Slide 1

Skewed data sets, ROC analysis, and learning to detect rooftops

Mark Maloof
Department of Computer Science
Georgetown University
maloof@cs.georgetown.edu
http://www.cs.georgetown.edu/~maloof

Joint work with:
Pat Langley (ISLE & Stanford)
Tom Binford (Stanford)
Ram Nevatia (USC)

Sponsors: DARPA through ONR, Sun Microsystems

GMU School of Computational Sciences
16 November 2000

Slide 2

Talk Overview

- Brief Introduction to in Machine Learning
- BUDDS: **Bu**ilding **D**etection and **D**escription **S**ystem
- Experiment I: Evaluating Methods Traditionally
- Skewed Data Sets and Unequal but Unknown Error Costs
- Experiment II: Cost-sensitive Learning and ROC Analysis
- A Statement to Generate Controversy

Slide 3

Learning from Examples

- One way humans (and computers) learn is from examples
- Imagine a child learning the concept ‘dog’

spaniel	dog	+	
husky	dog	+	
retriever	dog	+	← positive example
cat	not a dog	−	← negative example
cow	not a dog	−	
wolf	not a dog	−	
boxer	dog	+	

Slide 4

What is Being Learned?

- How does the child know its a dog? What *features* does the child use to recognize the dog? Its shape? color? fur? sound?
- What is the child learning?
  - is she learning the features that are predictive of a dog?
“This animal has fur and barks, so it is a doggie”
  - is she remembering specific cases?
“This animal sounds more like Lassie than Garfield, so its a doggie”
  - is she doing a little of both?
  - should machines do a little of both?

Slide 5

Testing and Generalization

- How do *we* know the child has learned ‘dog’?

Show her a poodle	Child: dog	Correct
Show her a lion	Child: not a dog	Correct
Show her a hyena	Child: dog	Oops, incorrect

- So we have the notions of *training* and *testing*
- We also have the notion of *generalization*:
  - she correctly identified the poodle and the lion but had never seen them before
  - over-generalization: everything is a dog!
  - under-generalization: nothing is a dog!

Slide 6

Accuracy and Error Costs

- By counting mistakes, we can *measure accuracy*:
  - true positive: saying ‘doggie’ to Lassie ‘item true negative: saying ‘not a doggie’ to Garfield
  - false positive: saying ‘doggie’ to a hyena
  - false negative: saying ‘not a doggie’ to a doberman
- How should performance change with more and more training?
  - hopefully it increases!
- Mistakes have different costs:
  - saying ‘doggie’ to a grizzly bear: HIGH COST!
  - saying ‘not a doggie’ to a poodle: low cost

Slide 7

Evaluation Methodology

- Like parents with children, ML researchers want to show their *learning algorithm* is best!
- How to do this in an unbiased way?
  - run experiments
  - IMPORTANT: Train on a randomly selected portion of the data and test on the remainder
  - plot accuracy: errors, types of errors
  - examine trade-offs
  - plot learning curves (i.e., accuracy over time)
- Other performance measures: time, space, understandability of learned concepts

Slide 8

Machine Learning for Image Understanding

- Existing software for image understanding is fragile and depends on manual setting of numerous parameters
- Machine learning holds the potential for:
  - acquiring the conditions for applying vision operators
  - determining the parameter values to use for particular images
  - inducing classifiers to identify visual objects
- Moreover, we predict that learned knowledge:
  - will be more accurate than handcrafted knowledge
  - will be constructed more efficiently than by manual methods
- However, this approach to embedding knowledge in vision system requires a source of training data

Slide 9

Opportunities for Visual Learning

- Lin and Nevatia (1996) present one approach to detecting buildings in RADIUS images; Budds uses knowledge at a number of levels:

  1. Grouping pixels into edge elements (i.e., edgels)
  2. Grouping edgels into lines
  3. Finding junctions and parallel lines
  4. Combining junctions and parallels into ‘Us’
  5. Grouping ‘Us’ into parallelograms (rooftop candidates)
  6. Verifying rooftop candidates (walls, shadows, overlap)
  7. Generating 3D building descriptions

- Learning can occur at any of these levels, but we have focused on rooftop detection (step 5)

