Data Mining in Metric Space: An Empirical Analysis of Supervised Learning Performance Criteria

Rich Caruana and **Alexandru Niculescu-Mizil** [1]

Abstract.
Many criteria can be used to evaluate the performance of supervised learning. Different criteria are appropriate in different settings, and it is not always clear which criteria to use. A further complication is that learning methods that perform well on one criterion may not perform well on other criteria. For example, SVMs and boosting are designed to optimize accuracy, whereas neural nets typically optimize squared error or cross entropy. We conducted an empirical study using a variety of learning methods (SVMs, neural nets, k-nearest neighbor, bagged and boosted trees, and boosted stumps) to compare nine boolean classification performance metrics: Accuracy, Lift, F-Score, Area under the ROC Curve, Average Precision, Precision/Recall Break-Even Point, Squared Error, Cross Entropy, and Probability Calibration. Multidimensional scaling (MDS) shows that these metrics span a low dimensional manifold. The three metrics that are appropriate when predictions are interpreted as probabilities: squared error, cross entropy, and calibration, lay in one part of metric space far away from metrics that depend on the relative order of the predicted values: ROC area, average precision, break-even point, and lift. In between them fall two metrics that depend on comparing predictions to a threshold: accuracy and F-score. As expected, maximum margin methods such as SVMs and boosted trees have excellent performance on metrics like accuracy, but perform poorly on probability metrics such as squared error. What was not expected was that the margin methods have excellent performance on ordering metrics such as ROC area and average precision. We introduce a new metric, SAR, that combines squared error, accuracy, and ROC area into one metric. MDS and correlation analysis shows that SAR is centrally located and correlates well with other metrics, suggesting that it is a good general purpose metric to use when more specific criteria are not known.

[1] Department of Computer Science, Cornell University, Ithaca, NY 14853 USA email {caruana,alexn}@cs.cornell.edu

KDD'04, August 22–25, 2004, Seattle, Washington, USA.

1 Introduction

In supervised learning, finding a model that could predict the true underlying probability for each test case would be optimal. We refer to such an ideal model as the *One True Model*. Any reasonable performance metric should be optimized (in expectation, at least) by the one true model, and no other model should yield performance better than it.

Unfortunately, we usually do not know how to train models to predict the true underlying probabilities. The one true model is not easy to learn. Either the correct parametric model type for the domain is not known, or the training sample is too small for the model parameters to be estimated accurately, or there is noise in the data. Typically, all of these problems occur together to varying degrees.

Even if magically the one true model were given to us, we would have difficulty selecting it from other less true models. We do not have performance metrics that will reliably assign best performance to the probabilistically true model given finite validation data.

In practice, we train models to minimize loss measured via a specific performance metric. Since we don't have metrics that could reliably select the one true model, we must accept the fact that the model(s) we select will necessarily be suboptimal. There may be only one true model, but there are *many* suboptimal models.

There are different ways that suboptimal models can differ from the one true model – tradeoffs can be made between different kinds of deviation from the one true model. Different performance metrics reflect these different tradeoffs. For example, ordering metrics such as area under the ROC curve and average precision do not care if the predicted values are near the true probabilities, but depend only on the relative size of the values. Dividing all predictions by ten does not change the ROC curve, and metrics based on the ROC curve are insensitive to this kind of deviation from truth. Metrics such as squared error and cross entropy, however, are greatly affected by scaling the predicted values, but are less affected by small changes in predicted values that might alter the relative ordering but not significantly change the deviation from the target values. Squared error and cross entropy reflect very different tradeoffs than metrics based on the ROC curve. Similarly, metrics such as accuracy depend on how the predicted values fall relative to a threshold. If predicted values are rescaled, accuracy will be unaffected if the threshold also is rescaled. But if small changes to predicted values are

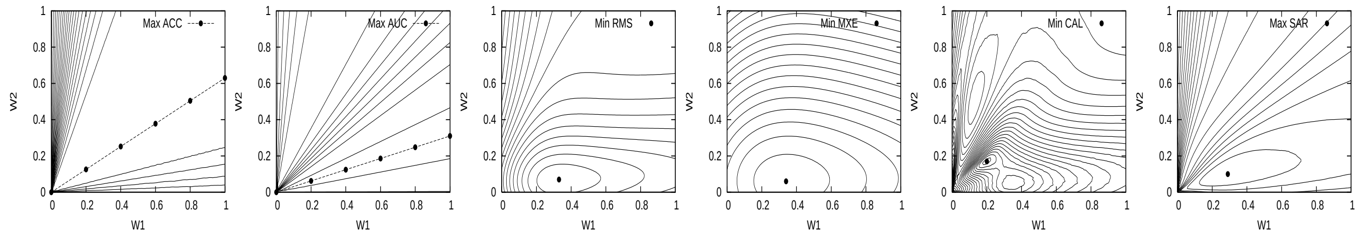

Figure 1. Level curves for six error metrics: ACC, AUC, RMS, MXE, CAL, SAR for a simple problem.

made for cases near the threshold, this can have large impact on accuracy. Accuracy reflects yet another tradeoff in how deviation from truth is measured.

The one true model, if available, would have (in expectation) the best accuracy, the best ROC curve, and the best cross entropy, and the different tradeoffs made by these metrics would not be important. But once we accept that we will not be able to find the one true model, and must therefore accept suboptimal models, the different tradeoffs made by different performance metrics become interesting and important. Unfortunately, little is known about how different performance metrics compare to each other.

In this paper we present results from an empirical analysis of nine widely used performance metrics. We perform this empirical comparison using models trained with seven learning algorithms: SVMs, neural nets, k-nearest neighbor, bagged and boosted trees, and boosted stumps. We use multidimensional scaling (MDS) and correlation analysis to interpret the results. We also examine which learning methods perform best on the different metrics. Finally, we introduce a new metric, SAR, that combines squared error, accuracy, and ROC area into a single, robust metric.

2 The Performance Metrics

We experiment with nine performance metrics for boolean classification: Accuracy (ACC), Lift (LFT), F-Score (FSC), Area under the ROC Curve (AUC), Average Precision (APR), the Precision/Recall Break-Even Point (BEP), Root Mean Squared Error (RMS), Mean Cross Entropy (MXE), and Probability Calibration (CAL). Definitions for each of the metrics can be found in Appendix A.

