Ensembles for Cost-Sensitive Learning

Thomas G. Dietterich
Department of Computer Science
Oregon State University
Corvallis, Oregon 97331
http://www.cs.orst.edu/~tgd

Outline

- Cost-Sensitive Learning
  - Problem Statement; Main Approaches
- Preliminaries
  - Standard Form for Cost Matrices
  - Evaluating CSL Methods
- Costs known at learning time
- Costs unknown at learning time
- Open Problems

Cost-Sensitive Learning

- Learning to minimize the *expected cost* of misclassifications
- Most classification learning algorithms attempt to minimize the *expected number* of misclassification errors
- In many applications, different kinds of classification errors have different costs, so we need cost-sensitive methods

Examples of Applications with Unequal Misclassification Costs

- Medical Diagnosis:
  - Cost of false positive error: Unnecessary treatment; unnecessary worry
  - Cost of false negative error: Postponed treatment or failure to treat; death or injury
- Fraud Detection:
  - False positive: resources wasted investigating non-fraud
  - False negative: failure to detect fraud could be very expensive

Related Problems

- Imbalanced classes: Often the most expensive class (e.g., cancerous cells) is rarer and more expensive than the less expensive class
- Need statistical tests for comparing expected costs of different classifiers and learning algorithms

Example Misclassification Costs Diagnosis of Appendicitis

- Cost Matrix: C(i,j) = cost of predicting class i when the true class is j

Predicted State of Patient	True State of Patient	
	Positive	Negative
Positive	1	1
Negative	100	0

Estimating Expected Misclassification Cost

- Let M be the *confusion matrix* for a classifier: M(i,j) is the number of test examples that are predicted to be in class i when their true class is j

Predicted Class	True Class	
	Positive	Negative
Positive	40	16
Negative	8	36

Estimating Expected Misclassification Cost (2)

- The expected misclassification cost is the Hadamard product of M and C divided by the number of test examples N:

$$\Sigma_{i,j} M(i,j) * C(i,j) / N$$

- We can also write the probabilistic confusion matrix: $P(i,j) = M(i,j) / N$. The expected cost is then $P * C$

Interlude: Normal Form for Cost Matrices

- Any cost matrix C can be transformed to an equivalent matrix C' with zeroes along the diagonal
- Let L(h,C) be the expected loss of classifier h measured on loss matrix C.
- Defn: Let h_1 and h_2 be two classifiers. C and C' are *equivalent* if

$$L(h_1,C) > L(h_2,C) \text{ iff } L(h_1,C') > L(h_2,C')$$

Theorem
(Margineantu, 2001)

- Let Δ be a matrix of the form

$$\begin{bmatrix} \delta_1 & \delta_2 & \dots & \delta_k \\ \delta_1 & \delta_2 & \dots & \delta_k \\ \dots & & & \\ \delta_1 & \delta_2 & \dots & \delta_k \end{bmatrix}$$

- If $C_2 = C_1 + \Delta$, then C_1 is equivalent to C_2

Proof

- Let $P_1(i,k)$ be the probabilistic confusion matrix of classifier h_1, and $P_2(i,k)$ be the probabilistic confusion matrix of classifier h_2
- $L(h_1,C) = P_1 * C$
- $L(h_2,C) = P_2 * C$
- $L(h_1,C) – L(h_2,C) = [P_1 – P_2] * C$

Proof (2)

- Similarly, $L(h_1,C') - L(h_2, C')$

  $= [P_1 - P_2] * C'$

  $= [P_1 - P_2] * [C + \Delta]$

  $= [P_1 - P_2] * C + [P_1 - P_2] * \Delta$

- We now show that $[P_1 - P_2] * \Delta = 0$, from which we can conclude that

  $L(h_1,C) - L(h_2,C) = L(h_1,C') - L(h_2,C')$

and hence, C is equivalent to C'.

Proof (3)

$$[P_1 - P_2] * \Delta = \Sigma_i \Sigma_k [P_1(i,k) - P_2(i,k)] * \Delta(i,k)$$

$$= \Sigma_i \Sigma_k [P_1(i,k) - P_2(i,k)] * \delta_k$$

$$= \Sigma_k \delta_k \Sigma_i [P_1(i,k) - P_2(i,k)]$$

$$= \Sigma_k \delta_k \Sigma_i [P_1(i|k) P(k) - P_2(i|k) P(k)]$$

$$= \Sigma_k \delta_k P(k) \Sigma_i [P_1(i|k) - P_2(i|k)]$$

$$= \Sigma_k \delta_k P(k) [1 - 1]$$

$$= 0$$

Proof (4)

- Therefore,

  $L(h_1,C) - L(h_2,C) = L(h_1,C') - L(h_2,C')$.

- Hence, if we set $\delta_k = -C(k,k)$, then C' will have zeroes on the diagonal

End of Interlude

- From now on, we will assume that $C(i,i) = 0$

Interlude 2: Evaluating Cost-Sensitive Learning Algorithms

- Evaluation for a particular C:
  - BCost and BDeltaCost procedures
- Evaluation for a range of possible C's:
  - AUC: Area under the ROC curve
  - Average cost given some distribution D(C) over cost matrices

Two Statistical Questions

- Given a classifier h, how can we estimate its expected misclassification cost?
- Given two classifiers h_1 and h_2, how can we determine whether their misclassification costs are significantly different?

Estimating Misclassification Cost: BCost

- Simple Bootstrap Confidence Interval
  - Draw 1000 bootstrap replicates of the test data
  - Compute confusion matrix M_b, for each replicate
  - Compute expected cost $c_b = M_b * C$
  - Sort c_b's, form confidence interval from the middle 950 points (i.e., from $c^{(26)}$ to $c^{(975)}$).

Comparing Misclassification Costs: BDeltaCost

- Construct 1000 bootstrap replicates of the test set
- For each replicate b, compute the combined confusion matrix $M_b(i,j,k)$ = # of examples classified as i by h_1, as j by h_2, whose true class is k.
- Define $\Delta(i,j,k) = C(i,k) - C(j,k)$ to be the difference in cost when h_1 predicts class i, h_2 predicts j, and the true class is k.
- Compute $\delta_b = M_b * \Delta$
- Sort the δ_b's and form a confidence interval $[\delta^{(26)}, \delta^{(975)}]$
- If this interval excludes 0, conclude that h_1 and h_2 have different expected costs

ROC Curves

- Most learning algorithms and classifiers can tune the decision boundary
  - Probability threshold: $P(y=1|x) > \theta$
  - Classification threshold: $f(x) > \theta$
  - Input example weights λ
  - Ratio of $C(0,1)/C(1,0)$ for C-dependent algorithms

ROC Curve

- For each setting of such parameters, given a validation set, we can compute the false positive rate:

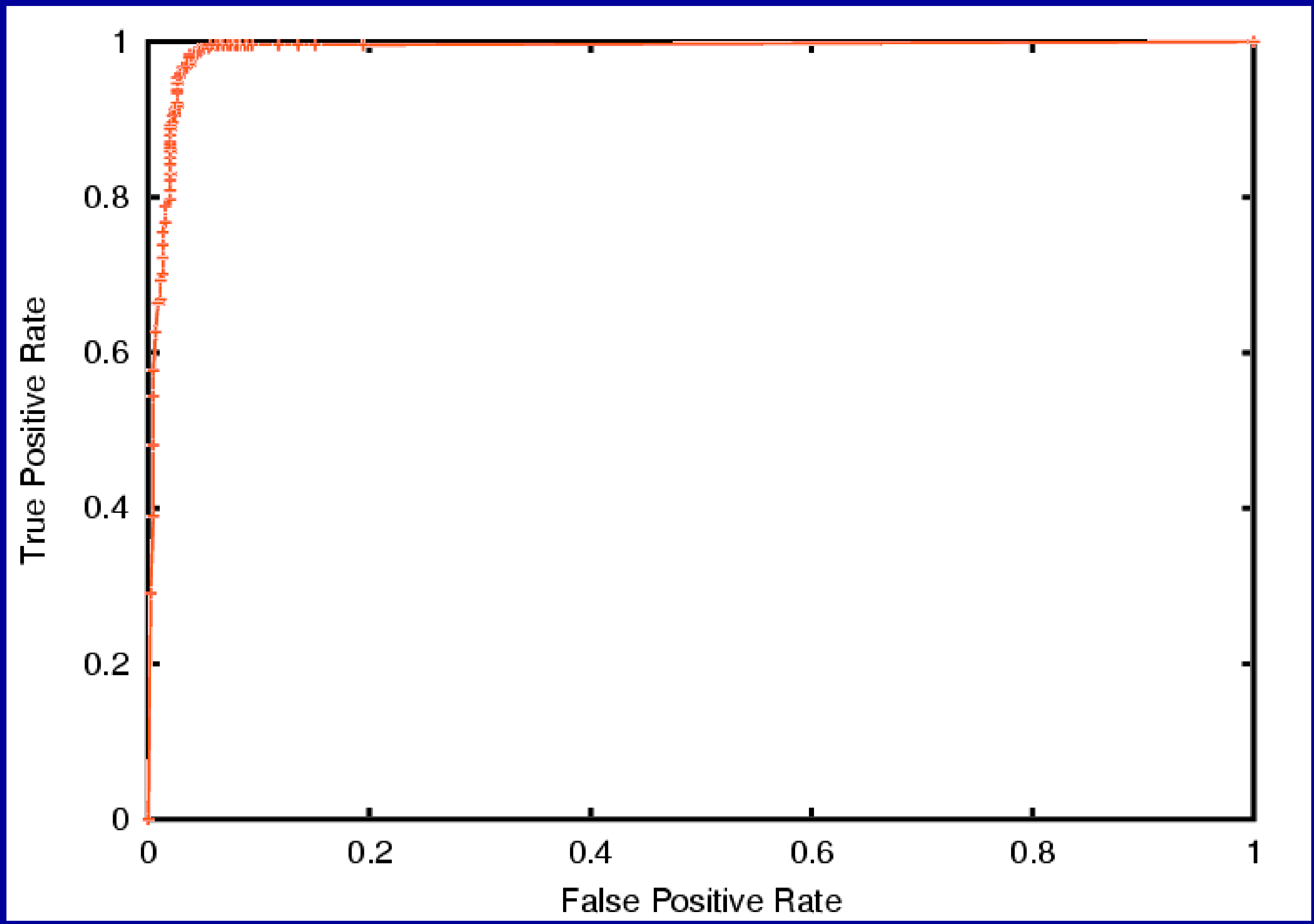
  fpr = FP/(# negative examples)

  and the true positive rate

  tpr = TP/(# positive examples)

  and plot a point (tpr, fpr)
- This sweeps out a curve: The ROC curve

Example ROC Curve



AUC: The area under the ROC curve

- AUC = Probability that two randomly chosen points x_1 and x_2 will be correctly ranked: $P(y=1|x_1)$ versus $P(y=1|x_2)$
- Measures correct ranking (e.g., ranking all positive examples above all negative examples)
- Does not require correct estimates of $P(y=1|x)$

Direct Computation of AUC (Hand & Till, 2001)

- Direct computation:
  - Let $f(x_i)$ be a scoring function
  - Sort the test examples according to f
  - Let $r(x_i)$ be the rank of x_i in this sorted order
  - Let $S_1 = \Sigma_{\{i:\ yi=1\}} r(x_i)$ be the sum of ranks of the positive examples
  - $AUC = [S_1 – n_1(n_1+1)/2] / [n_0\ n_1]$

    where n_0 = # negatives, n_1 = # positives

Using the ROC Curve

- Given a cost matrix C, we must choose a value for θ that minimizes the expected cost
- When we build the ROC curve, we can store θ with each (tpr, fpr) pair
- Given C, we evaluate the expected cost according to

  $$\pi_0 * \text{fpr} * C(1,0) + \pi_1 * (1 - \text{tpr}) * C(0,1)$$

  where π_0 = probability of class 0, π_1 = probability of class 1
- Find best (tpr, fpr) pair and use corresponding threshold θ

End of Interlude 2

- Hand and Till show how to generalize the ROC curve to problems with multiple classes
- They also provide a confidence interval for AUC

Outline

- Cost-Sensitive Learning
  - Problem Statement; Main Approaches
- Preliminaries
  - Standard Form for Cost Matrices
  - Evaluating CSL Methods
- Costs known at learning time
- Costs unknown at learning time
- Variations and Open Problems

Two Learning Problems

- Problem 1: C known at learning time
- Problem 2: C not known at learning time (only becomes available at classification time)
  - Learned classifier should work well for a wide range of C's

Learning with known C

- Goal: Given a set of training examples $\{(x_i, y_i)\}$ and a cost matrix C,
- Find a classifier h that minimizes the expected misclassification cost on new data points (x^*, y^*)

Two Strategies

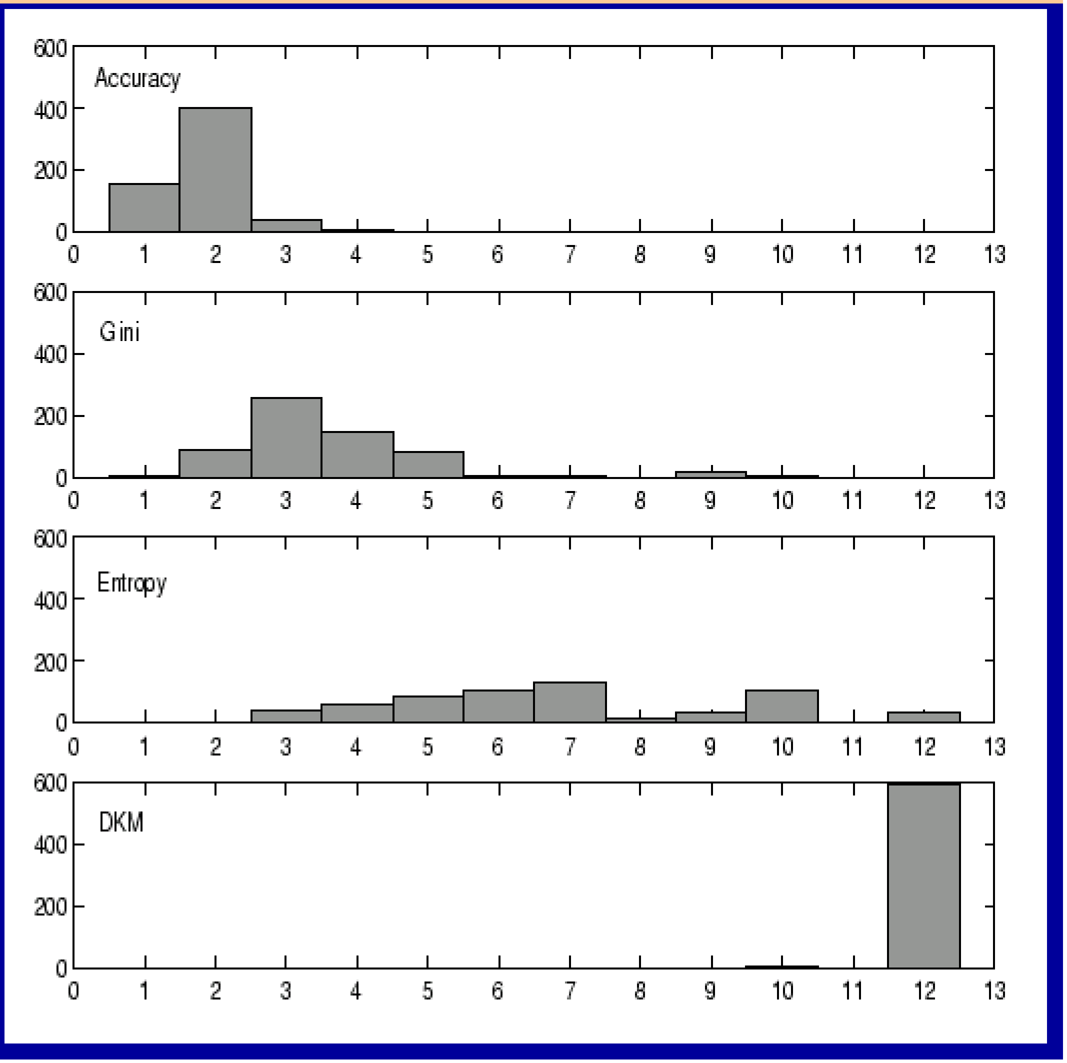
- Modify the inputs to the learning algorithm to reflect C
- Incorporate C into the learning algorithm

Strategy 1: Modifying the Inputs

- If there are only 2 classes and the cost of a false positive error is λ times larger than the cost of a false negative error, then we can put a weight of λ on each negative training example
- $\lambda = C(1,0) / C(0,1)$
- Then apply the learning algorithm as before

Some algorithms are insensitive to instance weights

- Decision tree splitting criteria are fairly insensitive (Holte, 2000)