Slide 10

Attributes for Representing Rooftop Candidates

- Budds uses nine continuous attributes to evaluate rooftop candidates:

  1. Support for corners
  2. Support for parallel lines
  3. Support for orthogonal trihedral vertices
  4. Support for corner shadows
  5. Gaps in the edges of the candidate
  6. Displacement of edge support
  7. Lines crossing the candidate
  8. Existence of adjacent L-junctions and T-junctions

- We included each of these features in the training and test descriptions used for learning

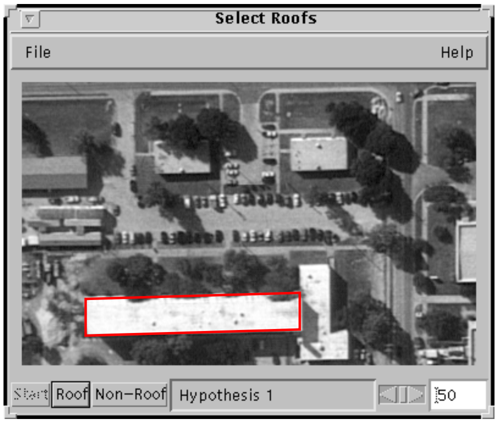
Slide 11

Generation of Training and Testing Data

- Our study focused on two images from the Fort Hood RADIUS library: FHOV1027 and FHOV625
- These images were of the same area but were taken from different aspects: nadir and oblique
- We identified three locations that contained concentrations of buildings, resulting in six images
- Used BUDDS to extract rooftop candidates from each image
- Used a visualization system to label the candidates as either ‘positive’ or ‘negative’
- Required 5 hours to label 17,829 candidates
- Produced 718 positive examples and 17,048 negative examples

Slide 12

Visualization Interface for Labeling Rooftop Candidates



Slide 13

Visualization System for Labeling Rooftop Candidates

- Displays rooftop candidates, which it overlays on the image
- User classifies candidates as either 'roof' or 'nonroof'
- Investigating visualization and feedback mechanisms:
  - users can view candidates in attribute space as well as the visual space
  - nearest neighbor classifies instances based on previously labeled candidates
  - displays 'roofs' as green rectangles, 'non-roofs' as blue rectangles, and 'unknowns' as red rectangles
  - 'unknowns' are those candidates with a distance greater than a user-defined threshold

Slide 14

Image and Data Set Characteristics

Image Number	Original Image	Location	Aspect	Positive Examples	Negative Examples
1	FHOV1027	1	Nadir	197	982
2	FHOV625	1	Oblique	238	1955
3	FHOV1027	2	Nadir	71	2645
4	FHOV625	2	Oblique	74	3349
5	FHOV1027	3	Nadir	87	3722
6	FHOV625	3	Oblique	114	4395
				718	17,829

Slide 15

Nearest Neighbor

- Stores training cases in memory
- Predicts the class of the case “nearest” to a query
- Compute similarity using a distance function (e.g., Euclidean)
- Variants poll the k-nearest neighbors and predict the class with the majority vote

Slide 16

Naive Bayes

- Induces a probabilistic concept description for each class
- Uses a training set to estimate:
  - the prior probability of each class: $Pr(C_i)$
  - the conditional probability each attribute value given the class: $Pr(v_j|C_i)$
- Assumes conditional independence of attributes, hence *naive*
- Predicts the class with the highest posterior probability as computed by Bayes’ rule:

$$\Pr(C_i|v_j) = \frac{\Pr(C_i)\Pr(v_j|C_i)}{\sum_k \Pr(C_k)\Pr(v_j|C_k)}$$

Slide 17

The Perceptron Classifier (aka. Linear Classifier)

- Represents concepts as a set of weights, $\mathbf{w}$, and a threshold, θ
- Predicts + if the weighted sum of the attribute values surpasses the threshold; predicts − otherwise

$$o = \begin{cases} + & \text{if } \sum_{i=1}^{n} w_i v_i > \theta \\ - & \text{otherwise} \end{cases}$$

- Determines the weights, $\mathbf{w}$, using a gradient descent technique that minimizes the error between the actual and the predicted output values

