kNN Approach to Unbalanced Data Distributions: A Case Study involving Information Extraction*

Jianping Zhang JZHANG@MITRE.ORG
The MITRE Corporation, M/S H305, 7515 Colshire Drive, McLean, VA 22102-7508

Inderjeet Mani IM5@GEORGETOWN.EDU
Georgetown University, Washington, DC 20057

Abstract

This paper describes an application of a simple kNN approach to a novel classification problem with an unbalanced class distribution. We discuss here one particular problem area, extracting protein names from the biological literature. Specifically, we empirically study the effects of under-sampling on the k nearest neighbor kNN approach and five different methods of choosing negative training examples. Our experimental results show that the kNN method is sensitive to the number of negative examples selected and the random selection of negative examples works better than three of the other four example selection methods.

Introduction

Classification is a well-studied problem in machine learning. Various classification techniques such as decision trees, neural networks, and rule induction have been developed and successfully applied to many domains. Many of these standard classification algorithms usually assume that training examples are evenly distributed among different classes. However, as indicated in (Japkowicz, 2000), unbalanced data sets often appear in many practical applications. In an *unbalanced data set*, the *majority class* is represented by a large portion of all the examples, while the other, minority class, has only a small percentage of all the examples. For a multi-class classification problem, there may be several minority classes. In some applications, all classes are minorities and this is especially true for many information extraction applications.

Studies (Weiss 1995; Kubat, Holte, & Matwin 1998; Fawcett & Provost 1997; Japkowicz & Stephen 2002) show that in many applications, unbalanced class distributions result in poor performances from standard classification algorithms. These classification algorithms generate classifiers that maximize the overall classification accuracy. When dealing with unbalanced data sets, this leads to either trivial classifiers that completely ignore the minority class or classifiers with many small (specific) disjuncts that tend to overfit the training sample. These small disjuncts are primarily responsible for errors on unseen cases (Holte, Acker, & Porter 1989). There have been some attempts at dealing with classification of unbalanced data sets (Japkowicz 2000). Methods include resizing training data sets, adjusting misclassification costs, and recognition-based learning (learning from the minority class).

Classifications with unbalanced data sets are also fairly common in problems of information extraction from text (Cardie and Howe 1997). Our case study here involves information extraction from the biomedical literature. Such information extraction is needed in order to speed up the creation of structured databases representing the latest scientific knowledge about specific objects such as proteins, genes, etc. We discuss here one particular problem area, that of extracting protein names from MEDLINE abstracts; this work is a part of a larger project at Georgetown University aimed at automatically inducing an ontology of protein names by fusing together information mined from MEDLINE abstracts, protein databases, and existing ontologies. Protein names show considerable variation because of the existence of multiple naming conventions and short forms used for convenience in running text. For example, the EphB2 receptor, a protein involved in signaling in the brain, was initially

* Workshop on Learning from Imbalanced Datasets II, ICML, Washington DC, 2003

referred to as 'Cek5', 'Nuk', 'Erk', 'Qek5 , 'Tyro6', 'Sek3', 'Hek5 , and 'Drt' before being standardized as 'EphB2' (Nature 1999). Existing information extraction approaches have used lexical resources such dictionaries as well supervised learning from labeled examples. While dictionaries are useful, they are costly to develop and maintain, and fail to handle new names; hence, machine-learning approaches are of interest. See (Hirschman et al. 2002) for a survey of bioinformatics information extraction methods.

In this paper, we describe a k nearest neighbor (kNN) learning approach to extracting protein names from MEDLINE abstracts. The learning task exhibits an unbalanced class distribution and only about 4% of all examples represent protein names. We study the effects of under-sampling on the k nearest neighbor approach and different ways of choosing negative training examples. We also show that the k nearest neighbor approach performs significantly better than C5.0 (Quinlan 1993) in this application.

2. Previous Work on Unbalanced Class Distributions

Since many practical machine learning applications involve unbalanced data sets, machine learning researchers have developed different methods for solving the class unbalance problem. Developed methods include resizing training data sets, adjusting misclassification costs, and recognition-based learning (learning from the minority class).

