Feature Selection on Imbalanced Data

Presented by Randall Wald

Introduction

- Data mining background
- What is feature selection?
- Why feature selection matters
- Traditional approaches
- Working with imbalanced data

Data Mining

- Many instances
  - Each has features
  - Each has a class
- The goal: Create a classifier based on the features which predicts the class

Numeric vs. Nominal

- Features and classes can each be numeric or nominal
- Numeric classes have certain classifiers:
  - Regressions
  - Module Order Modeling
- Nominal classes: good/bad, red/green, categories
- Features also are numeric or nominal

Classifier: Naïve Bayes

- For each feature, what fraction of instances of each value are in each class?
- When evaluating instances, add together the probabilities of being in each class
- Assign to class based on probability

Classifier: Decision Tree

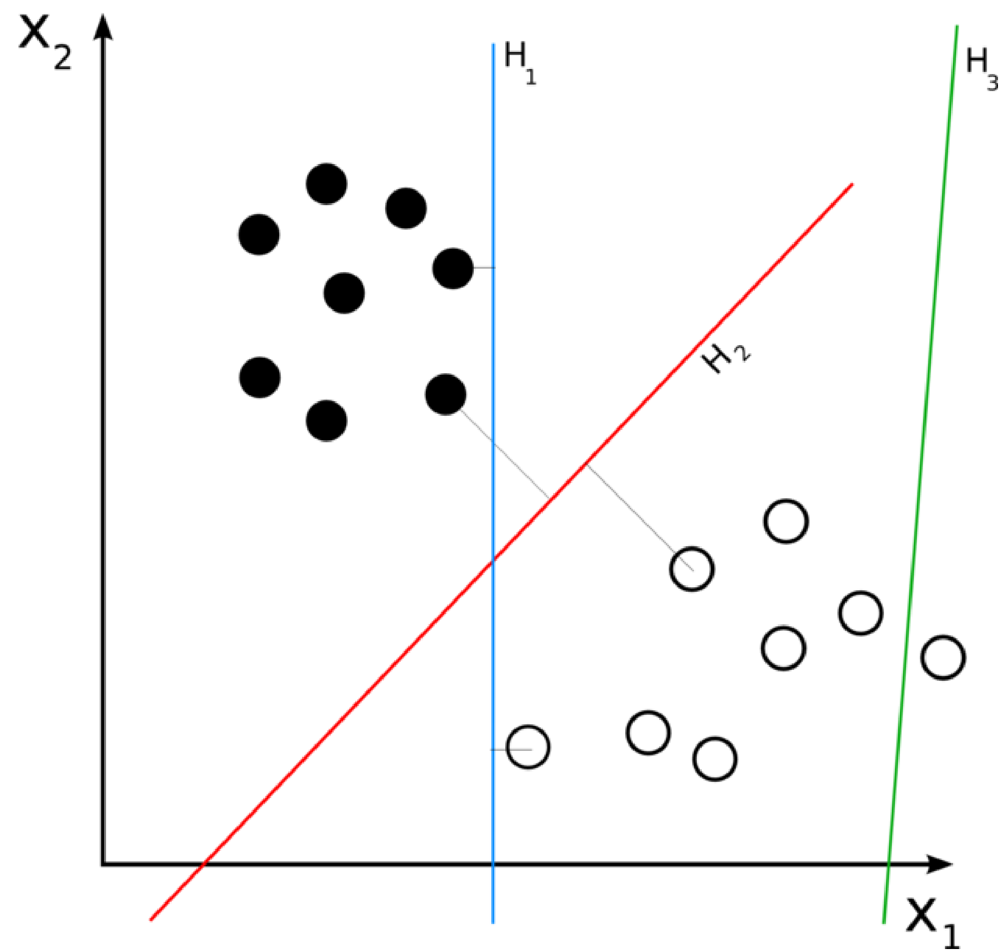
- Find feature which best predicts which class each instance is in
- Make a flow chart: instances start at the top, and go to a different child node based on what value they have for that feature
- Iterate to create further children

