Supplemental Material to "Classifier-Adjusted Density Estimation for Anomaly Detection and One-Class Classification"

8 Implementation Details

8.1 Density Estimation In our implementation of the kernel density estimator (KDE), we fit a univariate KDE to each attribute independently. The bandwidth for each dimension is chosen using a plug-in selector, via the ks package in R [1], and the kernel itself is Gaussian. The joint density estimate is defined to be the product of the independent marginal densities. A common alternative formulation is to use one full-dimensional Gaussian kernel per data point. Our KDE is equivalent to a special case of that formulation, one in which we constrain the kernel's covariance to be diagonal and choose the bandwidth for each dimension as above. When sampling from the KDE, we use an efficiency trick suggested in Hastie et al.: we draw each attribute not from its corresponding KDE but directly from the training data [2].

For nominal attributes, we provide only two marginally independent options. When the density estimate is described as uniform, it is also uniform for the nominal attributes. When it is described as Gaussian or KDE, we use a frequency estimate of the attribute values. For density estimates that combine attributes independently, discrete attributes can be readily mixed with continuous.

The Bayesian network implementation we use is R's bnlearn package [6]. This package does not handle mixed attribute types, so for data sets containing both types, we make the simplifying assumption that the two sets of attributes are independent: we build one Bayes net for numeric attributes and one for nominal, then multiply the probabilities together.

8.2 Local Outlier Factor With LOF, we use Weka's default settings: $minPtsLB = 10$, $minPtsUB = 40$. For bagged LOF, we use the settings described by Lazarevic and Kumar [3]: we create 10 bags, each using a random selection of $\lfloor \frac{d}{2} \rfloor$ to $d - 1$ attributes, where d is the original dimensionality. We aggregate the scores using their *Cumulative Sum* approach, in which the final score for an item is the sum of its scores in the 10 bags.

LOF is usually described as an unsupervised approach, run on a mixture of normal and anomalous data. To make a fair comparison with CADE, we give both methods the same "training data" in every experiment. For CADE, this training data is used in conjunction with generated artificial anomalies for training the classifier. For LOF, these instances are the ones used as nearest neighbors by subsequent query points.

9 Assembling CADE Components

While one of the strengths of CADE is its ease of construction from commonly available tools, there are a number of easily-overlooked details. This section describes some considerations for making the components function well together.

9.1 Classifier Properties It is important that the classifier be able to learn a class distinction from the training data. Obviously, a near-random classifier will not be a useful contribution to the density estimate. Unfortunately, it is less obvious how easy it is to create such a near-random classifier in this setting. Section 2.1 mentioned one way this can happen: if the initial density estimate exactly matches the positive data. In that case, an ideal classifier would predict 0.5 on all test data, but a non-ideal classifier may overfit and make sub-random predictions. Another way this can happen is when (a) attribute by attribute, the initial density estimate matches the training data exactly, and (b) the classifier only considers one attribute at a time. Property (a) holds when, for the artificial negatives, we sample each attribute directly from the training data, as we do with KDEs. Property (b) holds with classifiers such as naive Bayes, logistic regression, or the C4.5 decision tree (when choosing its root node). With these combinations, we found poor performance. Beyond avoiding these particular cases, we suggest a robust way to check if the classifier is helpful: calculate the classifier's performance at separating the training classes (on a held-out sample). If the performance is near-random or below, the classifier adjustment to the initial density estimate is likely to be detrimental.

Another potential pitfall is the use of classifiers with good predictive performance but poor probability estimates. With some classifiers, we observed large blocks of tie scores in the classifier's predictions. Since we evaluate performance based on the position of the anomalies within the ranking, rankings with large numbers of ties are at a disadvantage. Although both KNN and random forest have been found to give consistent estimates [4], the initial settings we tried did not. With KNN, an initial $k = 20$ was far too small, and with random forest, when using 100 trees grown to purity (with no pruning or minimum leaf size), the average of the trees' probability estimates was often exactly 0 or 1. Experimenting informally, we found better results with KNN using $k = 200$ and weighting neighbors by the inverse of their distance. With random forest, increasing the minimum leaf size had little effect, but backfitting helped. In backfitting, after the trees are grown to purity, a held-out fraction of training data is added to the trees, not changing their structure but adjusting the probability distributions at the leaves. These changes reduced the number of ties and to some extent improved AUC. More generally, we conjecture that producing more consistent classifier predictions—beyond merely avoiding tie scores—will improve performance when the classifiers are used with non-uniform density estimates.

We also observe that the optimal settings for classifiers vary with the distribution of artificial negatives. As an extreme example, with decision trees, using C4.5 with Laplace smoothing [5] gives better results than CART when artificial negatives are uniform, but worse when the artificial negatives are sampled from the training data. In the latter situation, C4.5 builds no trees at all, finding no features significant, but CART builds useful trees.

9.2 Properties of Naive Bayes If the naive Bayes classifier is ineffective in combination with a KDE initial density, its behavior with a uniform or Gaussian density is oddly familiar. As remarked above, naive Bayes treats each attribute independently. For each attribute, it estimates the distributions of the positive and the negative training data. To compute the odds term needed in Eq. (2.1), $\frac{P(C=T|X)}{1-P(C=T|X)} = \frac{P(C=T|X)}{P(C=A|X)}$, it takes the ratio of the two distributions at the location of the test point.