Resizing training sets includes over-sampling minority class examples (e.g., Ling & Li 1998) and/or under-sampling the majority class (e.g., Kubat & Matwin 1997). Ling & Li (1998) over-sampled the minority class by adding copies of the minority examples to the training set. Over-sampling does not increase information, but it does increase the misclassification cost (see below) of the minority examples. In under-sampling, examples removed could be randomly selected, or near miss examples, or examples that are far from minority class examples. Kubat & Matwin (1997) studied several different methods for reducing majority class examples. But down-sizing the majority class results in a loss of information that may result in overly general rules.

Cost-sensitive classifiers (Pazzani et al. 1994; Domingos, 1999) have been developed to handle the problems with different misclassification error costs. Cost-sensitive classifiers may be used for unbalanced data sets by setting a high cost to the misclassifications of a minority class example. This has a similar effect to over-sampling the minority class and may end up with over specific rules or rules overfitting training.

Recognition-based learning approaches learn rules from the minority class examples with or without using the examples of the majority class. It guarantees some rules will be learned for the minority class. Japkowicz et al. (1995) developed recognition-based multi-layer perceptrons for unbalanced data sets. Kubat et al. (1998) developed a recognition-based rule learning system, SHRINK, and applied it to oil spill detection from satellite radar images. SHRINK generates a single rule, which is a list of intervals of attributes, for the minority class. In identifying a minority class example, partial matching is applied to the example and the rule. Other recognition-based learning systems include Gold-Digger and Brute (Riddle et al. 1994) and the PNrule method developed by Agarwal and Joshi (2000).

3. Approach

3.1 The Problem

This case study uses a data set of 300 MEDLINE abstracts corresponding to 300 database entries, which were randomly picked from ~5000 human-curated database entries from the Protein Information Resource[2]. The 300 abstracts have been annotated based on a set of protein-tagging guidelines (Hu 2003) using MITRE's Alembic Workbench (Day et al. 1997). The annotator tagged nearly 3300 protein names in these abstracts.

The learning approach uses a word-based representation, where each word token is given a class (protein-start, protein-end, protein-middle or none, based on overlap with the annotator's protein tag). The approach extracts word-level features. These features include syntactic features based on part-of-speech tags from the Alembic tagger (Aberdeen et al. 1995) and semantic features based on several subdictionaries: a list of macromolecule terms (1047) (mt), biomedical_terms (852) (bt), chemical terms (806) (ct), common English terms related to protein names, and other relevant non_word_tokens (2212) (nwt). Positional features based on a context window are also extracted.

In summary, each word position in the text gives rise to a feature vector consisting of:

1. Word token
2. Part-of-speech tag
3. Subdictionary tag – macromolecule, biomedical, chemical term, common English term, non-word tokens, or nil

[2] pir.georgetown.edu

4. Protein tag – class
5. Start of protein tag – class
6. End of protein tag – class
7. Features (part-of-speech tag and subdictionary tag) of all 3 words to left and all 3 words to right

Table 1. Number of training and test vectors out of 300 abstracts

	Start	Protein	End
# of positive training examples	2434	2555	2434
# of negative training examples	56255	58593	56255
# of positive test examples	505	509	505
# of negative examples	11812	11808	11812

There are 15 independent variables and three classification tasks: one for protein tag, one for start of protein tag, and one for end of protein tag. Although we view these as separate tasks, in the protein tagging system the three classifiers are combined to determine the text extent of each protein tag. Of these 300 abstracts, the first 50 abstracts are used as test data and the remaining 250 abstracts are used as training data. Table 1 shows the number of training and test examples for the three classification tasks. It can be seen that this word based model generates lots of data but it is heavily skewed with only 4% positive examples.

3.2 Statistical Classifiers

A simple kNN algorithm is applied to the classification tasks. The similarity metric of the kNN algorithm uses the longest common subsequence (LCS) for the partial match of the Word feature and a simple binary match on all other features. All features except the Word feature are weighted equally, with the Word feature being weighted twice as important as the other features. Specifically, the following similarity metric is used:

$$Similarity(x, y) = \frac{2 \times LCS(x_0, y_0)}{\max(|x_0|, |y_0|)} + \sum_{i=1}^{14} match(x_i, y_i),$$

where x and y are the two vectors corresponding to two word positions, x_0 (y_0) is the value of the word feature and x_i (y_i), for i = 1, …, 14, are values of the remaining 14 features, LCS(x_0, y_0) is the length of the longest common subsequence of x_0 and y_0, $|x_0|$, ($|y_0|$) is the length of x_0 (y_0), and *match*(x_i, y_i) returns 1 if x_i and y_i are equal and 0 otherwise.