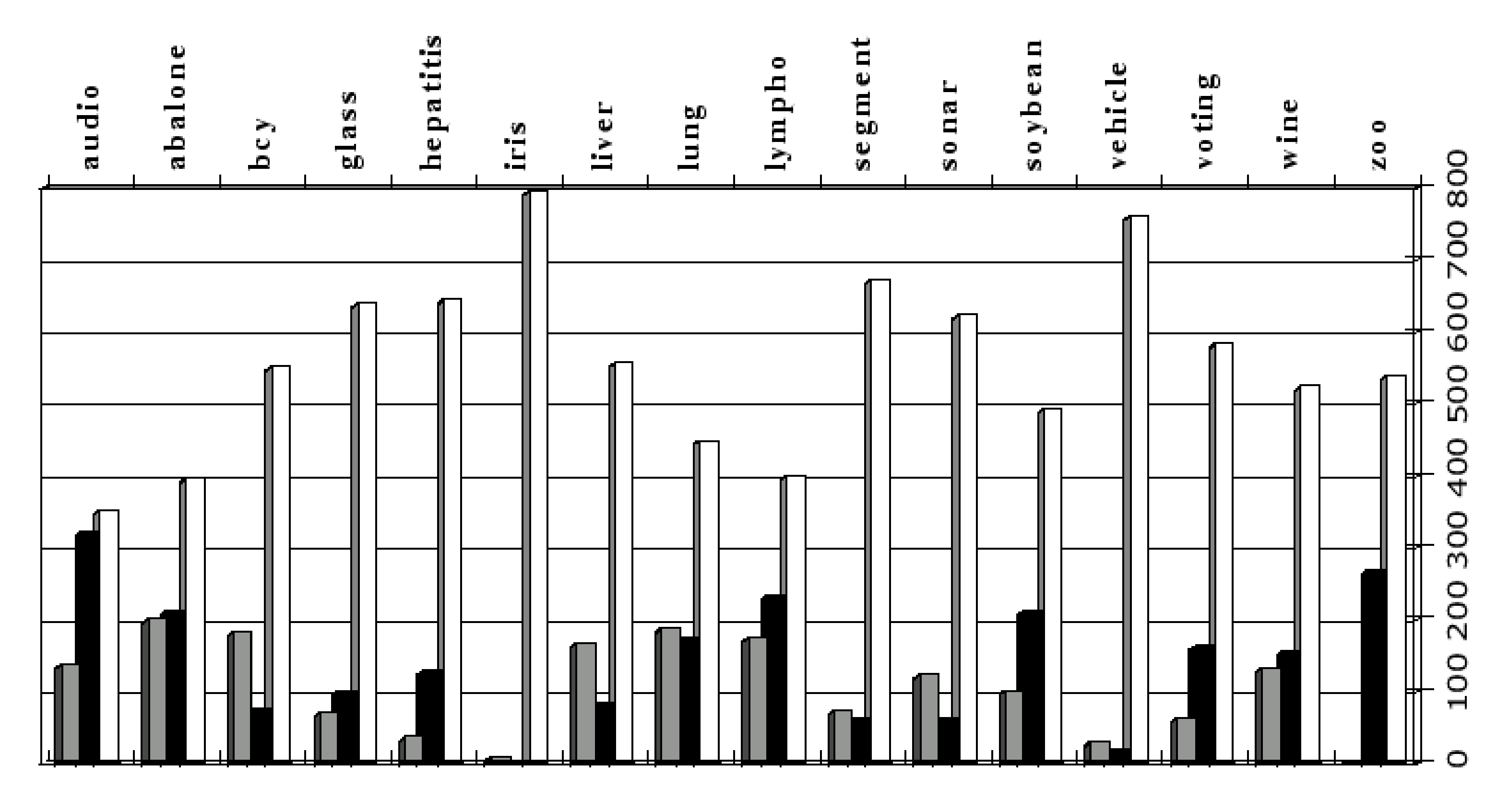
Setting λ By Class Frequency

- Set λ / $1/n_k$, where n_k is the number of training examples belonging to class k
- This equalizes the effective class frequencies
- Less frequent classes tend to have higher misclassification cost

Setting λ by Cross-validation

- Better results are obtained by using cross-validation to set λ to minimize the expected error on the validation set
- The resulting λ is usually more extreme than C(1,0)/C(0,1)
- Margineantu applied Powell's method to optimize λ_k for multi-class problems

Comparison Study



Grey: CV λ wins; Black: ClassFreq wins; White: tie

800 trials (8 cost models * 10 cost matrices * 10 splits)

Conclusions from Experiment

- Setting λ according to class frequency is cheaper gives the same results as setting λ by cross validation
- Possibly an artifact of our cost matrix generators

Strategy 2: Modifying the Algorithm

- Cost-Sensitive Boosting
- C can be incorporated directly into the error criterion when training neural networks (Kukar & Kononenko, 1998)

Cost-Sensitive Boosting (Ting, 2000)

- Adaboost ("confidence weighted")
  - Initialize $w_i = 1/N$
  - Repeat
    - Fit h_t to weighted training data
    - Compute $\varepsilon_t = \Sigma_i\, y_i\, h_t(x_i)\, w_i$
    - Set $\alpha_t = ½ * \ln (1 + \varepsilon_t)/(1 - \varepsilon_t)$
    - $w_i := w_i * \exp(-\alpha_t\, y_i\, h_t(x_i))/Z_t$
  - Classify using $\mathrm{sign}(\Sigma_t\, \alpha_t\, h_t(x))$

