Masters Project

Spatial Supervised Learning

Dan Vega

June 2005

In Acroread, CTRL-L switches
between full screen and window mode.
Or run xpdf -fullscreen masters-presentation.pdf

1 – Introduction

- Traditionally, machine learning algorithms learned functions $y = f(x)$ where
  - training examples are of the form (x_i, y_i)
  - x_i is a fixed length feature vector of feature values
  - y_i is a corresponding class label
- Over the past few years people have begun studying *structural supervised learning problems* where
  - x_i is a graph consisting of nodes and arcs
  - Each node s denoted as $x_{i,s}$ is described by a vector of features
  - Each arc defines a relationship between two nodes
  - y_i denotes the labels assigned to the graph x_i
  - Each node s also has a corresponding class label $y_{i,s}$
  - Each arc defines a relationship between two nodes
- The learning problem is to learn a function F that takes a new input graph x_i and outputs a new label assignment $y_i = F(x_i)$
- The simplest example of this is *sequential supervised learning* where each x_i is a simple sequence of objects (e.g. words in a sentence), and each y_i is a sequence of labels (e.g. parts of speech)
- In sequential supervised learning, many complex methods have been studied, but one of the simplest, recurrent sliding windows, is very fast to train and has gives very good performance

2 – Sequential Supervised Learning and RSW

2.1 – Simple Sliding Windows

- *Simple Sliding Windows* slides a window over the input context, which is made up of the input features of the sequential dataset
- Sliding a window across the sequential data constructs a new dataset, with the sequential nature of the data captured in each instance
  - Left Input Context (LIC) is the amount of input context to include from the *left* of x_i
  - Right Input Context (RIC) is the amount of input context to include from the *right* of x_i

Simple Sliding Windows

(X,Y)	$x_1\ x_2\ x_3\ x_4\ x_5\ x_6$	$y_1\ y_2\ y_3\ y_4\ y_5\ y_6$
w_1	$-\ -\ -\ x_1\ x_2\ x_3\ x_4$	y_1
w_2	$-\ -\ x_1\ x_2\ x_3\ x_4\ x_5$	y_2
w_3	$-\ x_1\ x_2\ x_3\ x_4\ x_5\ x_6$	y_3
w_4	$x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ -$	y_4
w_5	$x_2\ x_3\ x_4\ x_5\ x_6\ -\ -$	y_5
w_6	$x_3\ x_4\ x_5\ x_6\ -\ -\ -$	y_6

Table 1: Sliding window focused on x_i with $LIC = RIC = 3$

2.2 – Recurrent Sliding Windows

- Recurrent Sliding Windows, adds a sliding window over output context to the input context provided by simple sliding windows
  - Left Output Context (LOC) is the amount of output context to include from the left of the focus
  - Right Output Context (ROC) is the amount of output context to include from the right of the focus
  - ROC and LOC are mutually exclusive
- Recurrence can be either beneficial or detrimental
  - Strength of the relationship between the $y_{i,s}$ labels
  - The classifier's ability to learn that relationship,
  - The quality of the $\hat{y}_{i,s'}$ output by the classifier

Recurrent Sliding Windows (RSW)

(X,Y)	$x_1\ x_2\ x_3\ x_4\ x_5\ x_6$	$y_1\ y_2\ y_3\ y_4\ y_5\ y_6$
w_1	$-\ -\ -\ x_1\ x_2\ x_3\ x_4\ -\ -$	y_1
w_2	$-\ -\ x_1\ x_2\ x_3\ x_4\ x_5\ -\ y_1$	y_2
w_3	$-\ x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ y_1\ y_2$	y_3
w_4	$x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ -\ y_2\ y_3$	y_4
w_5	$x_2\ x_3\ x_4\ x_5\ x_6\ -\ -\ y_3\ y_4$	y_5
w_6	$x_3\ x_4\ x_5\ x_6\ -\ -\ -\ y_4\ y_5$	y_6

Table 2: Recurrent Sliding Windows focused on x_i with $LIC = RIC = 3$ and $LOC = 2$

2.3 – Structural Supervised Learning in 2-D

- Another instance of a structural supervised learning is when
  - each x_i is a 2-D rectangular mesh
  - in particular suppose x_i is an image
  - each node $x_{i,s}$ is a pixel with links to its north, south, east, and west neighbors
- This type of problem arises in remote sensing and ecology
  - in ecology: landcover classification (grassland, forest, agriculture, urban, etc.)
  - in agriculture: predicting what crop is growing at each pixel location
  - features typically include passively and actively-collected data
    - Passive: reflected sunlight (in different frequency bands)
    - Active: reflected microwaves, synthetic aperture radar readings
- The Feltwell dataset is a particular example of this type of problem
- Feltwell is discussed in detail on slide 11
- How do we extend Recurrent Sliding Windows to 2-D meshes?

3 – Extending the RSW Methodology

- Extending the RSW methodology to 2-D meshes is one of the goals of this project
- Extending the idea of input context is straightforward
  - LIC and RIC become *Input Context (IC)*
  - IC is the length in pixels of one side of the context square
  - this context square gets an *input mask* applied to it, and creates an x_i
  - the center of the IC window is the focus pixel, s
- Extending the idea of output context is a little more complicated
  - LOC and ROC become *Output Context (OC)*
  - OC is the length in pixels of one side of the context square
  - this context square gets an *output mask* applied to it, and adds more $\hat{y}_{i,s'}$ values as features in x_i
  - the center of the OC window is the focus pixel, s, and the corresponding $y_{i,s}$ is its label
- Scan-order in 1-D is either left-to-right, or right-to-left
  - In 2-D the scan-order is *raster order*
  - This is the sequence of focus pixels over which sliding windows pass