Figure 1 shows level curves for six of the ten performance metrics for a model with only two parameters ($W1$ and $W2$) trained on a simple synthetic binary problem. Peak performance in the first two plots occurs along a ridge in weight space. In the other four plots peak performance is indicated by solid dots. Peak performance for some metrics nearly coincide: RMS and MXE peak at nearly the same model weights. But other metrics peak in different places: CAL has a local optimum near the optima for RMS and MXE, but its global optimum is in a different place. Also, the ridges for optimal ACC and optimal AUC do not align and do not cross the optima for the other four metrics. Optimizing to each of these metrics yields different models, each representing different tradeoffs in the kinds of errors the models make. Which of these tradeoffs is best depends on the problem, the learning algorithm, and how the model predictions ultimately will be used.

We originally divided the nine metrics into three groups: threshold metrics, ordering/rank metrics, and probability metrics. The three threshold metrics are accuracy (ACC), F-score (FSC) and lift (LFT). F-score is the harmonic mean of precision and recall at some threshold. Lift measures the true positive rate in the fraction of cases that fall above threshold. (See Appendix A for a definition of lift, and [3] for a description of Lift Curves. Lift is the same as precision at some threshold, but scaled so that it can be larger than 1.) Usually ACC and FSC use a fixed threshold. In this paper we use 0.5. With lift, often the threshold is adjusted so that a fixed percent, p, of cases are predicted as positive, the rest falling below threshold. Usually p depends on the problem. For example, in marketing one might want to send fliers to 10% of customers. Here we somewhat arbitrarily set $p = 25\%$ for all problems. Note that for all threshold metrics it is not important how close a prediction is too a threshold, only if the predicted value is above or below threshold.

The ordering/rank metrics look at predictions differently from the threshold metrics. If cases are ordered by predicted value, the ordering/rank metrics measure how well the ordering ranks positive cases above negative cases. The rank metrics can be viewed as a summary of the performance of a model across all possible thresholds. The rank metrics we use are area under the ROC curve (AUC), average precision (APR), and precision/recall break even point (BEP). See [10] for a discussion of ROC curves from a machine learning perspective. Rank metrics depend only on the ordering of the predictions, not the actual predicted values. If the ordering is preserved it makes no difference if the predicted values range between 0 and 1 or between 0.29 and 0.31.

Although we group Lift with the threshold metrics, and BEP with the ordering metrics, BEP and Lift are similar to each other in some respects. Lift is proportional to BEP if Lift is calculated at p equal to the proportion of positives in the data set. This threshold also is the break-even point where precision equals recall. BEP and Lift are similar to the ordering metrics because the threshold depends implicitly on the ordering, but also are similar to the threshold metrics because neither is sensitive to the orderings on either side of the threshold once that threshold has been defined. Results presented later suggest that both Lift and BEP are more similar to the ordering metrics than to the threshold metrics.

The three probability metrics depend on the predicted values, not on how these values fall relative to a threshold or relative to each other. The probability metrics are uniquely minimized (in expectation) when the predicted value for each case coincides with the true underlying probability of that case being positive. The probability metrics we consider are squared error (RMS), cross entropy (MXE) and calibration

(CAL). CAL measures the calibration of a model: if a model predicts 0.85 for a large number of cases, about 85% of those cases should prove to be positive if the model is well calibrated. See Appendix A for details of how CAL is calculated.

We also experiment with a new performance metric, SAR, that combines squared error, accuracy, and ROC area into one measure: $SAR = (ACC + AUC + (1 - RMS))/3$. SAR behaves somewhat differently from ACC, AUC, and RMS alone, and is a robust metric to use when the correct metric is unknown. SAR is discussed further in Section 8.

3 Normalizing the Scores

Performance metrics such as accuracy or squared error have range $[0, 1]$, while others (lift, cross entropy) range from 0 to q where q depends on the data set. For some metrics lower values indicate better performance. For others higher values are better. Metrics such as ROC area have baseline rates that are independent of the data, while others such as accuracy have baseline rates that depend on the data. If baseline accuracy is 0.98, an accuracy of 0.981 might be poor performance, yet on another problem, if the Bayes optimal rate is 0.60, achieving an accuracy of 0.59 might be excellent performance.

In order to compare performance metrics in a meaningful way, all the metrics need to be placed on a similar scale. One way to do this is to scale the performances for each problem and metric from 0 to 1, where 0 is poor performance, and 1 is good performance. For example, we might place baseline performance at 0, and the Bayes optimal performance at 1. Unfortunately, we cannot estimate the Bayes optimal rate on real problems. Instead, we can use the performance of the best observed model as a proxy for the Bayes optimal performance. We calculate baseline rate as follows: predict p for every case, where p is the percent of positives in the test set. We normalize performances to the range $[0, 1]$, where 0 is baseline and 1 represents best performance. If a model performs worse than baseline, its normalized score will be negative. See Table 1 for an example of normalized scores. The disadvantage of normalized scores is that recovering the raw performances requires knowing the performances that define the top and bottom of the scale, and as new best models are found the top of the scale changes.

Table 1. Accuracy on ADULT problem

MODEL	ACC	NORM SCORE
BST-STMP	0.8556	1.0000
BAG-DT	0.8534	0.9795
DT	0.8503	0.9494
SVM	0.8480	0.9267
BST-DT	0.8464	0.9113
ANN	0.8449	0.8974
KNN	0.8320	0.7731
BASELINE	0.7518	0.0000

CAL, the metric we use to measure probability calibration, is unusual in that the baseline model that predicts p for all cases, where p is the percent of positives in the test set, has excellent calibration. (Because of this, measures like CAL typically are not used alone, but are used in conjunction with other measures such as AUC to insure that only models with good discrimination *and* good calibration are selected. See Figure 1 for a picture of how unusual CAL's error surface is compared with other metrics.) This creates a problem when normalizing CAL scores because the baseline model and Bayes optimal model have similar CAL scores. This does not mean CAL is a poor metric – it is effective at distinguishing poorly calibrated models from well calibrated models. We address this problem later in the paper.

4 Experimental Design

The goal of this work is to analyze how the ten metrics compare to each other, and also how well different learning algorithms perform on the different metrics. To do this we train many different kinds of models on seven test problems and calculate the performance of each of these models on the ten metrics for each problem.