Classifier: Nearest Neighbor

- Decide upon a decision metric to say how far apart two instances are
  - Geometric distance
  - Manhattan distance
- For new instances, find to nearest neighbors in each class
- Assign to class with nearest neighbor

Classifier: SVMs

- Support Vector Machines
- Plot data in n-dimensional space
- Find hyperplane which best separates the data

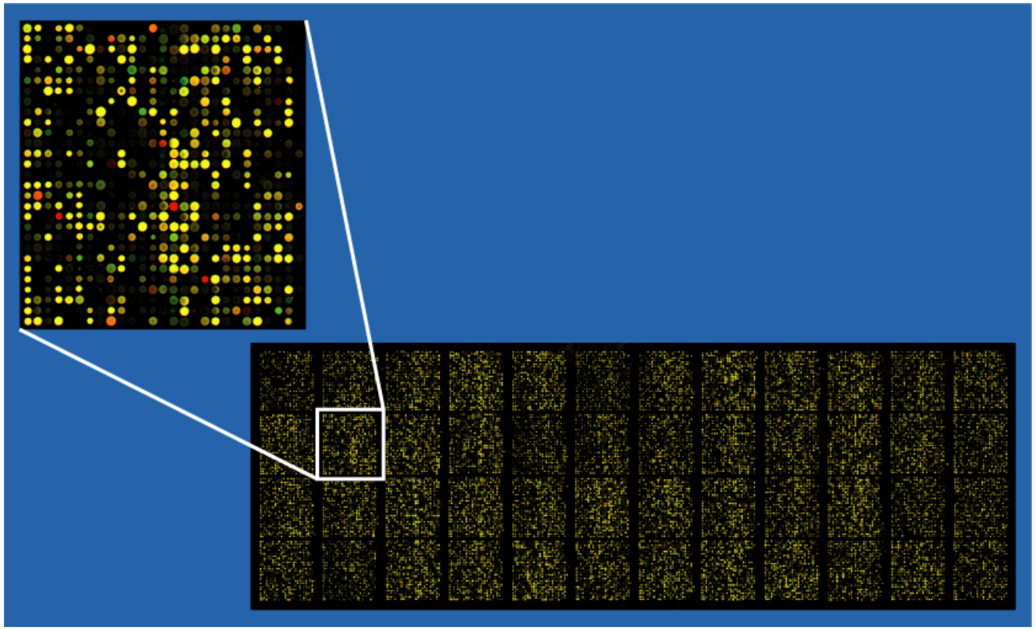


Feature Selection

- Many real-world data sets have more features than instances:
  - Gene chip data
  - Risk assessment
  - Text mining
  - Performance evaluation

Gene chips

- Thousands of wells
- Each a separate gene to test
- One sample is run against all genes
- Sample = instance
- Genes = features
- Chips are expensive, so few samples



Risk assessment

- Dangerous vs. not dangerous
  - Terrorism
  - Security
  - Financial
- Many factor to check
  - Buying habits
  - Movement patterns
  - Communication
- Few samples to work with

Text mining

- Each distinct word is a feature
  - Only two states: present and absent
- Relatively few documents, at least compared to the number of total words
- Goal: Use words to determine which documents are in various categories
- Applications to web search, etc.