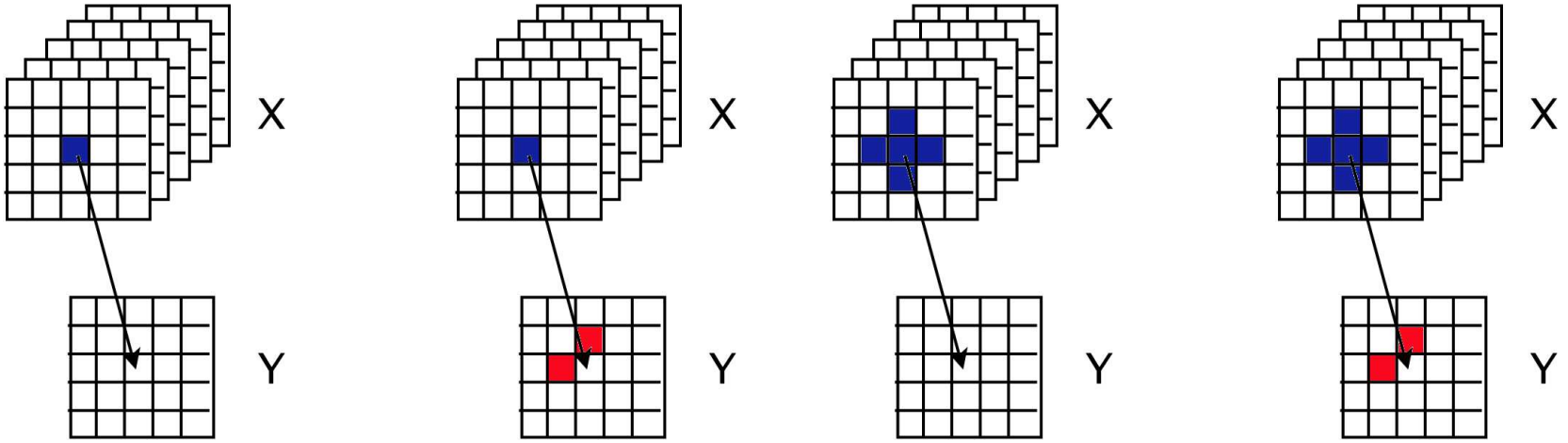
3.1 – IC and OC defined

$$P(\hat{y}_m|x_m) \quad (1)$$

$$P(\hat{y}_m|x_m, \hat{y}_{m-1}, \hat{y}_{m-w}) \quad (2)$$

$$P(\hat{y}_m|x_m, x_{m-1}, x_{m+1}, x_{m+w}, x_{m-w}) \quad (3)$$

$$P(\hat{y}_m|x_m, x_{m-1}, x_{m+1}, x_{m+w}, x_{m-w}, \hat{y}_{m-1}, \hat{y}_{m-w}) \quad (4)$$



Traditional (1,1): eqn (1) OC (1,3): eqn (2) IC (3,1): eqn (3) IC and OC (3,3): eqn (4)

Table 3: Illustration of (input context, output context) equations

3.2 – IC and OC window masks

IC or OC value	IC mask	OC mask	#IC features	#OC features
1	**1**	0	15	0
3	0**1**0 **111** 0**1**0	0**1**0 **1**00 000	15×5	2
5	0 0 **1** 0 0 0 **1 1 1** 0 **1 1 1 1 1** 0 **1 1 1** 0 0 0 **1** 0 0	0 0 **1** 0 0 0 **1 1 1** 0 **1 1** 0 0 0 0 0 0 0 0 0 0 0 0 0	15×13	6

Table 4: Selected Input and Output Context mask examples. The pixels which pass through the mask denoted as '**1**', withheld pixels are denoted as '0'.

- Shape of mask (circular) chosen to make IC and OC step sizes small.
- Increasing IC increases the number of features proportional to IC^2.

4 – A Study of RSW and Spatial Ensembles on the Feltwell Dataset

- Data is from 15 bands; each node $x_{i,s}$ consists of a feature vector with 15 features
  - 6 from Airborne Thematic Mapper (ATM) - together a *Spectral signature* [6]
  - 9 from Synthetic Aperture Radar (SAR) - gives coarseness
- Each corresponding $y_{i,s}$ is a type of land cover
- Both the training and test sets consist of 43750 instances.
- 36% total unlabeled pixels, training set 30% and test set 43%

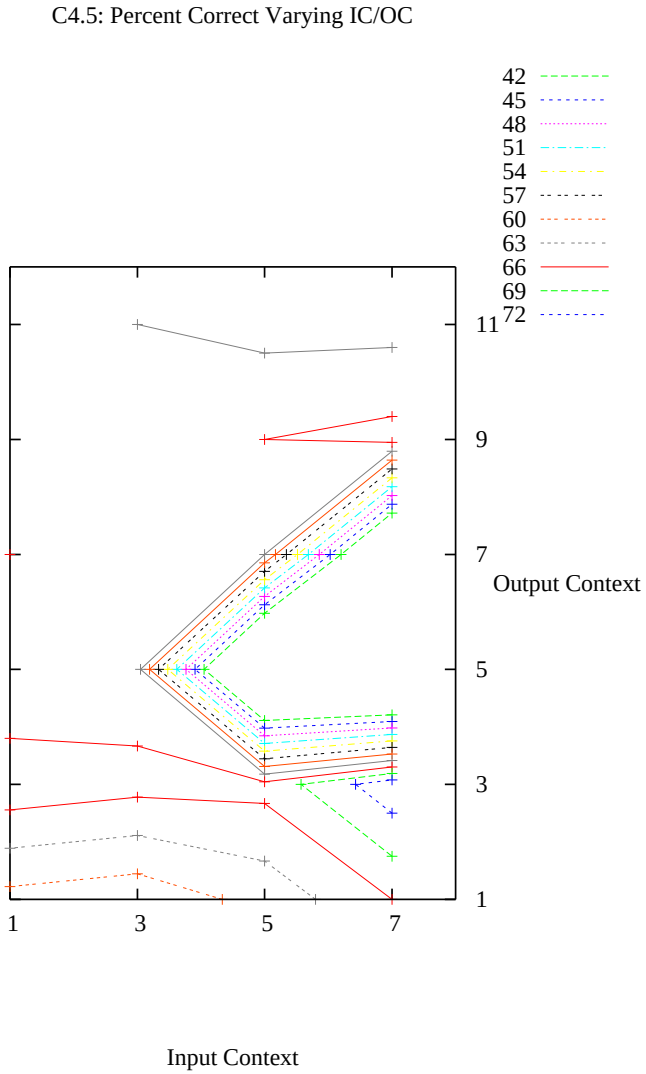
Class Label	Ground Cover Type	Training Set Instances	Testing Set Instances
?	Unknown	13249	18594
1	sugar beets	6959	8082
2	stubble	8702	4626
3	bare soil	3986	1266
4	potatoes	6842	5339
5	carrots	4012	5843

Table 5: Ground Truth labels for the Feltwell dataset [2] [3]

5 – Results and Discussion

- Contour Plots
- Data Tables (reorganized from paper)
  - Now, same orientation as contour plot
- IC values range from 1 to 7
- OC values range from 1 to 11
- Note
  - Settings for best accuracy achieved in each table
  - Transitions due to ensemble classifiers
  - Adaboost for 10, 20, and 30 iterations
  - Bagging for 10, and 20 iterations
  - Bottom left-hand corner is “bare” classifier

5.1 – C4.5



- Max at (7,3) shown in **red**
- Min neighbor at (7,5)