Slide 18

C5.0 (commercial successor of C4.5)

- Concept representation:
  - internal nodes correspond to attributes
  - leaf nodes correspond to classes
  - links between nodes correspond to attribute values
- Classification of an instance:
  - start at the tree's root
  - follow links based on the instance's attribute values
  - upon reaching a leaf node, return its class as the prediction
- Inducing decision trees:
  - select the attribute that best divides the training examples into their correct classes (information theoretic measures)
  - create a node for that attribute
  - create links for each attribute value
  - create a data set for each link by selecting on attribute value
  - proceed recursively with each new data set
  - "pruning" algorithm prevents overtraining

Slide 19

The Budds Classifier (aka. Linear Classifier)

- Represents concepts as a set of weights, $\mathbf{w}$, and a threshold, θ
- Predicts + if the weighted sum of attribute values surpasses the threshold; predicts − otherwise:

$$o = \begin{cases} + & \text{if } \sum_{i=1}^{n} w_i v_i > \theta \\ - & \text{otherwise} \end{cases}$$

- The weight vector for Budds was determined by handcrafting

Slide 20

Experiment I: Evaluating the Methods Traditionally

- What is the best classification method for this problem?
  - Combined all available image data
  - Randomly selected 60% for training, 40% for testing
  - Conducted 10 learning and testing runs
  - Ran Budds classifier, C5.0, naive Bayes, perceptron, nearest neighbor, and k-NN, for $k = 1 \ldots 19$
  - Used percent correct as the performance measure

Slide 21

Learning to Predict the Majority Class

Method	Overall	TP Rate	FP Rate
C5.0	**96.27±0.25**	0.23±0.02	0.01±0.001
k-NN ($k = 17$)	96.07±0.13	0.19±0.02	0.004±0.0003
Perceptron	95.68±0.12	0.02±0.01	**0.0001±0.0001**
BUDDS Classifier	91.70±0.15	0.55±0.02	0.07±0.0008
Naive Bayes	90.76±0.35	**0.56±0.008**	0.08±0.004

($p < 0.01$)

Slide 22

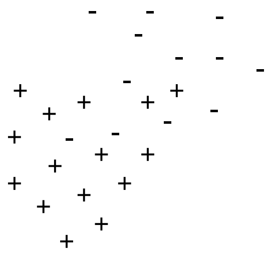
Error Costs and Skewed Data Sets

- For many applications, such as vision, fraud detection, computer security, medical diagnosis, mistakes are not equal
- In BUDDS, it is better to keep a false positive than to remove a false negative (i.e., a misclassified rooftop)
- Most learning algorithms assume equal error costs among classes
- Skewed data sets complicate this issue:
  - BUDDS generated thousands of non-rooftops and only a handful of rooftops
  - By adding examples to a class, we can increase its prior probability *and* its error costs (Breiman et al. 1984)
  - As a result, many methods learn to predict the majority class
  - But it is the *minority class* that is more important!

Slide 23

Cost-sensitive Learning Algorithms

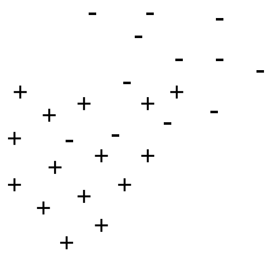
- Incorporate a cost heuristic into a method to change the decision boundary at which the method selects one class over the other
- This lets us bias the algorithm toward the preferred class
- If we know the costs of errors, we can find the minimum-cost classifier (e.g., Pazzani et al. 1994)
- Problem: We often do not know the costs of errors



Slide 24

Unequal but Unknown Error Costs

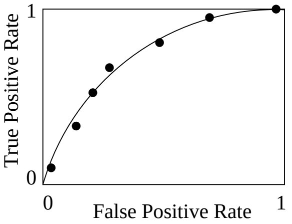
- Precise cost analyses may be impractical or may not be exist
- Here, we only have an informal notion that false positives are more costly than false negatives
- Cost analyses change (e.g., peacetime vs. wartime)
- Solution: Run algorithms over all possible costs!
- New Problem: Evaluation methodology?



