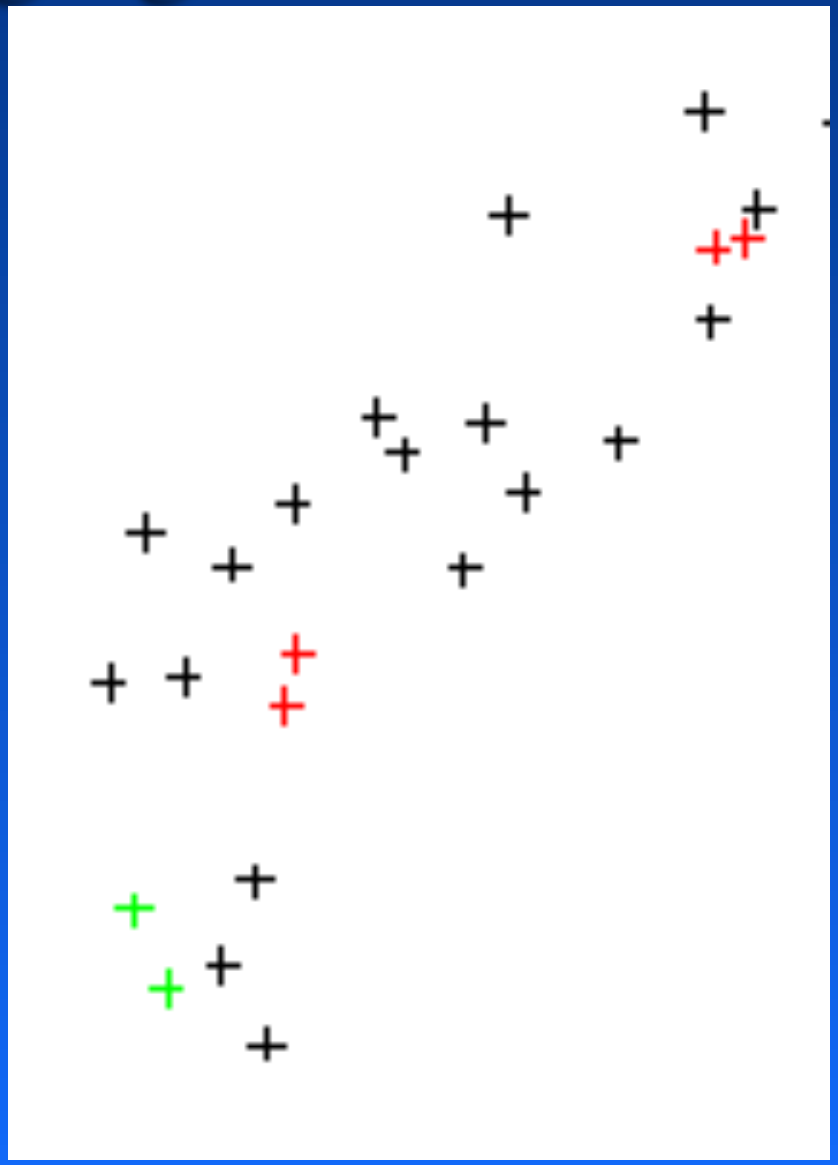
Munging

	x	y	d
1	-1.09	-0.09	-
2	-0.94	0.24	0.35
3	-0.56	0.76	0.99
4	0.82	0.56	2.02
5	-0.23	-0.90	1.18
…			
19	-1.06	-0.19	0.11
…			
61	-0.53	0.77	1.02
…			
89	-0.95	0.17	0.28



Munging

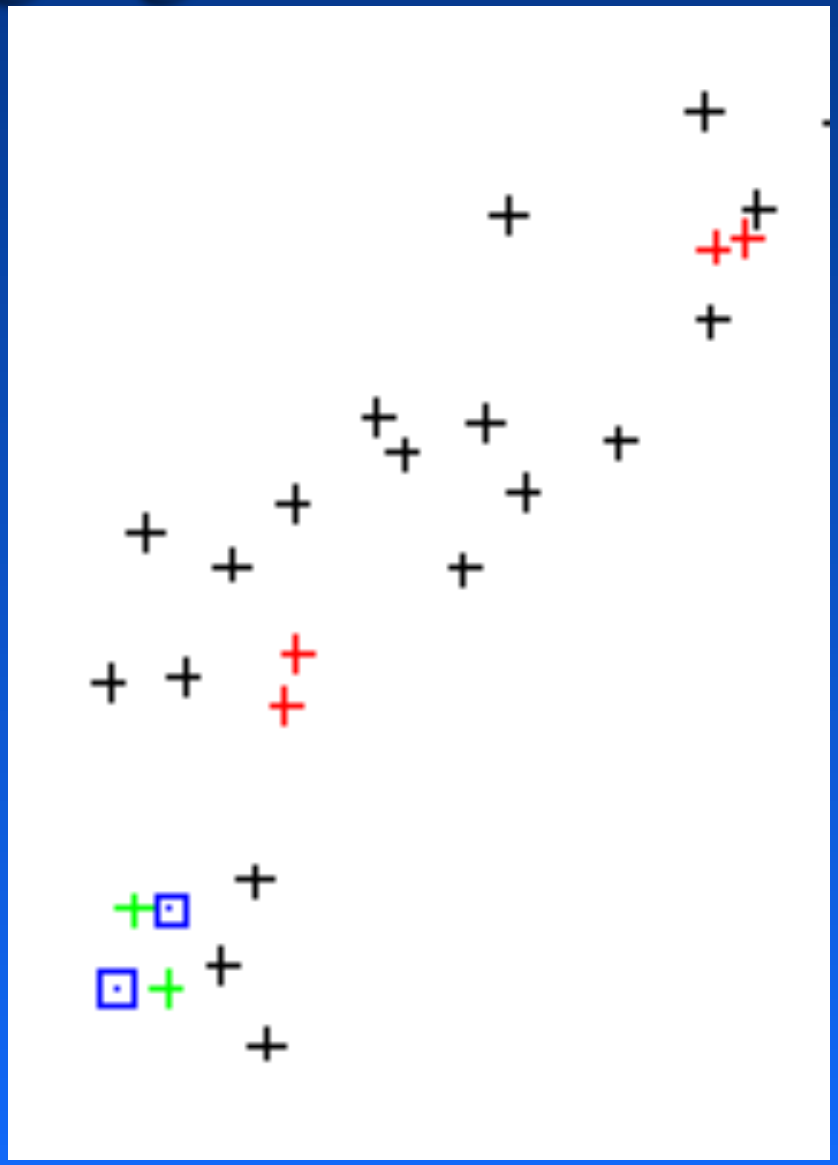
	x	y
1	-1.09	-0.09
2	-0.94	0.24
3	-0.56	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.06	-0.19
…		
61	-0.53	0.77
…		
89	-0.95	0.17
	yes	no



Munging

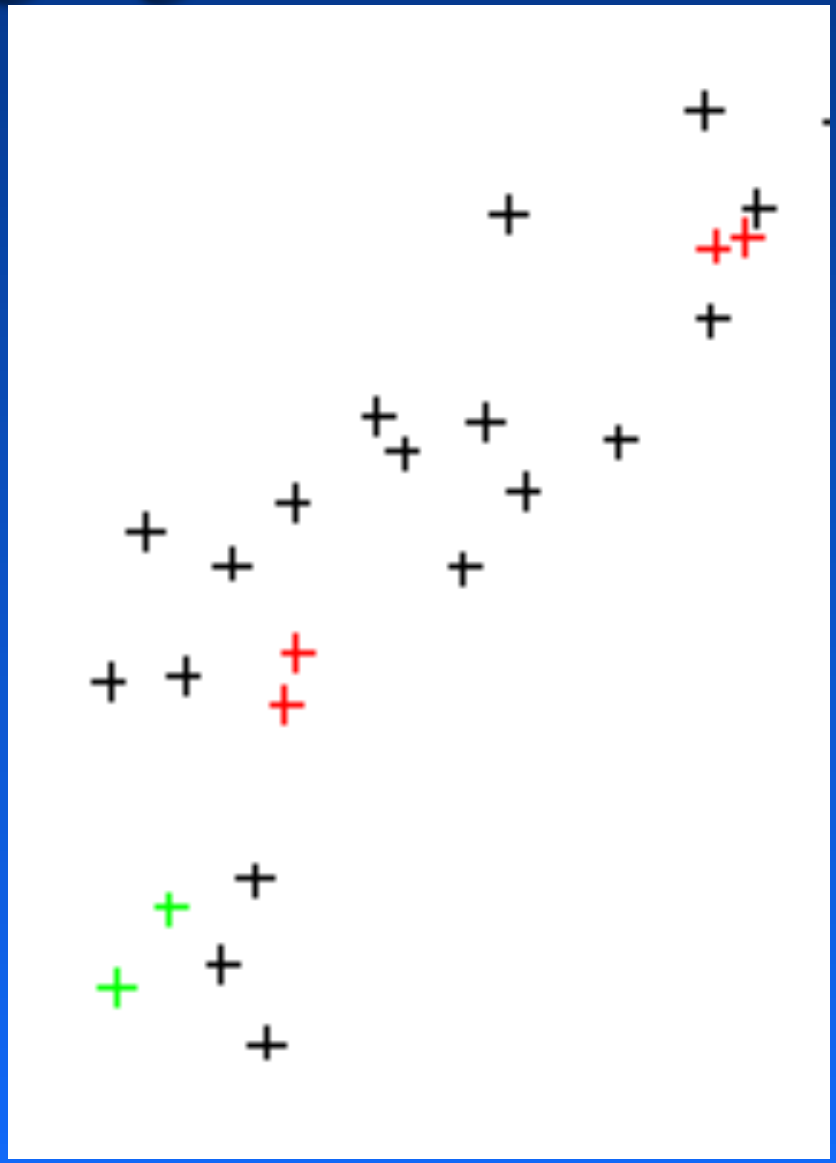
	x	y	x'	y'
1	-1.09	-0.09	-1.06	-0.09
2	-0.94	0.24		
3	-0.56	0.76		
4	0.82	0.56		
5	-0.23	-0.90		
…				
19	-1.06	-0.19	-1.11	-0.19
…				
61	-0.53	0.77		
…				
89	-0.95	0.17		

yes no



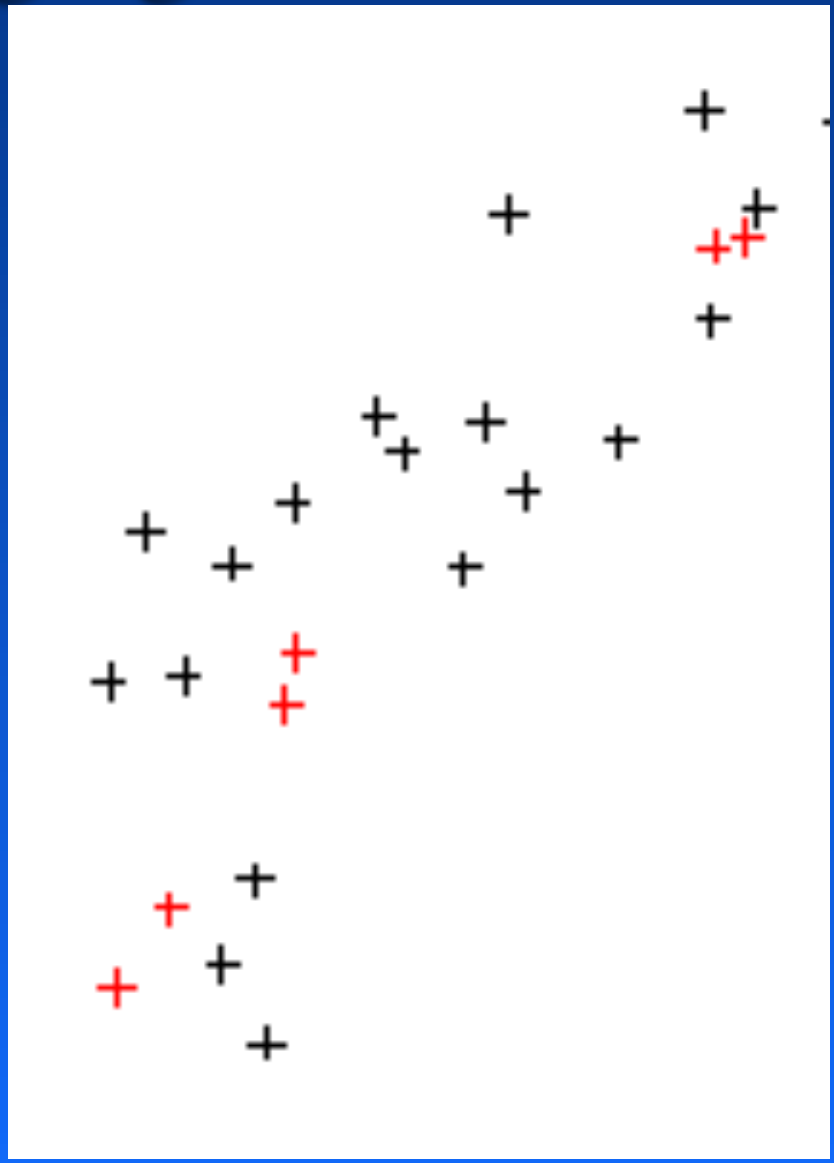
Munging

	x	y
1	-1.06	-0.09
2	-0.94	0.24
3	-0.56	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.11	-0.19
…		
61	-0.53	0.77
…		
89	-0.95	0.17



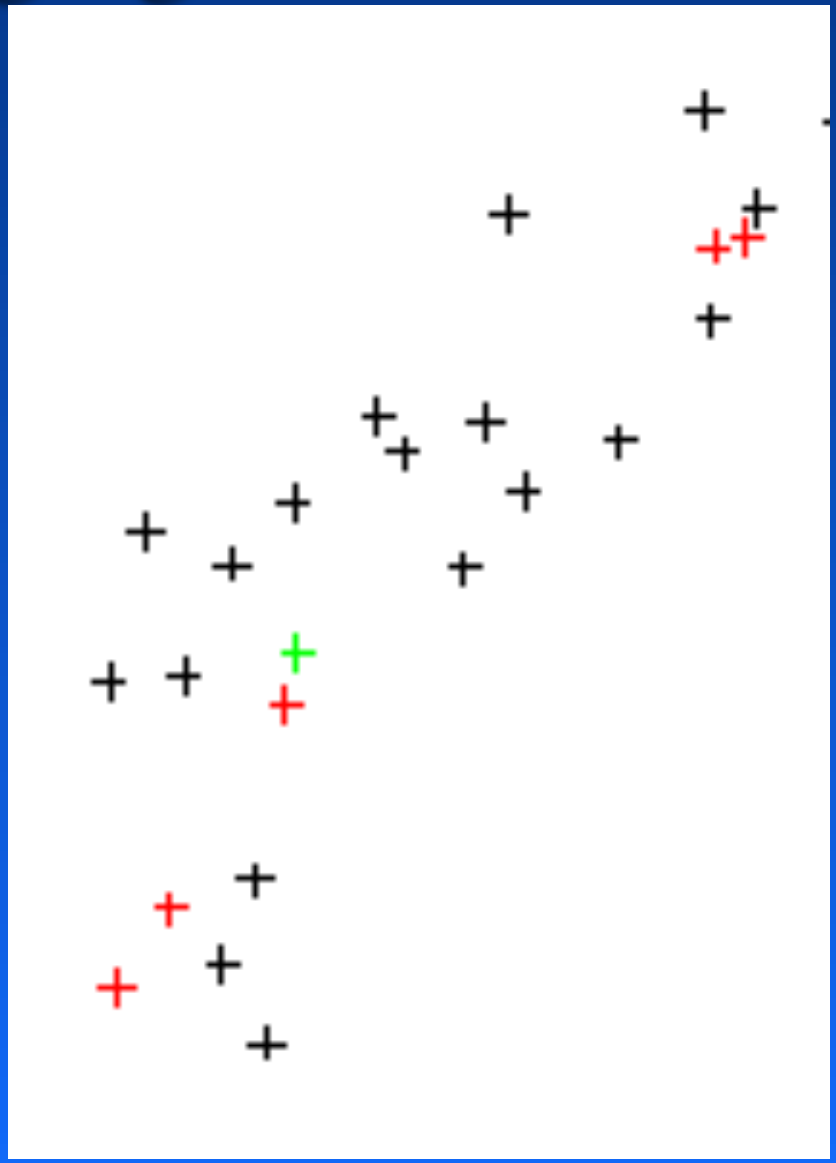
Munging

	x	y
1	-1.06	-0.09
2	-0.94	0.24
3	-0.56	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.11	-0.19
…		
61	-0.53	0.77
…		
89	-0.95	0.17



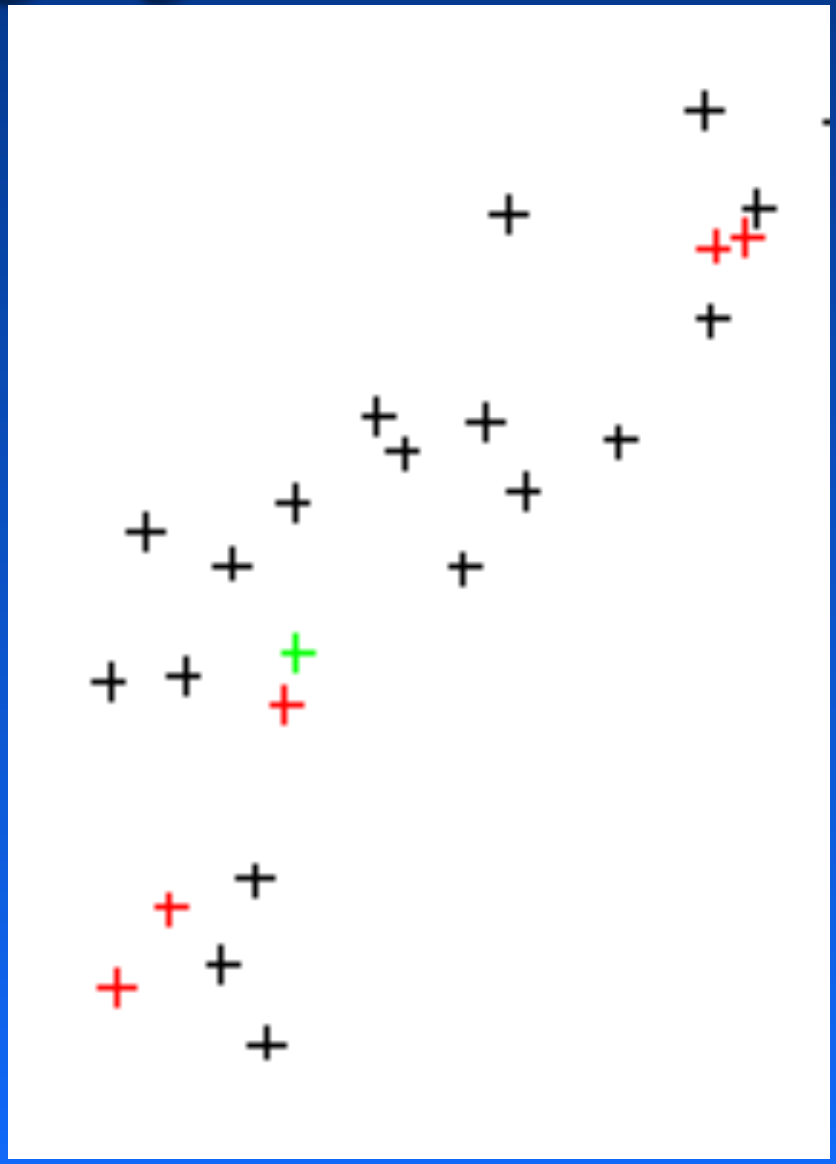
Munging

	x	y
1	-1.06	-0.09
2	-0.94	0.24
3	-0.56	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.11	-0.19
…		
61	-0.53	0.77
…		
89	-0.95	0.17



