Learning when Negative Examples Abound

Miroslav Kubat, Robert Holte, and Stan Matwin

Department of Computer Science, University of Ottawa
150 Louis Pasteur, Ottawa, Ontario, K1N 6N5 Canada
{mkubat,holte,stan}@csi.uottawa.ca

Abstract. Existing concept learning systems can fail when the negative examples heavily outnumber the positive examples. The paper discusses one essential trouble brought about by imbalanced training sets and presents a learning algorithm addressing this issue. The experiments (with synthetic and real-world data) focus on 2-class problems with examples described with binary and continuous attributes.

1 Introduction

The specific problem addressed here is learning from *imbalanced training sets* where examples from one class heavily outnumber examples from the other class. Highly imbalanced training sets occur in applications where the classifier is to detect an infrequent, albeit important, event: a fraudulent telephone call (Fawcett and Provost, 1996), an unreliable telecommunications customer (Ezawa, Singh, and Norton, 1996), or a rare diagnosis such as the thyroid diseases in the UCI repository. Extremely imbalanced classes also arise in information retrieval (Lewis and Catlett, 1994).

Performace of concept learning is customarily assessed with *accuracy:* the percentage of testing examples correctly classified by the induced classifier. In the case of imbalanced training sets, though, this is inappropriate. For instance, Kononenko and Bratko (1991) report a domain where a medical specialist achieved 64% accuracy, while 80% examples represented the majority class. Should he decide to always predict only the majority class, the expert would improve accuracy but probably lose his patients in the process.

Other performance indicators are needed. The information retrieval community uses *precision* and *recall* and combines them into the so-called F-measure (Lewis and Gale, 1994). Another good idea is the information-based criterion suggested by Kononenko and Bratko (1991). Swets (1988) measures the area under the curve that depicts the relation between the error rate observed on negative examples and the accuracy observed on positive examples. In our research on the detection of oil spills, we wanted to maximize the geometric mean (*g-mean*) $g = \sqrt{acc+ \cdot acc-}$, where $acc+$ is the percentage of positive examples correctly recognized and $acc-$ is the percentage of negative examples correctly recognized. Geometric mean is high when both $acc+$ and $acc-$ are high *and* when the difference between $acc+$ and $acc-$ is small. This criterion was chosen because it was consistent with the requirements of our customer; however, most of the ideas presented below will be valid with other criteria as well.

What do such criteria betray about the behavior of learning algorithms in applications with imbalanced training sets? This is best illustrated with a ficticious experiment. Choose randomly n positive and n negative examples, and run C4.5 on them. Then increase the number of negatives in increments of n (with the same n positives) and repeat the learning. In many applications, g-mean measured on an independent testing set (with the same distribution of positives and negatives) sooner or later significantly drops.

This problem has been noted in the neural-network community, where suggested solutions duplicate examples, create new examples, or increase the learning rate for examples of the underrepresented class (DeRouin et al. 1991). In the symbolic learning community, the problem has been addressed by weighing training examples (Pazzani et al., 1994), by windowing (Catlett, 1991), and by sampling (Lewis and Catlett, 1994). A natural approach exploits distinct costs assigned to false positives and false negatives (Pazzani et al., 1994).

In Section 2, we offer a hypothesis why abundant negatives hurt: the reason is that the positive and negative classes in real-world domains often overlap. The hypothesis underlies the simple learning algorithm SHRINK described in Section 3. Experiments illustrating its behavior are reported in Section 4.

2 Why Abundant Negatives Hurt

We assume a scenario where the agent is provided with a set of pairs $[(\mathbf{x}, c(\mathbf{x})]$ where $\mathbf{x}$ is a vector of attribute values (binary or continuous) and $c(\mathbf{x})$ is the corresponding concept label. To keep the work focussed, we consider only two-class problems.

Our point can be illustrated by the case of two *overlapping* classes. As the number of negative examples grows, so does the likelihood that the nearest neighbor of *any* example in the region of overlap will be negative. The *k-nearest-neighbor* rule will thus correctly recognize most examples of the majority class while failing on the minority class.