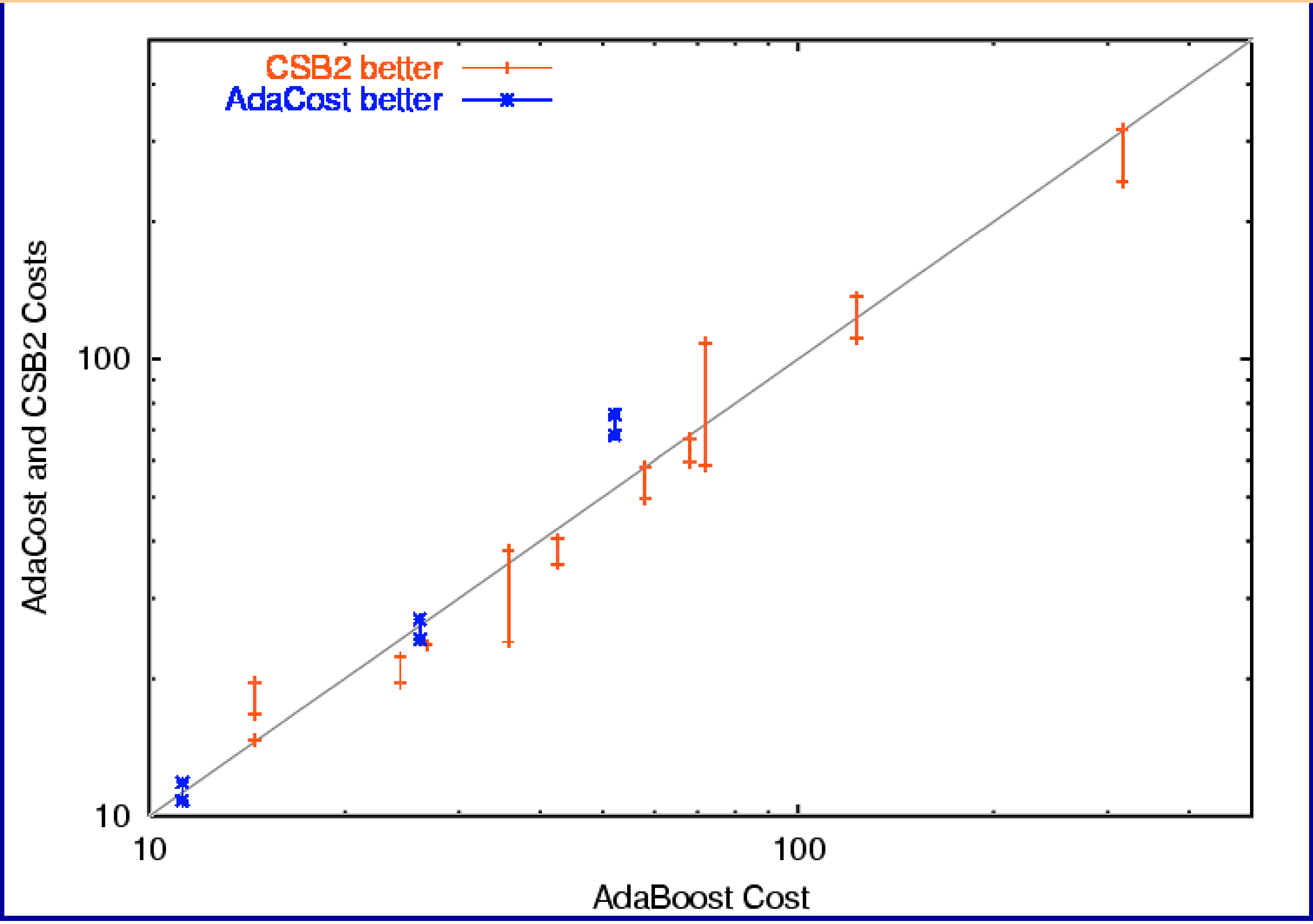
Three Variations

- Training examples of the form (x_i, y_i, c_i), where c_i is the cost of misclassifying x_i
- AdaCost (Fan et al., 1998)
  - $w_i := w_i * \exp(-\alpha_t\, y_i\, h_t(x_i)\, \beta_i)/Z_t$
  - $\beta_i = ½ * (1 + c_i)$ if error
    $= ½ * (1 - c_i)$ otherwise
- CSB2 (Ting, 2000)
  - $w_i := \beta_i\, w_i * \exp(-\alpha_t\, y_i\, h_t(x_i))/Z_t$
  - $\beta_i = c_i$ if error
    $= 1$ otherwise
- SSTBoost (Merler et al., 2002)
  - $w_i := w_i * \exp(-\alpha_t\, y_i\, h_t(x_i)\, \beta_i)/Z_t$
  - $\beta_i = c_i$ if error
  - $\beta_i = 2 - c_i$ otherwise
  - $c_i = w$ for positive examples; $1 - w$ for negative examples

Additional Changes

- Initialize the weights by scaling the costs c_i
  - $w_i = c_i / \Sigma_j c_j$
- Classify using “confidence weighting”
  - Let $F(x) = \Sigma_t \alpha_t h_t(x)$ be the result of boosting
  - Define $G(x,k) = F(x)$ if $k = 1$ and $-F(x)$ if $k = -1$
  - predicted $y = \text{argmin}_i \Sigma_k G(x,k) C(i,k)$

Experimental Results:
(14 data sets; 3 cost ratios; Ting, 2000)



Open Question

- CSB2, AdaCost, and SSTBoost were developed by making *ad hoc* changes to AdaBoost
- Opportunity: Derive a cost-sensitive boosting algorithm using the ideas from LogitBoost (Friedman, Hastie, Tibshirani, 1998) or Gradient Boosting (Friedman, 2000)
- Friedman's MART includes the ability to specify C (but I don't know how it works)

Outline

- Cost-Sensitive Learning
  - Problem Statement; Main Approaches
- Preliminaries
  - Standard Form for Cost Matrices
  - Evaluating CSL Methods
- Costs known at learning time
- Costs unknown at learning time
- Variations and Open Problems

Learning with Unknown C

- Goal: Construct a classifier h(x,C) that can accept the cost function at run time and minimize the expected cost of misclassification errors wrt C
- Approaches:
  - Learning to estimate P(y|x)
  - Learn a "ranking function" such that $f(x_1) > f(x_2)$ implies $P(y=1|x_1) > P(y=1|x_2)$

Learning Probability Estimators

- Train h(x) to estimate $P(y=1|x)$
- Given C, we can then apply the decision rule:
  - $y' = \text{argmin}_i \sum_k P(y=k|x)\, C(i,k)$

Good Class Probabilities from Decision Trees

- Probability Estimation Trees
- Bagged Probability Estimation Trees
- Lazy Option Trees
- Bagged Lazy Option Trees

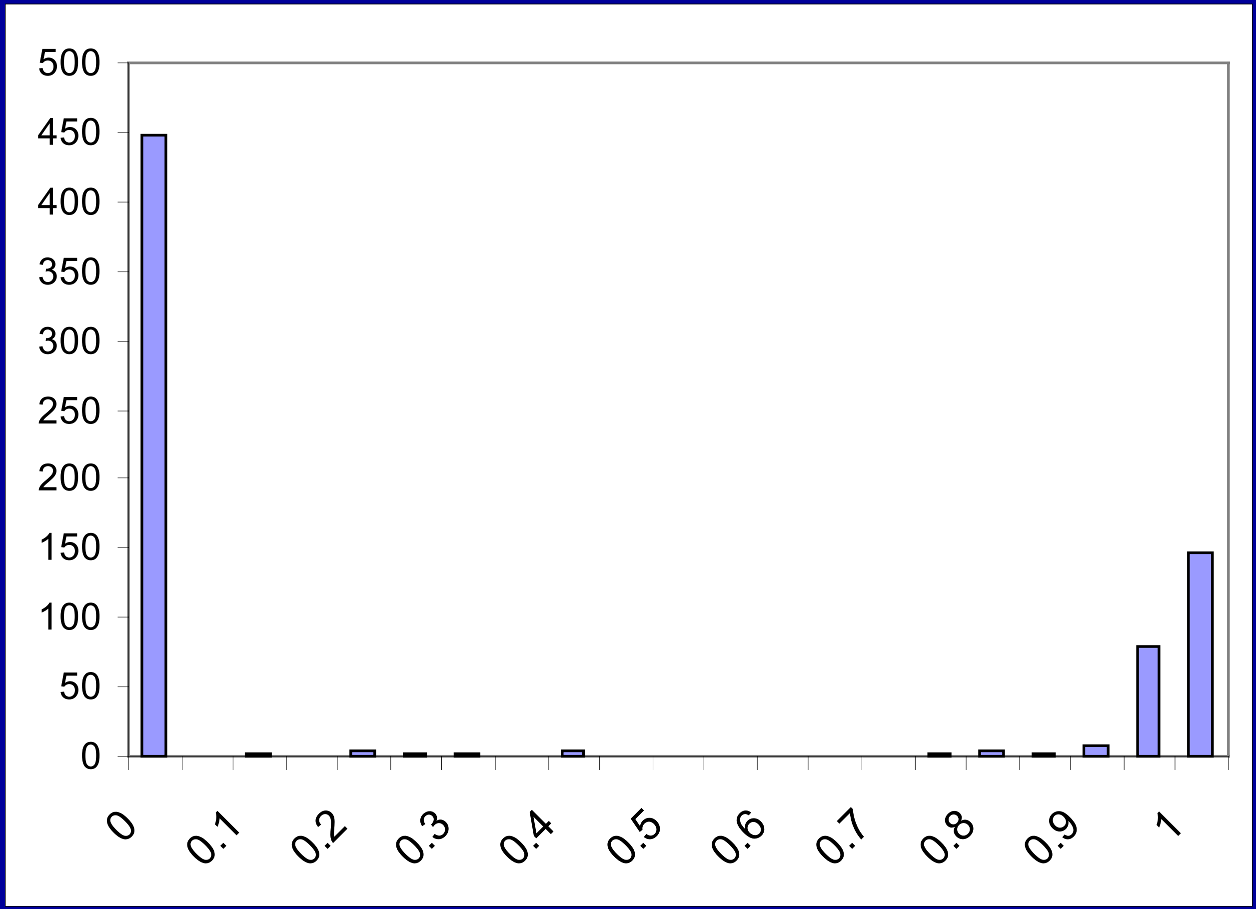
Causes of Poor Decision Tree Probability Estimates

- Estimates in leaves are based on a small number of examples (nearly pure)
- Need to sub-divide “pure” regions to get more accurate probabilities