We train the models using seven learning algorithms: SVM, Neural Nets (ANN), Boosted Decision Trees (BST-DT), Boosted Decision Stumps (BST-STMP), Bagged Decision Trees (BAG-DT), single Decision Trees (DT) and Memory Based Learning (KNN). For each algorithm we train many variants and many parameter settings. For example, we train ten styles of decision trees, neural nets of different sizes, SVMs using many different kernels, etc. A total of 2000 models are trained and tested on each problem. See Appendix B for a description of the parameter settings we use for each learning method. While this strategy won't create every possible model, and won't create a uniform sample of the space of possible models, we feel that this is an adequate sample of the models that often will be trained in practice.

For each problem, the 2000 models are trained on the same train set of 4000 points. The performance of each model is measured on the same large test set for each of the ten performance metrics. In order put the performances on the same scale across different metrics and different problems, we transform the raw performance to normalized scores as explained in Section 3. In total, across the seven problems, we have $2000 * 7 = 14,000$ models and for each model we have it's score on each of the 10 performances metrics.

5 Data Sets

We compare the algorithms on seven binary classification problems. ADULT, COV_TYPE and LETTER are from UCI Machine Learning Repository [1]. ADULT is the only problem that has nominal attributes. For ANNs, SVMs and KNNs we transform nominal attributes to boolean. Each DT, BAG-DT, BST-DT and BST-STMP model is trained twice, once with the transformed attributes and once with the original attributes. COV_TYPE has been converted to a binary problem by treating the largest class as the positive and the rest as negative. We converted LETTER to boolean in two ways. LETTER.p1 treats the letter "O" as positive and the remaining 25 letters as negative, yielding a very unbalanced binary problem. LETTER.p2 uses letters A-M as positives and the rest as negatives, yielding a well balanced problem. HYPER_SP is the IndianPine92 data set [4] where the difficult class Soybean-mintill is the positive class. SLAC is a problem from collaborators at the Stanford Linear Accelerator and MEDIS is a medical data set. The characteristics of these data sets are summarized in Table 2.

Table 2. Description of problems

PROBLEM	#ATTR	TRAIN SIZE	TEST SIZE	% POS.
ADULT	14/104	4000	35222	25%
COV_TYPE	54	4000	25000	36%
LETTER.P1	16	4000	14000	3%
LETTER.P2	16	4000	14000	53%
MEDIS	63	4000	8199	11%
SLAC	59	4000	25000	50%
HYPER_SP	200	4000	4366	24%

6 MDS in Metric Space

Training 2000 models on each problem using seven learning algorithms gives us 14,000 models, each of which is evaluated on ten performance metrics. This gives us 14,000 sample points to compare for each performance metric. We build a 10x14,000 table where lines represent the performance metrics, columns represent the models, and each entry in the table is the score of the model on that metric. For MDS, we treat each row in the table as the coordinate of a point in a 14,000 dimension space. The distance between two metrics is calculated as the Euclidean distance between the two corresponding points in this space. Because the coordinates are strongly correlated, there is no curse-of-dimensionality problem with Euclidean distance in this 14,000 dimensional space.

We are more interested in how the metrics compare to each other when models have good performance than when models have poor performance. Because of this, we delete columns representing poorer performing models in order to focus on the "interesting" part of the space where good models lie. For the analyses reported in this paper we delete models that perform below baseline on any metric (except CAL).

Ten metrics permits $10*9/2 = 45$ pairwise comparisons. We calculate Euclidean distance between each pair of metrics in the sample space, and then perform multidimensional scaling on these pairwise distances between metrics.

MDS is sensitive to how the performance metrics are scaled. The normalized scores described in Section 3 yield well-scaled performances suitable for MDS analysis for most metrics. Unfortunately, as discussed in Section 3, normalized scores do not work well with CAL. Because of this, we perform MDS two ways. In the first, we use normalized scores, but exclude the CAL metric. In the second, we include CAL, but scale performances to mean 0.0 and standard deviation 1.0 instead of using normalized scores. Scaling by standard deviation resolves the problem with CAL for MDS, but is somewhat less intuitive because scores scaled by standard deviation depend on the full distribution of models instead of just the performances that fall at the top and bottom of each scale.

Figure 2 shows the MDS stress as a function of the number of MDS dimensions (when CAL is included). The ten metrics appear to span an MDS space of about 3 to 5 dimensions. In this section we examine the 2-D MDS plots in some detail.

Figure 3 shows two MDS plots for the metrics that result when dimensionality is reduced to two dimensions. The plot on the left is MDS using normalized scores when CAL is excluded. The plot on the right is MDS using standard deviation scaled scores when CAL is included.

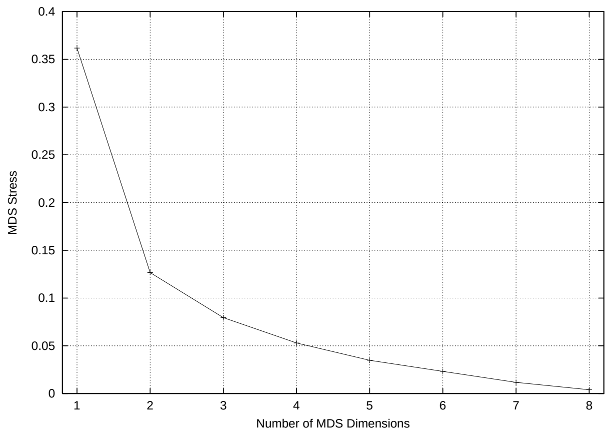

Figure 2. MDS stress vs. number of dimensions

Both MDS plots show a similar pattern. The metrics appear to form 4-5 somewhat distinct groups. In the upper right hand corner is a group that includes AUC, APR, BEP, LFT, and SAR. The other groups are RMS and MXE, ACC (by itself, or possibly with FSC), FSC (by itself, or possibly with ACC), and CAL (by itself). It is not surprising that squared error and cross entropy form a cluster. Also, presumably because squared error tends to be better behaved than cross entropy, RMS is closer to the other measures than MXE. We are somewhat surprised that RMS is so centrally located in the MDS plots. Perhaps this partially explains why squared error has proved so useful in many applications.

It is somewhat surprising that accuracy does not appear to correlate strongly with any of the other metrics, except possibly with FSC. ACC does not fall very close to other metrics that use thresholds such as Lift and F-Score, even though F-Score uses the same 0.5 threshold as accuracy in our experiments. (The threshold for Lift is adjusted dynamically so that 25% of the cases are predicted as positive.) Accuracy is surprisingly close to RMS, and closer to RMS than to MXE, again suggesting that part of the reason why RMS has been so useful is because of its close relationship to a metric such as ACC that has been so widely used.