We found the best performance with naive Bayes by enabling Weka's option for kernel density estimation. When the initial density is uniform, the two distributions estimated by naive Bayes are then kernel density estimates of the positive data and of the uniformly distributed artificial negatives. The odds term, once all dimensions have been multiplied together, comes out to $\frac{KDE(X|T)}{P(X|A)}$. Here, $KDE(X|T)$ is a naive-Bayes-generated kernel density estimate of the training data, and $P(X|A)$ is (approximately) a constant that will cancel out the matching term in Eq. (2.1).

The result of this is that the score CADE produces using Uniform + NB is identical, apart from artifacts of sampling and estimation, to that produced by a KDE initial density (with no classifier adjustment). The same should hold with a Gaussian (or any marginally independent) initial density, even though $P(X|A)$ is no longer constant: Gaussian + NB should be equivalent to an unadjusted KDE. If the naive Bayes implementation used a Gaussian estimate internally instead of a KDE, CADE's output with Uniform + NB or Gaussian + NB would match the unadjusted Gaussian density estimate instead. In practice, we did generally observe Uniform + NB to match the unadjusted KDE; however, contrary to the claim above, Gaussian + NB performed worse.

9.3 Combining Probability Estimates One way to lose information when multiplying terms together is to allow any of the terms to be zero or infinity. A zero can originate either from the classifier ($P(C = T|X) = 0$) or from the density estimate ($P(X|A) = 0$). When marginal density estimates are combined independently, a zero can come from any dimension and is more likely as the dimensionality increases. An infinity is caused by the classifier predicting $P(C = T|X) = 1$. These values cause the rest of the terms to be ignored and result in unnecessary ties in the ranking. To avoid this problem, we smooth the values before multiplying them, changing zeros and ones to ϵ and $(1 - \epsilon)$, respectively, choosing ϵ as a function of the minimum and maximum values otherwise seen in the data. One benefit of smoothing is that, when the density estimate is calculated as a product of marginals, items having $P(X|A) = 0$ in k dimensions are generally scored as more anomalous than those having $P(X|A) = 0$ in $k - 1$ dimensions. In our experiments, smoothing rarely hurts performance and in some cases improves it greatly.

Overall, when choosing components for CADE, a uniform initial density is the simplest to implement, and it only requires the classifier to produce a good ranking, not a consistent probability estimate. On the other hand, with a more complex initial density estimate, the classifier can focus on deviations from that estimate, and if the classifier predicts tie scores, the density estimate term will give them an ordering.

10 Additional Experiments

Section 5.1 showed the robustness of CADE and LOF when faced with irrelevant attributes. In that experiment, each set of training data was sampled from a

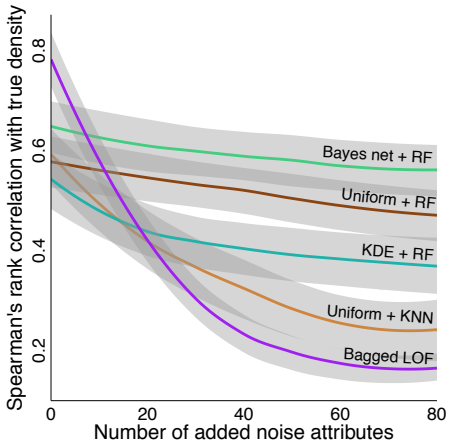

Figure 5: Synthetic experiments evaluating performance as uniform noise attributes are added. Experimental setup is the same as for Fig. 3, but test data is drawn from a uniform distribution that matches the range of the positive data.

distribution comprising a five-dimension ~~Gaussian~~ [Ed.: mixture of three Gaussians] plus noise attributes, and each test set was sampled from the same distribution as its training set. In this section we repeat this experiment using a different test set. Once again, the ground truth ranking of points is defined as their density according to the training distribution. But this time, the test set—the data being ranked—is sampled uniformly from a hypercube defined by the range of training data. Such a test set contains more low-density points than before. Fig. 1 shows the results of this experiment. Similar to Fig. 3, LOF degrades the most steeply, followed by Uniform + KNN, and the random forest methods are most robust. However, in this setup LOF actually performs better than CADE when there are few noise attributes. In addition, Bayes net + RF now performs best as noise attributes are added, above both KDE and Uniform.

References

[1] T. Duong. ks: Kernel density estimation and kernel discriminant analysis for multivariate data in R. *Journal of Statistical Software*, 21(7):1–16, 10 2007.

[2] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag, 2001.

[3] A. Lazarevic and V. Kumar. Feature bagging for outlier detection. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 157–166, New York, NY, USA, 2005. ACM.

[4] J. D. Malley, J. Kruppa, A. Dasgupta, K. G. Malley, and A. Ziegler. Probability machines: Consistent probability estimation using nonparametric learning machines. *Methods of Information in Medicine*, 51(1):74–81, 2012.

[5] F. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, 2003.

[6] M. Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010.