Probability Estimates are Extreme

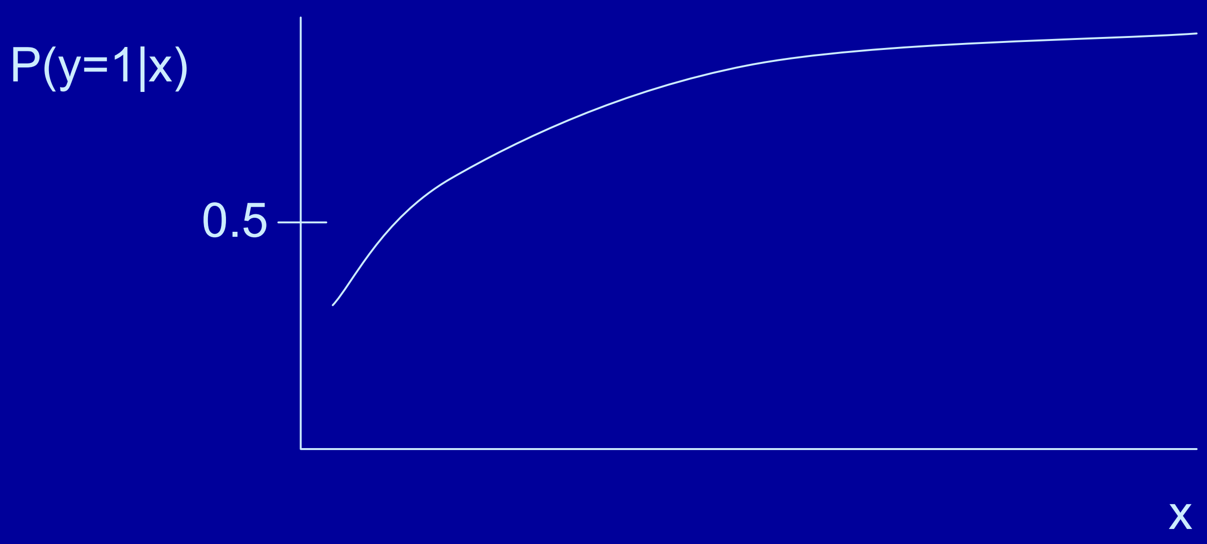
Single decision tree;

700 examples



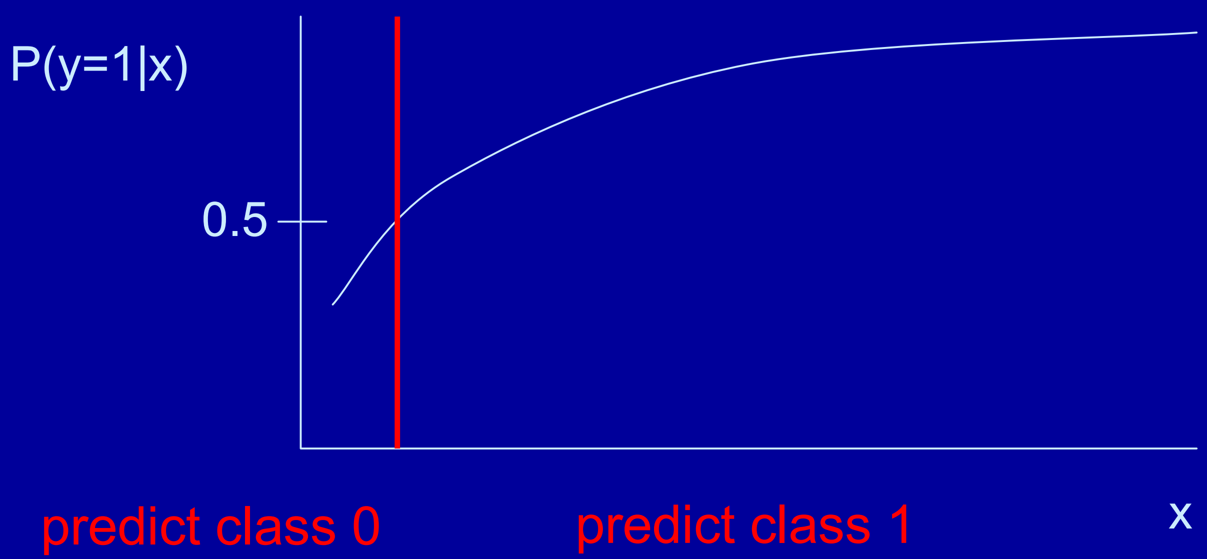
Need to Subdivide “Pure” Leaves

Consider a region of the feature space X. Suppose P(y=1|x) looks like this:



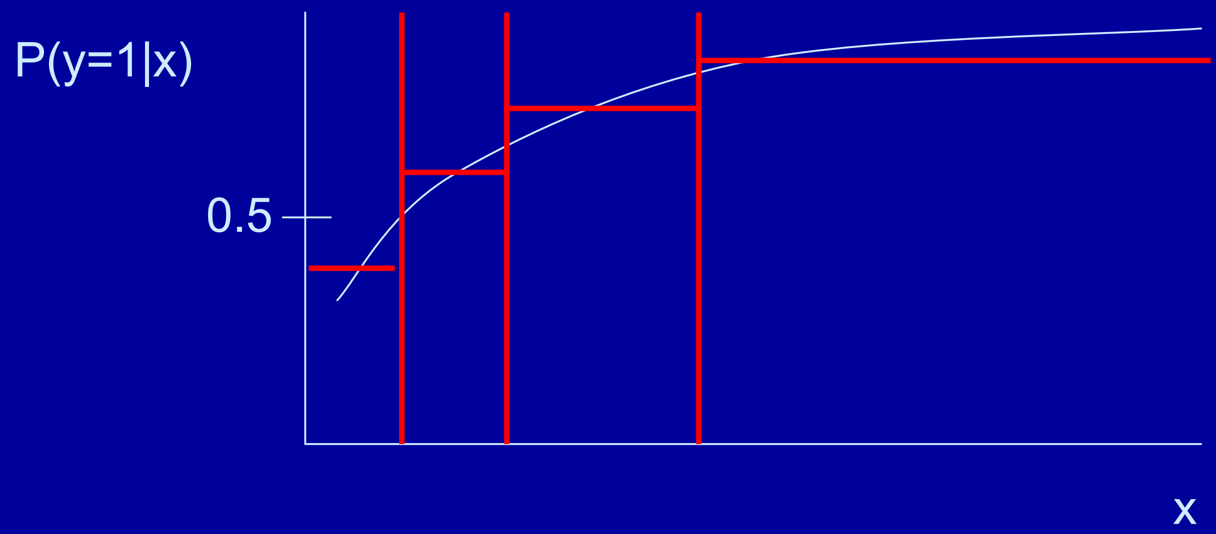
Probability Estimation versus Decision-making

A simple CLASSIFIER will introduce one split



Probability Estimation versus Decision-making

A PROBABILITY ESTIMATOR will introduce multiple splits, even though the decisions would be the same



Probability Estimation Trees
(Provost & Domingos, in press)

- C4.5
  - Prevent extreme probabilities:
    - Laplace Correction in the leaves

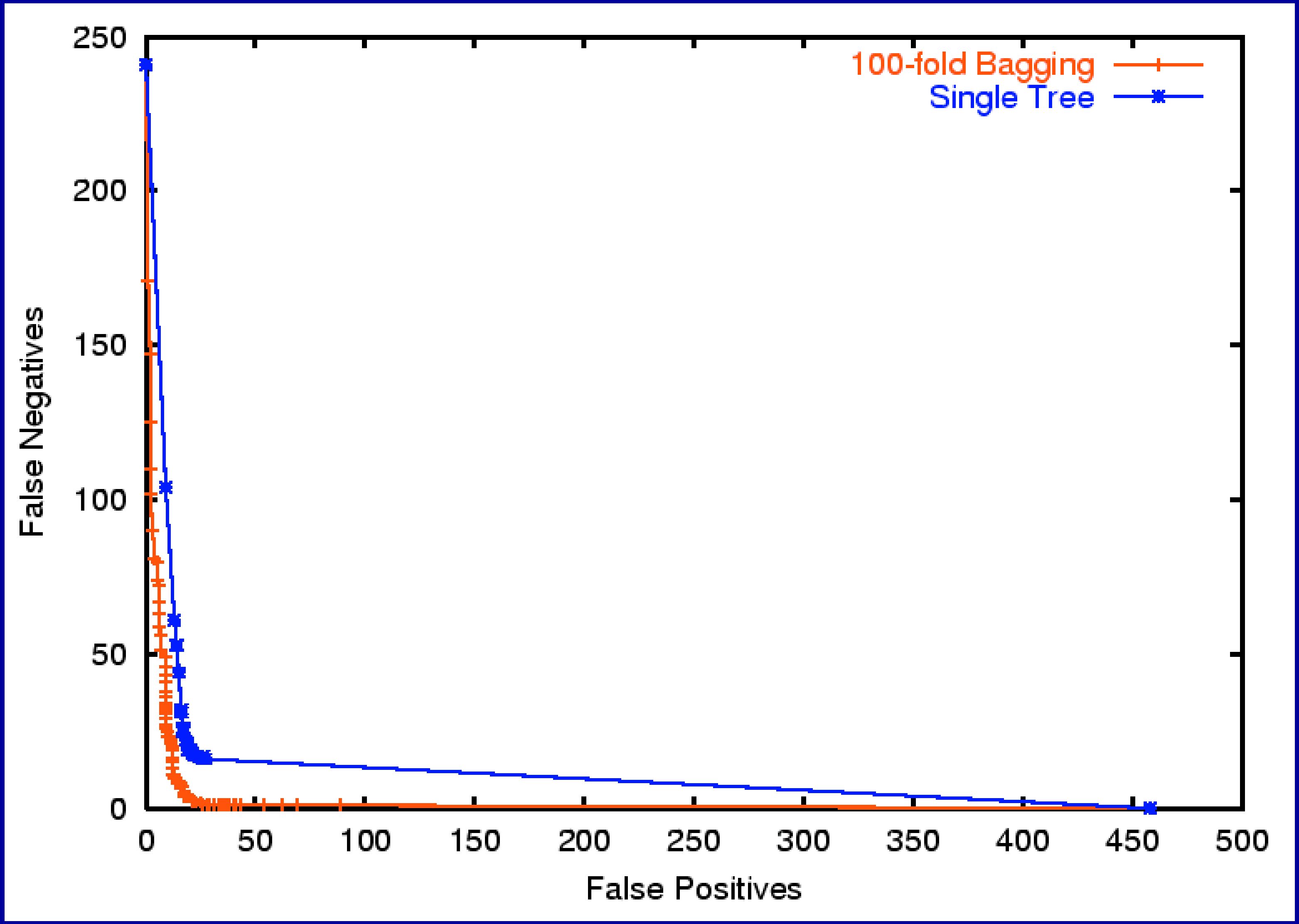
      $P(y=k|x) = (n_k + 1/K) / (n + 1)$
  - Need to subdivide:
    - no pruning
    - no "collapsing"