Performance evaluation

- Predicting the quality of future systems based on existing systems
- Many properties to describe a system
  - Hardware specs
  - Software properties
- Few systems to compare when planing new systems

Feature Selection

- Many algorithms become computationally intensive
  - Naïve Bayes is O(n)
  - Decision Tree is $O(n^2)$ for complete tree
- Extraneous, noisy features drown out noise
  - That many features, some random ones will have a pattern
  - Hard to decide weightings

Feature Selection

- Solution: Feature selection
- Pick some subset of the features
- Only run the classifier on those features
- Results are often similar to using all features

Choosing features

- Expert knowledge
- Filter
  - Feature ranking
  - Subset evaluation
- Wrapper
- Hybrid

Expert knowledge

- Some attributes are known to matter more than others
- Experts can eliminate obviously-useless features
- Also aids identification of linked features in advance

Filter: Feature Ranking

- Perform evaluation on each feature
  - Information gain
  - Odds ratio
  - Chi-squared
  - OneR
- Rank features by performance
- Select the top few features
  - Number to use found via statistical measure or pilot study

Information gain

- Amount of entropy the data have with and without the feature in question
- Shanon entropy: How many bits it would take to encode the data in question
- Intuitively, how much we know
- If removing a feature removes information, it's a good feature

Information gain

- Calculation of entropy from a given state: e(a, b, c, ...) =

$-\log(a/n) - \log(b/n) - \log(c/n) \ldots$

- a is number of instances in class a, b is number of instances in class b, etc.
- n = a + b + c ... = total number of instances in current state

Information gain

- Calculation of entropy for a given feature:
  - For each value of that feature, calculate entropy (that is, use the earlier equation for instances with that value for that feature)
  - Add up entropies for all values, weighted by number of instances with that value

Information gain

- Information gain for a given feature
  - Calculate entropy of initial state (that is, the entire set)
  - Information gain = total entropy – entropy of that feature
- Rank features by information gain, higher is better

Odds ratio

- Only works with binary features (those which have just two values, present and absent) and binary classes (true and false)
- Ratio of the odds of a feature being present in the true instance versus those of it being in the false instances

Odds ratio

- Odds of being present in true instances = probability of being in true instances / probability of not being in true instance

= tpr / (1 – tpr)

- Odds of being present in false instances = fpr / (1 – fpr)
- Odds ratio = tpr (1 – fpr)/
  (1 – tpr) tpr

Chi-squared

- Assume that feature distribution is unrelated to class distribution
- Calculate the expected distribution of feature values
- Determine how far away from this expectation the actual feature distribution is
- Higher chi-squared is better

OneR metric

- Extremely simple classifier
  - For each value, predict the class which predominates amongst instances with that value
  - For numeric attributes, break into intervals
- Classify all instances
- Evaluate accuracy of classification
- Better classification = better feature

When to stop selection?

- Based on the statistical measure
  - Information Gain, Odds Ratio, etc. get too low to be useful
- Based on ad-hoc number
  - We only want the top 20 features
- Pilot study
  - On a smaller sample, try 5, 10, …, 75 features, and measure the performance of the resulting classifiers

Filter: Subset evaluation

- Select subsets to examine
  - Exhaustive search
  - Greedy stepwise
  - Best first
- Perform evaluation on each subset
  - Correlation-based feature selection
  - Consistency
  - Markov blanket
- Stop search when a threshold is met and use just those features

Search techniques

- Exhaustive search
  - Just try all possible subsets
  - Impractical in most situations
- Greedy stepwise
  - Start with either empty or full set
  - Add or remove best or worst feature
- Best first
  - Start with empty set
  - Add best feature
  - Compare with previous sets; if new feature doesn't help, backtrack

Correlation (CFS)

- Two competing goals
  - Features which predict (correlate) highly with the class
  - Features which do not correlate highly with one another
- Metric to evaluate performance: Pearson Correlation Coefficient

Pearson correlation

- For two variables, the variance in Y which is accounted for by the variance of X
- For k variables, more complex:

$$M_S = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}$$

- M_S is merit, r_{cf} is correlation between features and class, and r_{ff} is pairwise correlation between features

CFS

- Actual methods of calculating the r_{cf} and r_{ff} values vary
- Numerator is how well features predict class
- Denominator is how much features correlate with one another

$$M_S = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}$$

Consistency

- Only works with random or exhaustive search
- Generate new set
- If it's smaller than the current best
  - See if subset is sufficiently consistent
  - If it is, it's the new best subset
  - Print this subset
- If it's identical in size to the best
  - See if subset is sufficiently consistent
  - Print this subset