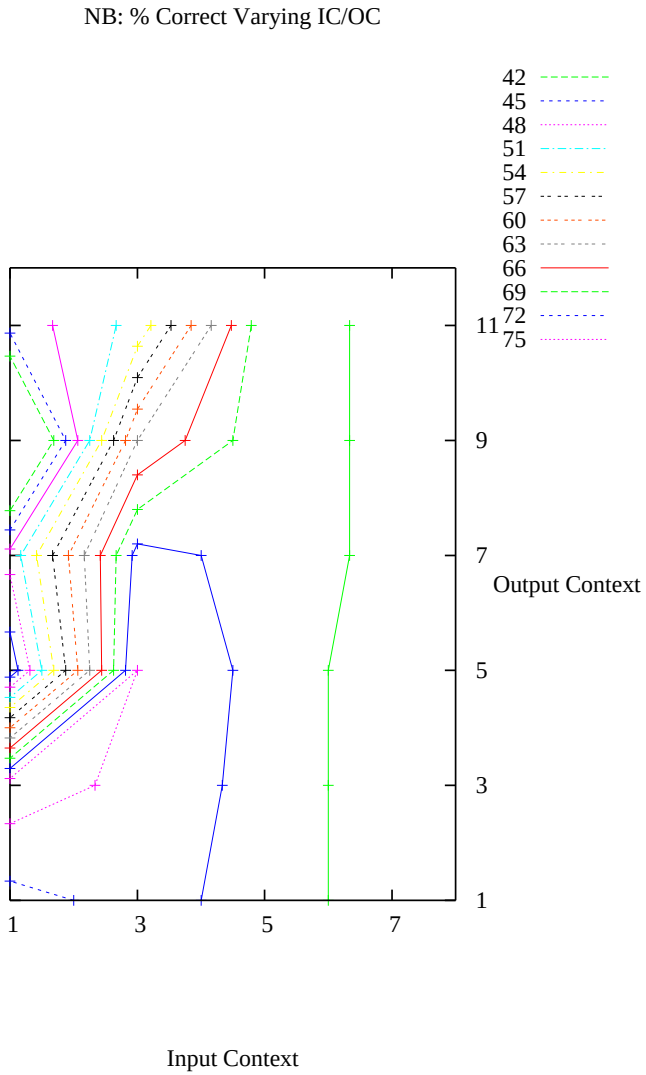
65	63	62	62	11
64	64	66	67	9
66	63	63	28	7
63	64	22	21	5
68	67	67	**74**	3
59	58	61	66	1
1	3	5	7	

OC

IC

Figure 1: Contour plot and data of the Accuracy of C4.5 at Various IC and OC Sizes

5.2 – Naive Bayes



- Max at (1,3) shown in **red**
- Min at (1,9)
- Didn't expect Naive Bayes to perform well with OC

46	52	71	68	11
31	63	71	68	9
49	73	71	68	7
43	75	71	67	5
77	74	71	67	3
71	73	71	67	1
1	3	5	7	

OC

IC

Figure 2: Accuracy of naive Bayes at Various IC and OC sizes

5.3 – Bagging C4.5

65	63	62	62	11
64	64	66	67	9
66	63	63	28	7
63	64	22	21	5
68	67	67	**74**	3
59	58	61	66	1
1	3	5	7	

OC

IC

Table 6: C4.5

- Bagging improves accuracy with C4.5
- Max moves from (7,3) to (1,3)!
- Not much change between 10 and 20 bags
- Contour plots on next slide

68	67	70	67	11
67	66	69	67	9
68	68	67	45	7
66	67	41	39	5
78	74	74	75	3
65	68	68	74	1
1	3	5	7	

OC

IC

Table 7: C4.5 Bagging (10)

68	66	69	67	11
66	65	69	67	9
68	68	68	44	7
66	67	40	36	5
78	77	73	76	3
64	67	66	74	1
1	3	5	7	

OC

IC

Table 8: C4.5 Bagging (20)

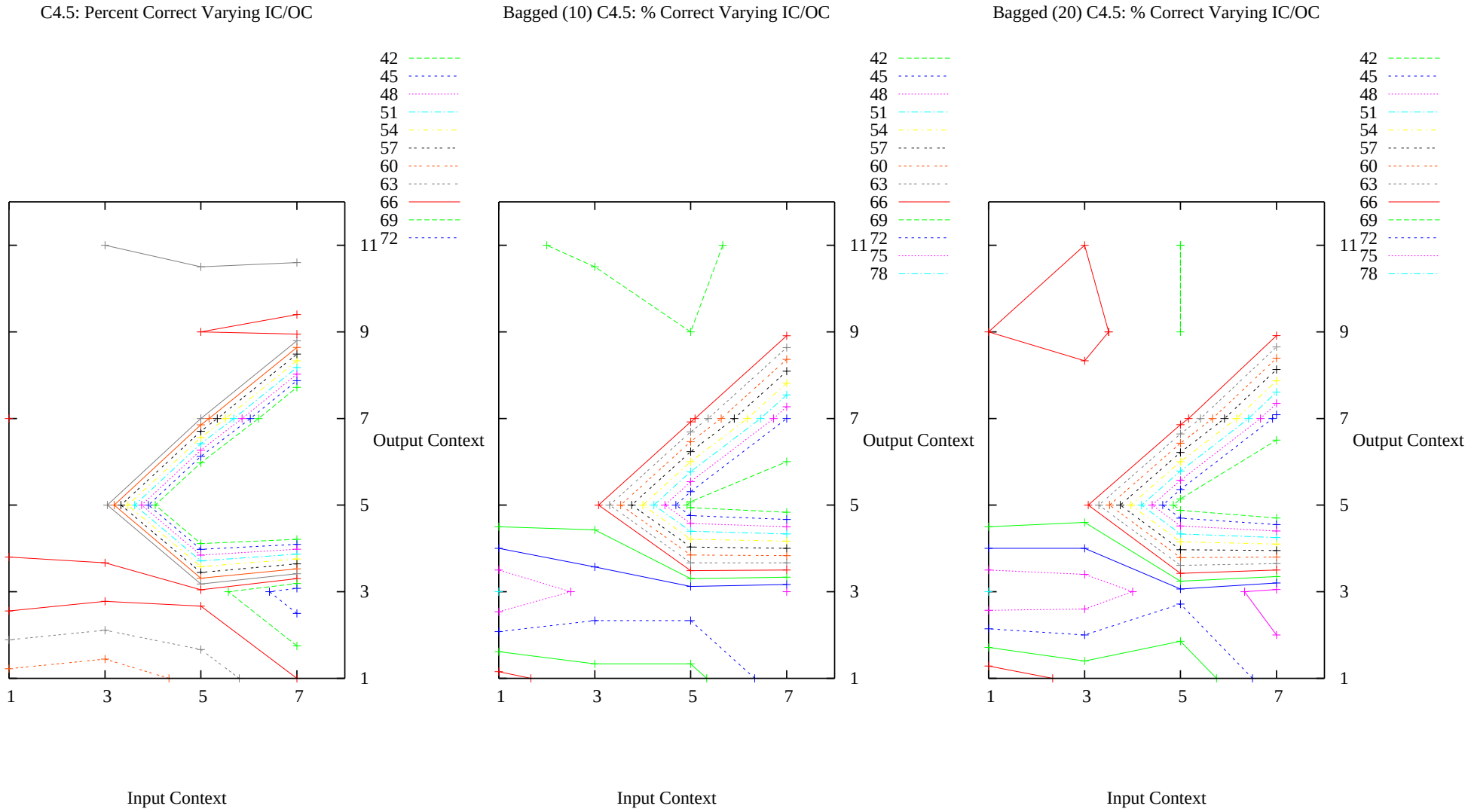


Figure 3: Effect of bagging using C4.5

5.4 – Bagging Naive Bayes

46	52	71	68	11
31	63	71	68	9
49	73	71	68	7
43	75	71	67	5
77	74	71	67	3
71	73	71	67	1
1	3	5	7	