Bagged PETs

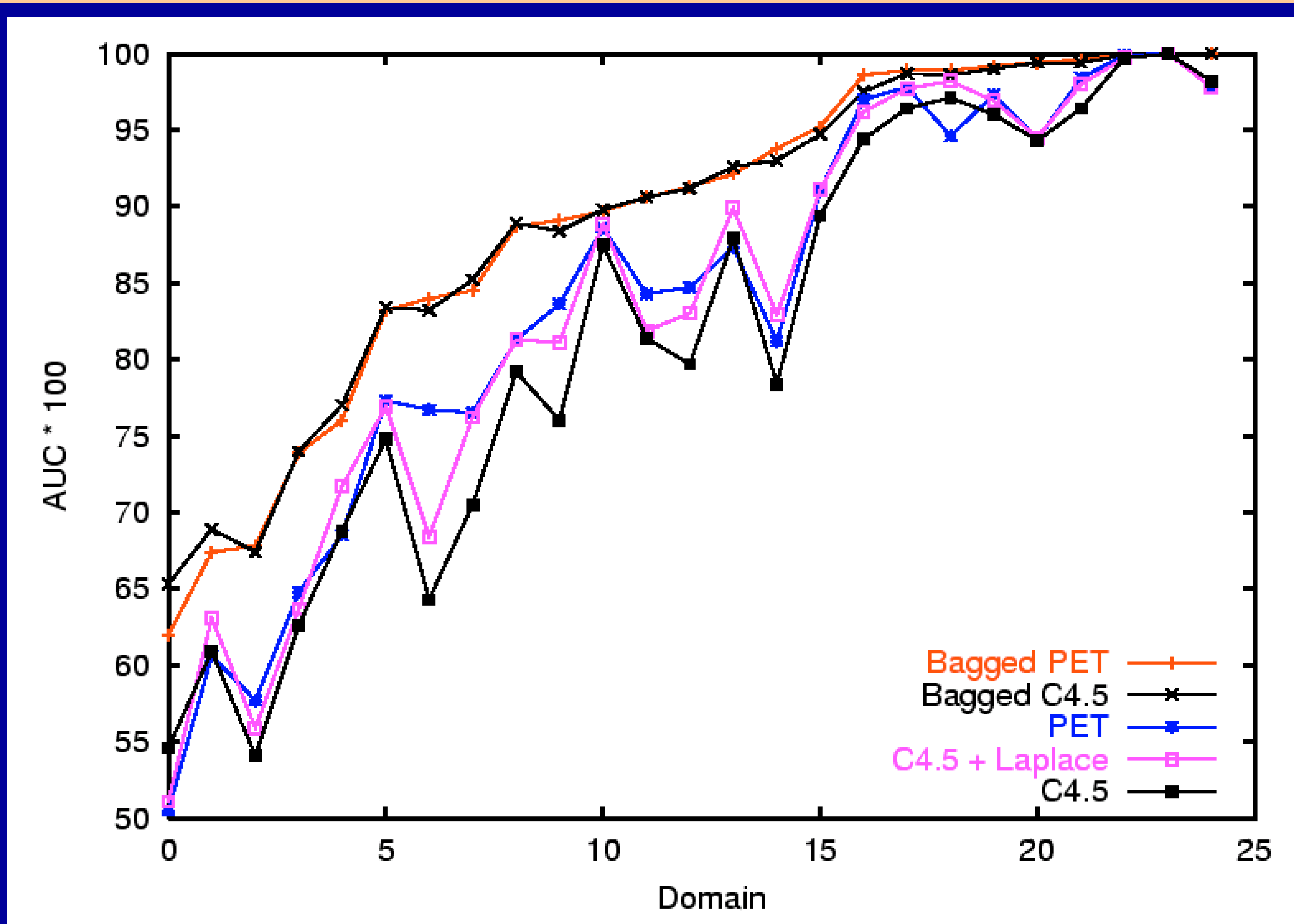
- Bagging helps solve the second problem
- Let $\{h_1, \ldots, h_B\}$ be the bag of PETs such that $h_b(x) = P(y=1|x)$

  estimate $P(y=1|x) = 1/B * \Sigma_b h_b(x)$

ROC: Single tree versus 100-fold bagging



AUC for 25 Irvine Data Sets
(Provost & Domingos, in press)



Notes

- Bagging consistently gives a huge improvement in the AUC
- The other factors are important if bagging is NOT used:
  - No pruning/collapsing
  - Laplace-corrected estimates

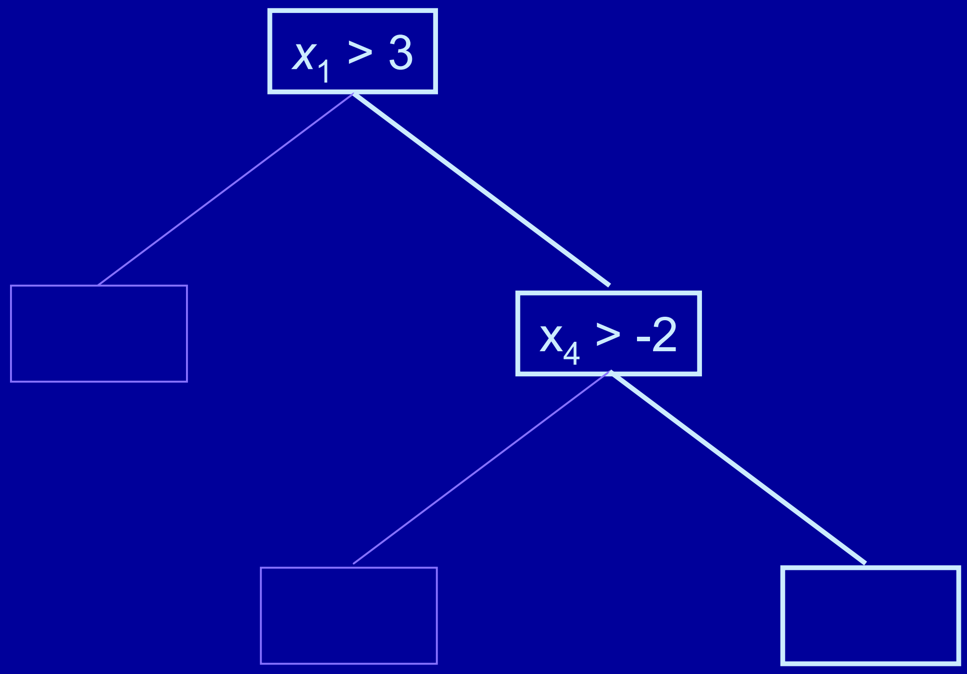
Lazy Trees

- Learning is delayed until the query point x* is observed
- An ad hoc decision tree (actually a rule) is constructed just to classify x*

Growing a Lazy Tree
(Friedman, Kohavi, Yun, 1985)

Only grow the branches corresponding to x*

Choose splits to make these branches "pure"