The most surprising pattern in the MDS plot that includes CAL is that CAL is distant from most other metrics. There appears to be an axis running from CAL at one end to the ordering metrics such as AUC and APR at the other end that forms the largest dimension in the space. This is surprising because one way to achieve excellent ordering is to accurately predict true probabilities, which is measured by the calibration metric. However, one can achieve excellent AUC and APR using predicted values that have extremely poor calibration, yet accurately predict the relative ordering of the cases. The MDS plot suggests that many models which achieve excellent ordering do so without achieving good probabilistic calibration. Closer examination shows that some models such as boosted decision trees yield remarkably good ordering, yet have extremely poor calibration. We believe maximum margin methods such as boosting tradeoff reduced calibration for better margin. See Section 9 for further discussion of this issue. One also can achieve good calibration, yet have poor AUC and APR. For example, decision trees with few leaves may be well calibrated, but the coarse set of values they predict do

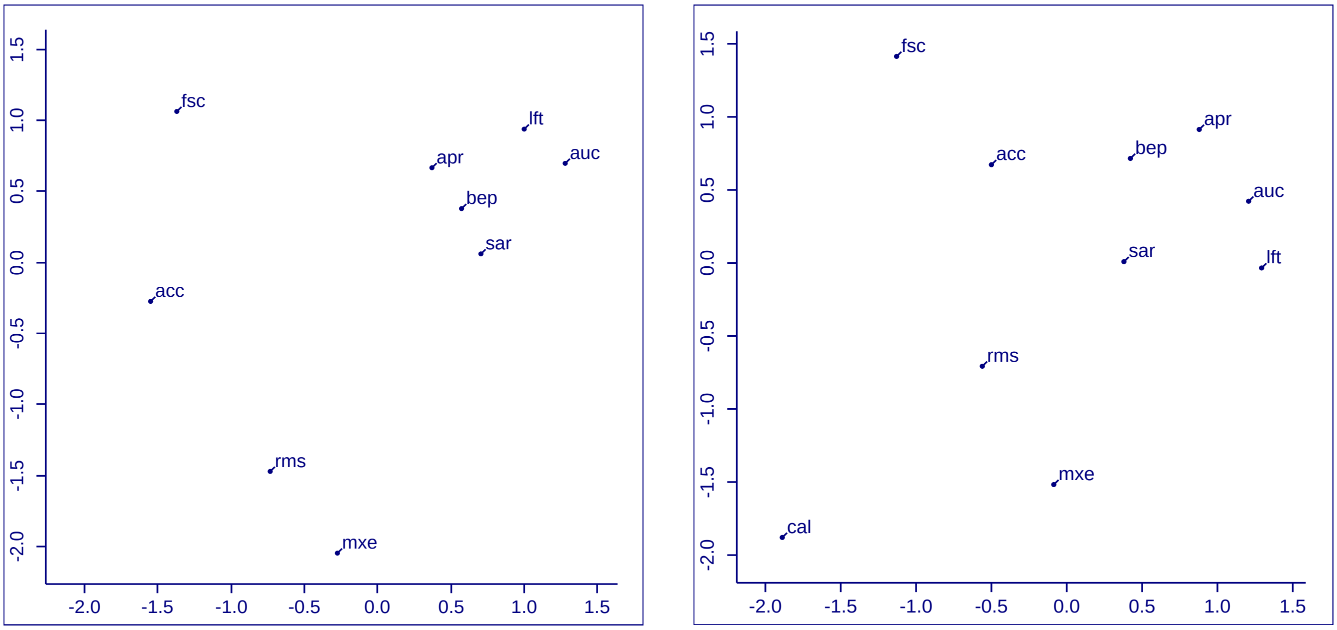

Figure 3. 2D MDS plot using normalized scores (left) and standard deviation scaling (right).

not provide a basis for good ordering.

Figure 4 shows 2-D MDS plots for six of the seven test problems. The seventh plot is similar and is omitted to save space. (The omitted plot is one of the two LETTER problems.) Although there are variations between the plots, the 2-D MDS plots for the seven problems are remarkably consistent given that these are different test problems. The consistency between the seven MDS plots suggests that we have an adequate sample size of models to reliably detect relationships between the metrics. Metrics such as ACC, FSC, and LFT seem to move around with respect to each other in these plots. This may be because they have different sensitivities to the ratio of positives to negatives in the data sets. For example, BEP is proprtional to LFT (and thus behaves similarly) when the percentage of positives in the dataset equals the fraction predicted above threshold (25% in this paper). Other than this, we have not been able to correlate differences we see in the individual plots with characteristics of the problems that might explain those differences, and currently believe that the MDS plots that combine all seven problems in Figure 3 represents an accurate summary of the relationships between metrics. Note that this does not mean that the performance of the different learning algorithms exhibits the same pattern on these test problems (in fact they are very different), only that the relationships between the ten metrics appear to be similar across the test problems when all the learning algorithms are considered at one time.

7 Correlation Analysis

As with the MDS analysis in the previous section, we used each of the ten performance metrics to measure the performance of the 2000 models trained with the different learning methods on each of the seven test problems. In this section we use correlation analysis on these models to compare metrics instead of MDS.

Again, to make the correlation analysis easier to interpret, we first scale performances to the range $[0, 1]$ so that the best performance we observed with that metric on each problem with any of the learning methods is performance 1, and baseline performance with that metric and data set is performance 0. This eliminates the inverse correlation between measures such as accuracy and squared error, and normalizes the scale of each metric.

Ten metrics permits $10*9/2 = 45$ pairwise correlations. We do these comparisons using both linear correlation (excluding CAL) and rank correlation. The results from the linear and rank correlation analyses are qualitatively similar. We present the results for non-parametric rank correlation because rank correlation makes fewer assumptions about the relationships between the metrics, and because rank correlation is insensitive to how CAL is scaled.

Table 3 shows the rank correlation between all pairs of metrics. Each entry in the table is the average rank correlation across the seven test problems. The table is symmetric and contains only 45 unique pairwise comparisons. We present the full matrix because this makes it easier to scan some comparisons. The final column is the mean of the rank correlations for each metric. This gives a rough idea how correlated each metric is on average to *all* other metrics.