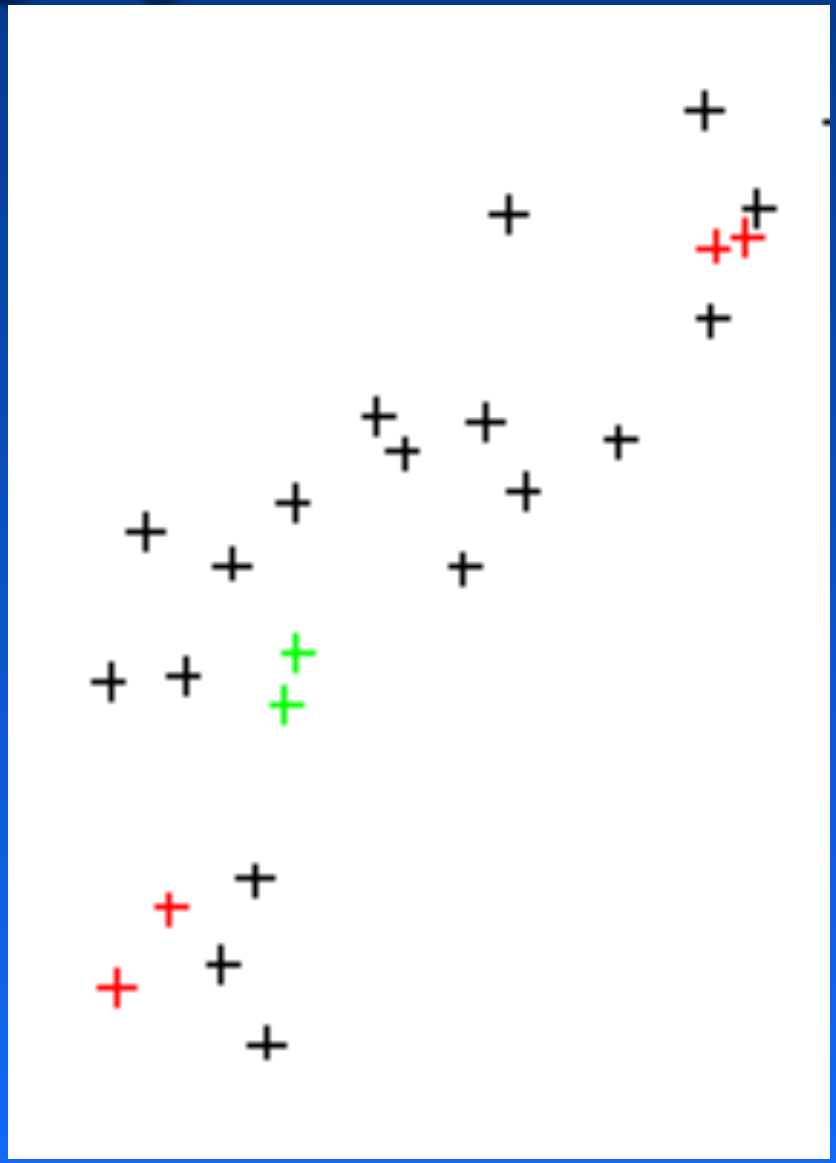
Munging

	x	y	d
1	-1.06	-0.09	0.35
2	-0.94	0.24	-
3	-0.56	0.76	0.65
4	0.82	0.56	1.79
5	-0.23	-0.90	1.34
…			
19	-1.11	-0.19	0.46
…			
61	-0.53	0.77	0.68
…			
89	-0.95	0.17	0.07



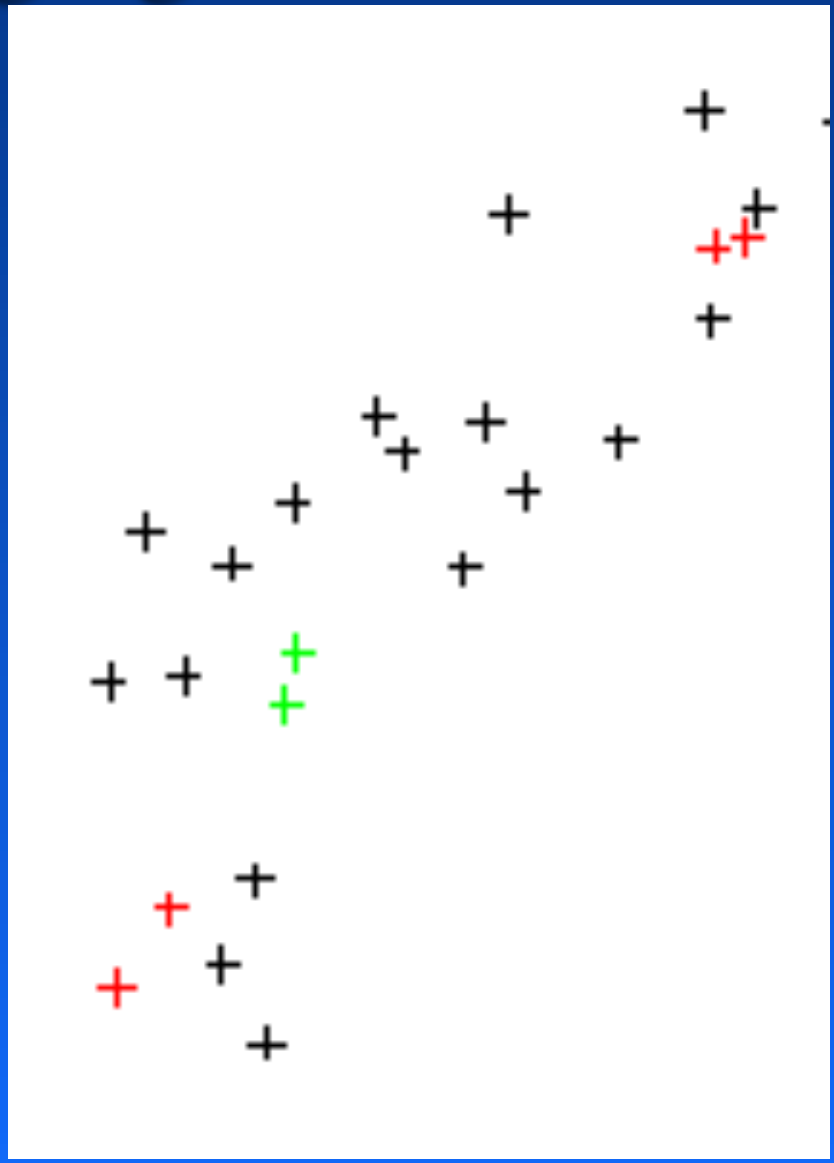
Munging

	x	y	d
1	-1.06	-0.09	0.35
2	-0.94	0.24	-
3	-0.56	0.76	0.65
4	0.82	0.56	1.79
5	-0.23	-0.90	1.34
…			
19	-1.11	-0.19	0.46
…			
61	-0.53	0.77	0.68
…			
89	-0.95	0.17	0.07



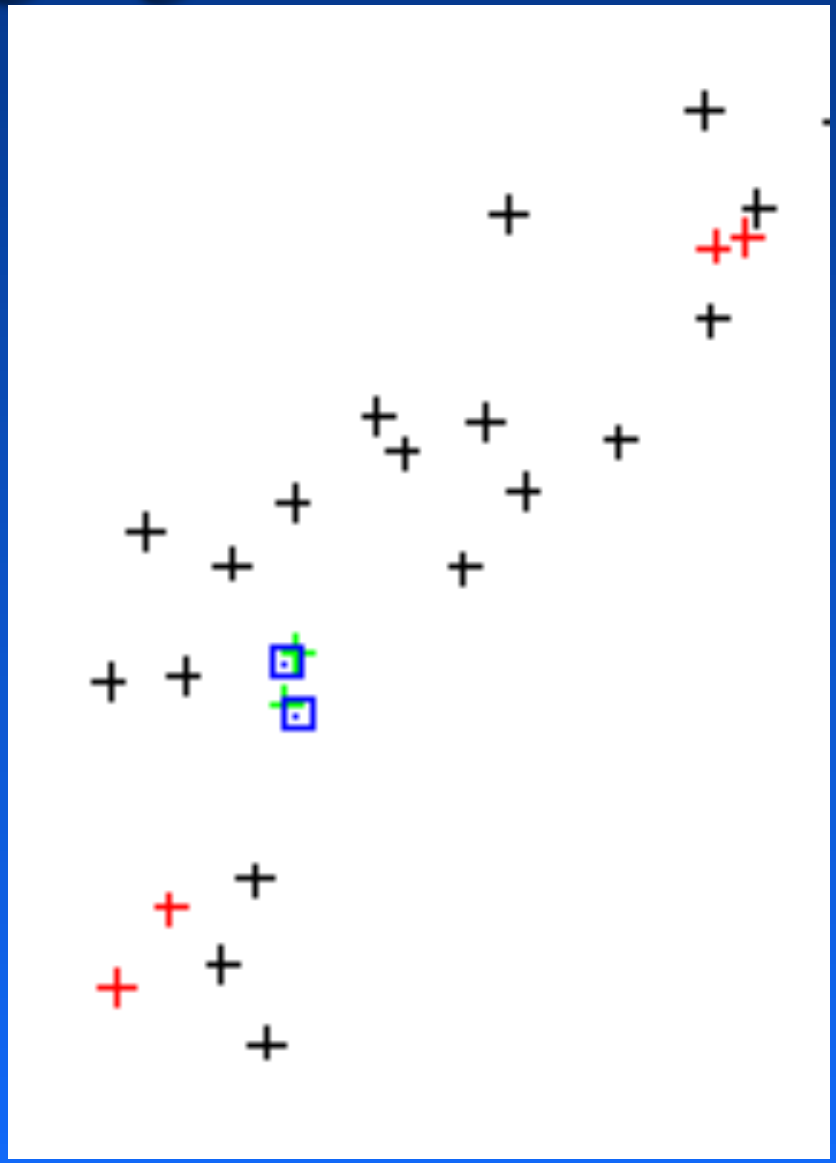
Munging

	x	y
1	-1.06	-0.09
2	-0.94	0.24
3	-0.56	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.11	-0.19
…		
61	-0.53	0.77
…		
89	-0.95	0.17
	no	yes



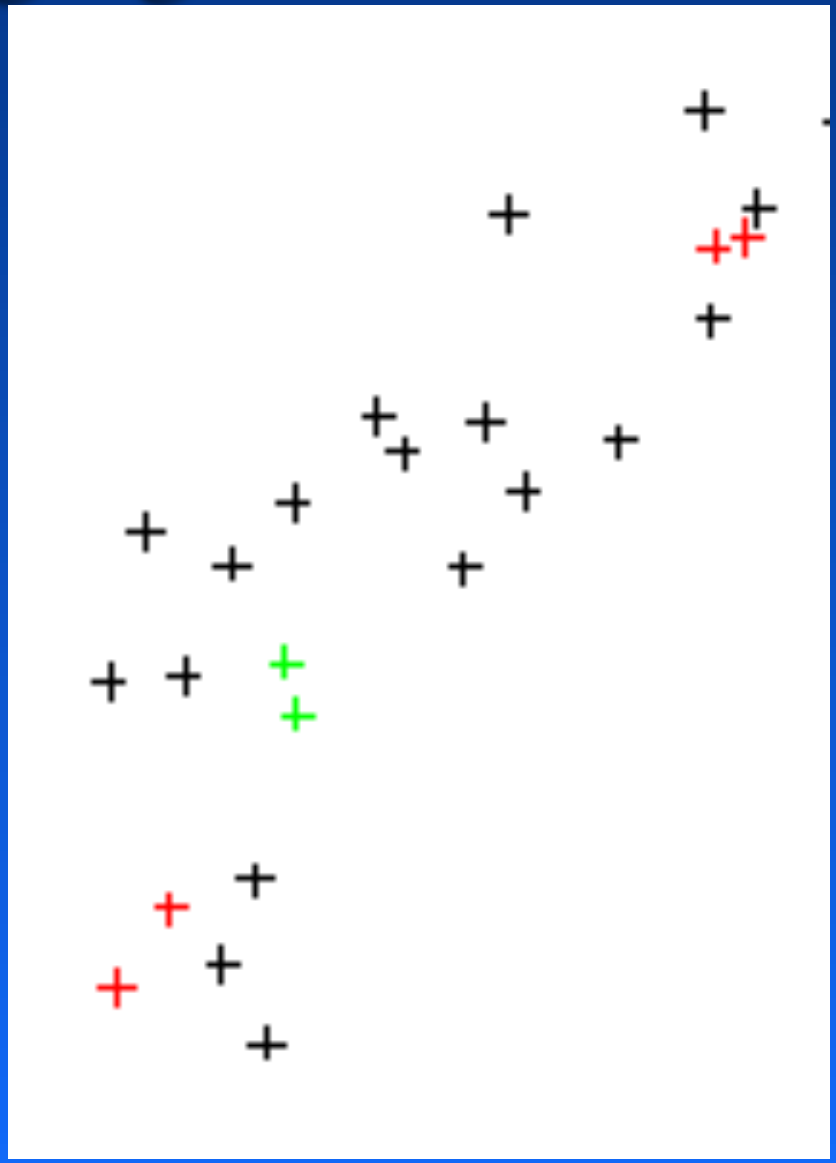
Munging

	x	y	x'	y'
1	-1.06	-0.09		
2	-0.94	0.24	-0.94	0.17
3	-0.56	0.76		
4	0.82	0.56		
5	-0.23	-0.90		
…				
19	-1.11	-0.19		
…				
61	-0.53	0.77		
…				
89	-0.95	0.17	-0.95	0.23
	no	yes		



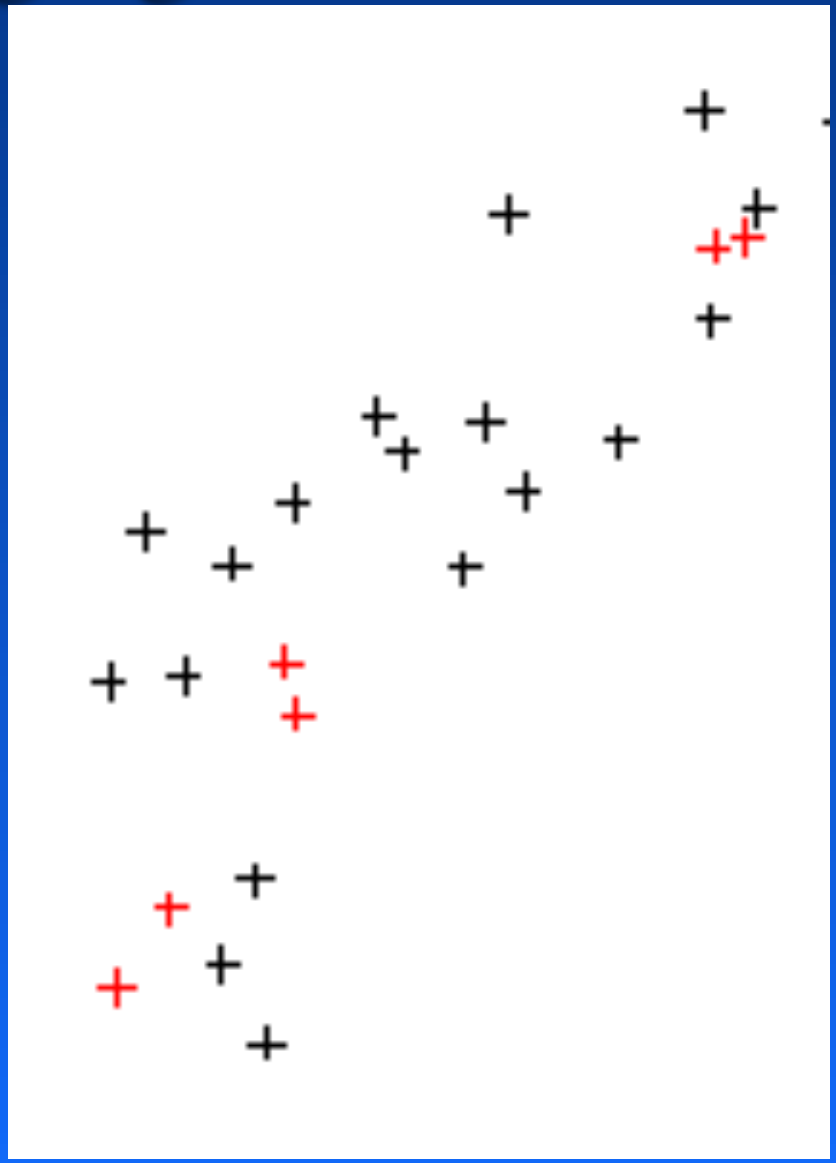
Munging

	x	y
1	-1.06	-0.09
2	-0.94	0.17
3	-0.56	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.11	-0.19
…		
61	-0.53	0.77
…		
89	-0.95	0.23