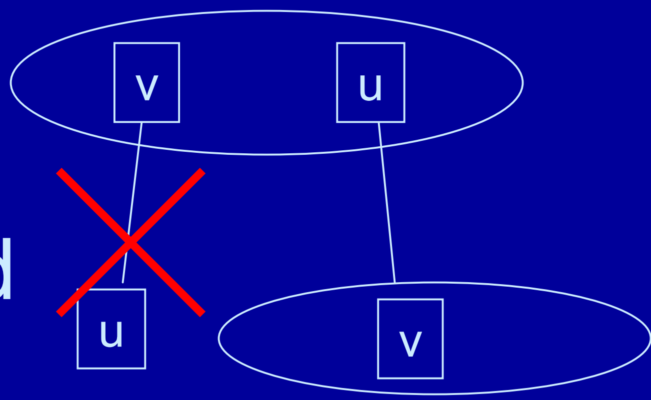


Option Trees
(Buntine, 1985; Kohavi & Kunz, 1997)

- Expand the Q best candidate splits at each node
- Evaluate by voting these alternatives

Lazy Option Trees
(Margineantu & Dietterich, 2001)

- Combine Lazy Decision Trees with Option Trees
- Avoid duplicate paths (by disallowing split on u as child of option v if there is already a split v as a child of u):



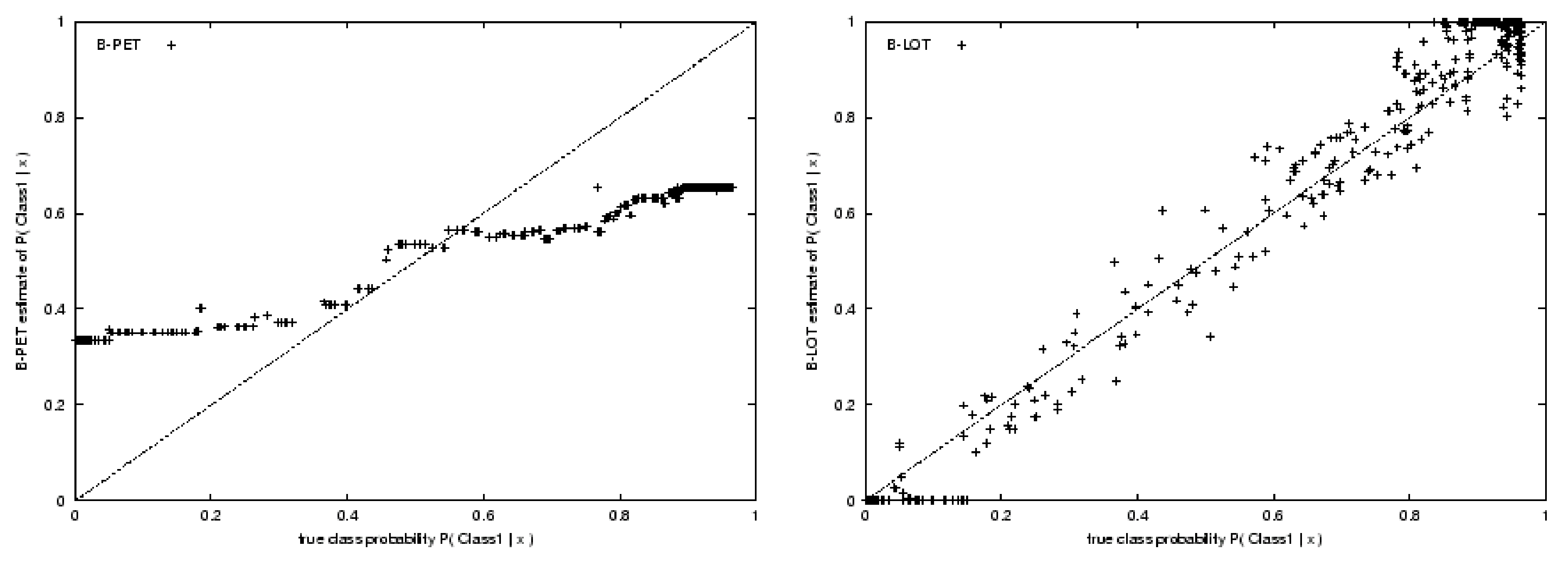
Bagged Lazy Option Trees (B-LOTs)

- Combine Lazy Option Trees with Bagging (expensive!)

Comparison of B-PETs and B-LOTs

- Overlapping Gaussians
- Varying amount of training data and minimum number of examples in each leaf (no other pruning)