OC

IC

Table 9: Naive Bayes

- Max value is (1,3) for each set of experiments
- Stable algorithms are not helped as much by Bagging
- The lack of improvement is visible in the contour plots
- IC=1 at the larger OC levels performance degrades

30	51	72	69	11
31	63	72	68	9
32	73	72	68	7
43	75	71	68	5
77	74	71	68	3
71	73	71	68	1
1	3	5	7	

OC

IC

Table 10: Naive Bayes Bagging (10)

30	52	72	69	11
31	63	72	68	9
32	73	72	68	7
43	75	71	68	5
77	74	71	68	3
71	73	71	68	1
1	3	5	7	

OC

IC

Table 11: Naive Bayes Bagging (20)

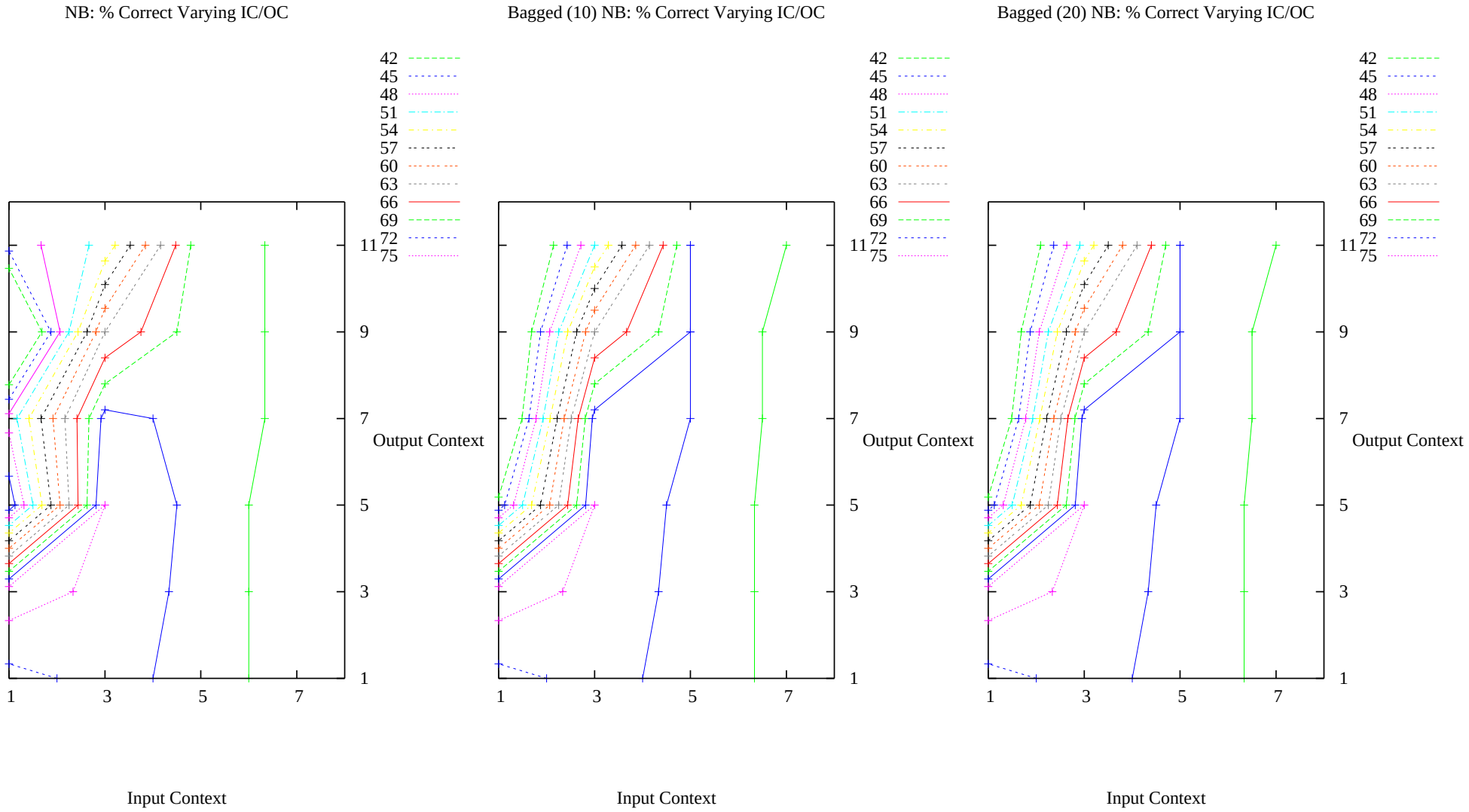


Figure 4: Effect of bagging using naive Bayes

5.5 – Boosting C4.5

65	63	62	62	11
64	64	66	67	9
66	63	63	28	7
63	64	22	21	5
68	67	67	**74**	3
59	58	61	66	1
1	3	5	7	

OC

IC

Table 12: C4.5

66	67	67	68	11
67	65	66	62	9
68	64	68	43	7
65	61	44	38	5
75	76	**78**	77	3
65	74	77	77	1
1	3	5	7	

OC

IC

Table 13: C4.5 Adaboost (10)

66	67	66	67	11
67	65	65	64	9
67	62	68	39	7
66	57	44	37	5
75	**82**	80	81	3
65	75	75	76	1
1	3	5	7	

OC

IC

Table 14: C4.5 Adaboost (20)

66	66	65	67	11
67	66	65	63	9
67	60	68	37	7
66	57	43	38	5
75	**81**	**81**	78	3
66	76	76	76	1
1	3	5	7	

OC

IC

Table 15: C4.5 Adaboost (30)