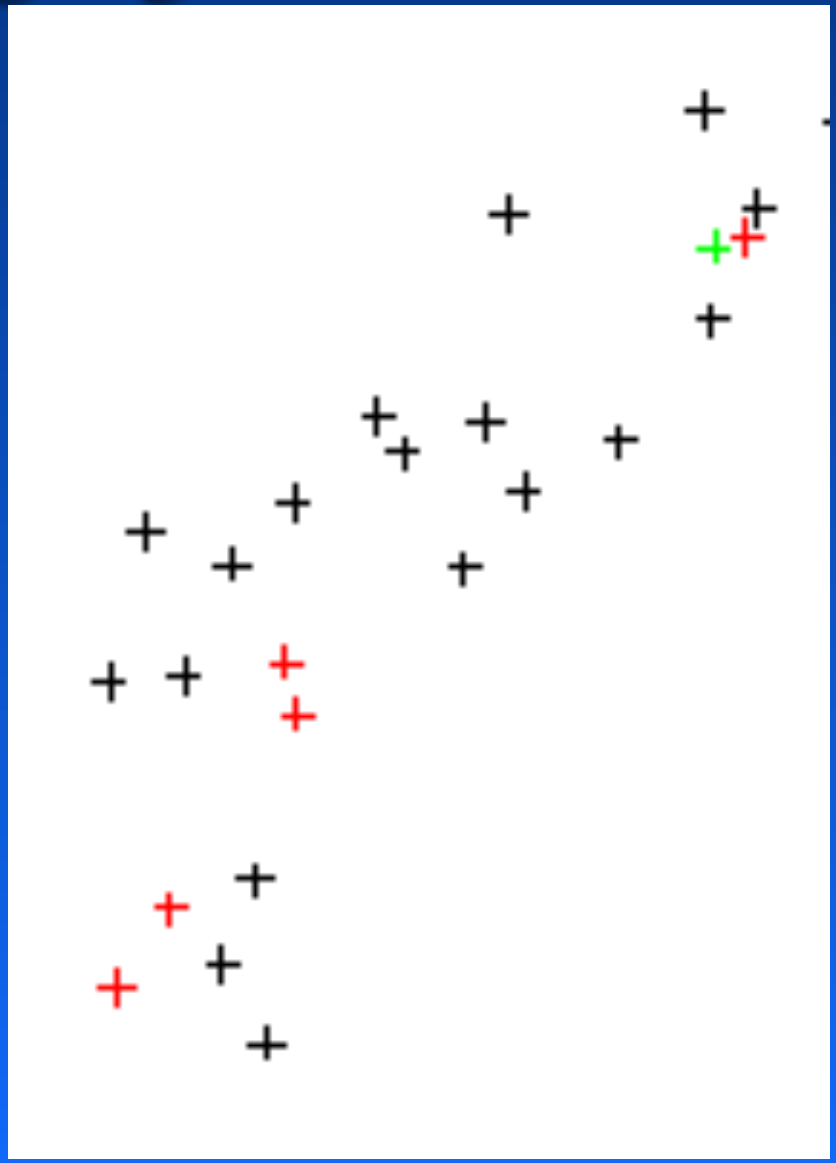
Munging

	x	y
1	-1.06	-0.09
2	-0.94	0.17
3	-0.56	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.11	-0.19
…		
61	-0.53	0.77
…		
89	-0.95	0.23



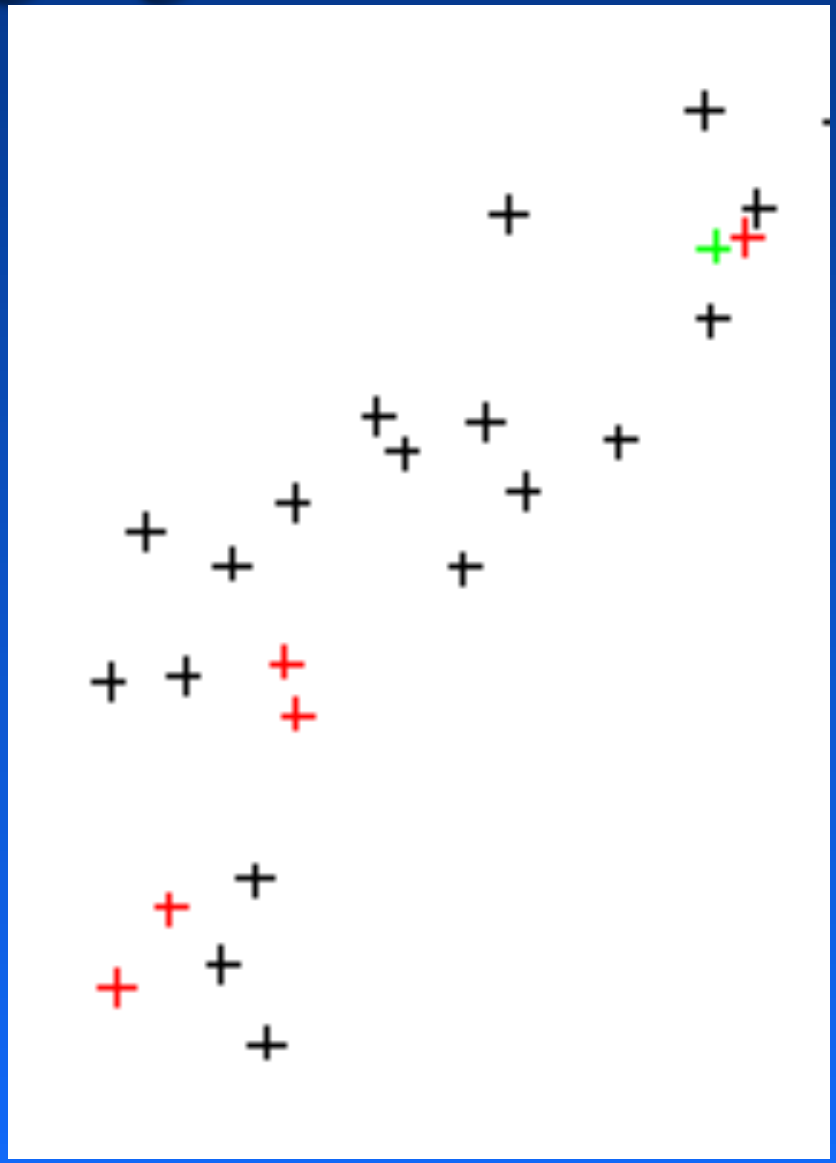
Munging

	x	y
1	-1.06	-0.09
2	-0.94	0.17
3	-0.56	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.11	-0.19
…		
61	-0.53	0.77
…		
89	-0.95	0.23



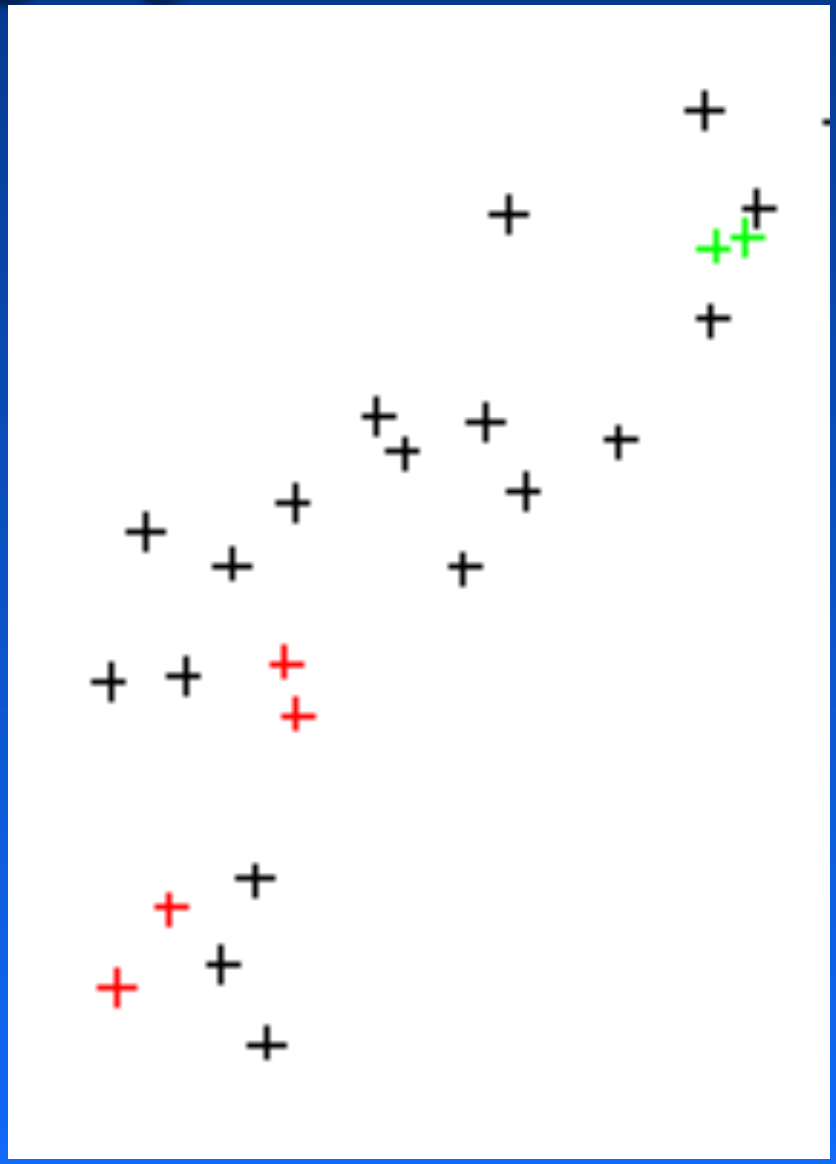
Munging

	x	y	d
1	-1.06	-0.09	0.99
2	-0.94	0.17	0.72
3	-0.56	0.76	-
4	0.82	0.56	1.39
5	-0.23	-0.90	1.69
…			
19	-1.11	-0.19	1.10
…			
61	-0.53	0.77	0.03
…			
89	-0.95	0.23	0.67



Munging

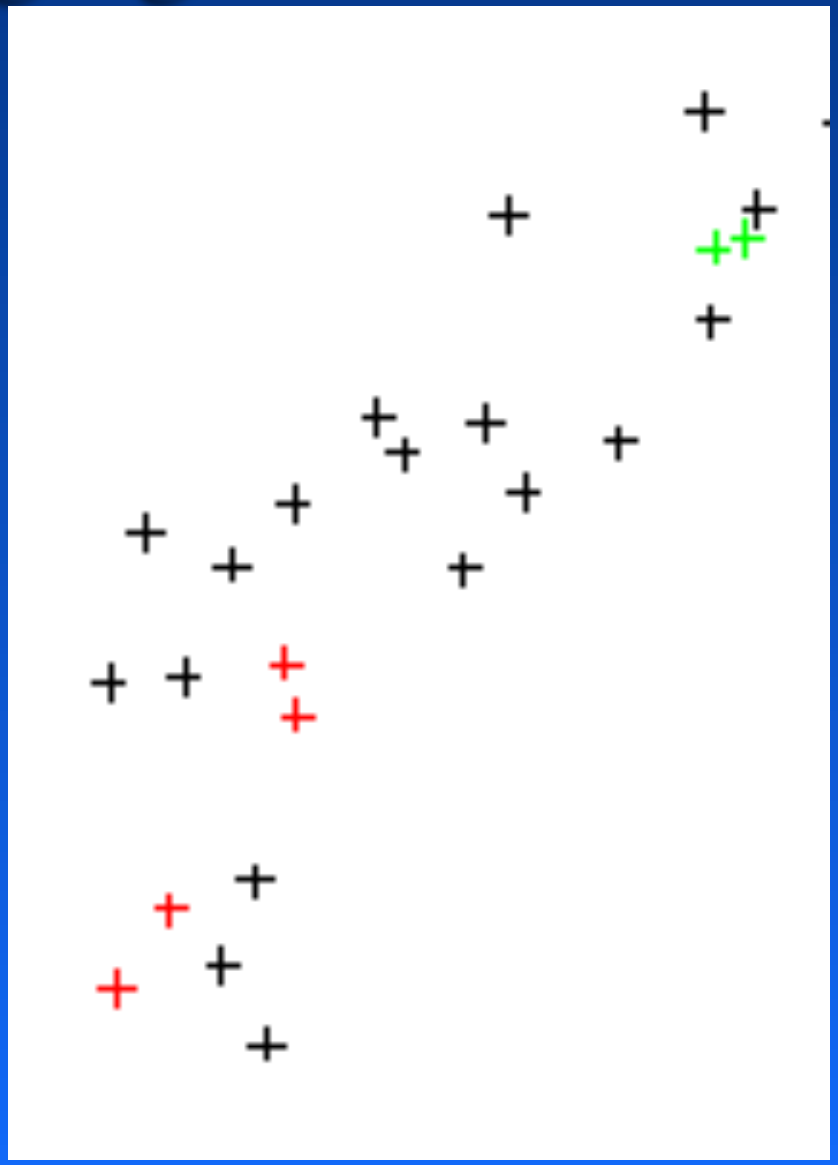
	x	y	d
1	-1.06	-0.09	0.99
2	-0.94	0.17	0.72
3	-0.56	0.76	-
4	0.82	0.56	1.39
5	-0.23	-0.90	1.69
…			
19	-1.11	-0.19	1.10
…			
61	-0.53	0.77	0.03
…			
89	-0.95	0.23	0.67



Munging

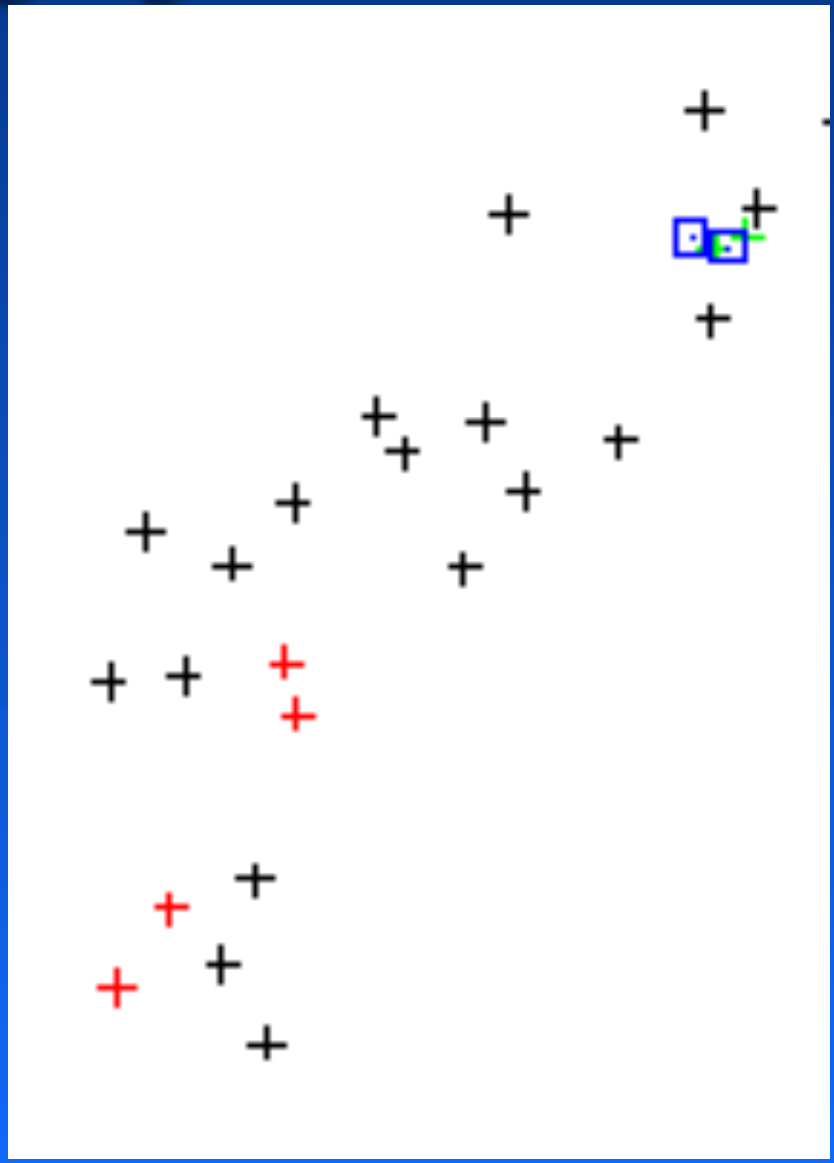
	x	y
1	-1.06	-0.09
2	-0.94	0.17
3	-0.56	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.11	-0.19
…		
61	-0.53	0.77
…		
89	-0.95	0.23

	yes	no



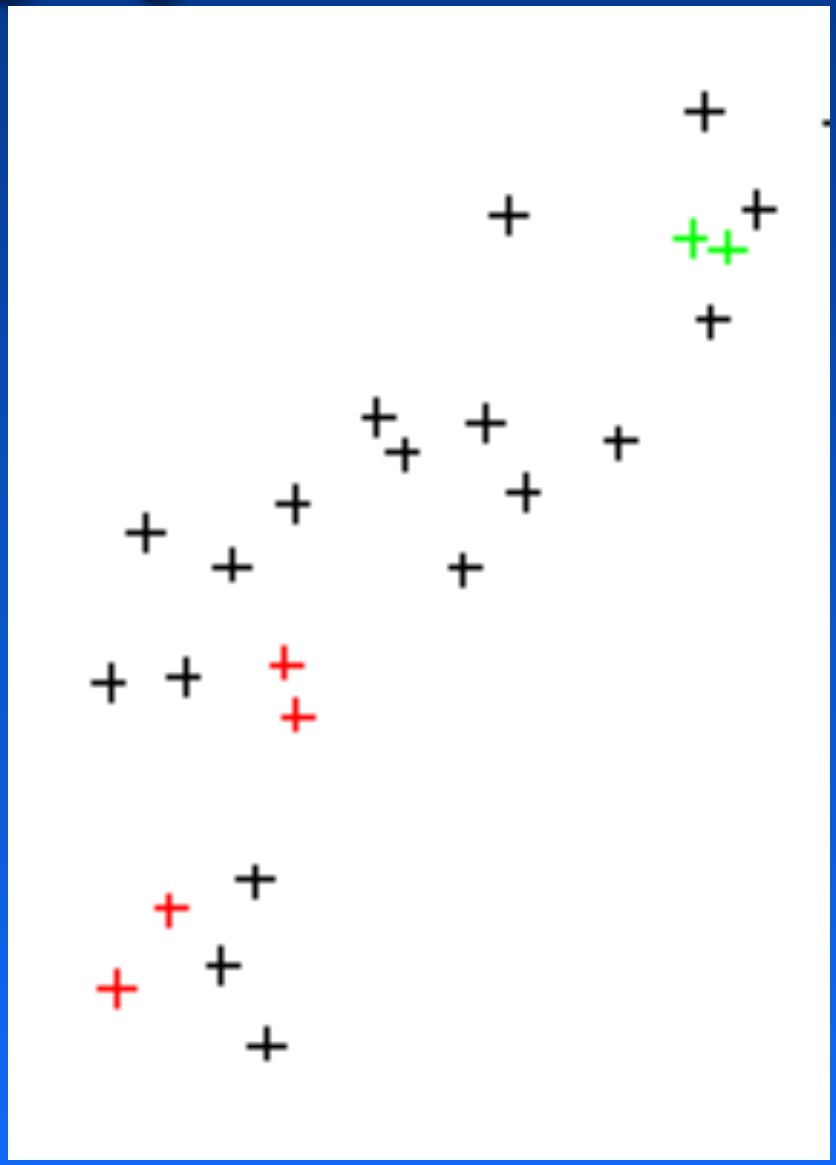
Munging

	x	y	x'	y'
1	-1.06	-0.09		
2	-0.94	0.17		
3	-0.56	0.76	-0.56	-0.54
4	0.82	0.56		
5	-0.23	-0.90		
…				
19	-1.11	-0.19		
…				
61	-0.53	0.77	-0.53	-0.58
…				
89	-0.95	0.23		
	yes	no		