A *decision-tree* generator partitions the instance space into regions labeled with the class that has majority in the region. With imbalanced classes, the regions with mixed positives and negatives will tend to be labeled with the preponderant class. By another perpective, each positive example is eventually separated from other positives by a "wall" of negatives and the tree generator either stops splitting, in which case negatives are a majority, or it keeps splitting until it forms a tiny region around each positive.

Consider an experimental testbed where random examples are generated according to normal distribution with $\mu_+ = [0, 0]$ and $\sigma_+ = [1, 1]$ for the positives, and with $\mu_- = [2, 0]$ and $\sigma_- = [2, 2]$ for the negatives. In all runs, the same 50 training positives are used, while the number of negatives grows from 50 to 800 in increments of 50. Performance is measured on an independent testing set with the same proportion of positive and negative examples as the training set. The results of C4.5 (Quinlan, 1993) and 1-NN are shown in Figure 1: with

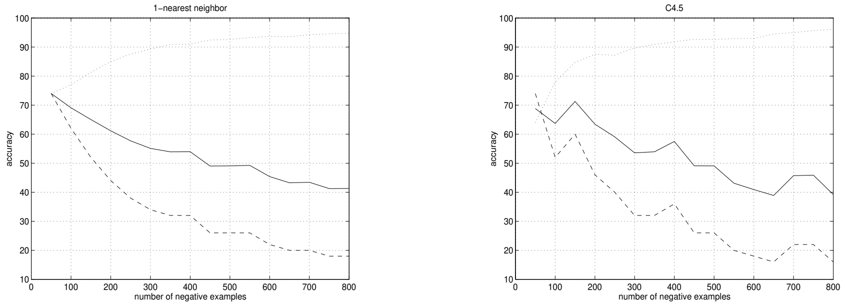

Fig. 1. Running the 1-NN rule (left) and of C4.5 (right) on the gauss data. Horizontal axis: the number of negatives. Dashed: accuracy on positives; dotted: accuracy on negatives; solid: g-mean.

abundant negatives, the performance on the majority class exceeds 90% and the performance on the minority class collapses.

(In C4.5, we used the default parameter setting while being aware that better results might be achieved by their more careful adjustment. The graphs show performances of unpruned trees: in the given domain, existing pruning techniques often degenerate the tree to a single leaf labeled with the majority class—the very event that we wanted to avoid.)

3 The System SHRINK

To cope with the domination of negative examples in mixed regions, our system SHRINK *insists* that a mixed region be labeled as positive, whether positive examples prevail in the region or not. That changes the learner's focus: search for the *best positive region,* one with the maximum ratio of positives to negatives.

The system is restricted to search for a *single* region to be labeled as positive, and is thus ill-suited for disjunctive concepts. The justification is that partitioning rare positives into two or more subsets leaves virtually nothing to be reliably reasoned about, considering the presence of freak outliers which, with such small numbers, could well be mistaken for additional disjuncts.

The concept will be represented by the *network of tests* depicted in Figure 2. The tests on numeric attributes have the form $x_i \in [\min a_i; \max a_i]$ where i indexes the attributes. For boolean attribues the tests will have the form of $x_i = 0$ or $x_i = 1$. Denote by h_i the output of the i-th test and let $h_i = 1$ if the test suggests a positive label and $h_i = -1$ otherwise. The example is classified as positive if $\sum_i h_i \cdot w_i > 0$, where w_i is the weight of the i-th test.