We have run 1NN, 3NN, 5-NN, 7NN, and 9NN on the data sets and differences in performance are not dramatic, with 5NN achieving the best results. The results reported in next section were obtained using 5NN. In addition to the kNN algorithm, C5.0, a decision tree learning algorithm, was applied to the same training and test sets.

3.3 Under-Sampling Negative Examples

In this work, we investigate the effect of under-sampling on the kNN approach. Study of over-sampling and cost-sensitive methods will be our future research. Under-sampling negative (here the majority class) examples is an often-used approach to deal with the problem caused by unbalanced data distribution. Instead of using the entire set of negative training examples, a small subset of negative examples is selected such that the resulting training data is less skewed. Different ways for selecting examples have been studied in the past. In our work, we have selected a given percentage of training negative examples in five different ways: random selection, selection of near-miss examples (three ways), and selection of most distant examples.

We have used three different methods to select near-miss examples. The first method (NearMiss-1) selects negative examples that are close to *some* of the positive examples. In this method, we select negative examples whose average distances to three closest positive examples are the smallest. The second method (NearMiss-2) selects negative examples that are close to *all* positive examples. In this method, examples are selected based on their average distances to three farthest positive examples. In the third method (NearMiss-3), we select a given number of the closest negative examples for each positive example. This method guarantees every positive example is surrounded by some negative examples. Finally, in selection of most distant negative examples, we choose the negative examples whose average distances to the three closest positive examples are the farthest. We expect the NearMiss methods should perform better than the random and distant methods, and the random method should work better than the distant method. We also expect that the NearMiss-3 method should achieve high precision and low recall while the distant method should achieve high recall and low precision.

4. Results

In this section, we report the results achieved by applying the simple 5NN algorithm to the data sets with different percentages of negatives examples selected using different methods. We use recall, precision, and F-measure as our performance measures. F-measure is a metric that is widely used in information retrieval. It is also used in some machine learning algorithms for unbalanced data sets (e.g., Weiss & Hirsh 2000). The two components of F-measure are *recall* and *precision*. The recall is the ratio of the number of positive examples correctly recognized

and the number of all positive examples. The precision is the ratio of the number of positive examples correctly recognized and the total number of examples (both positive and negative) recognized. F-measure (F1) is defined below.

$$F-measure(r)=\frac{2\times recall\times precision}{recall+precision}$$

Predictive accuracy is a reasonable metric when the user's objective function assigns the same cost to false positives and false negatives. When the numbers of false positives, true positives, false negatives, and true negatives are about equal, predictive accuracy tends to agree with precision and recall, but when false negatives predominate there can be large disagreements.

Figure 1 shows the 5NN F-measure scores with the different percentages of randomly selected negative examples. For all three classification tasks: protein, protein start, and protein end, the highest score was achieved at 10% randomly selected negative examples. This is due to the fact that the best balance between precision and recall is achieved at 10%.

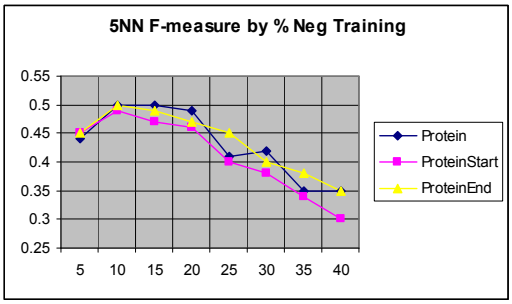

Figure 1. F-measure scores for randomly selected negative examples

Figure 2 is the 5NN precision-recall curve for different percentages of randomly selected negative examples, while Table 2 reports detailed results for 5NN and C5.0. Precision increases with the increase of the percentage of negative training examples, whereas recall decreases with it. These are true for both 5NN and C5.0 and showed that both algorithms are sensitive to the amount of negative training examples in this application. Therefore, undersampling may be used to adjust the trade-off between recall and precision, but it has to be used carefully.