Consistency measure

- Want subsets that “make sense” as descriptions of the class
- Subsets are inconsistent if two instances with identical features have different classes
- For each set of *instances* with matching feature values, find how many are inconsistent
- # of inconsistent instances / total # of instances = inconsistency

Markov blanket

- Based on conditional probabilities: probability of being in a given class based on what we know about the features
- Want to remove features which do not add any additional information to the current feature set

Markov blanket

- A feature F_i is “conditionally independent” of some other subset C of features if the conditional probabilities for C + F_i are identical to those for C alone
- In this case, F_i is an unnecessary feature, since it adds no new information

Markov blanket

- In practice, finding conditional probabilities for large sets of features is hard
- Instead, test with many small subsets of features

Markov blanket

- Start with full set of features
- For each feature F_i, find feature subset of size k which best correlates with F_i (not including F_i itself)
- Find how conditionally independent F_i is from that subset
- After trying all F_i, remove the worst one and iterate on those remaining

Wrapper

- Select subsets (same as subset evaluation)
- Use a classifier with just those features; test performance
- Performance of classifier on subset is quality of subset
- Stop search when a threshold is met

Wrapper

- “Wrapped” classifier is usually identical to the target classifier, but not always
- Different performance metrics can rank the subsets

Comparisons

- Feature ranking is much cheaper than subset evaluation or wrapper
- Subset evaluation and wrapper avoid the chance that two highly-ranked features don't add new information together
- Wrapper is tightly bound to the target classifier, which is good and bad

Hybrid approaches

- Some feature selection methods excel at removing completely useless features
- Others are better at comparing among important features to determine most important
- Solution: apply more than one filter

Hybrid approaches

- Especially useful for using a feature ranker to filter out the really bad features, then subset selection or a wrapper on the remainder
- Example: Odds ratio followed by Markov blanket

Feature Selection

- Not all approaches work with all classifiers
- Many parameters that need to be varied to maximize performance
- Overfitting a perennial risk

Imbalanced data

- Frequently interested in binary classes
  - Sick/healthy
  - Risky/safe
  - Fault-prone/not fault-prone
- Many more examples of one class than the other
- Care more about the minority class

Imbalanced data

- Many classifiers seek to balance Type I and Type II errors
  - Type I: false positives
  - Type II: false negatives
- For imbalanced data, this causes all positive instances to be classified as negative instances
  - Even if all positive instances classified as negative, still small compared to any false positives

Imbalanced data

- Some classifiers can be modified to handle this
  - Naïve Bayes: require lower probability to find positive class
  - Nearest Neighbor: give positive class multiplier on distance
- These approaches require fine-tuning

Imbalanced data

- The same data sets which have many features often have imbalanced data
- Feature selection techniques also have problems on such data
  - Information Gain and Odds Ratio favor negative features
  - Subset evaluation and wrapper usually favor subsets which balance Type I and Type II error

Imbalanced data

- Solutions:
  - Use different feature ranking metrics
    - Bi-normal separation
    - FAST
  - Sampling
    - Effectiveness varies based on feature selection and classifier

Bi-normal separation

- BNS: feature ranking method designed for imbalanced data
- Only works for binary features
- Thus, useful for text mining and evaluating classifiers

Bi-normal separation

- Find # of false positives, # of true positives, total # of positives, total # of negatives
- #tp / total positives = true positive rate
- #fp / total negatives = false positive rate
- Find inverse normal CDF for fpr and tpr
- Difference is BNS

Bi-normal separation

Figure 1. Two views of Bi-Normal Separation using the Normal probability distribution: (left) Separation of thresholds. (right) Separation of curves (ROC analysis).