The *weights* are determined by an algorithm by Freund and Schapire (1995): Denote by $\mathbf{p}$ a vector assigning to each example its "importance." The vector is fixed during learning. In the simplest case, $p_j = 1$ for any j. Alternatively, p_j can be set to a higher value for a positive example, and to a lower value for a

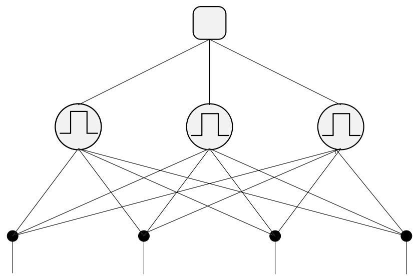

Fig. 2. A classifier in the form of a network of tests.

negative example. Define the loss vector $\mathbf{l_i} = [l_{1i}, \ldots, l_{ji}, \ldots]$ so that $l_{ji} = 0$ if the j-th example is classified correctly by the i-th test, and $l_{ji} = 1$ otherwise. The overall error of the i-th test is calculated as $e_i = \mathbf{l_i} \cdot \mathbf{p}^T$ and the corresponding weight is obtained as $w_i = \log(e_i/(1 - e_i))$. The intuition behind this expression is to assign higher weights to tests with less errors on the training set.

In the search for the tests, SHRINK begins by establishing the "best" interval along each attribute. To find it, the program begins with the smallest interval containing all positive examples and on every iteration shrinks the interval by eliminating either the left or right endpoint, whichever results in the better g-mean score. This produces a set of nested intervals that are scanned for the one with the maximum score. The intervals thus shrink from one attribute value to the next, hence the program's name.

When the intervals have been found, SHRINK discards tests with $g > 0.5$ (so as to get rid of less relevant attributes), and then calculates the weights of each of the remaining tests using the formula mentioned above. The procedure is summarized in Table 1.

Table 1. Control structure of SHRINK

1) For each attribute:
   - Sort the examples by the attribute's value;
   - The initial interval is $[\min a_i, \max a_i]$ where $\min a_i$ and $\max a_i$ are the min and max values observed for the i-th attribute in positive examples;
   - Remove either $\min a_i$ or $\max a_i$, whichever reduces more radically the number of negative examples in the interval; record the value of g-mean (g) for the new interval;
   - Repeat the previous step as long as there is at least one positive example in the interval, and then return the interval with maximum g.
5) Discard tests with $g < 0.5$.
6) Calculate the weight of each test.

SHRINK pays a price for its simplicity: it will fail on disjunctive concepts, and, as it was developed specifically for overlapping classes, its advantage will disappear if the classes do not overlap.

4 Experimental Evidence

SHRINK's results in the gaussian domain from Section 2 are depicted in Figure 3, using the same format as in Figure 1 and 2. The learner's performance does not appear to be affected by the growing number of negatives: whereas g-mean for C4.5 dropped with the growing number of negatives from about 70% to less than 40%, SHRINK kept a steady performance slightly above 75%.

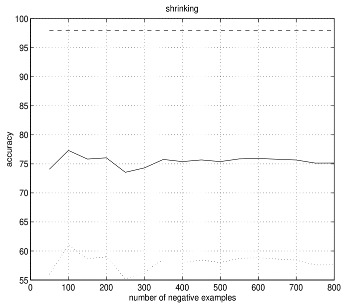

Fig. 3. Performance of SHRINK on the 2-dim data

The claim that SHRINK overcomes the problems with imbalanced classes is corroborated by experiments with the following real-world domains.

Oil-slick data. (our current research) Oil slicks I: 21 positives, 350 negatives; 39 numeric attributes; 7-fold cross-validation. Oil slicks II: 24 positives, 400 negatives; 44 numeric attributes; 8-fold cross-validation.

Sleep data. (An earlier project of one of the authors, see Kubat, Pfurtscheller, and Flotzinger, 1994). The original task was modified to recognize the occurence of class REM. Two files, slightly adapted from the original, were used: KR2 with 150 positives and 750 negatives; BR2 with 140 positives and 700 negatives. 15 numeric attributes; 5-fold cross-validation.

Euthyroid and Hypothyroid from the UCI repository (Murphy and Aha, 1994). We deleted all examples `age` and `sex` had missing values and we removed attribute `TBG_measured` because its values were frequently missing. From Euthyroid, we randomly selected 240 positives and 2400 negatives. From Hypothyroid, we selected 120 positives and 2400 negatives. 18 boolean and 6 numeric attributes; 5-fold cross-validation.