Spatial Supervised Learning: Results and Discussion

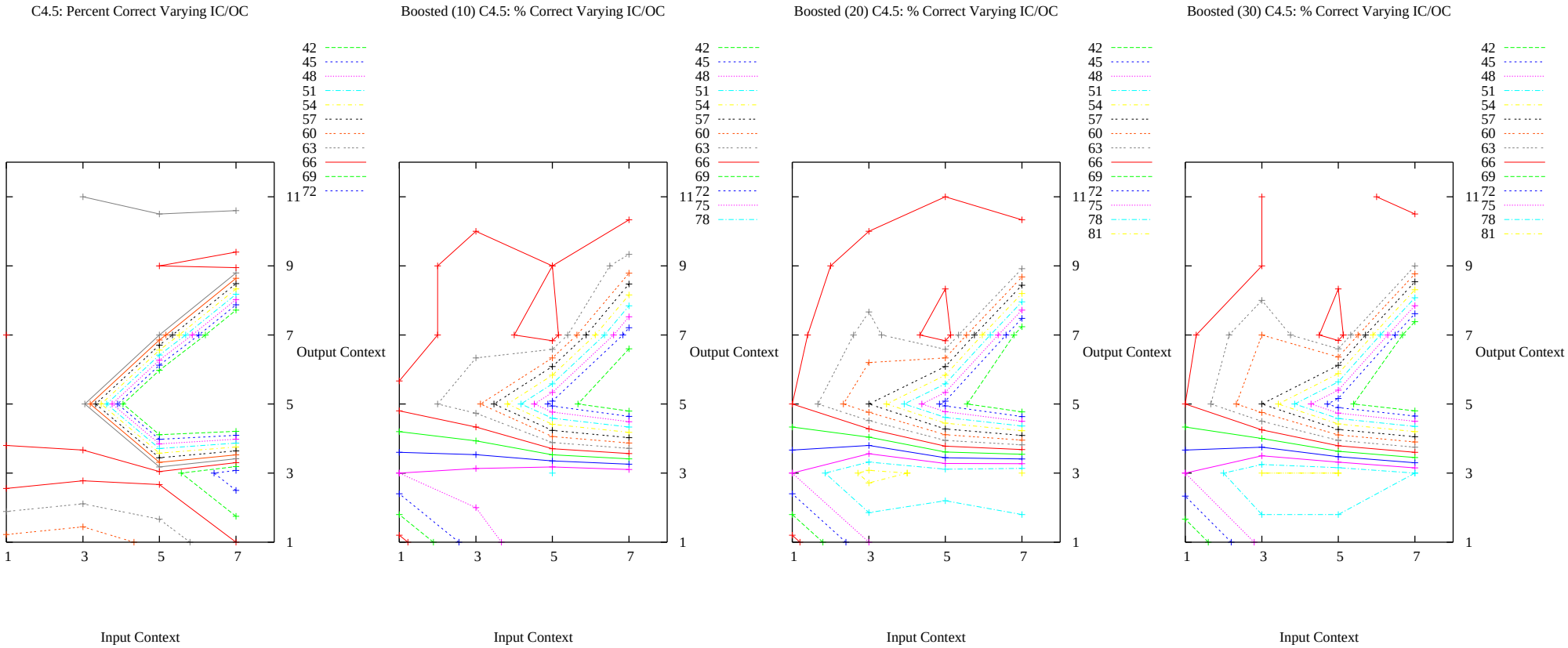


Figure 5: Effect of Adaboost using C4.5

- C4.5 shows considerable improvement using Adaboost
- Adaboost reduces the amount of IC necessary, while $OC = 3$ for each

5.6 – Boosting Naive Bayes

46	52	71	68	11
31	63	71	68	9
49	73	71	68	7
43	75	71	67	5
77	74	71	67	3
71	73	71	67	1
1	3	5	7	

OC

IC

Table 16: Naive Bayes

31	59	72	72	11
32	65	75	73	9
32	75	75	72	7
69	**77**	74	70	5
77	74	72	68	3
71	74	71	68	1
1	3	5	7	

OC

IC

Table 17: Naive Bayes Adaboost (10)

31	59	72	72	11
32	65	75	73	9
32	75	76	72	7
75	**81**	74	70	5
77	74	71	68	3
71	74	71	68	1
1	3	5	7	

OC

IC

Table 18: Naive Bayes Adaboost (20)

31	59	72	72	11
32	65	75	73	9
32	75	76	72	7
76	**81**	74	70	5
78	75	72	68	3
72	74	72	68	1
1	3	5	7	

OC

IC

Table 19: Naive Bayes Adaboost (30)

Spatial Supervised Learning: Results and Discussion

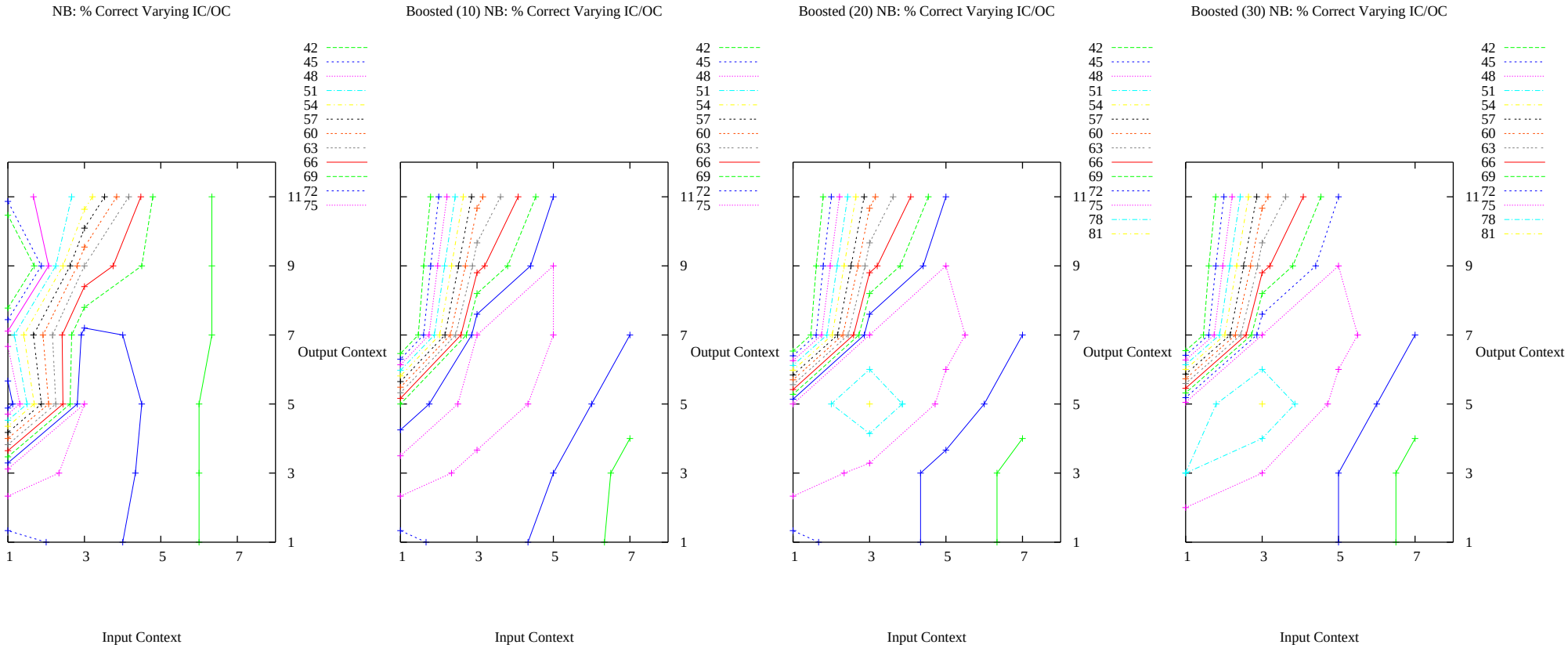


Figure 6: Effect of Adaboost using naive Bayes

- Boosting moves Max from (1,3) to (3,5)
- Not much change between 20 and 30 iterations of Adaboost