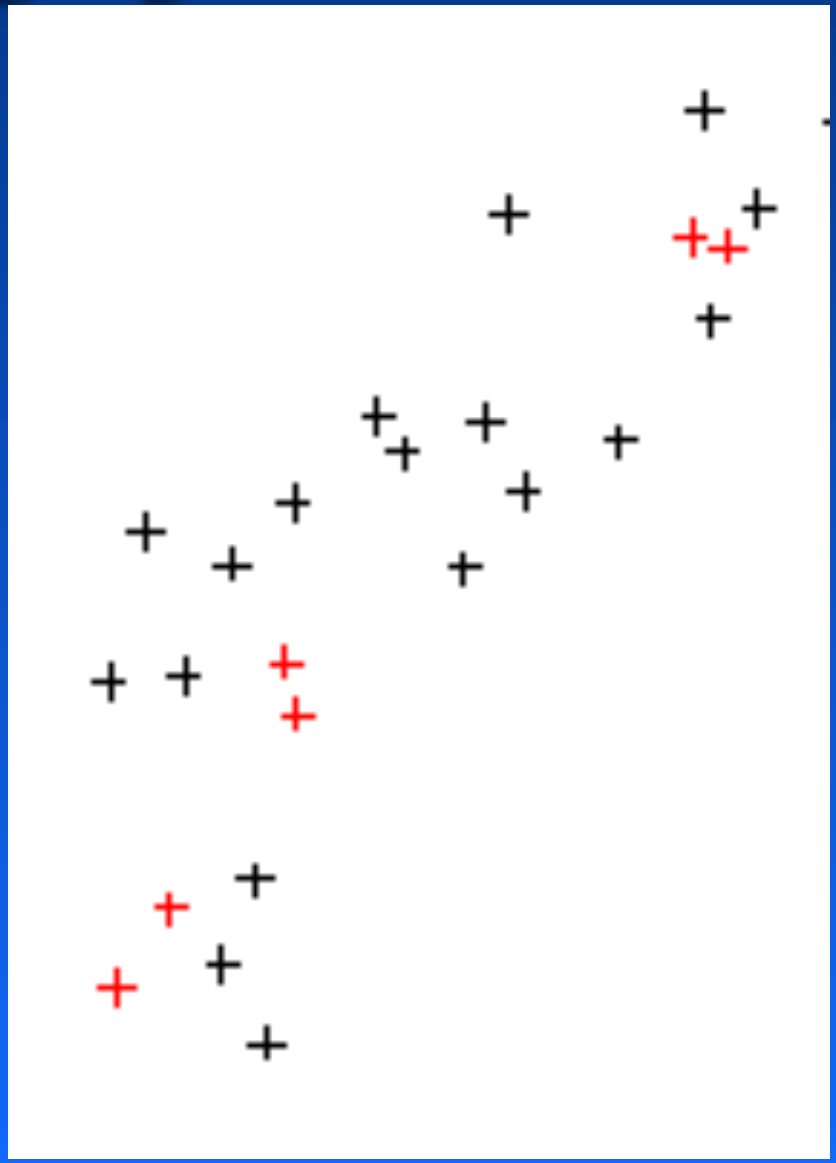
Munging

	x	y
1	-1.06	-0.09
2	-0.94	0.17
3	-0.54	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.11	-0.19
…		
61	-0.58	0.77
…		
89	-0.95	0.23



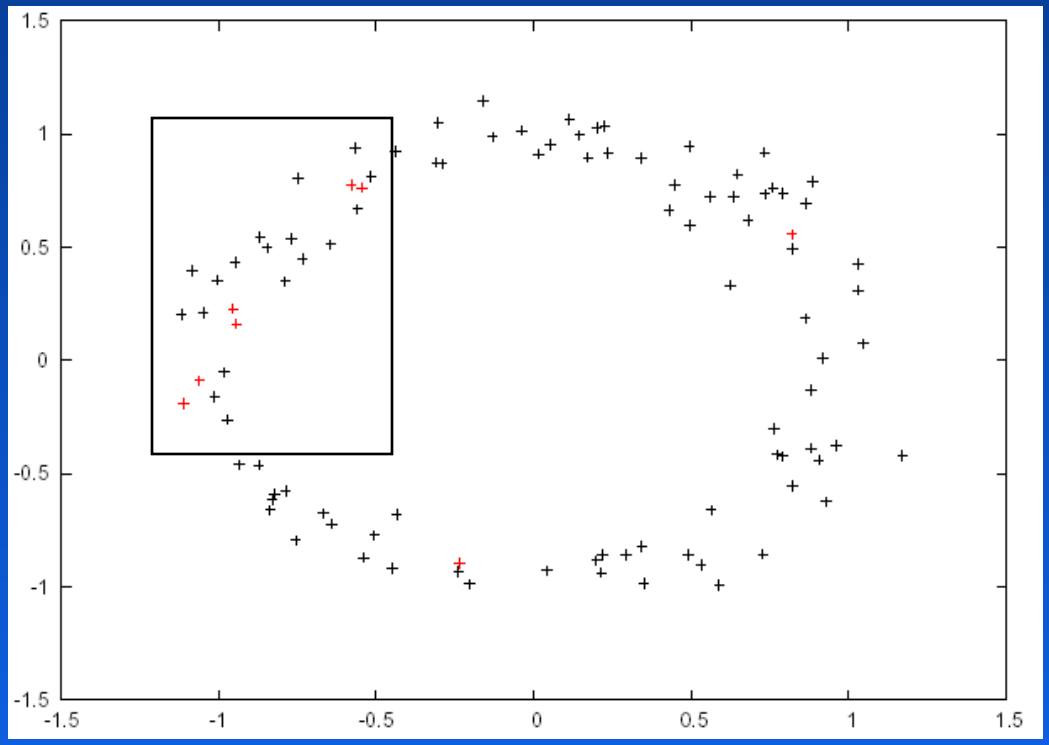
Munging

	x	y
1	-1.06	-0.09
2	-0.94	0.17
3	-0.54	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.11	-0.19
…		
61	-0.58	0.77
…		
89	-0.95	0.23



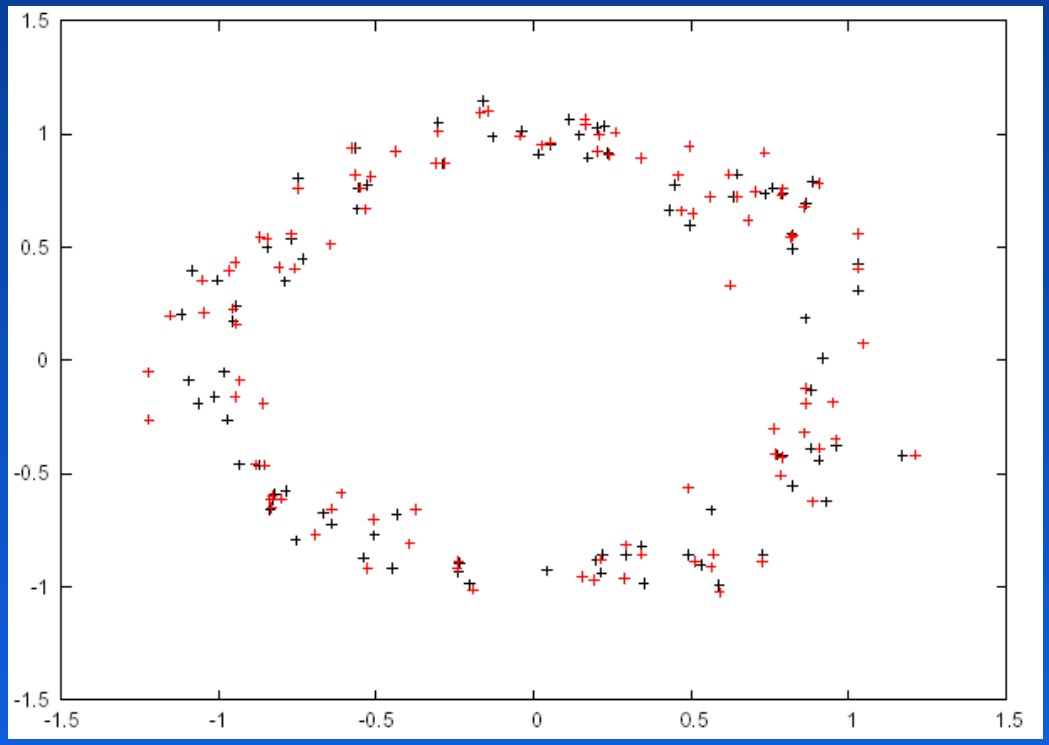
Munging

	x	y
1	-1.06	-0.09
2	-0.94	0.17
3	-0.54	0.76
4	0.82	0.56
5	-0.23	-0.90
…		
19	-1.11	-0.19
…		
61	-0.58	0.77
…		
89	-0.95	0.23

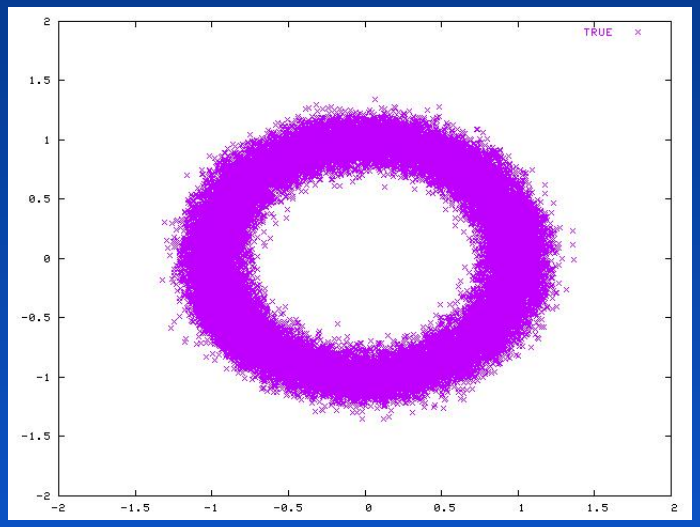


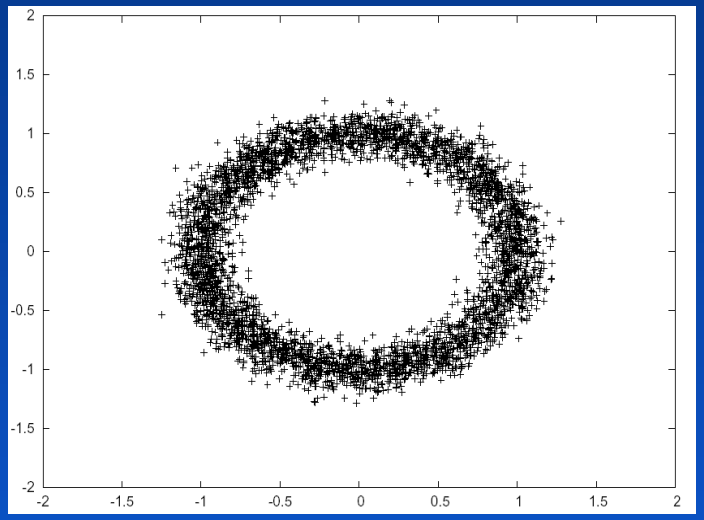
Munging

	x	y
1	-0.93	-0.09
2	-0.94	0.16
3	-0.55	0.76
4	0.82	0.55
5	-0.19	-1.01
…		
19	-0.85	-0.19
…		
61	-0.57	0.82
…		
89	-0.95	0.23