Metrics with pairwise rank correlations near one behave more similarly than those with smaller rank correlations. Ignoring the SAR metric which is discussed in the next section, seven metric pairs have rank correlations above 0.90:

0.96: Lift to ROC Area
0.95: ROC Area to Average Precision
0.93: Accuracy to Break-even Point
0.92: RMS to Cross-Entropy
0.92: Break-Even Point to ROC Area
0.92: Break-Even Point to Average Precision
0.91: Average Precision to Lift

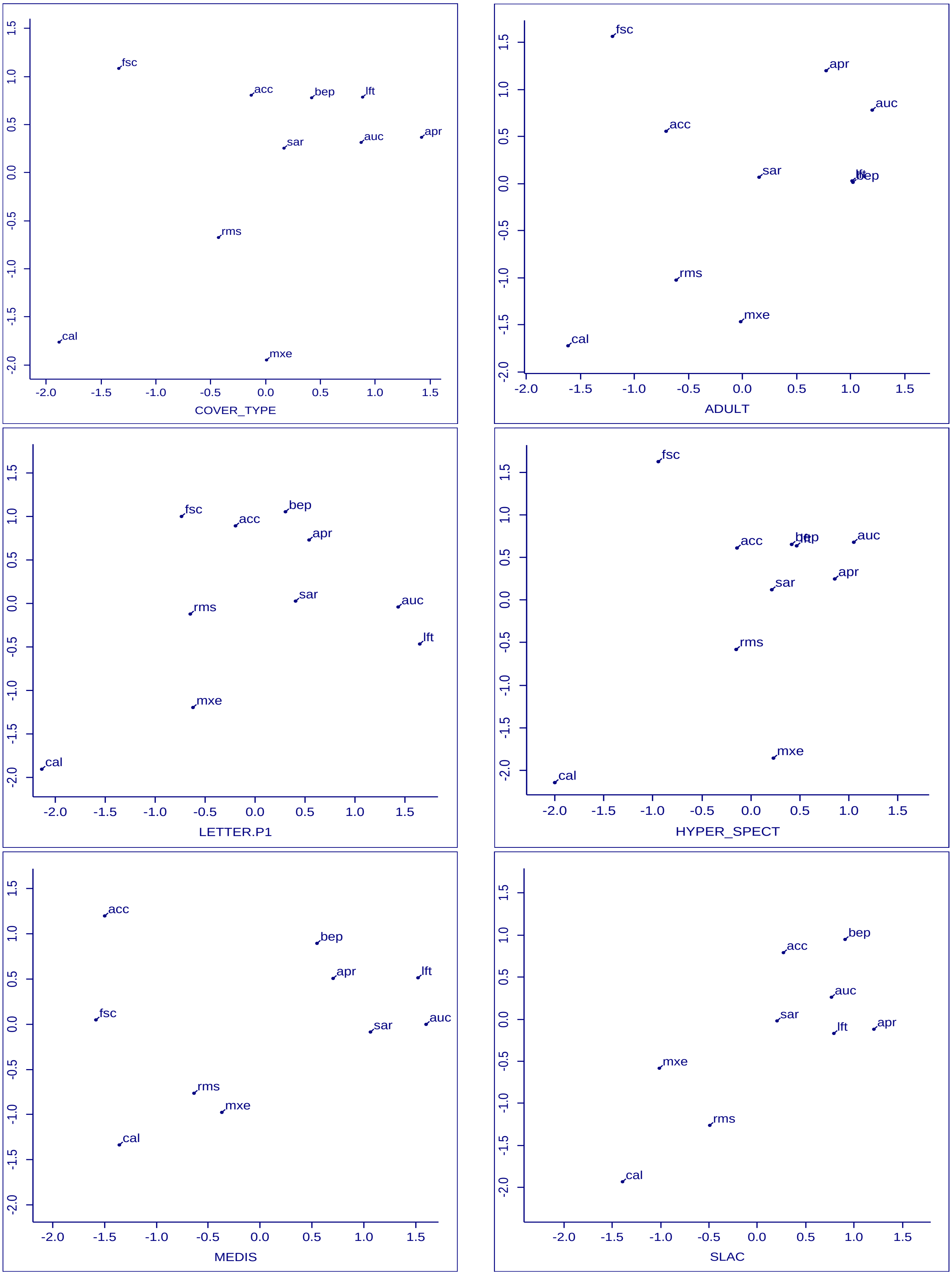

Figure 4. 2-D MDS plots for six of the seven test problems. The seventh problem yields a similar plot and is omitted only to save space. The missing plot is for one of the LETTER problems.

Table 3. Average rank correlations between metrics

	ACC	FSC	LFT	AUC	APR	BEP	RMS	MXE	CAL	SAR	MEAN
ACC	1.00	0.87	0.85	0.88	0.89	0.93	0.87	0.75	0.56	0.92	0.852
FSC	0.87	1.00	0.77	0.81	0.82	0.87	0.79	0.69	0.50	0.84	0.796
LFT	0.85	0.77	1.00	0.96	0.91	0.89	0.82	0.73	0.47	0.92	0.832
AUC	0.88	0.81	0.96	1.00	0.95	0.92	0.85	0.77	0.51	0.96	0.861
APR	0.89	0.82	0.91	0.95	1.00	0.92	0.86	0.75	0.50	0.93	0.853
BEP	0.93	0.87	0.89	0.92	0.92	1.00	0.87	0.75	0.52	0.93	0.860
RMS	0.87	0.79	0.82	0.85	0.86	0.87	1.00	0.92	0.79	0.95	0.872
MXE	0.75	0.69	0.73	0.77	0.75	0.75	0.92	1.00	0.81	0.86	0.803
CAL	0.56	0.50	0.47	0.51	0.50	0.52	0.79	0.81	1.00	0.65	0.631
SAR	0.92	0.84	0.92	0.96	0.93	0.93	0.95	0.86	0.65	1.00	0.896

We expected AUC and average precision to behave very similarly and thus have high rank correlation. But we are surprised to see that Lift has such high correlation to AUC. Note that because Lift has high correlation to AUC, and AUC has high correlation to average precision, it is not surprising that Lift also has high correlation to average precision. As expected, break-even point is highly correlated with the other two ordering metrics, AUC and average precision. But the high correlation between accuracy and break-even point is somewhat surprising and we currently do not know how to explain this.

The weakest correlations are all between the calibration metric (CAL) and the other metrics. On average, CAL correlates with the other metrics only about 0.63. We are surprised how low the correlation is between probability calibration and other metrics, and are currently looking at other measures of calibration to see if this is true for all of them.