C5.0 achieved higher recalls than those of 5NN with significantly lower precisions than 5NN. F-measure scores of C5.0 are lower than those of 5NN in almost all cases. It has been noticed that decision tree algorithms are sensitive to unbalanced class distributions (Japkowicz & Stephen 2002). C5.0 achieved the low precisions for the small amount of negative examples. This is probably due the overgeneralization of the positive class. It is known that the 5NN method is better than rule induction (or decision tree learning) methods in handling small disjuncts (Zhang 1990). When the percentages of negative examples are low, the differences are significant. With the increases of percentages of negative examples, C5.0 recalls drop quickly; this is because C5.0 tries to maximize the overall classification precision.

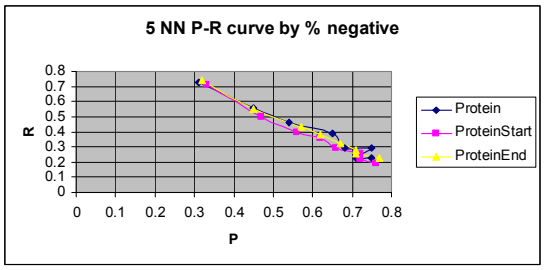

Figure 2. 5NN Precision-recall curve by % negative examples (randomly selected)

Table 3 shows the results for the five different negative example selection methods using 5NN. The result is the average of five different runs on different training and test sets from the protein data. Four fifth of the data was used for training and the remaining one fifth was used for test. The number in the parenthesis is the standard deviation. Surprisingly, the random method performed as well as the NearMiss-2 method and much better than the other three methods. The distribution of randomly selected negative examples should be similar to the distribution of all negative examples, other methods enforce the change of the negative example distribution.

The NearMiss-1 method selects negative examples based on their distances to three of the closest positive examples. This method performed surprisingly poorly on both precision and recall. Its poor performance may be explained as follows. Negative examples selected in this method may not be evenly distributed around positive examples. Namely, the NearMiss-1 method may select many negative examples around some positive examples, and very few around other examples. Positive examples with many selected negative neighbors cause low recall, while examples with very few selected negative neighbors are the reason of low precision. The results of the NearMiss-3 method support this claim.

The performance of the NearMiss-2 method is comparable with the performance of the random method. Negative examples selected in this method are close to all positive examples and they could be evenly distributed among positive examples. In the NearMiss-3 method, every positive example is surrounded by some selected negative examples. Therefore, its precision is high (up to 80%), but its recall is low. Its best F-measure score is achieved with 5% negative examples. If precision is

preferred, the Near-Miss-3 method could be a good option. Finally, as expected, the recall of the distant method is extremely high, but its precision is very low.

Table 2. Detailed results for randomly selected negative examples

% of training negative examples		Protein		Start		End	
		C5.0	5NN	C5.0	5NN	C5.0	5NN
	Precision	0.16	0.31	0.14	0.33	0.14	0.32
	Recall	0.77	0.73	0.82	0.71	0.80	0.74
	F-measure	0.27	0.44	0.24	0.45	0.24	0.45
10%	Precision	0.19	0.45	0.21	0.47	0.19	0.45
	Recall	0.62	0.56	0.68	0.50	0.64	0.55
	F-measure	0.29	0.50	0.32	0.49	0.29	0.50
15%	Precision	0.25	0.54	0.26	0.56	0.24	0.57
	Recall	0.49	0.46	0.54	0.40	0.48	0.43
	F-measure	0.33	0.50	0.35	0.47	0.32	0.49
20%	Precision	0.29	0.65	0.26	0.62	0.28	0.62
	Recall	0.49	0.39	0.45	0.36	0.44	0.38
	F-measure	0.37	0.49	0.33	0.46	0.34	0.47
25%	Precision	0.32	0.68	0.31	0.66	0.31	0.67
	Recall	0.41	0.29	0.43	0.29	0.40	0.32
	F-measure	0.36	0.41	0.36	0.40	0.35	0.45
30%	Precision	0.36	0.75	0.33	0.72	0.31	0.71
	Recall	0.36	0.29	0.39	0.26	0.30	0.28
	F-measure	0.36	0.42	0.36	0.38	0.31	0.40
35%	Precision	0.42	0.71	0.34	0.72	0.40	0.71
	Recall	0.30	0.23	0.30	0.22	0.34	0.26
	F-measure	0.35	0.35	0.32	0.34	0.37	0.38
40%	Precision	0.43	0.75	0.39	0.76	0.44	0.77
	Recall	0.15	0.23	0.28	0.19	0.16	0.23
	F-measure	0.22	0.35	0.33	0.30	0.24	0.35