In the oil-slick domains, the ratio between the positive and negative examples is very high. For this reason, the importance vector $\mathbf{p}$ was set so that p_i for a positive example was r times higher than in the case of a negative example. The value of r is determined as $r = n_n/n_p$ where n_n is the number of negative examples and n_p is the number of positive examples. In the other domains (sleep and thyroid), $p_i = 1$ for all i. Auxiliary experiments (not reported here) have shown that SHRINK'S performance was not very sensitive to precise values in the importance vector.

Table 2. g-means in oil-slicks I

# neg.	C4.5	1-NN	SHRINK
140	66.5	40.0	66.9
210	51.0	43.5	68.2
280	52.6	29.8	68.0
350	48.2	27.6	67.5

Table 3. g-means in oil-slicks II

# neg.	C4.5	1-NN	SHRINK
160	83.9	55.9	75.8
240	82.3	59.8	74.9
320	83.2	49.1	72.6
400	80.5	44.8	73.8

Table 4. g-means in sleep data: KR

# neg.	C4.5	1-NN	SHRINK
150	81.0	76.3	68.6
300	79.4	75.1	71.7
450	76.9	73.0	74.7
600	72.3	70.3	73.8
750	72.0	67.8	75.9

Table 5. g-means in sleep data: BR

# neg.	C4.5	1-NN	SHRINK
140	84.5	84.5	69.7
280	83.9	86.0	71.0
420	83.2	85.7	70.2
560	78.7	83.5	72.2
700	78.4	78.9	73.1

Table 6. g-means in euthyroid data

# neg.	C4.5	1-NN	SHRINK
480	94.3	71.5	71.7
960	94.3	67.0	78.1
1440	94.7	63.2	74.6
1920	91.1	62.2	73.7
2400	88.2	60.8	74.0

Table 7. g-means in hypothyroid

# neg.	C4.5	1-NN	SHRINK
360	95.7	93.1	93.5
840	96.3	93.0	94.8
1320	95.4	91.0	94.4
1800	95.6	89.2	94.7
2280	93.6	88.9	95.0

The results are summarized in Tables 2 through 7 for growing numbers of negatives. For reference, the tables give also results achieved by C4.5 and 1-NN. (The poor results of 1-NN in some domains, such as oil-slick II, are caused by a high number of irrelevant attributes.) In all domains, the performance of 1-NN decreases with the growing number of negatives. The performance of C4.5 drops in all domains save for the hypothyroid data file, where it remains virtually unchanged regardless of the number of negatives.

SHRINK's performance steadily increases with the growing number of negatives in oil slicks I, and in both sleep-data domains. In the hypothyroid domain, the improvement is marginal. Only in the euthyroid domain did SHRINK's performance drop with increasing number of negatives. When presented with *all* available examples, SHRINK outperformed C4.5 in three (out of 6) domains and outperformed 1-NN in five domains. In some domains, the limits of SHRINK's representation language were reached even with small numbers of negatives and could not be improved by providing more negatives. As an aside, our algorithm tended to yield better results on positive examples than on negative examples. A more detailed study would exceed the scope of this brief contribution and will appear in a full-length paper currently under preparation.

To obtain evidence that the performance of SHRINK can be attributed to the chosen evaluation criterion, we have run the program on the same data, this time using mean accuracy. Expectedly, SHRINK now turned out to be useless, invariably relapsing to 100% on negatives and 0% on positives whenever the ratio between the positives and negatives exceeded 3, in some domains even earlier.

The approach should not be viewed as a panacea. In some domains, such as the glass data from the UCI repository, the performance of C45 and 1-NN does not degrade with increasing numbers of negatives. SHRINK will also lose its edge if the concept is disjunctive.

5 Conclusion

The poor behavior of existing learners in domains with imbalanced training sets can be caused by the fact that examples of the majority class can "infest" the region of the minority class. This can be due to class-label noise or to the overlap between the two classes. In this paper, we described a novel technique that is robust against this phenomenon.