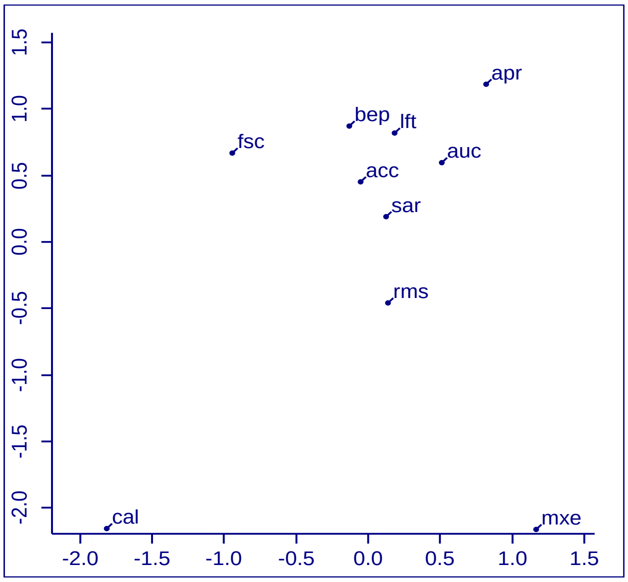

Figure 5. MDS using rank correlation

Figure 5 shows an MDS plot for the metrics when distance between metrics is calculated as $1 - rank\_correlation$, making MDS insensitive to how the metrics are scaled. (Distances based on $1 - rank\_correlation$ do not respect the triangle inequality so this is not a *proper* metric space.) The overall pattern is similar to that observed in the MDS plots in Figure 3. CAL is at one end of the space far from the other metrics. Cross-entropy is closest to RMS, though not as close as in the other plots. Cross-entropy and RMS have high rank correlation, but because cross-entropy has lower rank-correlation to other most metrics than RMS, it is pushed far from RMS which is close to other metrics in the MDS plot. APR and AUC are at the other end of the space farthest from CAL. FSC is in the upper left side of the space. ACC and RMS are near the center of the space.

8 SAR: A General Purpose Metric

When applying supervised learning to data, a decision must be made about what metric to train to and what metric to use for model selection. Often the learning algorithm dictates what metrics can be used for training, e.g. it is difficult to train a neural net for metrics other than RMS or MXE. But there usually is much more freedom when selecting the metric to use for model selection, i.e. the metric used to pick the best learning algorithm and the best parameters for that algorithm.

If the correct metric for the problem is known, model selection probably should be done using that metric even if the learning algorithm cannot be trained to it. What should be done when the *correct* metric is not known? The MDS plots and correlation analysis suggest that RMS is remarkably well correlated with the other measures, and thus might serve as a good general purpose metric to use when a more specific optimization criterion is not known.

We wondered if we could devise a new metric more centrally located than RMS and with better correlation to the other metrics. Rather than devise a completely new metric, we tried averaging several of the well behaved metrics into a new metric that might be more robust than each one individually. SAR combines **S**quared error, **A**ccuracy, and **R**OC area into one measure: $SAR = (ACC + AUC + (1 - RMS))/3$. We chose these metrics for SAR for three reasons:

1. we wanted to select one metric from each metric group: the threshold metrics, the ordering metrics, and the probability metrics
2. ACC, AUC, and RMS seemed to be the most popular metric from each of these groups, respectively
3. these three metrics are well correlated to the other metrics in their groups, and in the MDS plots lie closest to the other metrics in their groups

As can be seen from the MDS plots and in the tables, SAR behaves differently from ACC, AUC, and RMS alone. In Table 3 SAR has higher mean rank correlation to other metrics than any other metric. In the MDS plots, SAR tends to be more consistently centrally located than other metrics. And in Table 4 it is the metric that best reflects the ordering by mean performance of the seven learning methods.

These results suggest that of the ten metrics we examined, SAR is the metric that on average is most correlated with the other metrics, both separately, and in groups. SAR is even more representative than RMS (though RMS also is *very* good). In an experiment where SAR was used for model selection, SAR outperformed eight of the nine metrics in selecting the models with the best overall, and tied with RMS. We believe our results suggest that SAR is a robust combination of three popular metrics that may bey appropriate when the correct metric to use is not known, though the benefit of SAR over RMS is modest at best. Attempts to make SAR better by optimizing the weights given to ACC, AUC, and RMS in the SAR average did not significantly improve SAR compared to equal weights for the three metrics. We are very impressed at how well behaved RMS alone is and are currently working to devise a better SAR-like metric that yields more improvement over RMS alone.

9 Performances by Metric

Table 4 shows the normalized performance of each learning algorithm on the nine metrics. (CAL is scaled so that the mean observed CAL score is 0.0 and the maximum observed CAL score is 1.0) For each test problem we find the best parameter settings for each learning algorithm and compute it's normalized score. Each entry in the table averages these scores across the seven problems. The last two columns are the mean normalized scores over the nine metrics, and the SAR performance. Higher scores indicate better performance. The models in the table are ordered by mean overall performance. We have written a separate paper to compare the performance of the learning methods to each other on these metrics, but there are a few interesting relationships between learning algorithms and metrics that are worth discussing in the context of this paper.

Overall, the best performing models are neural nets, SVMs, and bagged trees. Surprisingly, neural nets outperform all other model types if one averages over the nine metrics. ANNs appear to be excellent general purpose learning methods. This is not to say that ANNs are the best learning algorithm – they only win on RMS and CAL, but because they rarely perform poorly on any problem or metric, they have excellent overall performance.

The SVMs perform almost as well as ANNs. Note that SVM predictions on $[-\infty, +\infty]$ are not suitable for measures like cross entropy, calibration, and squared error. SVMs do well on these metrics because we use Platt's method [8] to transform SVM predictions to calibrated probabilities. Like neural nets, SVMs appear to be a safe, general purpose, high performance learning method once their predictions have been calibrated by a method such as Platt scaling.

Although single decision trees yield very poor performance, bagged trees yield performance nearly as good as neural nets and SVMs. Bagging improves decision tree performance on all metrics, and yields particularly large improvements on the probability metrics. Like neural nets and SVMs, bagged trees appear to be a safe, general purpose, high performance learning method.

Boosted trees outperform all other learning methods on ACC, LFT, ROC, APR, and BEP. Boosting wins 2 of 3 threshold metrics and 3 of 3 rank metrics, but performs poorly on the probability metrics: squared error, cross entropy, and calibration. Maximum margin methods such as boosted trees yield poorly calibrated probabilities. (SVMs perform well on these because Platt scaling "undoes" the maximum margin.) Overall, boosting wins 5 of the 6 metrics for which it is well suited, and would easily be the top performing learning method if we consider only the 6 threshold and ordering metrics.