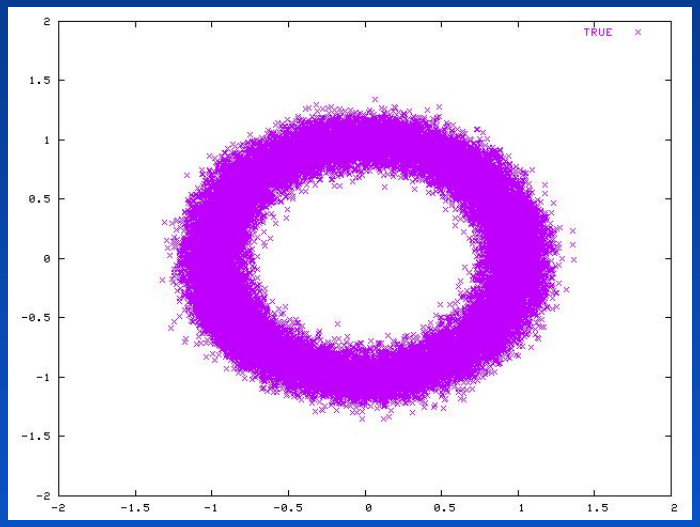


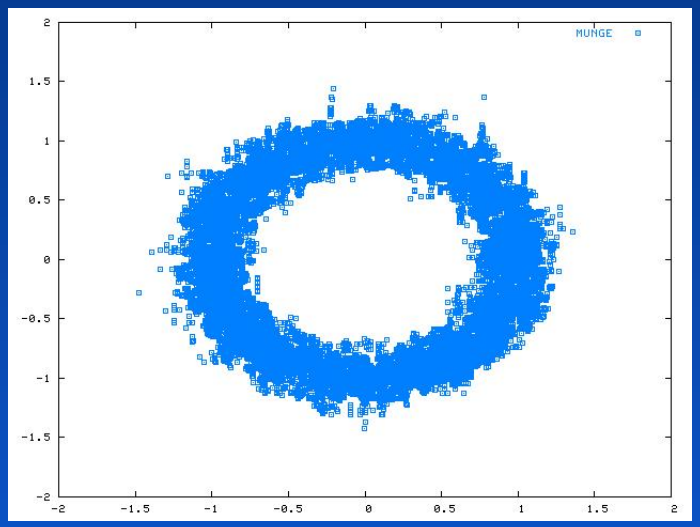
Synthetic Data: Munging



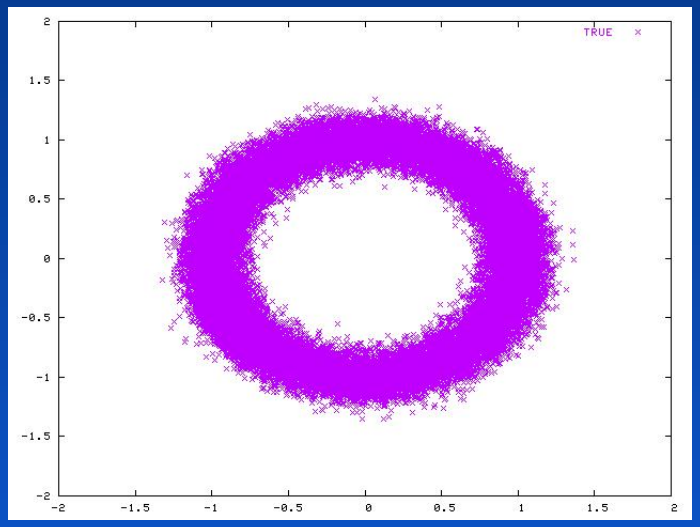


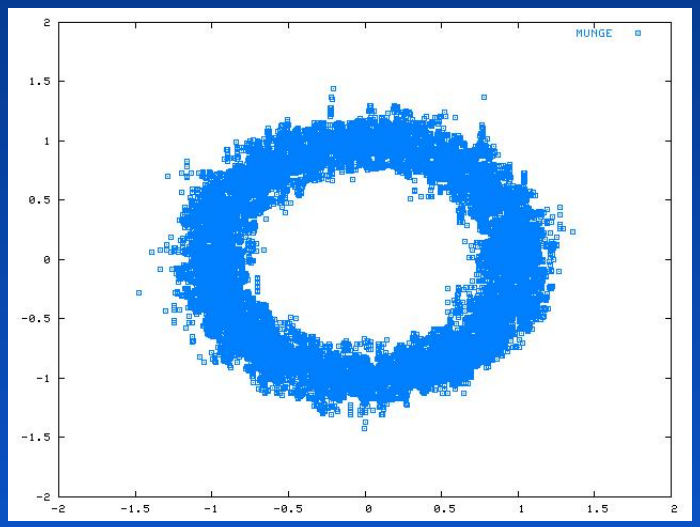
Synthetic Data: Munging

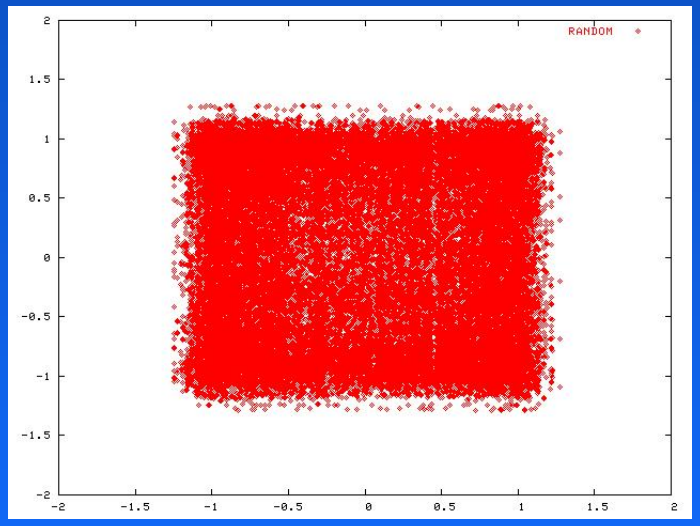




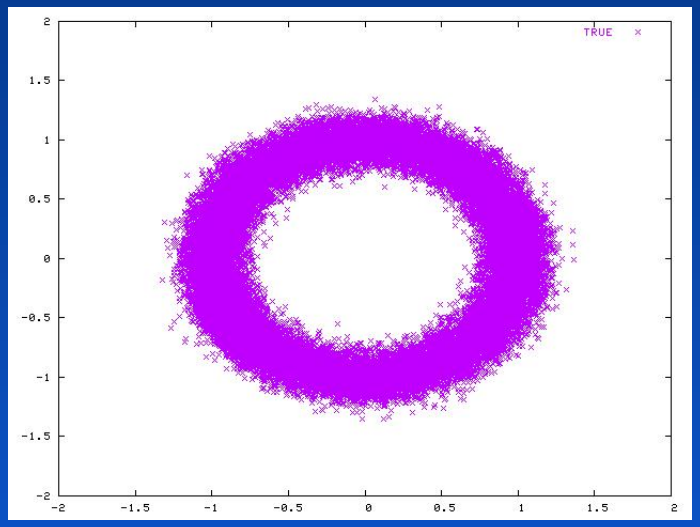
Synthetic Data: Munging

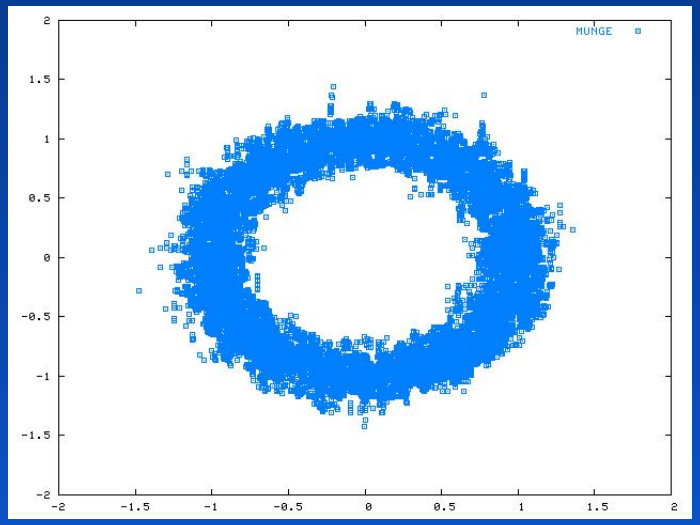


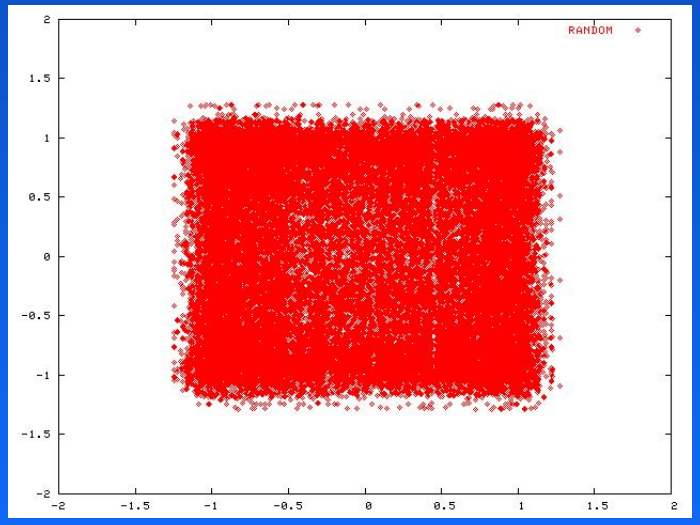


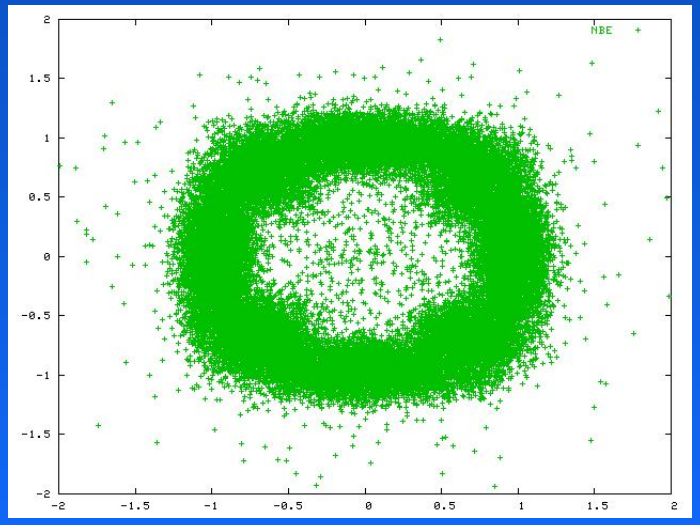


Synthetic Data: Munging

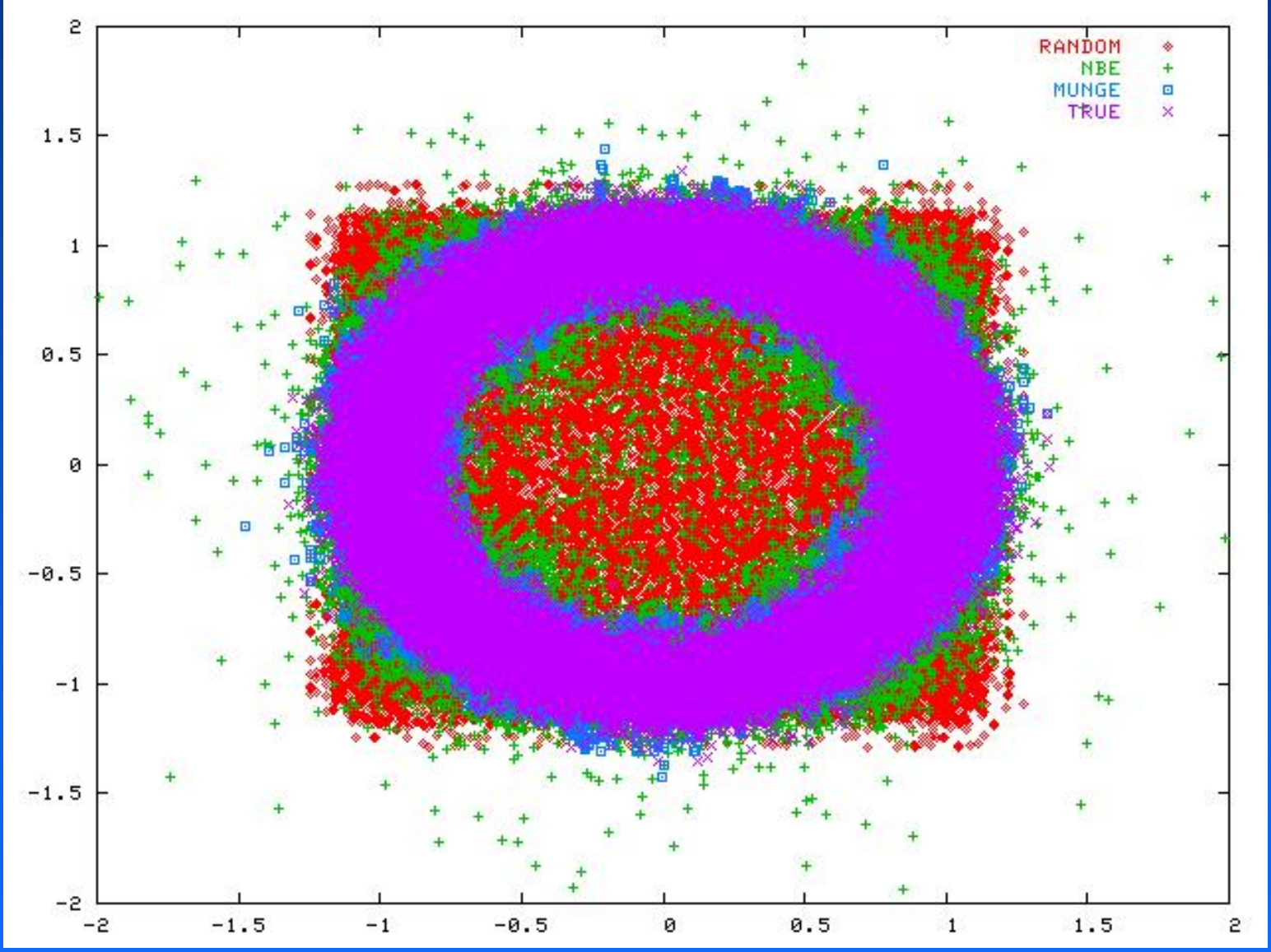








Synthetic Data

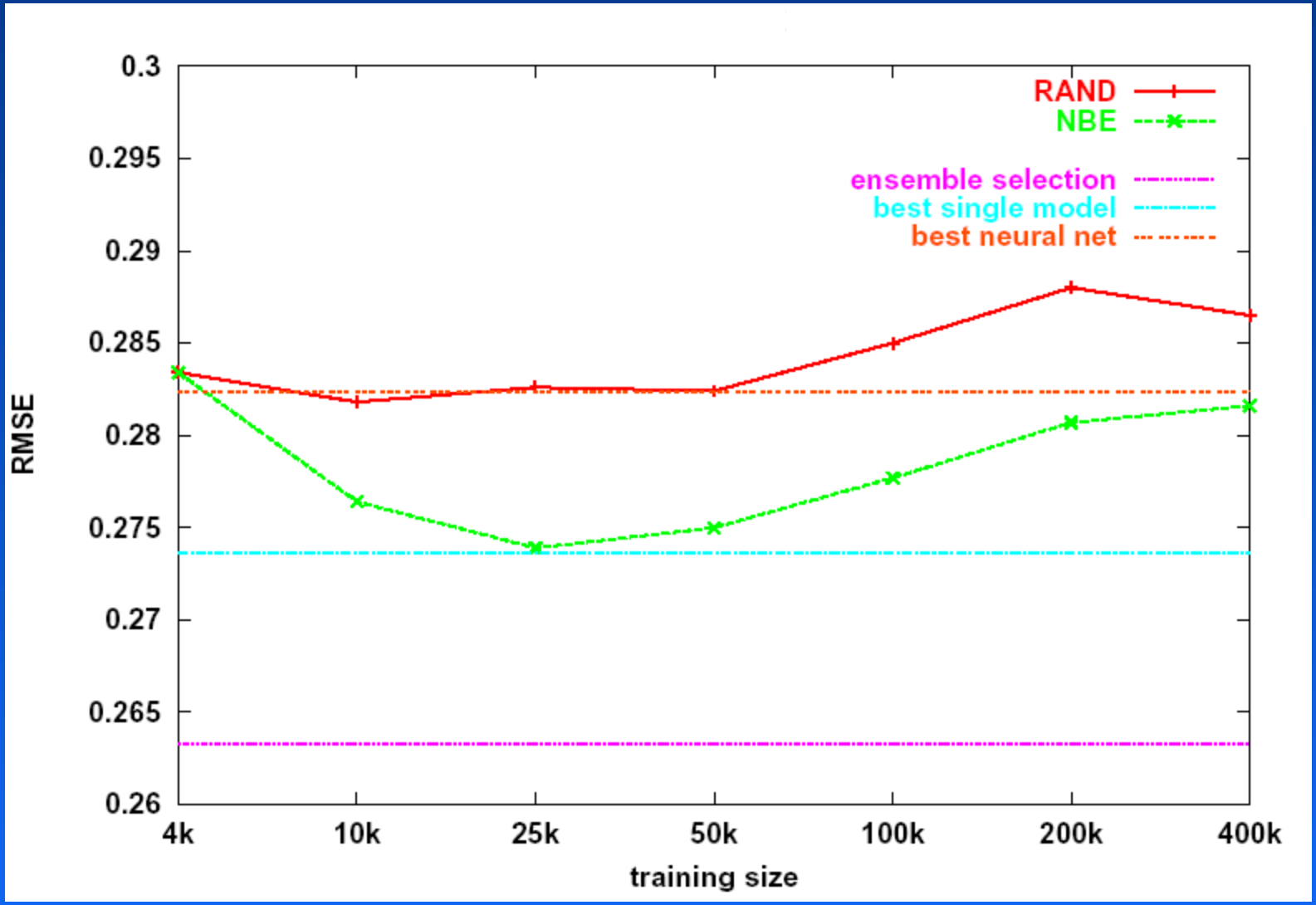


Why Does Munging Work?

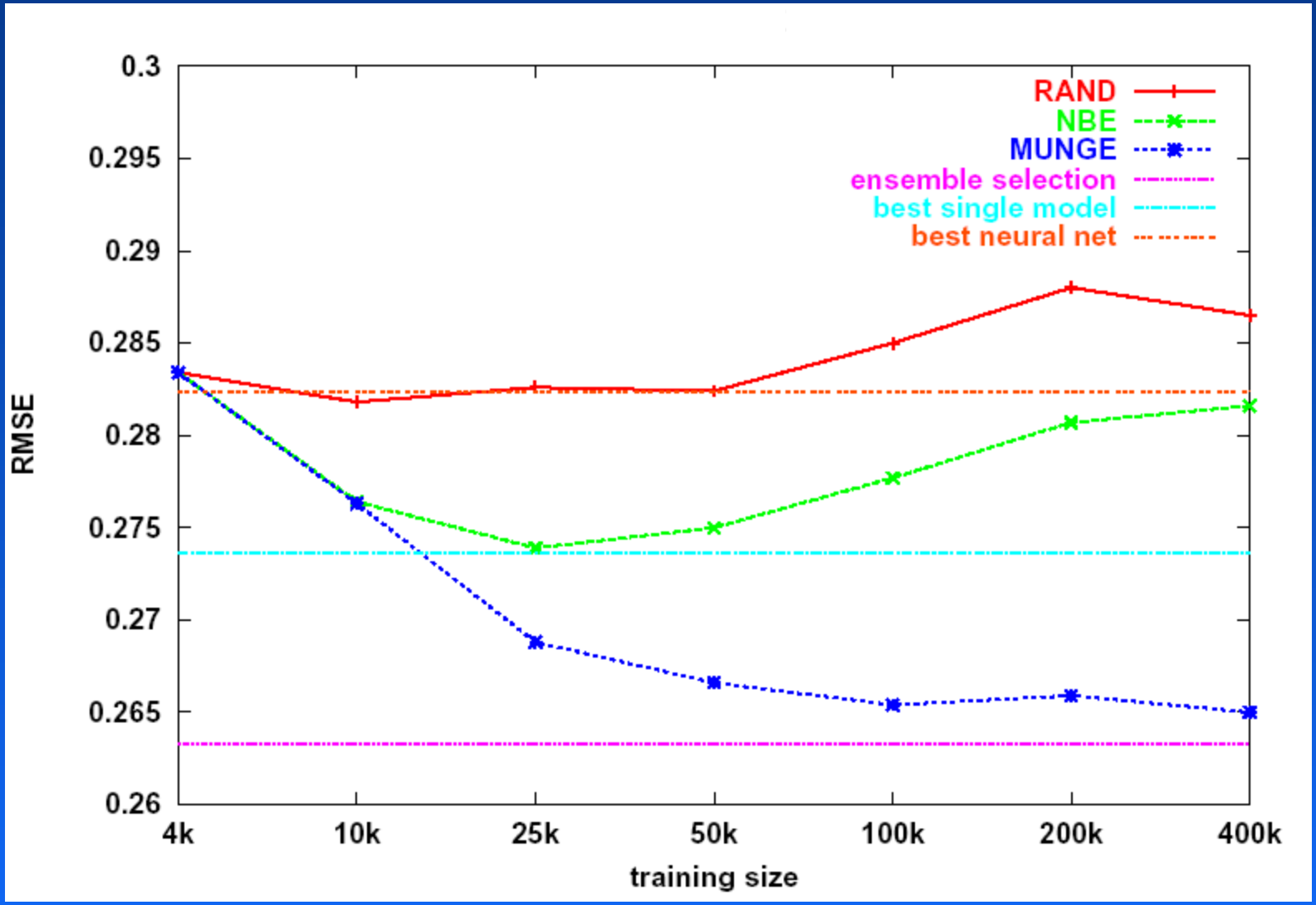
- Deceptively simple, yet effective, algorithm
- Originally developed munging to protect privacy in data without changing the character of the data too much
- Never explicitly models p(x)
- Like KNN, the data *IS* the model
- Swapping values between nearest neighbors is critical to preserve the conditional structure
- Can use kernels instead of Euclidean Distance neighbors
- Surprisingly insensitive to parameters
- Probably falls apart in high dimensions…

Does Munging Work for Model Compression?