- Assumes that fpr and tpr are fixed values
- Probability of feature being present is normally distributed

Bi-normal separation

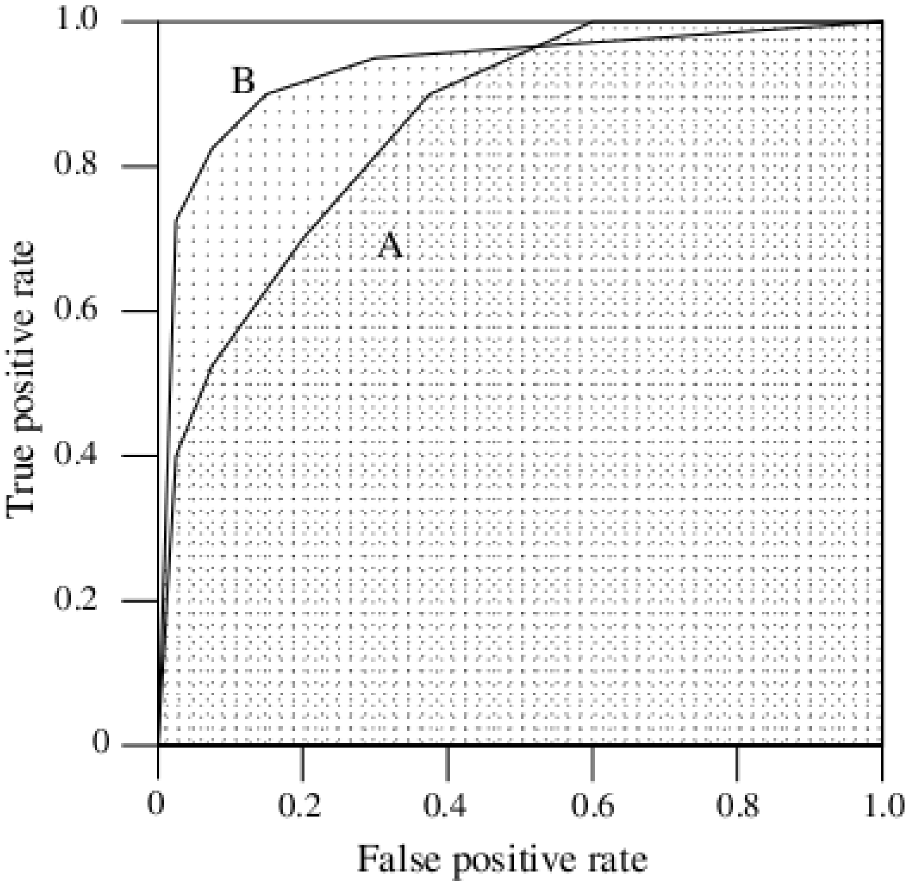
Figure 1. Two views of Bi-Normal Separation using the Normal probability distribution: (left) Separation of thresholds. (right) Separation of curves (ROC analysis).

- Difference between tpr and fpr is real classification power of feature
- Selects both positive and negative features