In *future research,* analytical and practical studies of essential learning algorithms should be addressed. One should also study decision-tree generators using (e.g. in attribute-value splitting) criteria maximizing g-mean or other criteria of this kind. Whereas our experiments focussed on two-class problems, multiclass domains present new challenges: $n-1$ underrepresented classes and one dominating class can be different case than $n-1$ balanced classes with a single underrepresented class. Techniques for sampling the examples from the majority class should be investigated.

Acknowledgements

The research was partially supported by Precarn Inc. and NSERC. The sleep data belong to the Department of Medical Informatics, Technical University Graz, and have been recorded and classified under a grant from the 'Fonds zur Förderung der wissenschaftlichen Forschung' (S49/03). Thanks are due to Gert Pfurtscheller for his kind permission to use these data.

References

Ambrosino, R., Buchanan, R., Cooper, G.F., and Fine, M. (1995). The Use of Misclassification Costs to Learn Rule-Based Decision Support Models for Cost-Effective Hospital Admission Strategies. *Proceedings of the 19th Annual Symposium on Computer Applications in Medical Care (SCAMC'95)* pp. 304–308

Bloedorn, E., Mani, I., and MacMillan, T.R. (1996). Machine Learning of User Profiles: Representational Issues. *Proceeding of the National Conference on Artificial Intelligence, AAAI'96* pp. 433–437

Catlett, J. (1991). Megainduction: A Test Flight. *Proceedings of the 8th International Workshop on ML* (pp.596–599), San Mateo, CA: Morgan Kaufmann

DeRouin, E., Brown, J., Beck, H., Fausett, L, and Schneider, M. (1991). Neural Network Training on Unequally Represented Classes. In Dagli, C.H., Kumara, S.R.T. and Shin, Y.C. (eds.): *Intelligent Engineering Systems Through Artificial Neural Networks*, ASME Press, New York, 135–145

Ezawa, K.J., Singh, M. and Norton, S.W. (1996). Learning Goal Oriented Bayesian Networks for Telecommunications Management. *Proceedings of the International Conference on Machine Learning, ICML'96* (pp. 139–147), Bari, Italy, Morgan Kaufmann

Fawcett, T and Provost, F. (1996). Combining Data Mining and Machine Learning for Effective User Profile. *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining* (pp. 8–13), Portland OR, AAAI Press

Freund, Y. and Schapire, R.E. (1995). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Proceedings of the 2nd Annual European Conference on Computational Learning Theory* (pp.23–37)

Kononenko, I. and Bratko, I. (1991). Information-Based Evaluation Criterion for Classifier's Performance. *Machine Learning*, 6, 67–80

Kubat, M., Pfurtscheller, G., and Flotzinger D. (1994). AI-Based Approach to Automatic Sleep Classification. *Biological Cybernetics*, 79, 443–448

Lang, K. (1995). Newsreader: Learning to Filter News. *Proceedings of the 12th International Conference on Machine Learning, ICML'95* (pp. 331–339), Tahoe Lake, CA, Morgan Kaufmann

Lewis, D. and Catlett, J. (1994). Heterogeneous Uncertainty Sampling for Supervized Learning. *Proceedings of the 11th International Conference on Machine Learning, ICML'94* (pp. 148–156), New Brunswick, New Jersey, Morgan Kaufmann

Lewis, D. and Gale, W. (1994). Training Text Classifiers by Uncertainty Sampling. *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*

Murphy, P. and Aha, D. (1994). UCI Repository of Machine Learning Databases [machine-readable data repository]. Technical Report, University of California, Irvine

Murthy, S., Kasif, S., & Salzberg, S. (1994). A System for Induction of Oblique Decision Trees. *Journal of Artificial Intelligence Research*, 2, 1–32

Pazzani, M, Merz, C., Murphy, P., Ali, K., Hume, T., and Brunk, C. (1994). Reducing Misclassification Costs. *Proceedings of the 11th International Conference on ML, ICML'94* (pp. 217–225), New Brunswick, New Jersey, Morgan Kaufmann

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

Swets, J.A. (1988). Measuring the Accuracy of Diagnostic Systems. *Science*, 240, 1285–1293