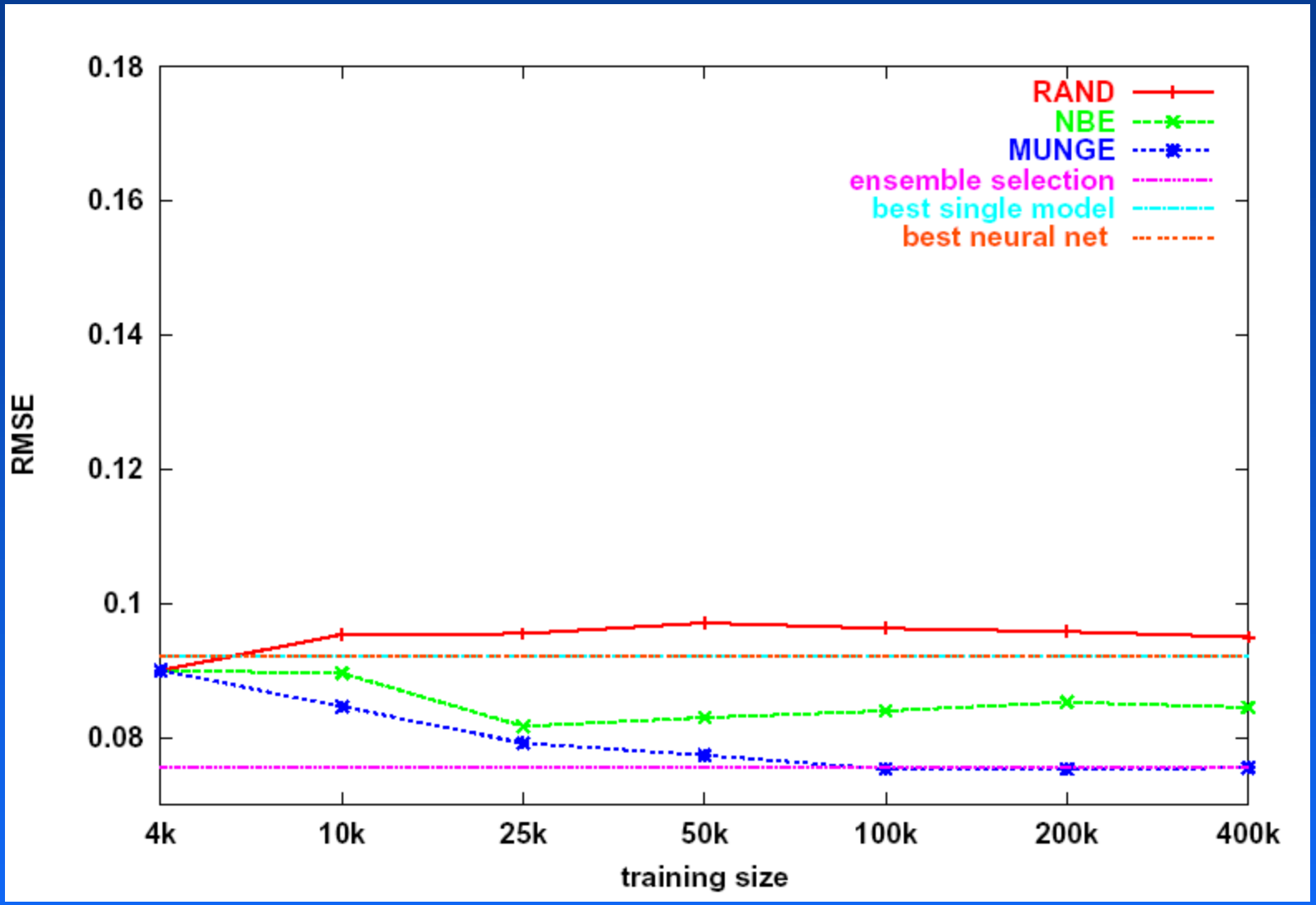
Average Results by Train Size



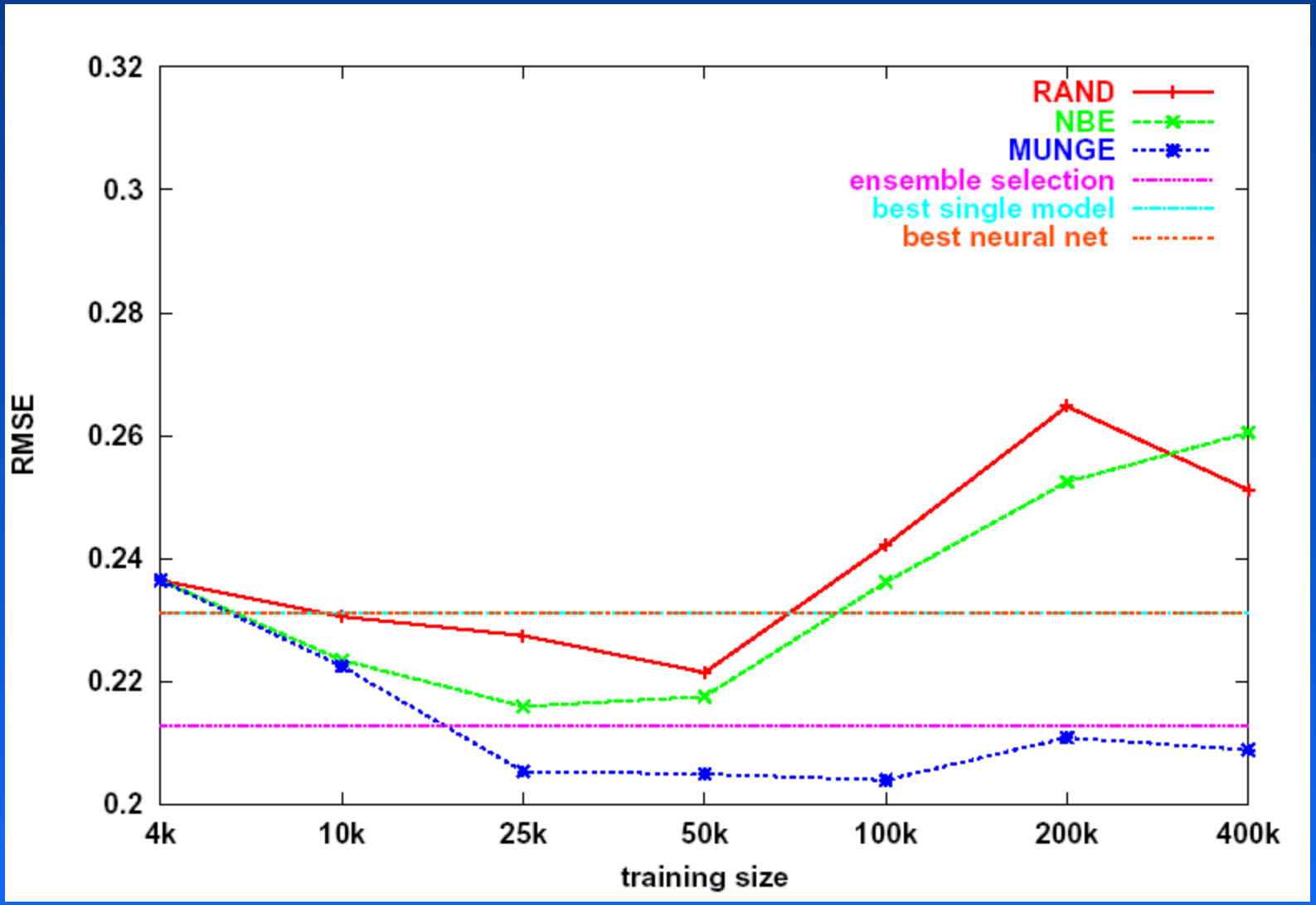
Average Results by Train Size



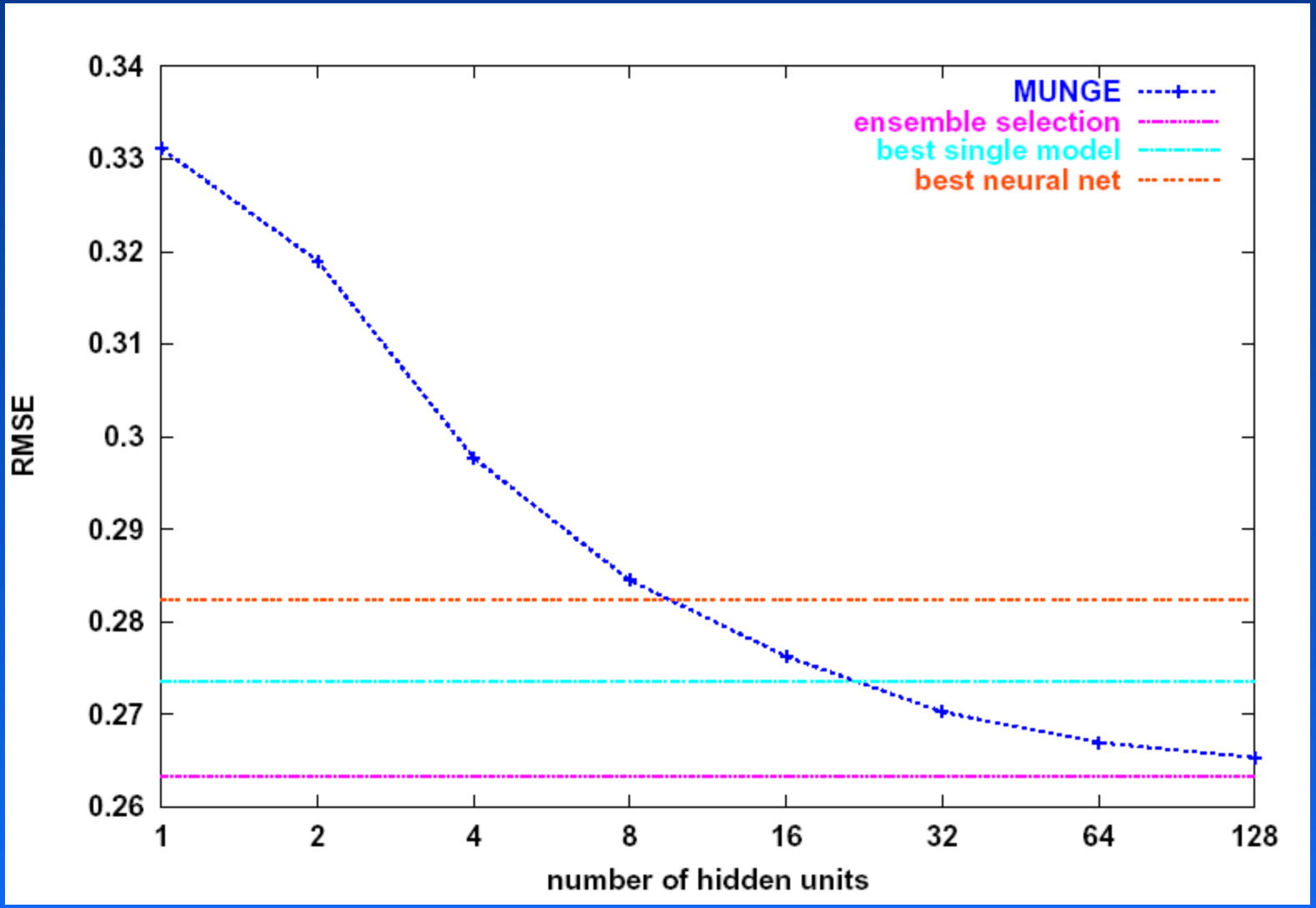
Letter.P1 Results



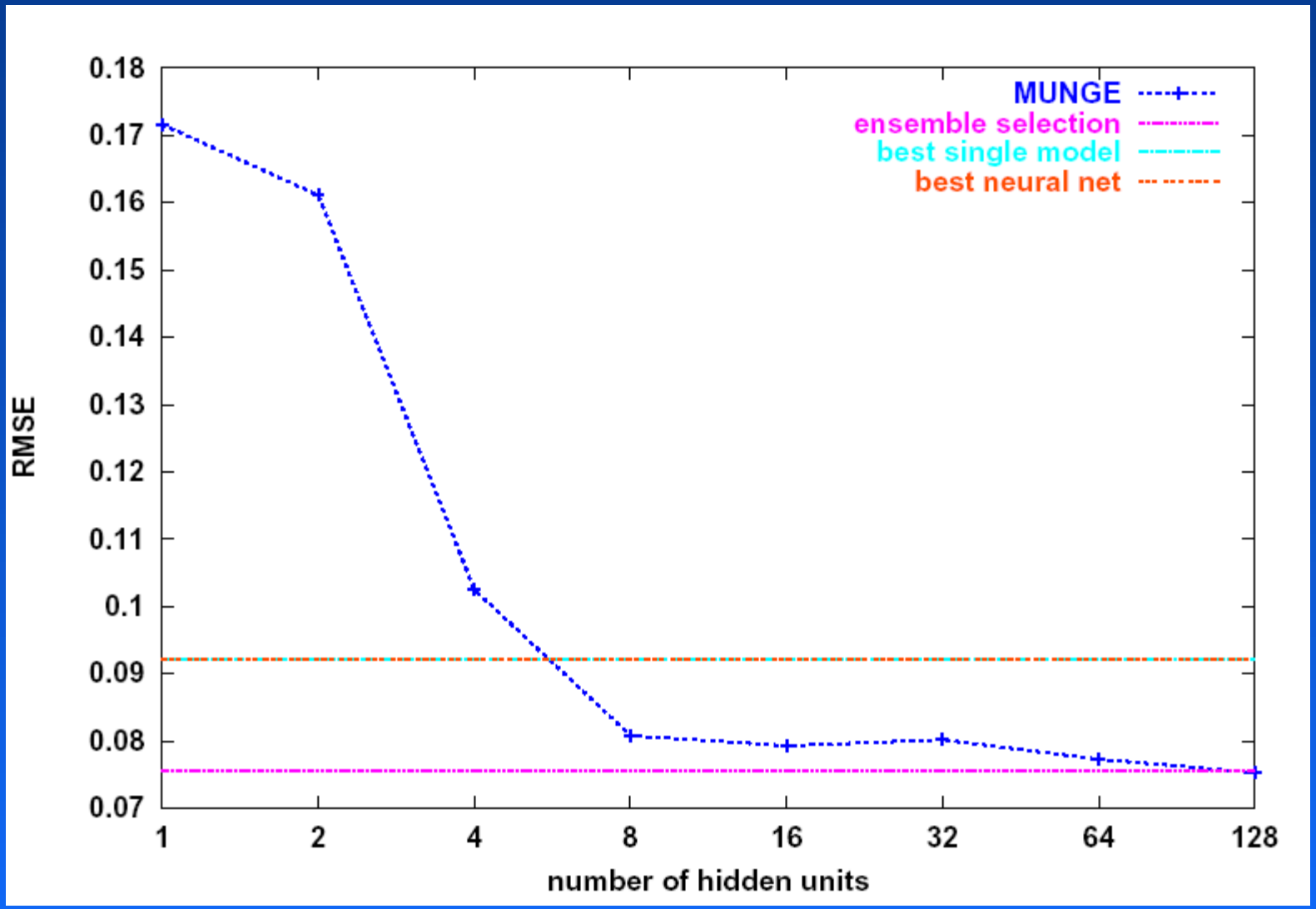
HyperSpectral Results



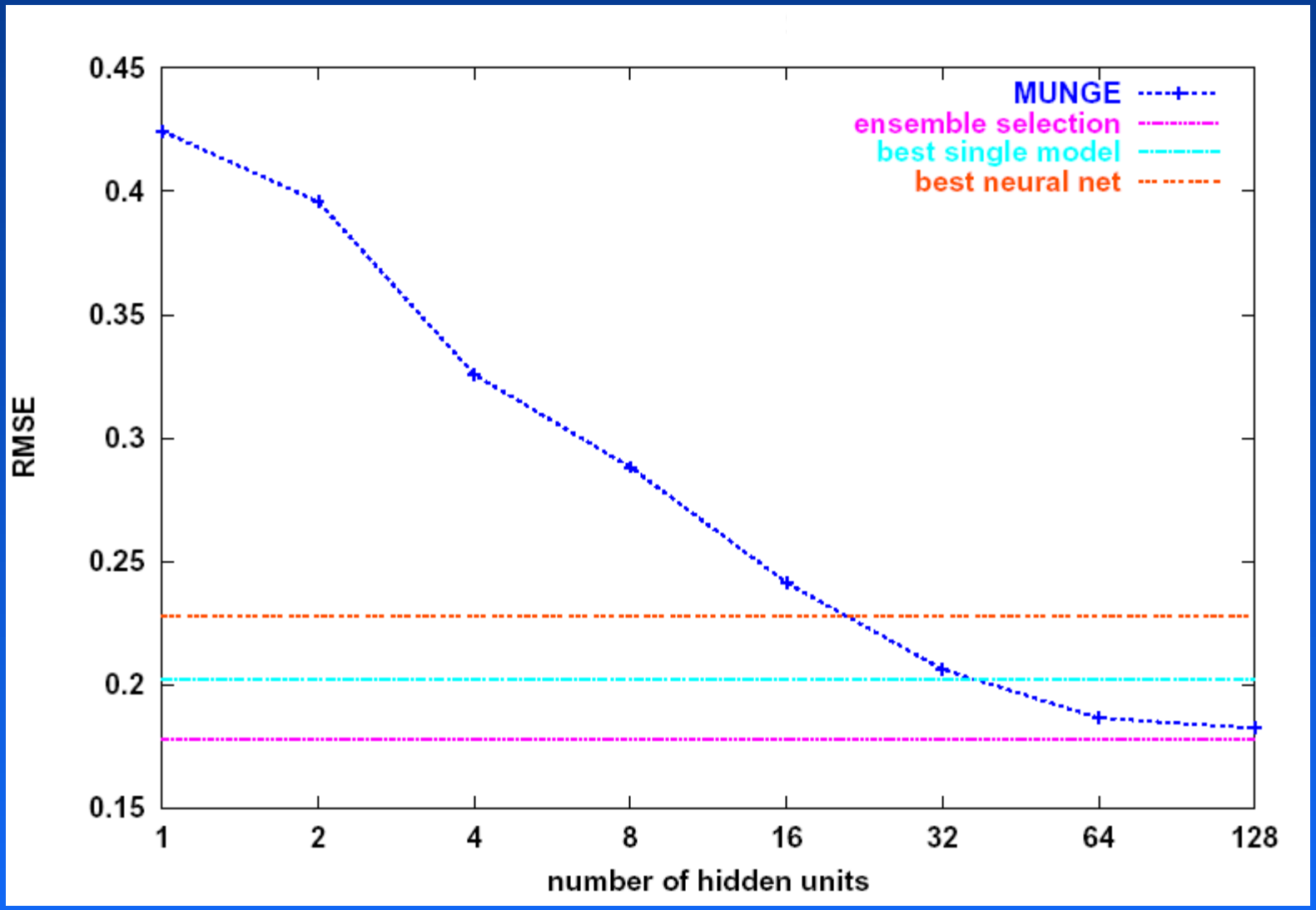
Average Results by HU



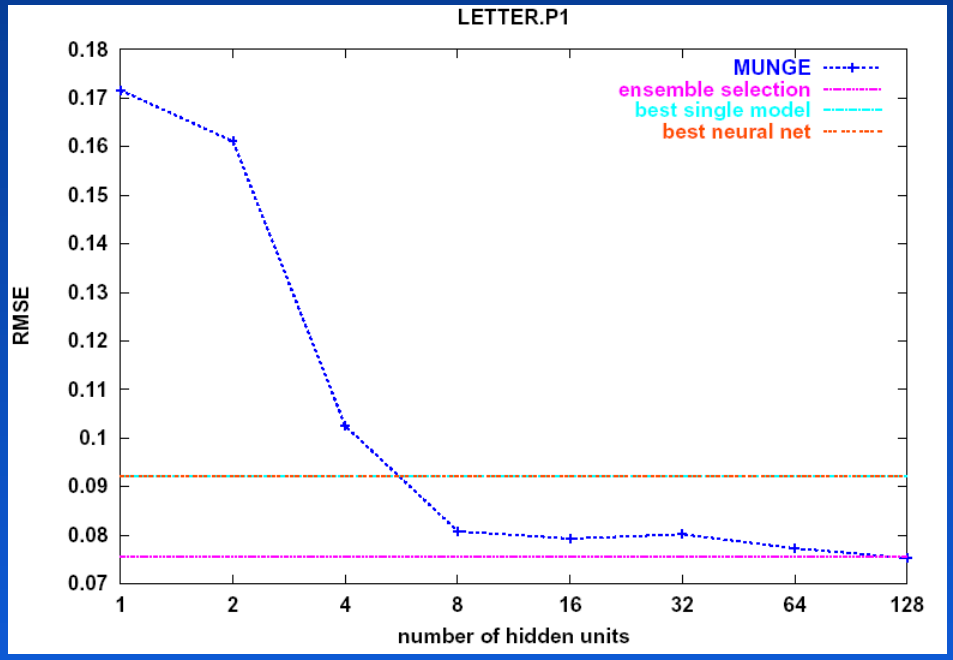
Letter.P1 Results

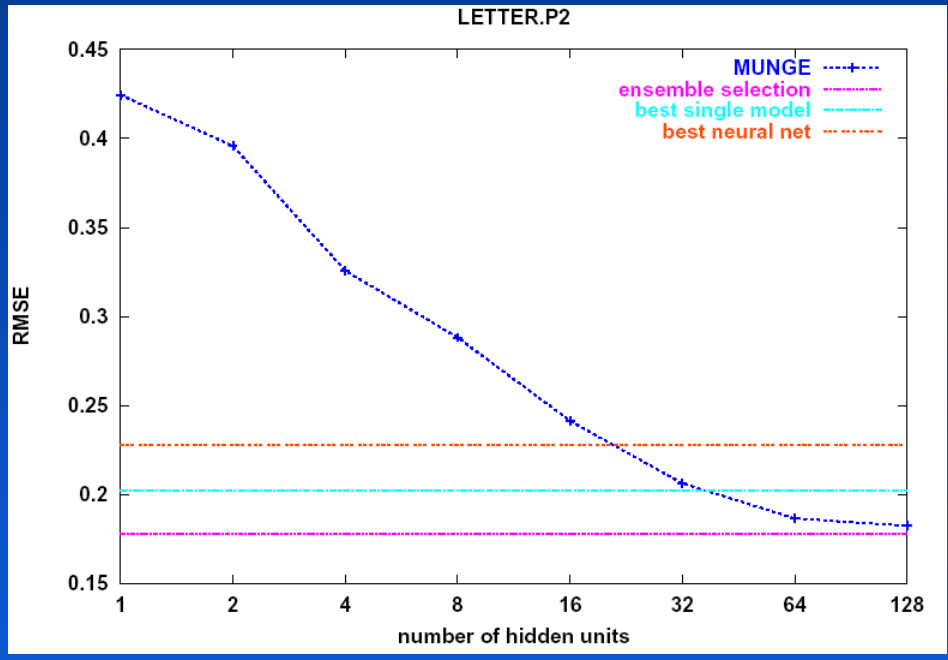


Letter.P2 Results



Letter Results

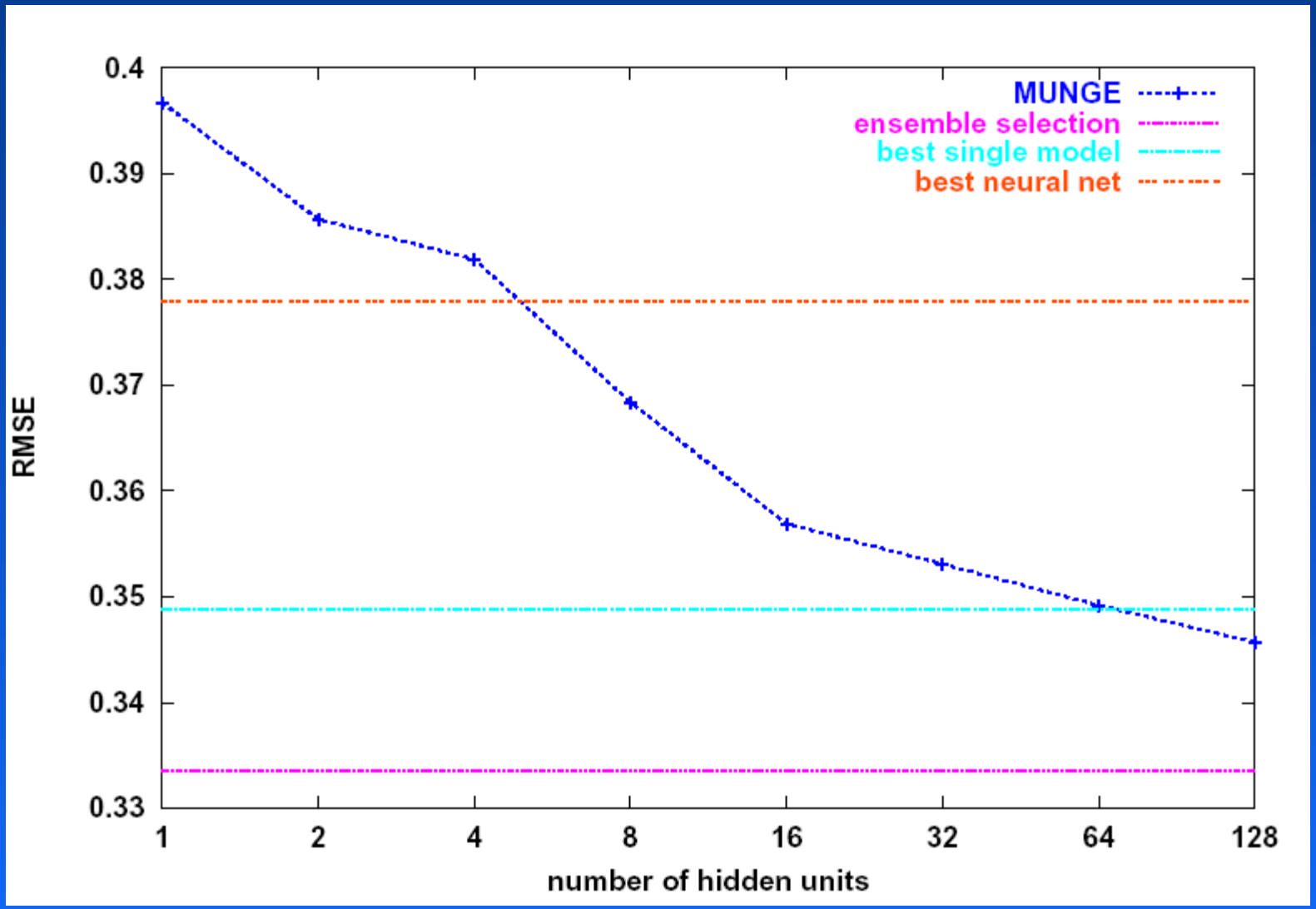




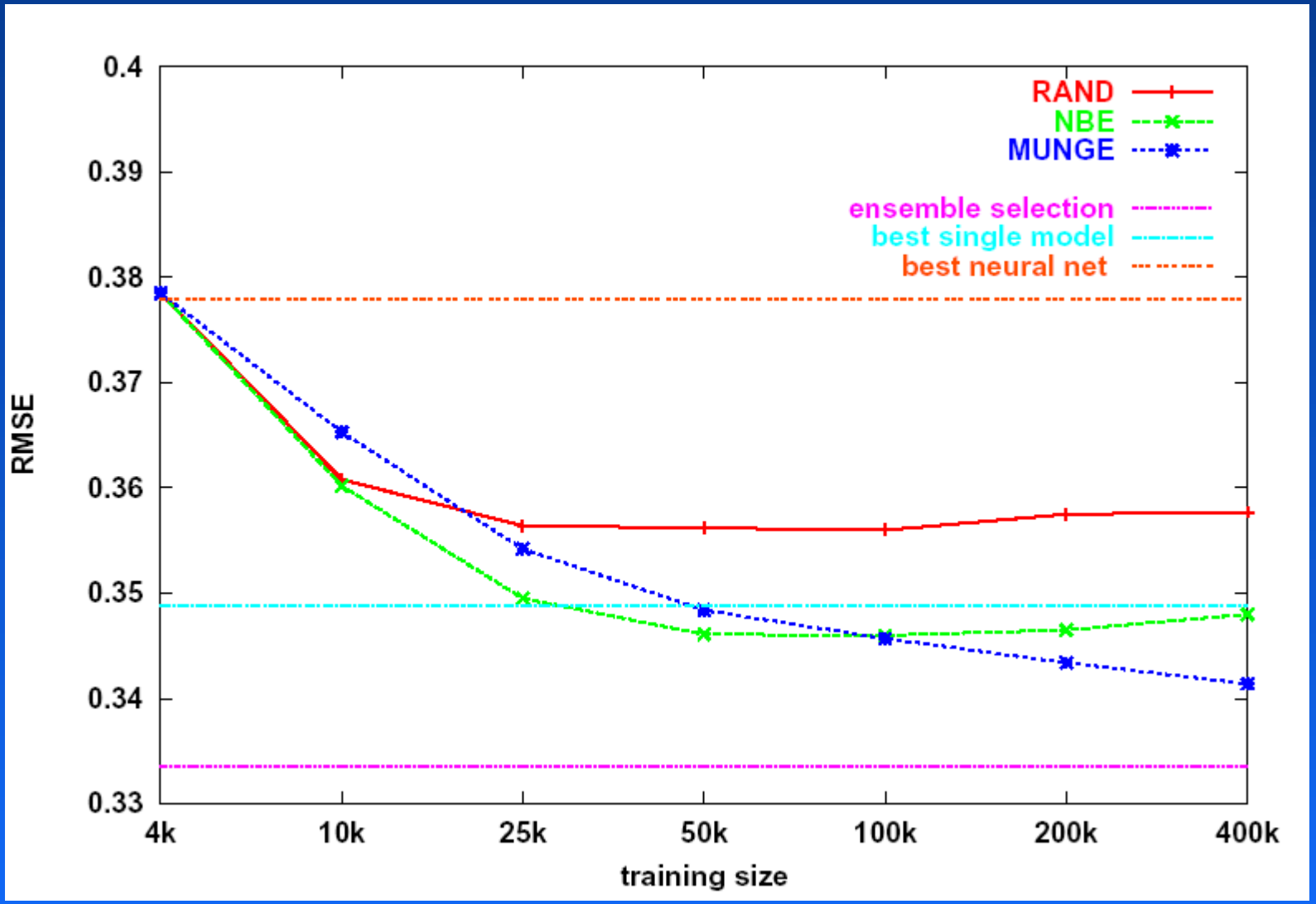
- Letter.p1: Distinguish letter O from the rest
- Letter.p2: Distinguish letters A-M from N-Z