Slide 25

ROC Analysis

- Lets us evaluate performance for a variety of error costs
- ROC curve plots the true positive and false positive rates over a range of misclassification costs for a given method
- The point (0, 1) is where classification is perfect, so we want curves that "push" toward this corner
- Traditional ROC analysis uses area under the curve as the measure of performance



Slide 26

Cost-Sensitive Nearest Neighbor

- Returns the class label of the closest neighbor to the query
- Defined a misclassification cost for a class on the range $[0, 1]$
- Numbers closer to one indicate a higher cost of error (i.e., a higher preference for the class)
- Modified the performance element of the method
- Computed the expected cost of a class as a function f of the distance to the closest example and the cost metric
- Minimize f for large values of the cost metric and small values of the distance to the closest example
- Has the effect of moving the query point closer to the nearest neighbor of the preferred class
- Predicts the class with the least expected cost

Slide 27

Cost-Sensitive Naive Bayes

- Induces a probabilistic concept descriptions and predicts the class with the highest posterior probability
- Defined a misclassification cost for a class on the range $[0, 1]$
- Numbers closer to one indicate a higher cost of error (i.e., a higher preference for the class)
- Modified the performance element of the method
- Computed the expected cost of a class as a function f of the posterior probability and the cost metric
- Minimize f for large values of the posterior probability and cost metric
- Has the effect of making the preferred class more probable
- Predicts the class with the least expected cost

Slide 28

Cost-Sensitive Perceptron

- Defined a misclassification cost for a class on the range $[0, 1]$
- Numbers closer to one indicate a higher cost of error
- Modified the performance element of the method
- Used the cost metric to adjust the threshold *farther* from the preferred class
- Has the effect of enlarging the decision region of the preferred class
- Predicts using the adjusted threshold, θ'

Slide 29

Costs in C5.0

- Uses a method similar to that of CART (Breiman et al., 1984)
- Grows decision trees in a way that minimizes error cost
- Estimates the prior probability of each class from the training set
- Computes altered priors from estimated priors and error costs for each class
- Uses altered priors to bias the selection of attributes during the tree growing process

Slide 30

Cost-Sensitive Budds Classifier

- Hand-configured linear classifier
- Defined a misclassification cost for a class on the range $[0, 1]$
- Numbers closer to one indicate a higher cost of error
- Modified the performance element of the method
- Used the cost metric to adjust the threshold *farther* from the more costly class
- Has the effect of enlarging the decision region of the more costly class, making mistakes on that class less likely
- Predicts using the adjusted threshold, θ'

Slide 31

Description of Experiments

- Conducted three experiments using error costs:
  1. Within-Image Learning
     – established a baseline performance condition
  2. Between-Image Learning
     – determined best-performing method for generalizing to unseen images
     – investigated performance degradation when generalizing to unseen images
  3. Learning with all of the image data
     – determined the best performing method
     – evaluated performance as the amount of training data increases

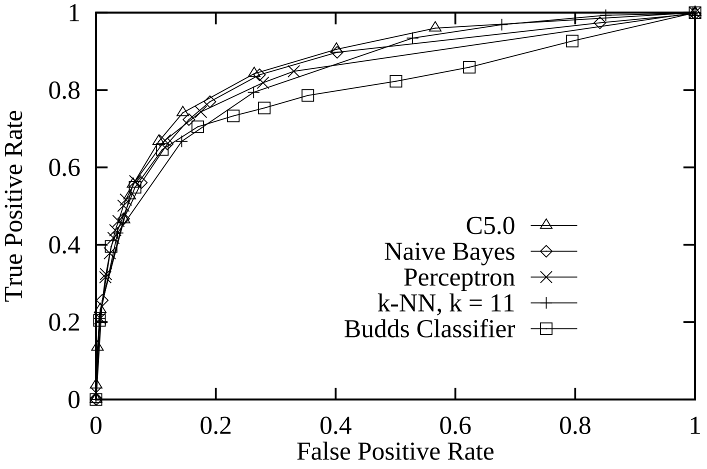
Slide 32

Experiment II: Cost-sensitive Algorithms and ROC Analysis

- What is the best classification method for this problem?
  – Combined all available image data
  – Selected 60% for training, 40% for testing
  – Conducted 10 learning and testing runs
  – Ran cost-sensitive versions of the BUDDS classifier, C5.0, naive Bayes, perceptron, nearest neighbor, and k-NN, for $k = 1 \dots 19$
  – Varied the decision thresholds
  – Plotted averaged results as ROC curves
  – Used area under curve as the performance measure
  – Tested significance using ANOVA and LabMRMC