The KNN methods were not competitive with the better algorithms, but might done better with larger train sets. Single decision trees also did not perform as well as most other methods, probably because recursive partitioning runs out of data quickly with 4k train sets, and because small trees are not good at predicting probabilities [9]. We tested many different kinds of decision trees, including smoothed unpruned trees, and then picked the best, so the poor performance of trees here is not due to any one tree type being inferior, but because all of the many tree types we tested did not perform as well as other methods.

Interestingly, boosting stump models does not perform as well as boosting full decision trees. Boosted stumps do outperform single trees on 5 of the 6 threshold and rank metrics. Their last-place ranking below decision trees is due to their extremely poor performance on the three probability measures.

10 Related Work

There is not a large literature comparing performance metrics. The closest work to ours is by Flach [7]. In this work Flach uses the ROC space to understand and compare different metrics. He analyzes accuracy, precision, weighted relative accuracy and several decision tree splitting criteria.

The STATLOG project [6] performed a large scale empirical evaluation of a number of learning algorithms in 1995. STATLOG compared the performance of the different algorithms, and also did an analysis of how the predictions made by the algorithms compared to each other. STATLOG, however, did not compare performance using different metrics.

11 Discussion and Conclusions

Our analysis allows us to draw a variety of conclusions which we summarize here. If the goal is to maximize accuracy, but the model needs a continuous performance metric (e.g. using backpropagation to train a neural net), it probably is better to train the model using squared error instead of cross entropy because squared error sits closer to accuracy in metric space. This result is surprising since cross entropy is the theoretically preferred loss function for binary classification. We suspect cross entropy is not as robust as squared error on real data sets because real data sometimes contains class noise that cross entropy is very sensitive to.

Table 4. Normalized scores for each learning algorithm by metric (average over seven problems)

MODEL	ACC	FSC	LFT	AUC	APR	BEP	RMS	MXE	CAL	MEAN	SAR
ANN	0.9236	0.9217	0.9645	0.9665	0.9449	0.9470	**0.8915**	0.8919	**0.9752**	**0.9363**	0.9107
SVM	0.8848	**0.9225**	0.9546	0.9632	0.9434	0.9492	0.8898	0.8951	0.9315	**0.9260**	**0.9119**
BAG-DT	0.8643	0.8733	0.9618	0.9708	0.9486	0.9304	0.8646	**0.8999**	0.8253	**0.9043**	0.9068
KNN	0.7978	0.8774	0.9316	0.9427	0.8968	0.9108	0.7846	0.7677	0.8921	**0.8668**	0.8633
BST-DT	**0.9337**	0.9187	**0.9773**	**0.9809**	**0.9688**	**0.9693**	0.6305	0.6357	0.5235	**0.8376**	0.8786
DT	0.6610	0.8360	0.8838	0.8845	0.8090	0.8267	0.6207	0.6681	0.7831	**0.7748**	0.7812
BST-STMP	0.7797	0.8050	0.9173	0.9236	0.8715	0.8576	0.3147	0.2983	0.4266	**0.6883**	0.6659

Squared error is a remarkably robust performance metric that has higher average correlation to the other metrics than any other metric except SAR. Squared error appears to be an excellent general purpose metric.

Many models achieve excellent performance on the ordering metrics AUC, APR, and BEP without making predictions that yield good probabilities. For example, the k-nearest neighbor models with the best ROC performance use values of K that are so large that most of the predictions are close to p, the fraction of positives in the data. This yields predictions that are poor when viewed as probabilities, yet small differences between these predicted values are sufficient to provide for good ordering.

As expected, maximum margin methods such as boosting and SVMs yield excellent performance on metrics such as accuracy for which they are designed. Surprisingly, however, the maximum margin methods also yield excellent performance on the ordering metrics. We had not expected that maximizing distances to decision boundaries would provide a good basis for ordering cases that fall far from those boundaries.

Although boosted trees perform well on accuracy and ROC, they perform poorly on probability metrics such as squared error and cross entropy. This poor performance on probability metrics is a consequence of boosting being a maximum margin method. SVMs do not exhibit this problem because we scale SVM predictions with Platt's method; Linearly scaling SVM predictions to $[0, 1]$ does not work well.

Neural nets trained with backpropagation have excellent overall performance because, unlike boosting, they perform well on all metrics including the probability metrics RMS, MXE, and CAL. We believe part of the reason why the neural nets perform so well is that they were trained with backpropagation on squared error, and as we have seen squared error is an excellent metric.

The three ordering metrics, AUC, APR, and BEP, cluster close in metric space and exhibit strong pairwise correlations. These metrics clearly are similar to each other and somewhat interchangeable. We originally grouped LFT with the threshold metrics ACC and FSC, but the results suggest that LFT behaves more like BEP, an ordering metric. We now would group LFT with BEP in the ordering metrics along with AUC and APR.

The metric space for the ten metrics has three or more significant dimensions. The ten metrics do not all measure the same thing. Different performance metrics yield different tradeoffs that are appropriate in different settings. No one metric does it all, and the metric optimized to or used for model selection does matter. The SAR metric that combines accuracy, ROC area, and squared error appears to be a good, general purpose metric, but RMS is so good that SAR may not provide much benefit over using RMS alone. We hope that additional research in this area will enable us to design better metrics, and will shed more light on which metrics are most appropriate to use in different settings.

12 Acknowledgments

Thanks to Geoff Crew and Alex Ksikes for help running some of the experiments. Thanks to the creators of XGVIS and XGOBI for the interactive MDS software used to generate the MDS plots. Thanks to collaborators at Stanford Linear Accelerator for the SLAC data, and to Tony Gualtieri at NASA Goddard for help with the Indian Pines data.