6 – Ensemble Effects

- Ensemble effects in
  - Training set expansion (Rotations and Reflections)
  - Voting of individual classifiers (with Adaboost and Bagging)
  - Spatial Ensemble (voting multiple image classification maps)
- How can we evaluate the individual contribution made by each ensemble?
- Compare with and without the ensemble in action.
- The best settings in Table 20 (next slide)
- Then the worst settings in Table 21
- Following that, we'll look at a couple of instances of the Spatial Ensemble in action

6.1 – Best Case: Ensemble Effects

Ensemble Algorithm	Spatial Ensemble	
	No	Yes
NB (3,5)	74	75
Boosted (20) NB (3,5)	77	**81**

Ensemble Algorithm	Spatial Ensemble	
	No	Yes
NB (1,3)	75	77
Bagged (10) NB (1,3)	74	**77**

Ensemble Algorithm	Spatial Ensemble	
	No	Yes
C4.5 (3,3)	63	67
Boosted (20) C4.5 (3,3)	76	**82**

Ensemble Algorithm	Spatial Ensemble	
	No	Yes
C4.5 (1,3)	62	62
Bagged (10) C4.5 (1,3)	75	**78**

Table 20: Accuracy of naive Bayes and C4.5 with and without ensemble effects. Values in **bold** are the **best** results for all IC/OC settings

- ☞ Left to right each table shows the effect of spatial ensembling
- ☞ Top to bottom each table shows the effect of classifier ensembling
- ☞ Both ensembles are generally beneficial, when the classifier alone does well

6.2 – Worst Case: Ensemble Effects

Ensemble Algorithm	Spatial Ensemble	
	No	**Yes**
NB (1,11)	46	44
Boosted (10) NB (1,11)	30	**31**

Ensemble Algorithm	Spatial Ensemble	
	No	**Yes**
NB (1,11)	46	44
Bagged (10) NB (1,11)	29	**30**

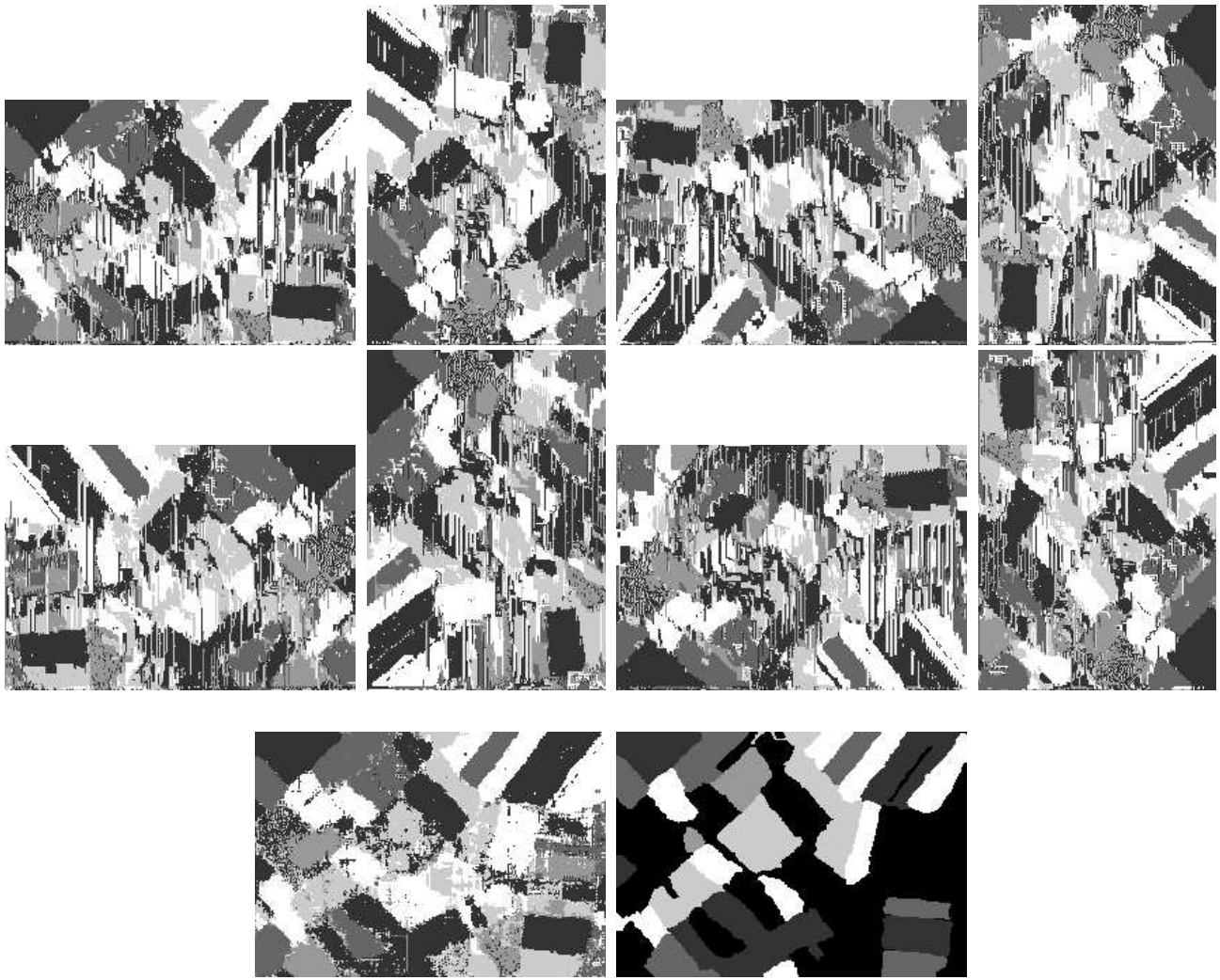
Ensemble Algorithm	Spatial Ensemble	
	No	**Yes**
C4.5 (7,5)	23	**21**
Boosted (20) C4.5 (7,5)	39	**37**

Ensemble Algorithm	Spatial Ensemble	
	No	**Yes**
C4.5 (7,5)	23	**21**
Bagged (20) C4.5 (7,5)	36	**36**

Table 21: Accuracy of naive Bayes and C4.5 with and without ensemble effects. Values in **bold** are the **worst** results for all IC/OC settings