FAST

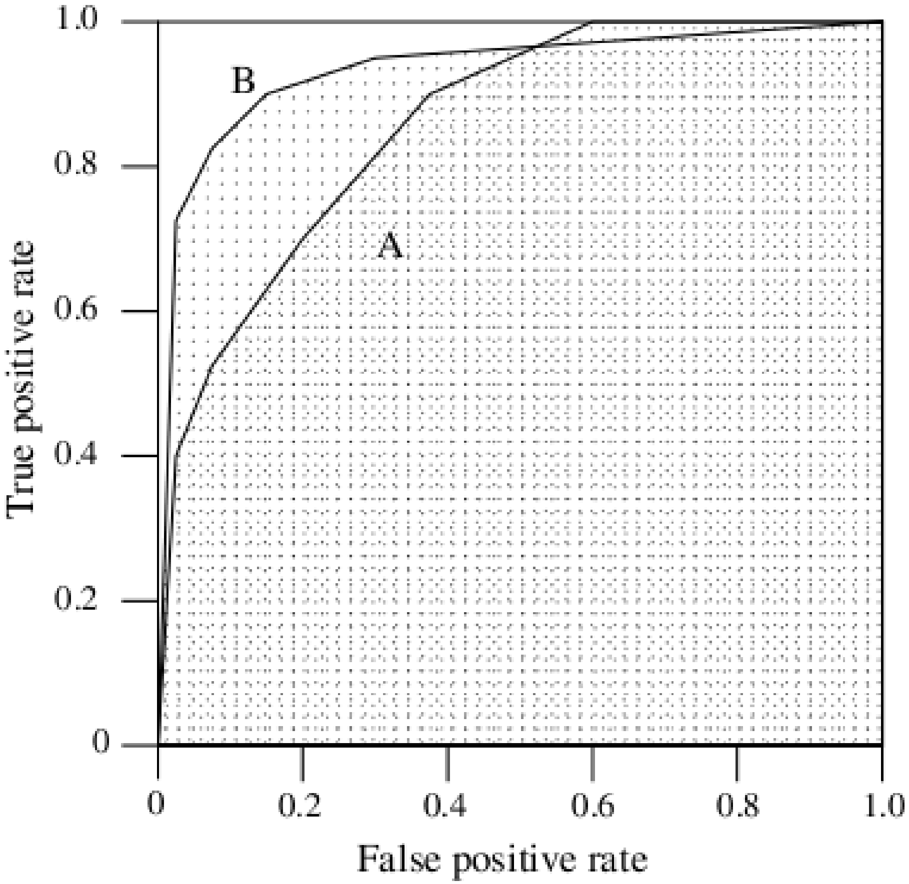
- Feature Assessment by Sliding Thresholds: approximate area under ROC curve for threshold-based classifiers run on each feature

ROC curves



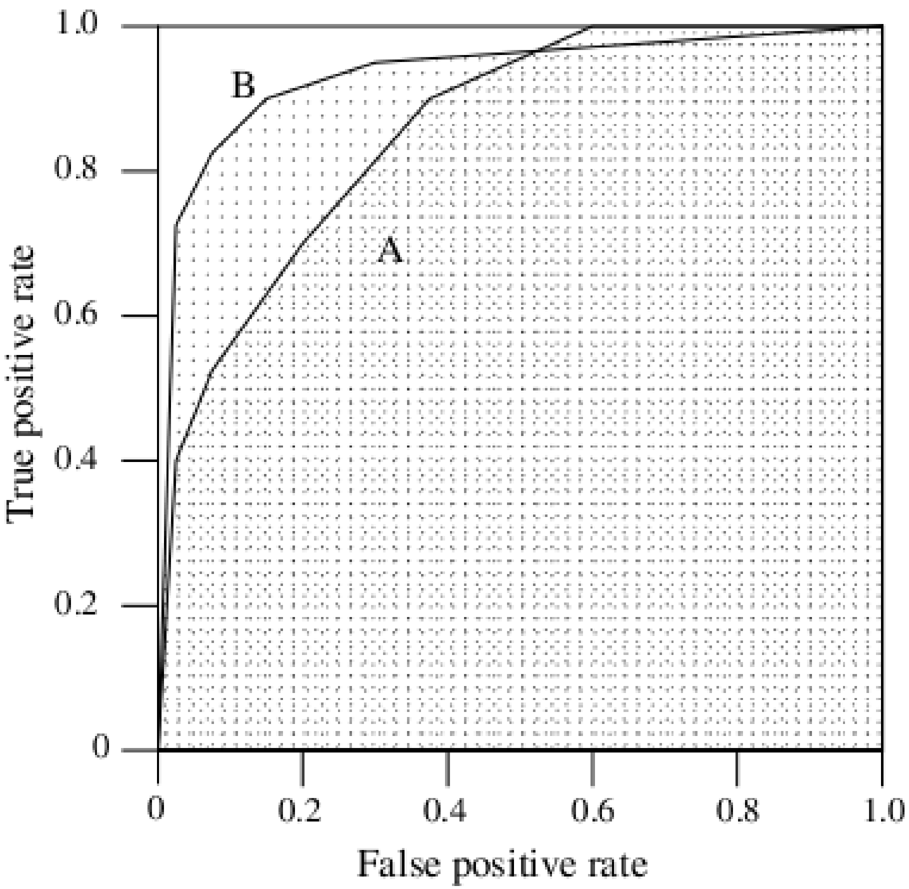
- tpr vs fpr
- (0,0) corresponds to always guessing negative
- (1,1) is always guessing positive
- x=y line is random guess