Doesn't Always Work As Well As We'd Like

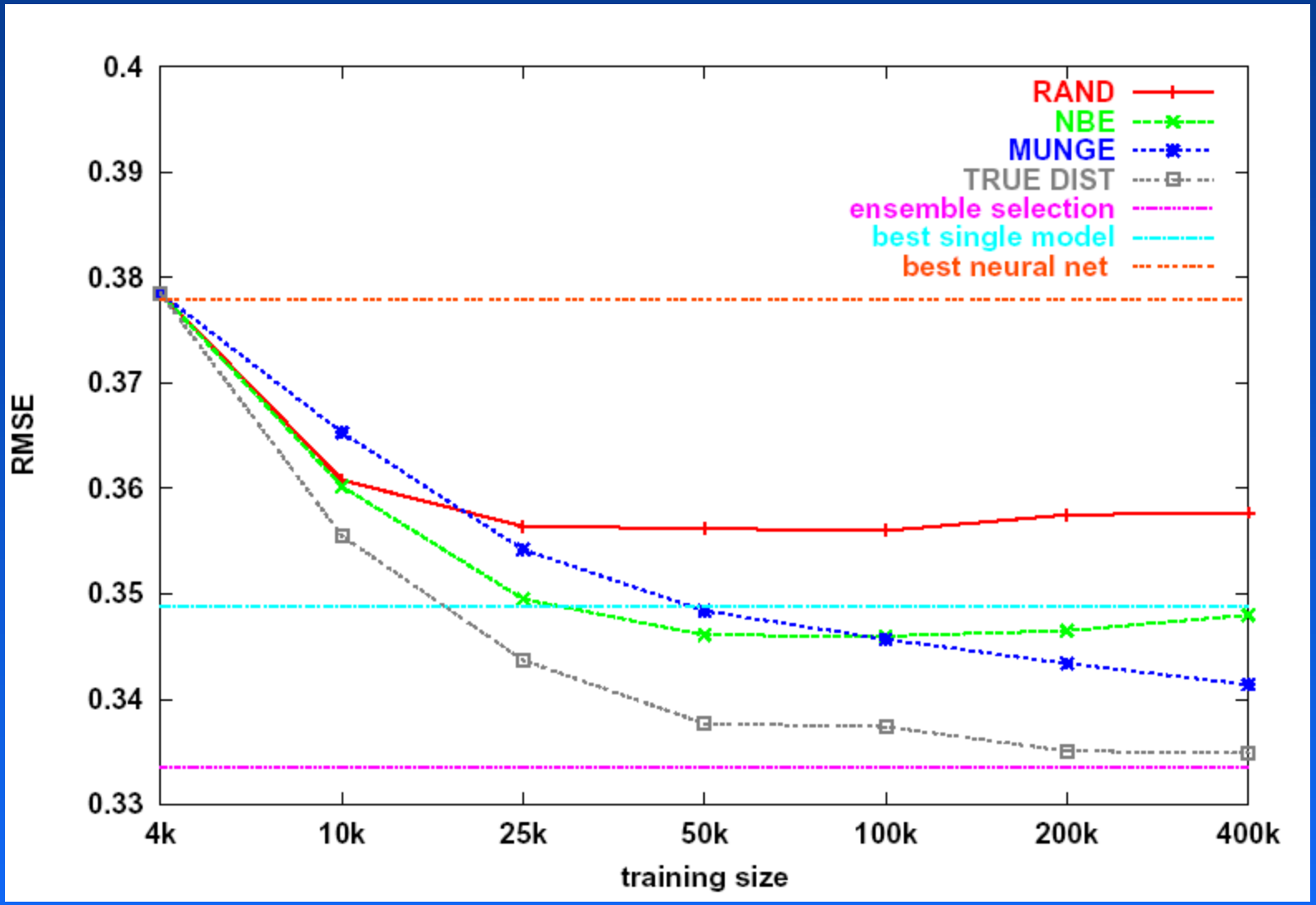
TreeCoverType Results



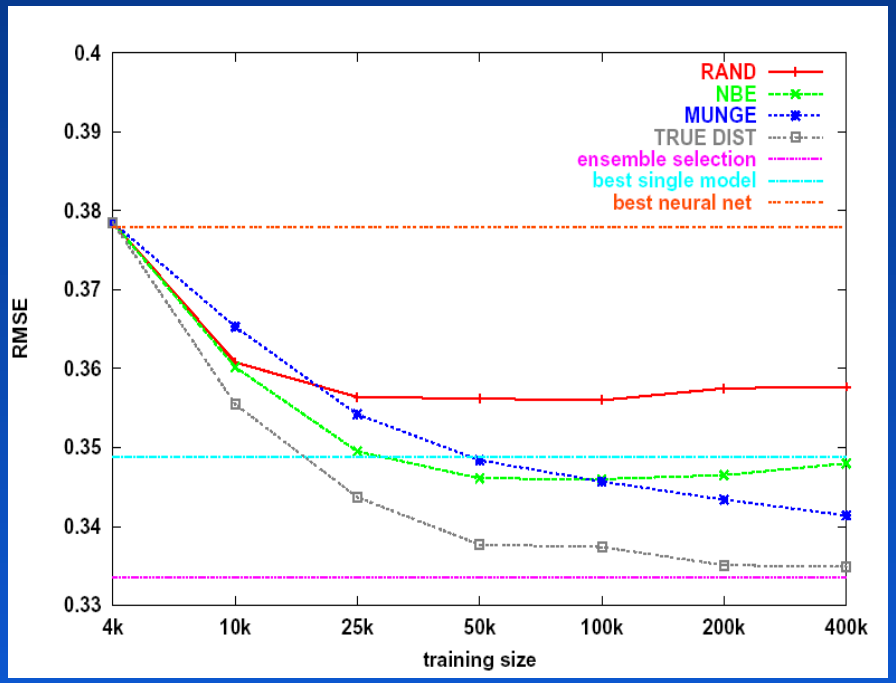
TreeCoverType Results

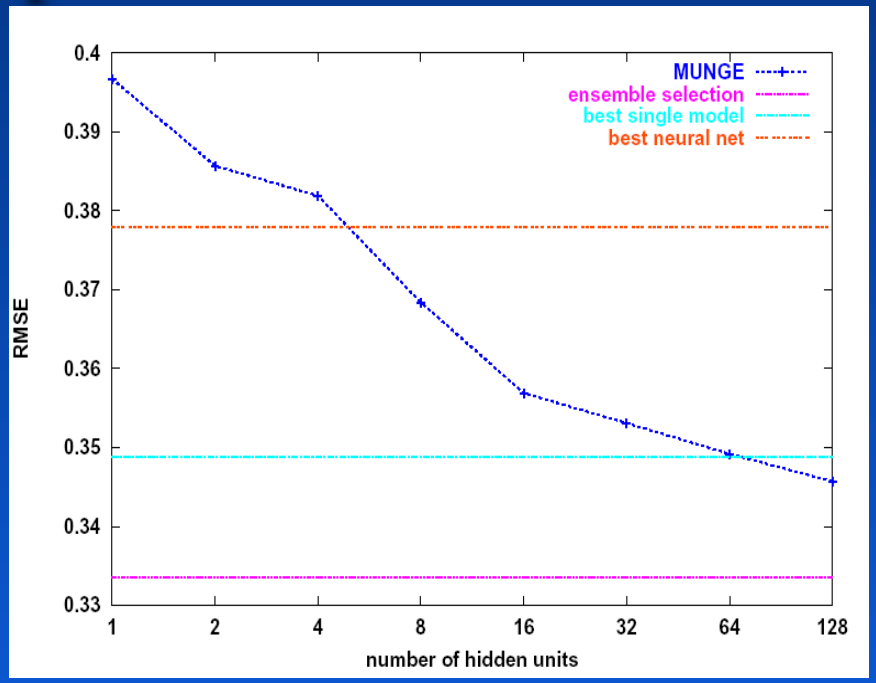


TreeCoverType Results



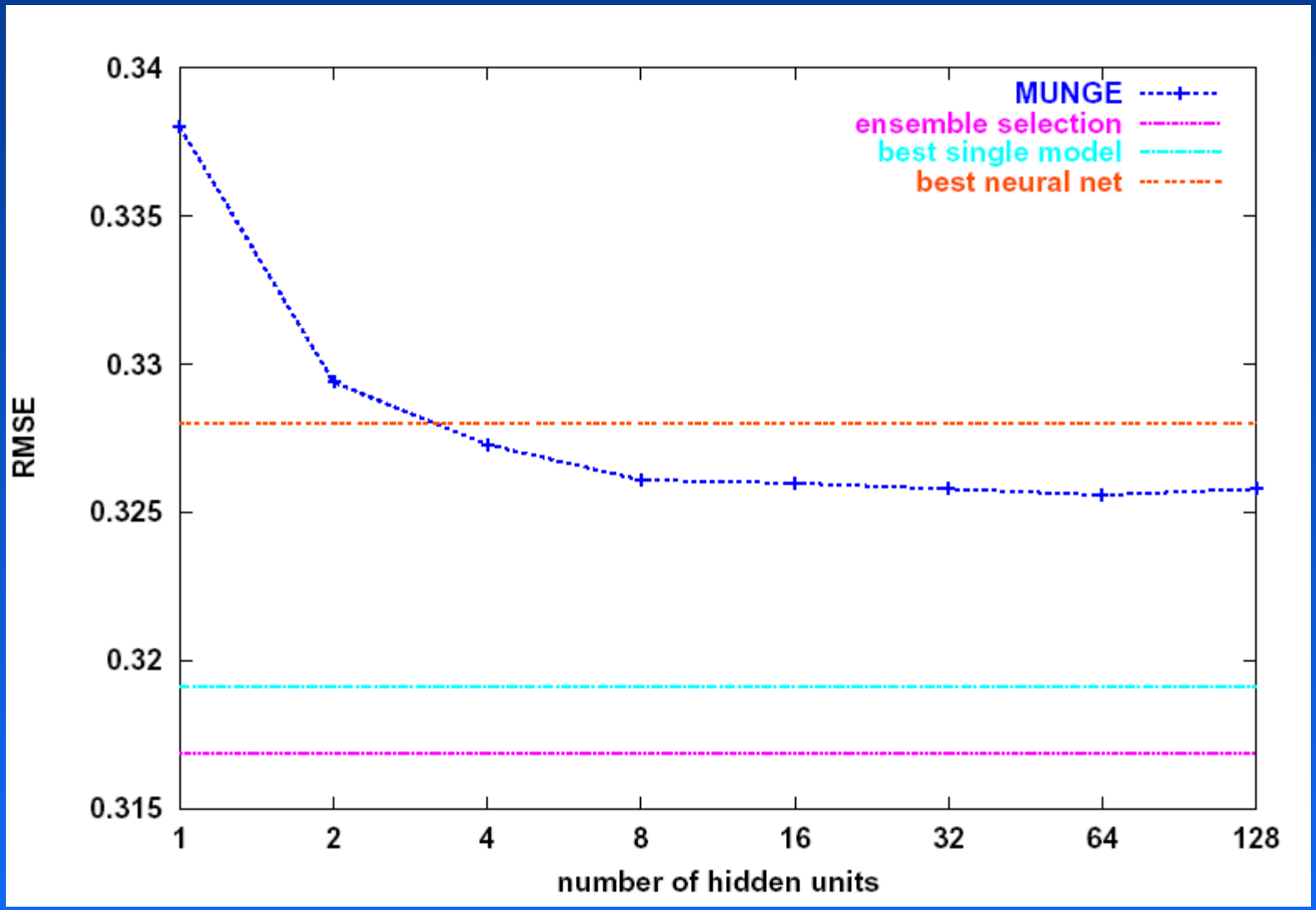
TreeCoverType Results



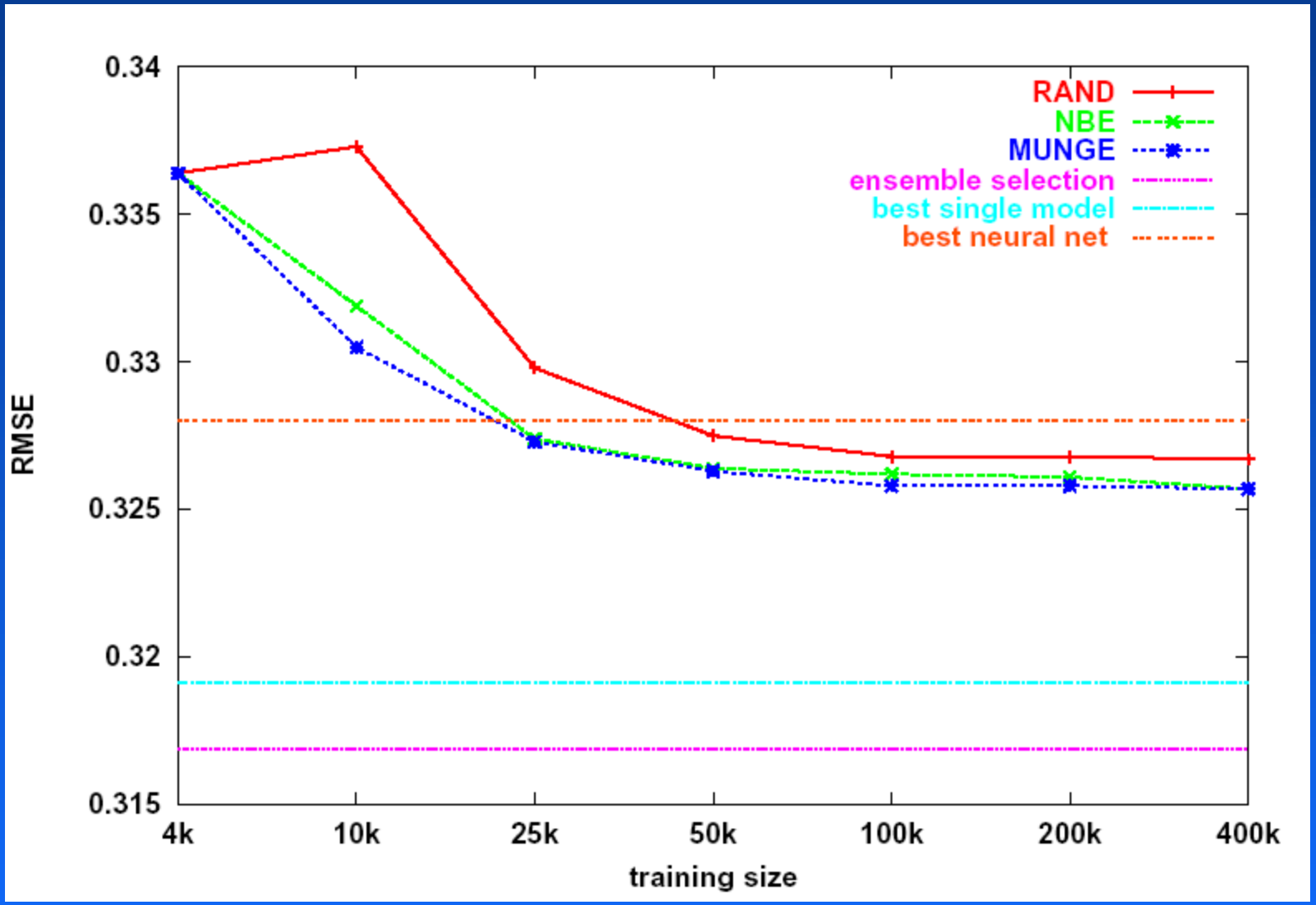


- More hidden units necessary for better mimic model (less compression)
- More synthetic unlabeled data might help
- Performance on TRUE DIST data is excellent --- munging isn't good enough! (this is NOT a problem when unlabeled data is available)

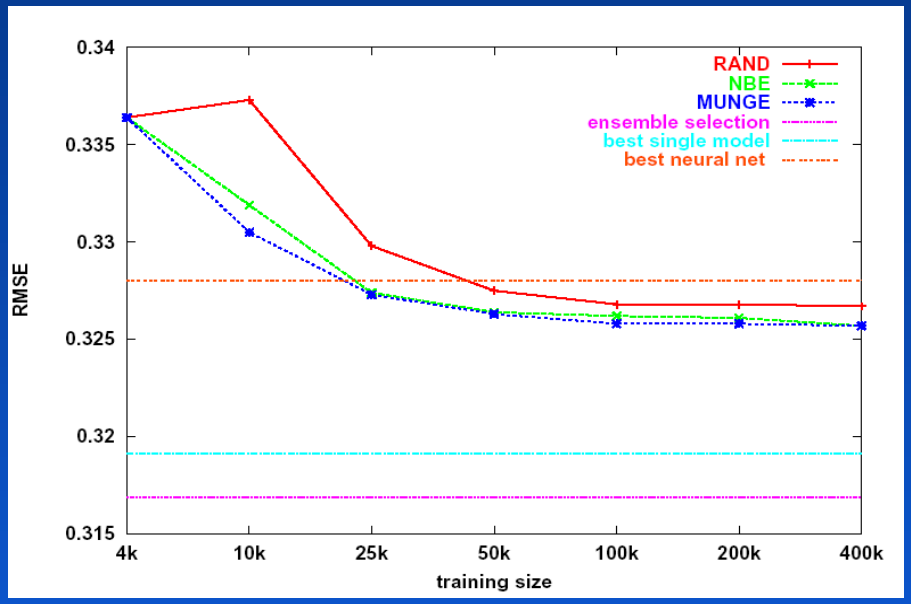
Adult Results

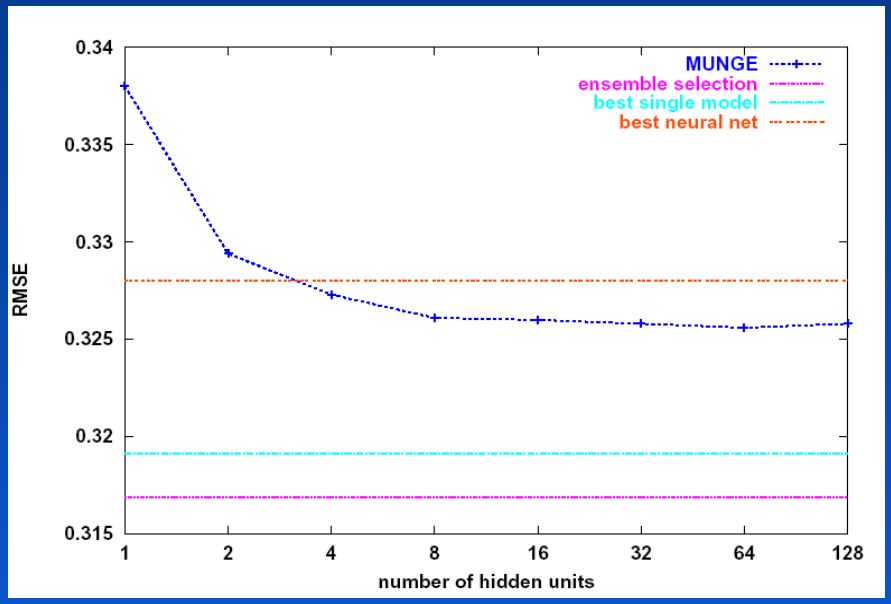


Adult Results



Adult Results

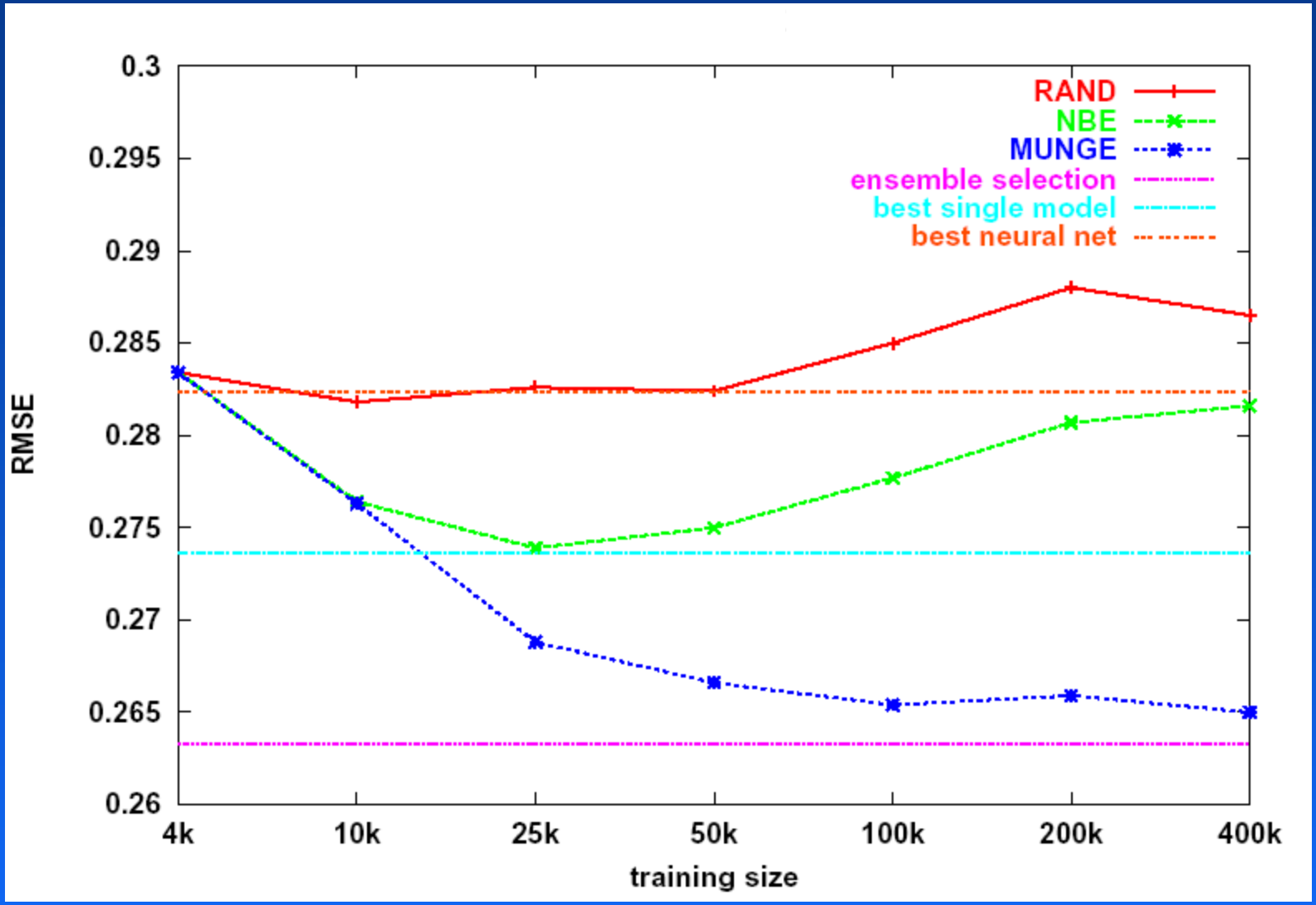




- More Munge data or more hidden units doesn't seem to help much
- Adult has a few high arity nominal attributes that when binarized increase the number of attributes from 14 to 104 sparse binary attributes
- Neural nets may not be well suited for this problem
- Munge may not be effective in generating good synthetic data for adult

Summary of Results

Average Results by Train Size



RMSE Results – 400K, 256 HU

	ENSEMBLE	NN	MIMIC NN	RATIO
ADULT	0.317	0.328	0.325	**0.29**
COVTYPE	0.334	0.378	0.340	**0.84**
HS	0.213	0.231	0.204	**1.47**
LETTER.P1	0.075	0.092	0.075	**1.01**
LETTER.P2	0.178	0.228	0.179	**0.98**
MEDIS	0.278	0.279	0.277	**2.29**
MG	0.287	0.295	0.288	**0.88**
SLAC	0.424	0.428	0.422	**1.69**
AVERAGE	**0.263**	**0.282**	**0.264**	**0.97**

RATIO = (MUNGE – ANN) / (ENSEMBLE – ANN)

Retaining 97% of Accuracy of Target Model,

but How Are We Doing on Compression?

Size of Models (MB)

	ENSEMBLE	NN	MIMIC NN	RATIO
ADULT	1234.72	0.22	0.45	**2744**
COVTYPE	1108.16	0.03	0.23	**4818**
HS	74.37	0.12	0.79	**94**
LETTER.P1	1.23	0.01	0.08	**15**
LETTER.P2	325.80	0.04	0.08	**4073**
MEDIS	5.24	0.14	0.27	**19**
MG	25.75	0.03	0.50	**52**
SLAC	1627.08	0.13	0.25	**6508**
AVERAGE	**550.29**	**0.09**	**0.33**	**2290**

RATIO = ENSEMBLE / MUNGE

Execution Time of Models

Time in seconds to classify 10,000 examples

	ENSEMBLE	NN	MIMIC NN	RATIO
ADULT	8560.61	3.94	7.88	**1086**
COVTYPE	3440.90	1.05	4.46	**772**
HS	1817.17	3.85	12.09	**150**
LETTER.P1	1630.21	0.25	2.59	**629**
LETTER.P2	2651.95	0.74	2.59	**1024**
MEDIS	190.18	2.85	4.78	**40**
MG	1220.04	1.80	6.98	**175**
SLAC	23659.03	2.85	3.60	**6572**
AVERAGE	**5396.27**	**2.17**	**5.62**	**1306**

RATIO = ENSEMBLE / MUNGE

Summary of Compression Results

	NN	ENSEMBLE	MIMIC NN	RATIO
RMSE	0.282	0.263	0.264	**0.97**
Size (Mb)	0.09	550.29	0.33	**2290**
Time (s)	2.17	5396.27	5.62	**1306**