Table 3. Results for five different negative example selection methods

% of training negative examples		Random	NearMiss1	NearMiss2	NearMiss3	Distant
5%	Precision	0.30 (0.021)	0.14 (0.011)	0.29 (0.021)	0.52 (0.042)	0.06 (0.004)
	Recall	0.78 (0.009)	0.68 (0.016)	0.78 (0.013)	0.53 (0.055)	0.99 (0.000)
	F-measure	0.42 (0.025)	0.23 (0.014)	0.42 (0.022)	0.52 (0.035)	0.12 (0.008)
10%	Precision	0.44 (0.030)	0.28 (0.026)	0.43 (0.030)	0.68 (0.039)	0.08 (0.006)
	Recall	0.64 (0.019)	0.52 (0.009)	0.64 (0.012)	0.35 (0.015)	0.97 (0.003)
	F-measure	0.52 (0.022)	0.36 (0.022)	0.51 (0.022)	0.46 (0.020)	0.14(0.010)
15%	Precision	0.51 (0.037)	0.40 (0.046)	0.51 (0.037)	0.75 (0.043)	0.09 (0.006)
	Recall	0.56 (0.016)	0.40 (0.012)	0.54 (0.015)	0.25 (0.012)	0.96 (0.002)
	F-measure	0.53 (0.021)	0.40 (0.022)	0.52 (0.021)	0.37 (0.017)	0.15 (0.013)
20%	Precision	0.58 (0.027)	0.48 (0.063)	0.57 (0.041)	0.78 (0.041)	0.10 (0.008)
	Recall	0.47 (0.044)	0.30 (0.011)	0.46 (0.017)	0.19 (0.013)	0.94 (0.005)
	F-measure	0.52 (0.019)	0.36 (0.019)	0.50 (0.020)	0.30 (0.020)	0.18 (0.012)
25%	Precision	0.62 (0.052)	0.53 (0.064)	0.65 (0.039)	0.80 (0.042)	0.11 (0.008)
	Recall	0.43 (0.019)	0.26 (0.013)	0.41 (0.017)	0.16 (0.010)	0.91 (0.008)
	F-measure	0.50 (0.020)	0.34 (0.019)	0.50 (0.022)	0.26 (0.016)	0.19 (0.013)
30%	Precision	0.63 (0.056)	0.56 (0.062)	0.68 (0.039)	0.82 (0.038)	0.13 (0.013)
	Recall	0.40 (0.028)	0.22 (0.013)	0.36 (0.019)	0.14 (0.009)	0.88 (0.007)
	F-measure	0.47 (0.026)	0.31 (0.014)	0.47 (0.023)	0.23 (0.015)	0.21 (0.013)
35%	Precision	0.67 (0.053)	0.62 (0.062	0.70 (0.040)	0.83 (0.044)	0.13 (0.009)
	Recall	0.36 (0.032)	0.20 (0.014)	0.32 (0.016)	0.12 (0.007)	0.85 (0.008)
	F-measure	0.45 (0.023)	0.30 (0.016)	0.43 (0.017)	0.21 (0.012)	0.22 (0.015)
40%	Precision	0.68 (0.059)	0.66 (0.055)	0.72 (0.044)	0.84 (0.046)	0.13 (0.010)
	Recall	0.34 (0.024)	0.18 (0.012)	0.30 (0.015)	0.11 (0.007)	0.83 (0.008)
	F-measure	0.45 (0.020)	0.28 (0.015)	0.42 (0.018)	0.18 (0.021)	0.23 (0.015)

5. Conclusion

In this paper, we described an application of a simple kNN approach to a novel classification problem in information extraction with unbalanced class distribution. Specifically, we empirically studied the effects of under-sampling on the kNN method and different negative example selection methods. In this application, we found that both the kNN and C5.0 are sensitive to the percentage of negative examples selected. Their recalls decrease with the increase in the percentage of selected negative examples, while precisions increase with the increase in the percentage of selected negative examples. Nevertheless, under-sampling is still a useful strategy for unbalanced class distribute in this application. It provides us a means for trading recall (or precision) for precision (or recall). The kNN method also outperforms C5.0.