Slide 33

ROC Curve for All Image Data



Slide 34

Areas Under the ROC Curves

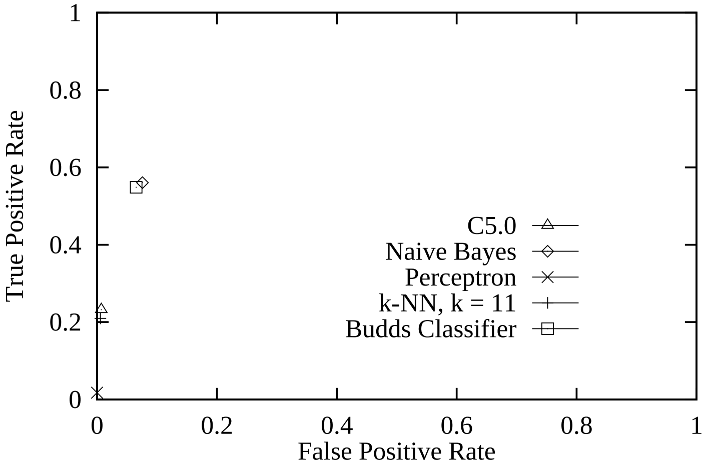
Classifier	Area under ROC Curve
C5.0	0.867±0.006
Naive Bayes	0.854±0.009
Perceptron	0.853±0.010
k-NN ($k = 11$)	0.847±0.006
BUDDS Classifier	0.802±0.014

ANOVA: $p < 0.01$, LabMRMC: $p < 0.001$*

* The current implementation of LabMRMC is limited to five treatments (i.e., learning algorithms), so we conducted this analysis for the best five and not all twelve.

Slide 35

Plotting Results from Experiment I in the ROC Space



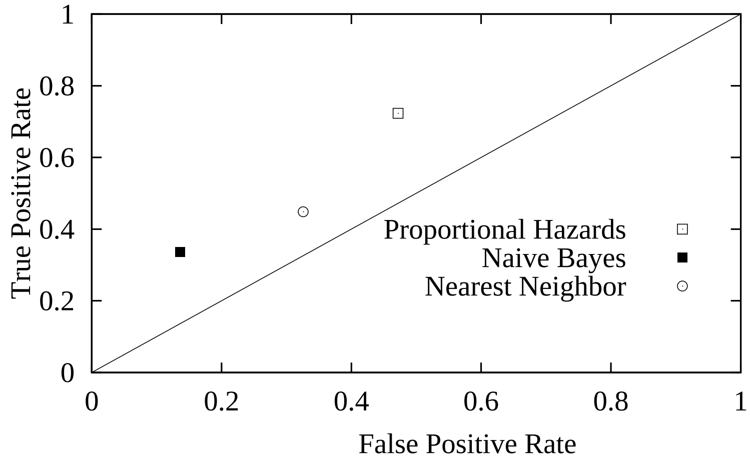
Slide 36

Summary and Implications of Results

- Under the same experimental conditions, each “standard” algorithm produces a point on some unknown ROC curve
- Because Algorithm 1’s point is higher on its unknown ROC curve than Algorithm 2’s does not mean that Algorithm 1’s ROC curve is of greater area than Algorithm 2’s
- Implication: May be able to find a cost—indeed, many costs—for Algorithm 2 such that its performance is better than that of Algorithm 1
- For all the past studies using percent correct, was the best algorithm really the best?
- Or did we simply run it using the wrong costs?
- Conjecture: Each algorithm operates under an inherent set of costs associated with its inductive bias