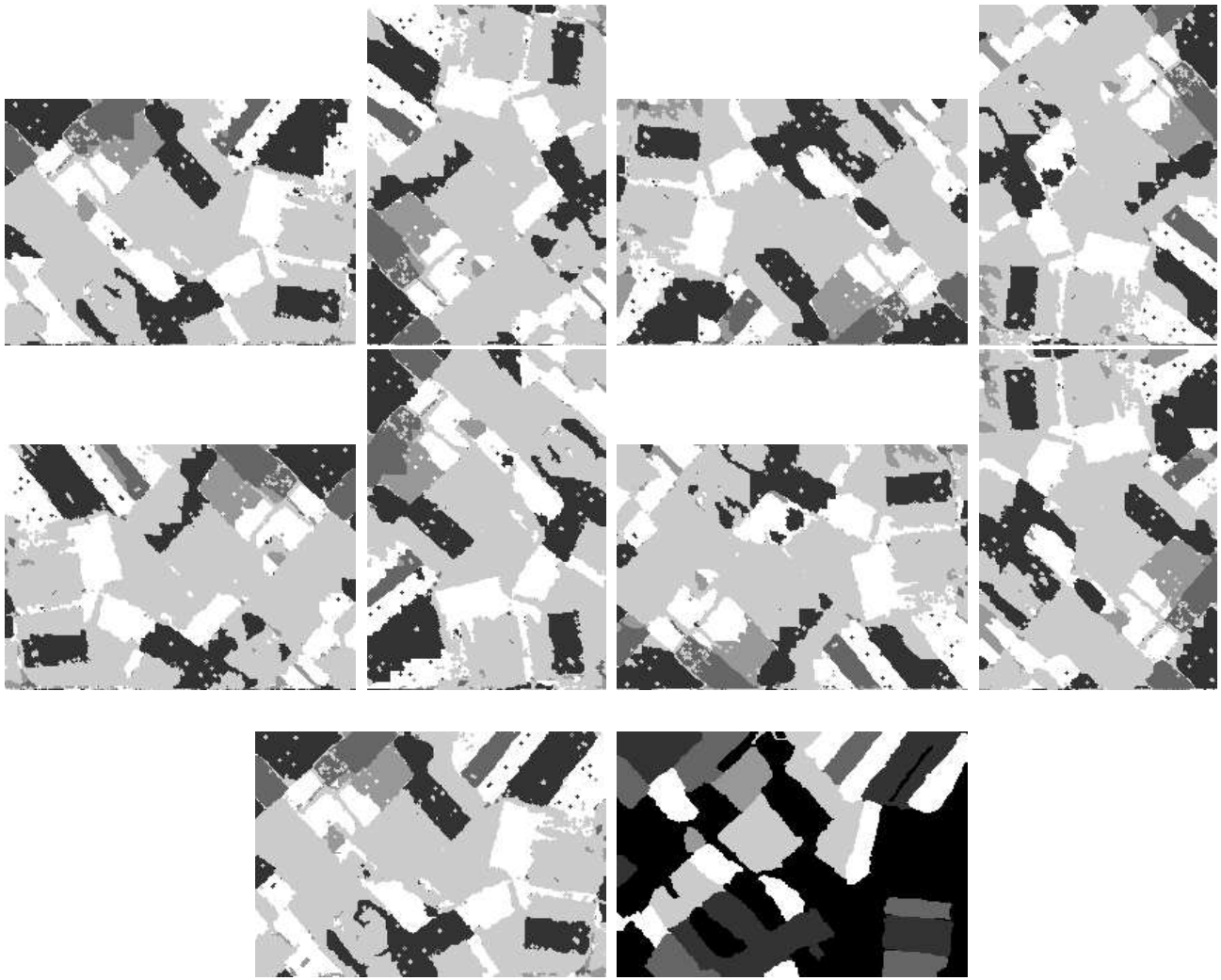
- ☞ Spatial ensembling either helps or hurts performance, but neither by very much.
- ☞ The ensemble classifier has a much greater (usually detrimental) impact than the spatial ensemble

6.3 – Spatial Ensemble with C4.5



Best C4.5 (7,3) results. Each of the eight images at the top corresponds to a recurrent sliding window classifier in a single scan order (the pictures have been rotated so that the scan order is left-to-right and top-to-bottom). We can see the “runny paint” effect whereby an error in one pixel is propagated for many pixels in the direction of the scan order. The left-bottom image is the result of the spatial ensemble voting. The voting has eliminated most of the runny paint to produce a much more accurate predicted image. Compare this to the right-bottom ground truth image.

6.4 – Spatial Ensemble with Naive Bayes



Naive Bayes (3,7). Each of the eight images at the top corresponds to a recurrent sliding window classifier in a single scan order (the pictures have been rotated so that the scan order is left-to-right and top-to-bottom). We can see the "stair-stepping" effect whereby an error in one pixel is propagated for many pixels and "stretches" the agricultural fields in the direction of the scan order. The bottom image is the result of the spatial ensemble voting. The voting has eliminated most of the stretched areas to produce a much more accurate predicted image. Compare this to the right-bottom ground truth image.

7 – Conclusions

7.1 – Recommendations

- Criteria
  - accuracy
  - robustness
  - training time
- Use Boosting; it improves both classifiers
- Bagging had little effect
- Naive Bayes is faster to train so try it first
- After boosting both algorithms are similarly robust
- No substitute for trying them all, just as Joshi [4] found
- Try using hill-climbing as a heuristic to save time

7.2 – Further Research

- large input and output contexts give better performance
- Pixels are small relative to the size of the features of the fields
- As window sizes increase so does the time and computing requirement
- To work around this need for context but not overwhelm ourselves we could
  - “Donut Context” - disk-shaped mask ignores local information
  - Reducing the resolution through subsampling
- The “donut context” research would show if distant pixels give us more information than local pixels
- Subsampling condenses the structure of the data [1].
- Smaller windows would contain more distant information and less local information.

7.3 – An Additional Control in this Experiment

- ☞ All experiments trained on all 8 rotations and reflections
- ☞ We should run the algorithm on just the original image
- ☞ We would call this the Training Spatial Ensemble

8 – Summary

- Recurrent sliding windows can be generalized to the 2-D case
- Classifier ensembles aide performance of these 2-D RSW
- Spatial ensembles further aide performance of these 2-D RSW

8.1 – Questions?

8.2 – Thank You

- My Major Professor: Dr. Dietterich
- My Committee: Dr. Tadepalli and Dr. Fern
- My Family

9 – Bibliography

References

[1] N. A. Dodgson. Image resampling. Technical Report UCAM-CL-TR-261, University of Cambridge, Computer Laboratory, August 1992.

[2] G. Giacinto, F. Roli, and L. Bruzzone. Combination of neural and statistical algorithms for supervised classification of remote-sensing images. *Pattern Recognition Letters*, 21(5):385–397, 2000.

[3] IEEE GRSS. Data fusion reference database data set grss_dfc_0006, 2001.

[4] S. Joshi and T. G. Dietterich. Calibrating recurrent sliding window classifiers for sequential supervised learning (revised), 2003.

[5] J. R. Quinlan. *C4.5: Programs for Empirical Learning.* Morgan Kaufmann, San Francisco, CA, 1993.

[6] N. M. Short Sr. The remote sensing tutorial (http://rst.gsfc.nasa.gov/), 2005.

[7] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, CA, 2000.