ROC curves



- ROC works on ranges of classifiers
  - Naïve Bayes with sliding thresholds
  - OneR with different thresholds

ROC curves



- Area under ROC curve corresponds to “goodness” of models
- B is better than A in this example

FAST

- Create a collection of models for each feature:
  - 10 threshold models
  - Each threshold is the mean of one quantile
- Find ROC for each feature
  - If you get an ROC of < 0.5, take the value (1 – ROC)
- ROC value is quality of feature

FAST

- Why use ROC and sliding thresholds?
  - Hard to pick one good static threshold
  - ROC is the final performance metric, so might as well use for feature selection
- FAST only uses 10 thresholds, but should approximate the use-all-possible-thresholds case

Sampling

- Modify the data set to not be imbalanced anymore
  - Oversampling: duplicate instances of positive class
  - Undersampling: remove instances of the negative class
- Randomized sampling, or focused on strengthening the boundary
  - Oversample near boundary
  - Undersample far from boundary

Sampling

- Oversampling can be applied both to feature selection and to building a classifier with those features
- Efficacy of oversampling depends on specific properties of feature selection or classifier

Sampling

- Feature selection
  - OR, BNS: Immune to oversampling
  - Chi^2, IG: Less biased with sampling
- Classifiers
  - Decision Tree: Immune to sampling of features, susceptible to sampling of itself
  - Naïve Bayes: Sampling of itself and features both help
  - SVMs: Only feature sampling helps

Future directions

- Wrapper with different performance metric
  - Instead of finding subsets with the best accuracy (I/II ratio), maximize BNS or ROC of the subset
- Novel imbalance-aware subset evaluator
- Combining existing approaches in new ways

Conclusion

- Both feature selection and imbalanced data are important problems in data mining
- Solving both of them together is essential to creating classifiers for a variety of applications

References

- X. Chen and M. Wasikowski, “FAST: A ROC-based Feature Selection Metric for Small Samples and Imbalanced Data Classification Problems,” *Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery & Data Mining* (KDD'08), 2008, pp. 124-132.
- G. Forman, “An Extensive Empirical Study of Feature Selection Metrics for Text Classification,” *J. Machine Learning Research*, vol. 3, May 2003, pp. 1289-1305.
- M. Hall, “Correlation-based Feature Selection for Machine Learning,” doctoral dissertation, Dept. Computer Science, Univ. of Waikato, 1999.
- R. C. Holte, “Very Simple Classification Rules Perform Well on Most Commonly Used Datasets,” *Machine Learning*, vol. 11, April 1993, pp. 63-91.
- D. Koller and M. Sahami, “Toward Optimal Feature Selection,” *Proc. 13th Int'l Conf. Machine Learning* (ICML'96), 1996.
- H. Liu and R. Setiono, “A Probabilistic Approach to Feature Selection – A Filter Solution,” *Proc. 13th Int'l Conf. Machine Learning* (ICML'96), 1996.
- E. Xing, M. Jordan, and R. Karp, “Feature Selection for High-Dimensional Geometric Microarray Data,” *Proc. 18th Int'l Conf. Machine Learning* (ICLM'01), 2001.