Among the five negative example selection methods, the random and NearMiss-2 methods performed the best. The other two near miss example selection methods do not work as well as the random method. The method based on selection of distant examples tends to have the positive class overgeneralized. To the best of our knowledge, the work reported in this paper is the first study of how the kNN method works with the under-sampling strategy for the problem of unbalanced class distributions.

Acknowledgement

We are grateful to Cathy Wu and Zhangzhi Hu of Georgetown University Medical Center for providing us with annotated abstracts and protein-related subdictionaries.

References

Aberdeen, J., Burger, J., Day, D., Hirschman, L., Robinson, P., and Vilain, M. (1995). "MITRE: Description of the Alembic System system as used for MUC-6". In Proceedings of the Sixth Message Understanding Conference (MUC-6), Columbia, Maryland.

Agarwal, R. & Joshi, M. V. (2000). PNrule: a new framework for learning classifier models in data mining: a case-study in network intrusion detection. IBM research report, April.

Cardie, C. and Howe, N. (1997). "Improving minority class predication using case-specific feature weights," Proceedings of the 14th International Conference on Machine Learning, Morgan Kaufmann, pp. 57-65.

Day, D., Aberdeen, J., Hirschman, L., Kozierok, R., Robinson, P., and Vilain, M. (1997). Mixed-Initiative Development of Language Processing Systems. In Proceedings of the Fifth Conference on Applied Natural Language Processing. Association for Computational Linguistics.

Domingos, P. (1999). MetaCost: A general method for making classifiers cost-sensitive. *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* (pp. 155-164), San Diego, CA.

Fawcett, T.E. & Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery* 1(3):291-316.

Hirschman, L., Park, J.C., Tsuji, J., Wong, L., and Wu, C.H.. Accomplishments and challenges in literature data mining for biology. , 2002/ Bioinformatics Review, 18, 12, 1553-1561.

Holte R.C., Acker, L., & Porter, B. (1989). Concept learning and the problem of small disjuncts. Proceedings of Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), 813-818.

Hu, Z. Guidelines for Protein Name Tagging, version 1.0. Technical Report, PIR, Georgetown University. 2003.

Japkowicz, N., Myers, C., & Gluck, M. (1995). A novelty detection approach to classification. *Proceedings of the Fourteenth Joint Conference on Artificial Intelligence (IJCAI-95),* 518-523.

Japkowicz, N. (2000). Learning from Imbalanced Data Sets: A comparison of various strategies. *Proceedings of Learning from Imbalanced Data Sets, AAAI Workshop*. Technical Report WS-00-05, 10-15.

Japkowicz N. and Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis*. 6(5).

Kubat, M. & Matwin, S. (1997). Addressing the curse of imbalanced data sets: One-sided sampleing. *Proceedings of the Fourteenth International Conference on Machine Learning*. 179-186. Morgan Kaufmann.

Kubat, M., Holte, R., & Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning* 30:195-215.

Ling, C.X., & Li, C. (1998). Data mining for direct marketing: Problems and solutions. *Proceedings of The Forth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 73-79. AAAI Press.

Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., & Brunk, C. (1994). Reducing misclassification costs. Proceedings of the Eleventh International Conference on Machine Learning, 217-225. Morgan Kaufmann.

Nature (editorial opinion).Wanted: A new order in protein nomenclature. Nature, 401, 411, 30 Sep 1999.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Riddle, P., Segal, R., & Etzioni, O. (1994). Representation of design and brute-force induction in a Boeing

manufacturing domain. Applied Artificial Intelligence, 8:125-147.

Weiss, G.M. (1995). Learning with rare cases and small disjuncts. *Proceedings of 12th International Conference on Machine Learning*, (pp. 558-565), Lake Tahoe, California.

Weiss, G.M. & Hirsh, H. (2000). Learning to predict extremely rare events. *Proceedings of Learning from Imbalanced Data Sets, AAAI Workshop*. Technical Report WS-00-05, 64-68.

Zhang, J. (1990) "A Method That Combines Inductive Learning with Exemplar-Based Learning," proceedings of the Second IEEE International Conference on Tools for Artificial Intelligence.