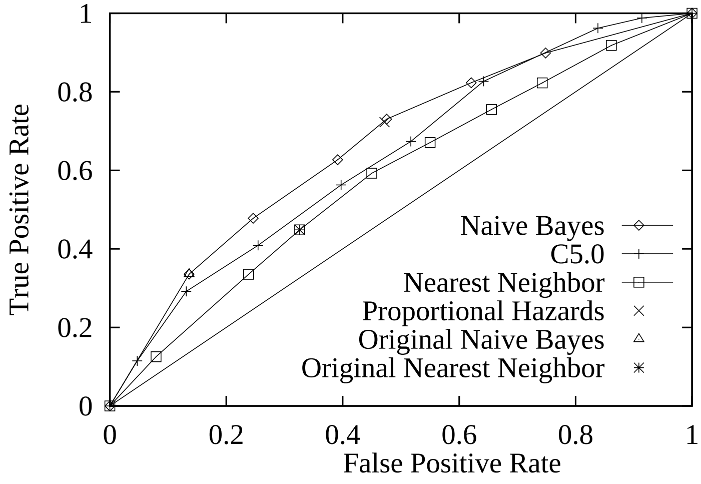
Slide 37

Results from a Recidivism Prediction Task



Slide 38

Results from a Recidivism Prediction Task



Slide 39

Plans for Future Work

- We intend to expand our studies of machine learning to include:
  - experiment using larger sets of RADIUS images
  - studies at different levels of BUDDS
  - simulation studies comparing ANOVA and LabMRMC
- In the longer term, we also hope to:
  - adapt our approach to learn contextual rules
  - embed learning within BUDDS
  - improve methods for labeling large amounts of training data
- Goal: Adaptive image understanding systems that use learning and feedback from users to improve performance

References

Ali, K., Langley, P., Maloof, M., Sage, S., & Binford, T. (1998). Improving rooftop detection with interactive visual learning. In *Proceedings of the Image Understanding Workshop* (pp. 479–492). San Francisco, CA: Morgan Kaufmann.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees.* Boca Raton, FL: CRC Press.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *IT-13*, 21–27.

Dorfman, D., Berbaum, K., & Metz, C. (1992). Receiver Operating Characteristic rating analysis: Generalization to the population of readers and patients with the jackknife method. *Investigative Radiology*, *27*, 723–731.

Duda, R., Hart, P., & Stork, D. (2000). *Pattern classification* (2nd ed.). New York, NY: John Wiley & Sons.

Firschein, O., & Strat, T. (Eds.). (1997). *RADIUS: image understanding for imagery intelligence.* San Francisco, CA: Morgan Kaufmann.

Lin, C., & Nevatia, R. (1996). Building detection and description from monocular aerial images. In *Proceedings of the Image Understanding Workshop* (pp. 461–468). San Francisco, CA: Morgan Kaufmann.

Maloof, M. (1999). *A machine learning researcher's foray into recidivism prediction* (Technical Report No. CS-99-2). Wasington, DC: Department of Computer Science, Georgetown University.

Maloof, M. (2000). An initial study of an adaptive hierarchical vision system. In *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 567–573). San Francisco, CA: Morgan Kaufmann.

Maloof, M., Langley, P., Binford, T., & Nevatia, R. (1998). Generalizing over aspect and location for rooftop detection. In *Proceedings of the Fourth IEEE Workshop on Applications of Computer Vision (WACV '98)* (pp. 194–199). Los Alamitos, CA: IEEE Press.

Maloof, M., Langley, P., Binford, T., & Sage, S. (1998). *Learning to detect rooftops in overhead imagery* (Technical Report No. 98-1). Palo Alto, CA: Institute for the Study of Learning and Expertise.

Maloof, M., Langley, P., Sage, S., & Binford, T. (1997). Learning to detect rooftops in aerial images. In *Proceedings of the Image Understanding Workshop* (pp. 835–845). San Francisco, CA: Morgan Kaufmann.

Mitchell, T. (1997). *Machine learning.* New York, NY: McGraw-Hill.

Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., & Brunk, C. (1994). Reducing misclassification costs. In *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 217–225). San Francisco, CA: Morgan Kaufmann.

Quinlan, J. (1993). *C4.5: Programs for machine learning.* San Francisco, CA: Morgan Kaufmann.

Swets, J., & Pickett, R. (1982). *Evaluation of diagnostic systems: Methods from signal detection theory.* New York, NY: Academic Press.