A Performance Metrics

accuracy: probably the most widely used performance metric in Machine Learning. It is defined as the proportion of correct predictions the classifier makes relative to the size of the dataset. If a classifier has continuous outputs (e.g. neural nets), a threshold is set and everything above this threshold is predicted to be a positive.

root-mean-squared-error (RMSE): widely used in regression, it measures how much predictions deviate from the true targets. [2]RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum(Pred(C) - True(C))^2} \qquad (1)$$

mean cross entropy (MXE): is used in the probabilistic setting when interested in predicting the probability that an example is positive (1). It can be proven that in this setting minimizing the cross entropy gives the maximum likelihood hypothesis. mean cross entropy is defined as:

$$\begin{aligned} MXE = \quad & -\tfrac{1}{N}\sum(True(C) * ln(Pred(C)) + \\ & (1 - True(C)) * ln(1 - Pred(C))) \end{aligned} \qquad (2)$$

(The assumptions are that $Pred(C) \in [0, 1]$ and $True(C) \in \{0, 1\}$)

receiver operating characteristic (ROC): has it's roots in WWII in the early days of radar where it was difficult to distinguish between true positives and false positives. ROC is a plot of sensitivity vs. (1-specificity) for all possible thresholds. Sensitivity is the defined as $P(Pred = positive|True = positive)$ and is approximated by the fraction of true positives that are predicted as positive (this is the same as recall). Specificity is $P(Pred =$

[2] Root-mean-squared error is applicable to binary classification settings where the classifier outputs predictions on $[0, 1]$ that are compared with the true target labels on $\{0, 1\}$.

$negative|True = negative$). It is approximated by the fraction of true negatives predicted as negatives. AUC, the area under the ROC curve, is used as a summary statistic. ROC has a number of nice properties that make it more principled than similar measures such as average precision. AUC is widely used in fields such as medicine, and recently has become more popular in the Machine Learning community.

lift: often used in marketing analysis, Lift measures how much better a classifier is at predicting positives than a baseline classifier that randomly predicts positives (at the rate observed for positives in the data). The definition is:

$$LIFT = \frac{\%\text{of true positives above the threshold}}{\%\text{of dataset above the threshold}} \quad (3)$$

Usually the threshold is set so that a fixed percentage of the dataset is classified as positive. For example, suppose a marketing agent wants to send advertising to potential clients, but can only afford to send ads to 10% of the population. A classifier is trained to predict how likely a client is to respond to the advertisement, and the ads are sent to the 10% of the population predicted most likely to respond. A classifier with optimal lift will get in this set as many clients as possible that will respond to the ads.

precision and recall : These measures are widely used in Information Retrieval. Precision is the fraction of examples predicted as positive that are actually positive. Recall is the fraction of the true positives that are predicted as positives. These measures are trivially maximized by not predicting anything, or predicting everything, respectively, as positive. Because of this these measures often are used together. There are different ways to combine these measures as described by the next 4 metrics.

precision-recall F-score: is the harmonic mean of the precision and recall at a given threshold.

precision at a recall level: as the name suggests, set the threshold such that you have a given recall and the precision for this threshold is computed.

precision-recall break-even point: is defined as the precision at the point (threshold) where precision equals recall.

average precision: usually is computed as the average of the precisions at eleven evenly spaced recall levels.

CAL is based on reliability diagrams [2]. It is calculated as follows: order all cases by their predicted value, and put cases 1-100 in the same bin. Calculate the percentage of these cases that are true positives. This approximates the true probability that these cases are positive. Then calculate the mean prediction for these cases. The absolute value of the difference between the observed frequency and the mean prediction is the calibration error for this bin. Now take cases 2-101, 3-102, ... and compute the errors in the same way for each of these bins. CAL is the mean of these binned calibration errors.

B Parameter Settings

We use the following parameter settings and algorithm variations for the seven learning methods:

KNN: we use 26 values of K ranging from $K = 1$ to $K = |trainset|$. We use KNN with Euclidean distance and Euclidean distance weighted by gain ratio. We also use distance weighted KNN, and locally weighted averaging. The kernel widths for locally weighted averaging vary from 2^0 to 2^{10} times the minimum distance between any two points in the train set.

ANN: we train nets with gradient descent backprop and vary the number of hidden units {1, 2, 4, 8, 32, 128} and the momentum {0, 0.2, 0.5, 0.9}. We don't use validation sets to do weight decay or early stopping. Instead, for each performance metric, we examine the nets at many different epochs.

DT: we vary the splitting criterion, pruning options, and smoothing (Laplacian or Bayesian smoothing). We use all of the tree models in Buntine's IND package: Bayes, ID3, CART, CART0, C4, MML, and SMML. We also generate trees of type C44 (C4 with no pruning), C44BS (C44 with Bayesian smoothing), and MMLLS (MML with Laplacian smoothing). See [9] for a description of C44.

BAG-DT: we bag at least 25 trees of each type. With **BST-DT** we boost each tree type. Boosting can overfit, so we consider boosted DTs after {2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048} steps of boosting. With **BST-STMP** we use stumps (single level decision trees) with 5 different splitting criteria, each boosted {2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192} steps.

SVMs: we use most kernels in SVMLight [5] {linear, polynomial degree 2 & 3, radial with width {0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2}} and vary the regularization parameter C by factors of ten from 10^{-7} to 10^3. The output range of SVMs is $[-\infty, +\infty]$ instead of $[0, 1]$. To make the SVM predictions compatible with other models, we use Platt's method to convert SVM outputs to probabilities by fitting them to a sigmoid [8]. Without scaling, SVMs would have poor RMS and it would not be possible to calculate MXE and CAL.

REFERENCES

[1] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[2] M.H. DeGroot and S.E. Fienberg, 'The comparison and evaluation of forecasters', *Statistician*, **32**(1), 12–22, (1982).

[3] P. Giudici, *Applied Data Mining*, John Wiley and Sons, New York, 2003.

[4] A. Gualtieri, S. R. Chettri, R.F. Cromp, and L.F. Johnson, 'Support vector machine classifiers as applied to aviris data', in *Proc. Eighth JPL Airborne Geoscience Workshop*, (1999).

[5] T. Joachims, 'Making large-scale svm learning practical', in *Advances in Kernel Methods*, (1999).

[6] R. King, C. Feng, and A. Shutherland, 'Statlog: comparison of classification algorithms on large real-world problems', *Applied Artificial Intelligence*, **9**(3), 259–287, (May/June 1995).

[7] P.A.Flach, 'The geometry of roc space: understanding machine learning metrics through roc isometrics', in *Proc. 20th International Conference on Machine Learning (ICML'03)*, pp. 194–201. AAAI Press, (January 2003).

[8] J. Platt, 'Probabilistic outputs for support vector machines and comparison to regularized likelihood methods', in *Advances in Large Margin Classifiers*, eds., A.J. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, pp. 61–74, (1999).

[9] F. Provost and P. Domingos, 'Tree induction for probability-based rankings', *Machine Learning*, **52**(3), (2003).

[10] Foster J. Provost and Tom Fawcett, 'Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions', in *Knowledge Discovery and Data Mining*, pp. 43–48, (1997).