10 – Appendix

10.1 – Hill Climbing

1. Initially run experiments on setting of (IC',OC') at (1,1)
2. repeat until (IC',OC') == (IC,OC):
   (a) Set: $IC = IC'$, $OC = OC'$
   (b) Run experiment on all eligible neighbors of (IC,OC)
   (c) Set (IC',OC') to settings with the maximum accuracy, taking new (IC,OC) settings in case of a tie

☞ Hill climbing would find best for all but one starting from (1,1)

☞ 9 average experiment out of 24 possible

Algorithm	Number of experiments to maximum
C4.5	Failed to reach global maximum
AdaBoosted (10) C4.5	11
AdaBoosted (20) C4.5	9
AdaBoosted (30) C4.5	12
Bagging (10) C4.5	6
Bagging (20) C4.5	6
naive Bayes	6
Adaboost (10) naive Bayes	11
Adaboost (20) naive Bayes	11
Adaboost (30) naive Bayes	11
Bagging (10) naive Bayes	6
Bagging (20) naive Bayes	6

Table 22: Number of different (IC,OC) settings to Maximum

10.2 – Handling Pixels Without Labels

- There are many unlabeled pixels, 36% of them.
  - 30% of training set
  - 43% of test set
- Collecting ground truth is expensive
- All features are known (cheaper than ground truth)
- With unknowns, there is less data from which to learn the relationship between the $y_{i,s}$ values.
- Naive Bayes handles unknowns in the x_i, s feature values gracefully as long as those instances have a known $y_{i,s}$ label.
- C4.5 behaves similarly [5].

10.3 – Spatially Correct Division of the Dataset

- The dataset is difficult to divide in a random fashion.
- To avoid overlapping of contexts the dataset was divided in half
- The 250 pixel $\times$ 350 pixel dataset became two contiguous sets
  - each is a 250 pixel $\times$ 175 pixel section
  - Easy notation, straight edges

10.4 – Algorithms

- Our spatial supervised learning meta-classifier was developed in Weka [7].
- The Weka implementations of the following algorithms were used in this project
  - naive Bayes
  - C4.5 decision tree
  - Boosting
  - Bagging

10.5 – Handling Edge Pixels

- Edge pixels have no predefined context
- The sliding window will extend **beyond** the boundaries of the image.
- When this happens what should we do?
- We considered four different approaches
  - treating feature values as unknown
    - fails to using spatial information that we do have
  - repeating the boundary value for unknowns beyond it
    - seems unnaturally unvarying
  - using a gibbs sampling technique to generate more pixels synthetically
    - requires more coding, and introduces another variable
  - using a reflection of the data across the boundary.
    - good tradeoff between data utilization and min added complexity

10.6 – Boundry Reflection

North

	$R_3C_3\ R_3C_2$ | $\mathbf{R_3C_1}\ R_3C_2\ R_3C_3$	
	$R_2C_3\ R_2C_2$ | $\mathbf{R_2C_1}\ R_2C_2\ R_2C_3$	
West	$\mathbf{R_1C_3\ R_1C_2}$ | $\mathbf{R_1C_1\ R_1C_2\ R_1C_3}$	East
	$R_2C_3\ R_2C_2$ | $\mathbf{R_2C_1}\ R_2C_2\ R_2C_3$	
	$R_3C_3\ R_3C_2$ | $\mathbf{R_3C_1}\ R_3C_2\ R_3C_3$	

South

Table 23: Reflection of Pixel Values in a 5 $\times$ 5 context window before mask is applied. The "corner" shown is the top left-hand corner of the image. The edge pixels shown in **red** are the axis of reflection.

- Prior to training the classifier, the edges are reflected
- During classification, the $\hat{y}_{i,s}$ values are inserted in real-time
- Propagation of belief is evident in Figures 6.3 and 6.4.

10.7 – Spatial Ensembles

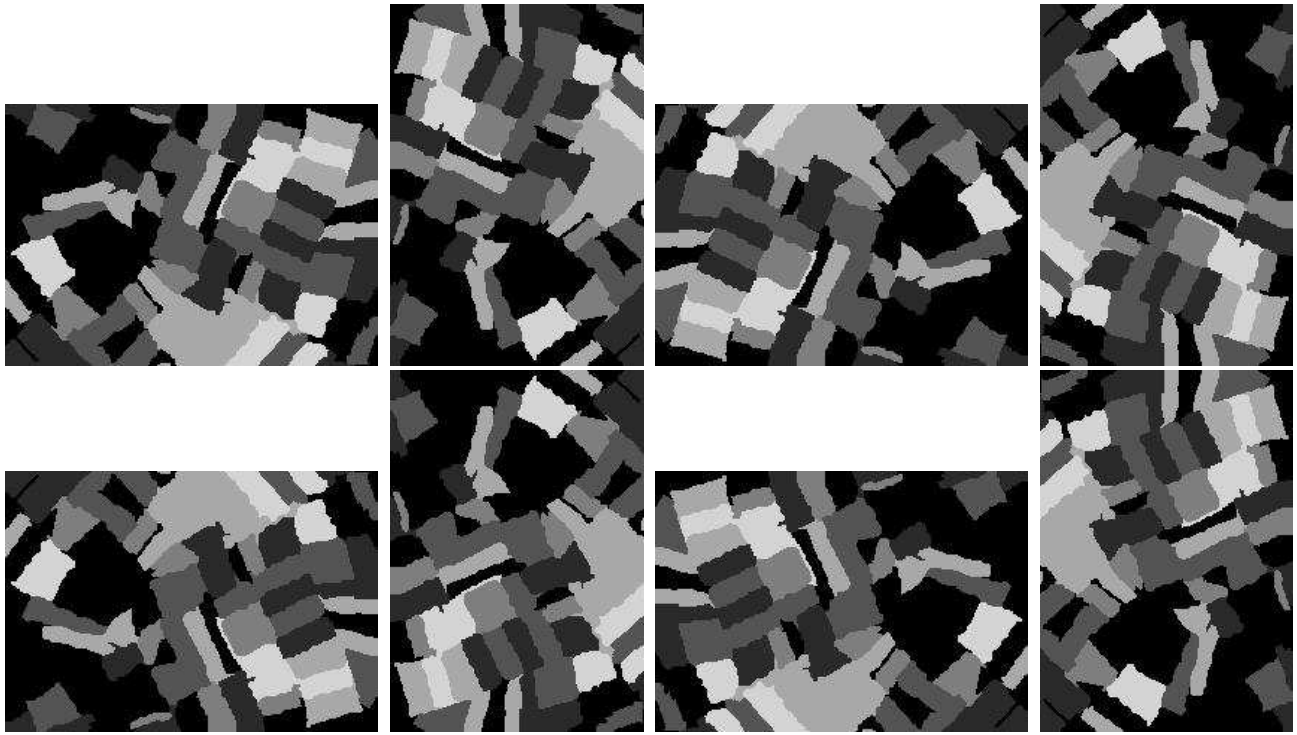


Figure 7: Eight images, the top-left image is the original training set and the other seven are its rotations and reflections

- We rotate and reflect our training data to create an 8-fold training set.
- Rotations could be any amount, as long as output context is used.
- All of our experiments used this 8-fold training set.