B-PET vs B-LOT



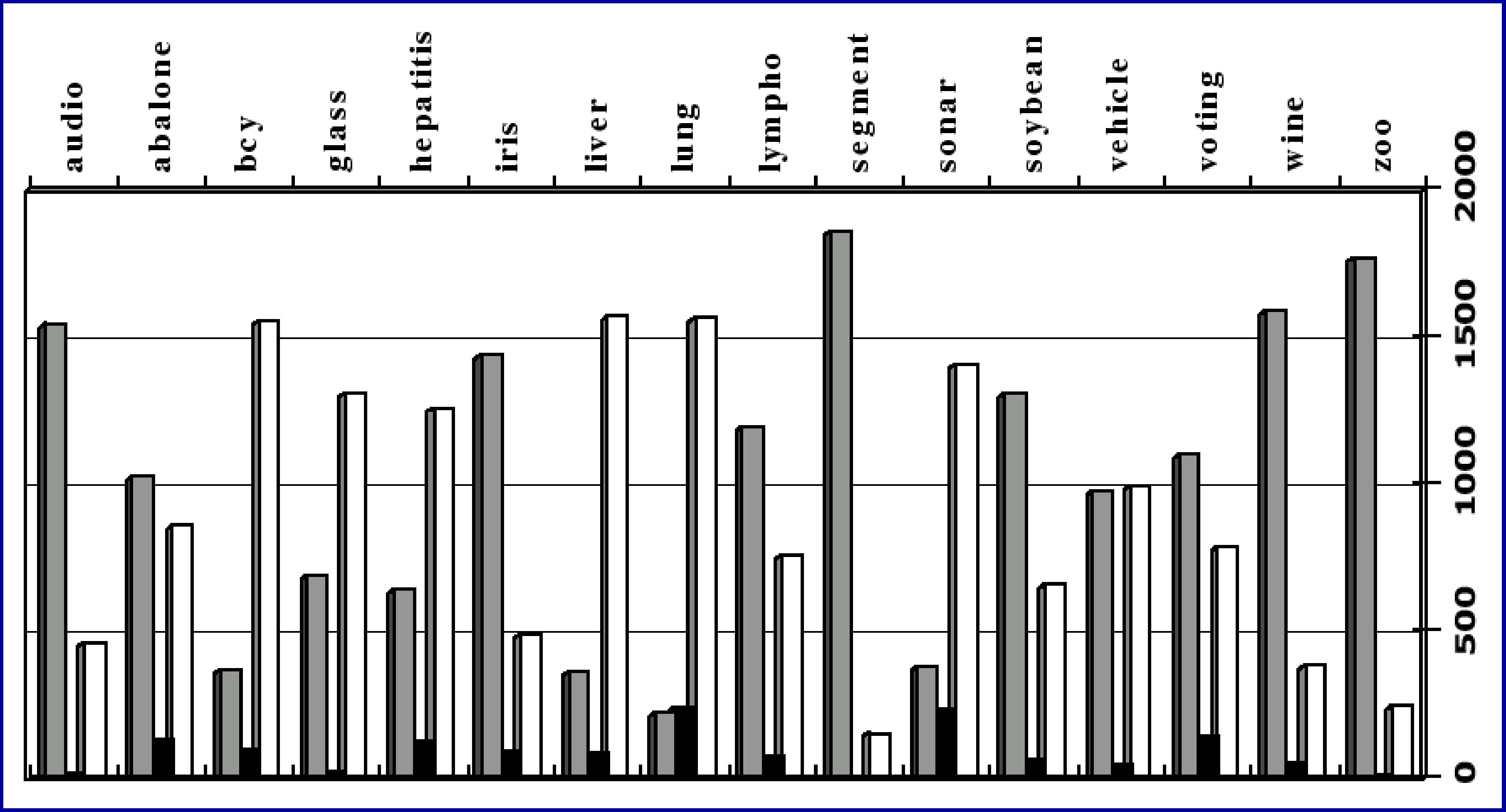
Bagged PETs

Bagged LOTs

Bagged PETs give better ranking

Bagged LOTs give better calibrated probabilities

B-PETs vs B-LOTs



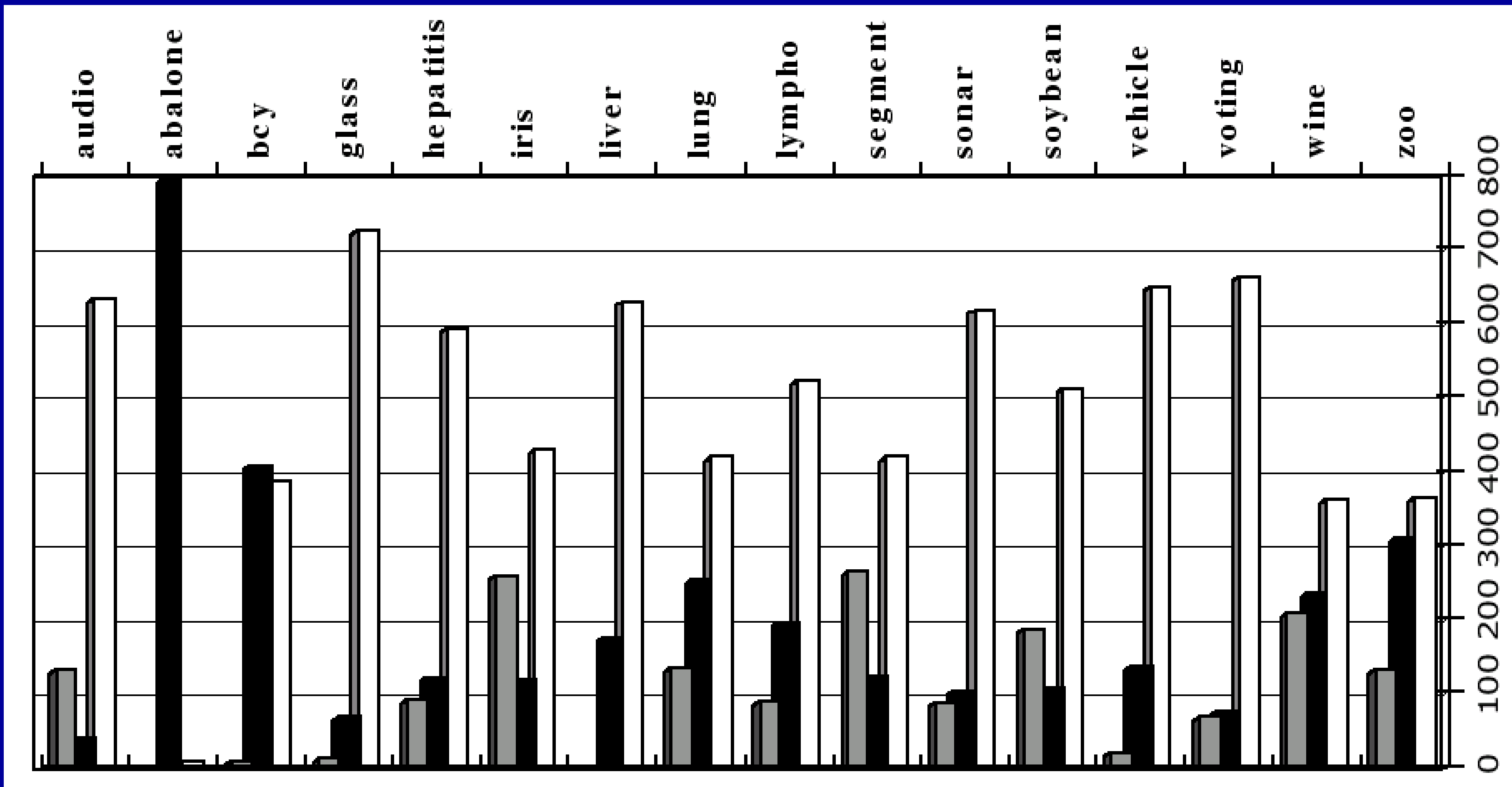
Grey: B-LOTs win Black: B-PETs win White: Tie

Test favors well-calibrated probabilities

Open Problem: Calibrating Probabilities

- Can we find a way to map the outputs of B-PETs into well-calibrated probabilities?
  - Post-process via logistic regression?
  - Histogram calibration is crude but effective (Zadrozny & Elkan, 2001)

Comparison of Instance-Weighting and Probability Estimation



Black: B-PETs win; Grey: ClassFreq wins; White: Tie

An Alternative: Ensemble Decision Making

- Don't estimate probabilities: compute decision thresholds and have ensemble vote!
- Let $\rho = C(0,1) / [C(0,1) + C(1,0)]$

  Classify as class 0 if $P(y=0|x) > \rho$
- Compute ensemble $h_1, \ldots, h_B$ of probability estimators

  Take majority vote of $h_b(x) > \rho$

Results
(Margineantu, 2002)

- On KDD-Cup 1998 data (Donations), in 100 trials, a random-forest ensemble beats B-PETs 20% of the time, ties 75%, and loses 5%
- On Irvine data sets, a bagged ensemble beats B-PETs 43.2% of the time, ties 48.6%, and loses 8.2% (averaged over 9 data sets, 4 cost models)

Conclusions

- Weighting inputs by class frequency works surprisingly well
- B-PETs would work better if they were well-calibrated
- Ensemble decision making is promising

Outline

- Cost-Sensitive Learning
  - Problem Statement; Main Approaches
- Preliminaries
  - Standard Form for Cost Matrices
  - Evaluating CSL Methods
- Costs known at learning time
- Costs unknown at learning time
- Open Problems and Summary

Open Problems

- Random forests for probability estimation?
- Combine example weighting with ensemble methods?
- Example weighting for CART (Gini)
- Calibration of probability estimates?
- Incorporation into more complex decision-making procedures, e.g. Viterbi algorithm?

Summary

- Cost-sensitive learning is important in many applications
- How can we extend “discriminative” machine learning methods for cost-sensitive learning?
- Example weighting: ClassFreq
- Probability estimation: Bagged LOTs
- Ranking: Bagged PETs
- Ensemble Decision-making

Bibliography

- Buntine, W. 1990. *A theory of learning classification rules.* Doctoral Dissertation. University of Technology, Sydney, Australia.
- Drummond, C., Holte, R. 2000. Exploiting the Cost (In)sensitivity of Decision Tree Splitting Criteria. ICML 2000. San Francisco: Morgan Kaufmann.
- Friedman, J. H. 1999. Greedy Function Approximation: A Gradient Boosting Machine. IMS 1999 Reitz Lecture. Tech Report, Department of Statistics, Stanford University.
- Friedman, J. H., Hastie, T., Tibshirani, R. 1998. Additive Logistic Regression: A Statistical View of Boosting. Department of Statistics, Stanford University.
- Friedman, J., Kohavi, R., Yun, Y. 1996. Lazy decision trees. Proceedings of the Thirteenth National Conference on Artificial Intelligence. (pp. 717-724). Cambridge, MA: AAAI Press/MIT Press.

Bibliography (2)

- Hand, D., and Till, R. 2001. A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. *Machine Learning*, 45(2): 171.
- Kohavi, R., Kunz, C. 1997. Option decision trees with majority votes. *ICML-97*. (pp 161-169). San Francisco, CA: Morgan Kaufmann.
- Kukar, M. and Kononenko, I. 1998. Cost-sensitive learning with neural networks. *Proceedings of the European Conference on Machine Learning.* Chichester, NY: Wiley.
- Margineantu, D. 1999. Building Ensembles of Classifiers for Loss Minimization, *Proceedings of the 31st Symposium on the Interface: Models, Prediction, and Computing.*
- Margineantu, D. 2001. *Methods for Cost-Sensitive Learning.* Doctoral Dissertation, Oregon State University.

Bibliography (3)

- Margineantu, D. 2002. Class probability estimation and cost-sensitive classification decisions. *Proceedings of the European Conference on Machine Learning.*
- Margineantu, D. and Dietterich, T. 2000. Bootstrap Methods for the Cost-Sensitive Evaluation of Classifiers. *ICML 2000.* (pp. 582-590). San Francisco: Morgan Kaufmann.
- Margineantu, D., Dietterich, T. G. 2002. Improved class probability estimates from decision tree models. To appear in *Lecture Notes in Statistics*. New York, NY: Springer Verlag.
- Provost, F., Domingos, P. In Press. Tree induction for probability-based ranking. To appear in *Machine Learning.* Available from Provost's home page.
- Ting, K. 2000. A comparative study of cost-sensitive boosting algorithms. *ICML 2000.* (pp 983-990) San Francisco, Morgan Kaufmann. (Longer version available from his home page.)

Bibliography (4)

- Zadrozny, B., Elkan, C. 2001. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. *ICML-2001*. (pp 609-616). San Francisco, CA: Morgan Kaufmann.