- Neural nets that mimic high performing ensembles
  - on average, capture 97% performance of target model
  - much better than any ANN we could train directly
  - More than 2000 times smaller than target ensemble
  - More than 1000 times faster than target ensemble
- Main challenge: better synthetic data when unlabeled data is not available

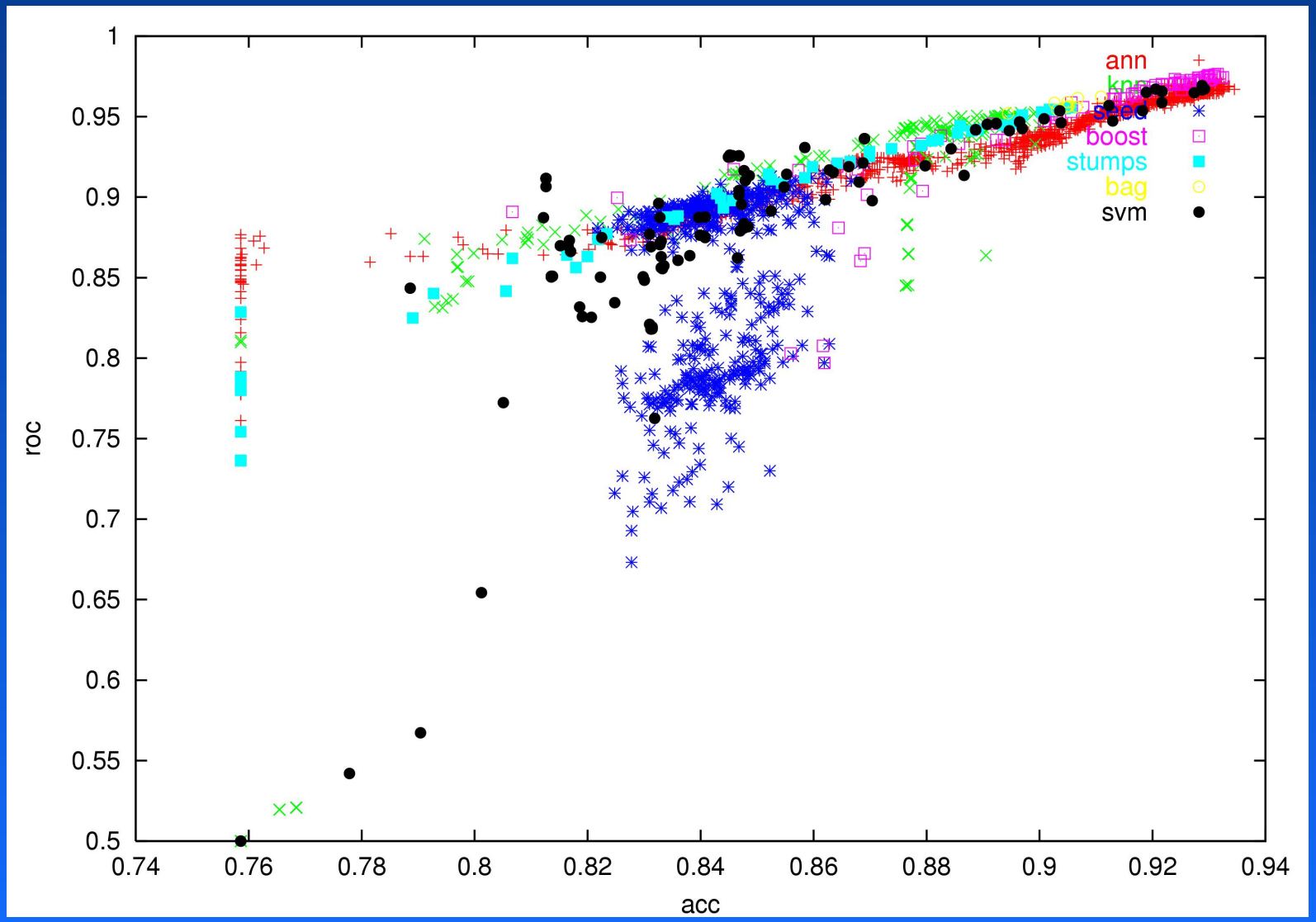
Why Does It Work?

- No extra labeled data required
- World's ugliest neural net training algorithm!!!
- Can make synthetic training set arbitrarily large so simple backprop doesn't overfit
- Mimic NN may provide additional regularization (train ensembles specifically for compression?)
- Clearly large ensemble ≠ complex function
- Ensembles just crude ways to reduce bias and variance in absence of "smarter" learning?

Limitations and Future Work

- Results for squared loss. What about other losses?

The "Funnel" of High Performance



Limitations and Future Work

- Results for squared loss. What about other losses?
- Measure problem complexity by # hidden units?
- When to mimic with other model types?
  - deep nets? neural nets not always best?
- Synthetic data and modeling p(x)
  - munging surprisingly effective --- improve it? theory?
  - other density estimation methods? rejection sampling?
  - what happens as dimensionality increases?
- Active Learning
  - reduce data that must be labeled and mimic train set size?
- Calibrate before or after compression (or both)?

Related Work

- TREPAN [Craven and Shavlik, 1996]
  - Extract tree-structured representations from neural nets
- Reduced-Set SVMs (prune support vectors) [Burgess, 1996]
- CMM (Combine Multiple Models) [Domingos, 1997]
  - Improve accuracy of C4.5 rules without losing intelligibility
  - Train C4.5 to mimic ensemble of C4.5 rules
  - Method for generating extra data specific for C4.5 rules
- Neural Nets Approximator [Zeng and Martinez, 2000]
  - Trained neural net to model ensemble of neural nets
- Pruning adaptive boosting [Margineantu and Dietterich 2000]
  - To compress ensemble, prune away some of the models
- Image Recognition via Boosted Cascades [Viola, 2001]
- DECORATE: Improve model diversity [Melville and Mooney, 2003]

Summary

- Breiman's Constant:
  - High accuracy models often large, expensive, and complex
- Model compression works!
  - 1000 times smaller and faster, with little loss in accuracy
  - Decouples function learning from deployment
- Density estimation/sampling important part of puzzle in domains where large unlabeled data not available

The “Big” Picture

“Model Compression let’s you have your cake and deploy it too”

Thank You.

Questions?