Iterative Relief

Bruce Draper
Computer Science Department
Colorado State University
Fort Collins CO 80523-1873
USA
draper@cs.colostate.edu

Carol Kaito
Computer Science Department
Colorado State University
Fort Collins CO 80523-1873
USA
kaito@cs.colostate.edu

José Bins
Faculdade de Informatica
Pontificia Universidade Catolica
Porto Alegre RS 90619-900
Brazil
bins@inf.pucrs.br

Abstract

Feature weighting algorithms assign weights to features according to their relevance to a particular task. Unfortunately, the best-known feature weighting algorithm, ReliefF, is biased. It decreases the relevance of some features and increases the relevance of others when irrelevant attributes are added to the data set. This paper presents an improved version of the algorithm, Iterative Relief, and shows on synthetic data that it removes the bias found in ReliefF. This paper also shows that Iterative Relief outperforms ReliefF on the task of cat and dog discrimination, using real images.

1   Introduction

In machine learning, feature weighting is the process of assigning weights to features, based on their relevance to a task. Once relevance weights have been assigned to features, the weights can be used in a number of ways, including feature selection (by discarding features below a threshold) and data compression (by projecting samples onto the weight vector). Many classifiers can also exploit feature relevance measures by weighting the features as part of an underlying distance measure.

Feature weighting algorithms can be divided roughly into two categories. Parametric approaches fit the data to known distributions, and then measure relevance in terms of that model. Linear discriminant analysis (LDA), for example, fits normal distributions to the samples in each class, and finds the feature weights that maximize the inter-class distance while minimizing the intra-class distance.

Non-parametric approaches make no assumptions about the distribution of the data.

Perhaps the best known non-parametric feature weighting algorithm is ReliefF (Kononenko 1994). ReliefF is a completely local algorithm. It measures the relevance of a feature at a sample in terms of the difference between the sample and other nearby samples of the same class (the so-called "hits") and nearby samples of other classes ("misses"). The relevance of a feature overall is the average relevance of the feature across all training samples.

Unfortunately, Bins showed that ReliefF is biased against non-monotonic features (Bins 2000; Bins and Draper 2001). The bias is particularly extreme in cases where the number of features greatly exceeds the number of training samples, as is common in many computer vision applications. But although Bins was able to clearly explain the bias, his approach to removing it was ad-hoc. This paper presents a better algorithm for unbiased non-parametric relevance estimation called Iterative Relief.

2   Background

Relief was first introduced by Kira and Rendell in 1992 (Kira and Rendell 1992). The basic idea is to measure the relevance of features in the neighborhoods around target samples. For each target sample, Relief finds the nearest sample in feature space of the same category, called the "hit" sample. It also finds the nearest sample of the other category, called the "miss" sample. The relevance of feature f near the target sample is measured as:

$$W_f += \frac{\left|f_{t\arg et} - f_{miss}\right| - \left|f_{t\arg et} - f_{hit}\right|}{Range(f)}$$

The overall relevance of feature f is the sum of its relevance for all target samples. Target samples may be selected randomly from the training set or, if the training set is small,

every training sample may be used as a target sample.

This basic algorithm was quickly extended by Kononenko (Kononenko 1994). His ReliefF algorithm is more robust than the original because it selects a set of nearby hits and a set of nearby misses for every target sample and averages their distances. Weights are updated according to:

$$W_f += \frac{\sum_{m \in misses} \left| f_{t\,\arg et} - f_m \right| - \sum_{h \in hits} \left| f_{t\,\arg et} - f_h \right|}{|misses| \cdot |hits| \cdot Range(f)}$$

This minimizes the effects of spurious samples. ReliefF also extends Relief to multi-class problems by defining a different set of "miss" samples for every category. (Kononenko shows that this works better than dividing samples into the target class and "other" (Kononenko 1994).)

In general, Relief and ReliefF assign relevance to features based on their ability to disambiguate similar samples, where similarity is defined by proximity in feature space. Relevant features accumulate high positive weights, while irrelevant features retain near-zero weights.

Relief is reasonably effective at measuring relevance, and it is very fast. Even if every training instance is used as a target sample to update the relevance weights, the complexity of ReliefF is still only $O(n^2 f)$, where n is the number of training instances and f is the number of features. This makes it ideal for domains where the number of features is much larger than the number of training samples. This is commonly the case in appearance-based vision, where every pixel (or every pair of pixels, in the case of probing) is considered a feature. It has also been noted that Relief works well even in the presence of strong but unknown dependencies among features (Caruana and Freitag 1994).

Despite its favorable performance, ReliefF has rarely been used for computer vision (although see (Ishiguro, Sato et al. 1996; Bins and Draper 2001)). This may be because of concerns about its semantics: what exactly do the weights computed by ReliefF represent? In the case of binary features, Kononenko related the weights to standard information measures (Kononenko 1994). More recently, Robnik-Šikonja and

Kononenko have related ReliefF weights to the probability that flipping a (binary) feature value would lead to a change in category for the sample (Robnik-Šikonja and Kononenko 2001). Unfortunately, this analysis cannot be extended to continuous valued features without making assumptions about the feature distributions, and if the distributions assumptions are valid it makes more sense to fit the data to the distribution model and use a parametric relevance weighting technique.

Instead, it is better to analyze ReliefF's weights directly in terms of what they maximize. The feature weights computed by ReliefF minimize the expected distance between a sample and the k nearest samples of the same class, while maximizing the expected distance between a sample and the k nearest samples of the other class. In other words, ReliefF's relevance measure optimizes the performance of a k-NN classifier. This is in contrast to LDA, which fits normal distributions to the categories, minimizing the within-class scatter and maximizing the between-class scatter. LDA therefore maximizes the performance of a traditional (Gaussian) maximum likelihood classifier.

3   Bias in Relief

Unfortunately, Bins showed that ReliefF is biased against non-monotonic features, and that this bias gets worse as the number of irrelevant features in the data set increases (Bins 2000; Bins and Draper 2001). More specifically, Bins looked at ReliefF in the context of a regression problem, where the goal is to predict the value of a (continuous) function, based on sample features. In this context, it is possible to make a 2D plot with feature f on the x-axis and the target function value on the y-axis. (Samples are points in this plot.) A feature is non-monotonic if its plot is neither non-decreasing nor non-increasing, i.e. if it has internal optima. Figure 1 shows synthetic monotonic and non-monotonic features. Bins showed that the weights computed for monotonic features were consistently higher than the weights computed for non-monotonic features. In addition, the discrepancy in weights between monotonic and non-monotonic features increases as more noise features are added to the data set.

The bias observed by Bins is large. Figure 2 shows the weights computed by ReliefF for the synthetic features in Figure 1, as more and more noise (i.e. purely random) features are added to the data set. Even before noisy features are added, the weight of the monotonic feature is twice the weight of the non-monotonic one, even though both features are noise-free and completely sufficient to predict the target function. As noisy features are added to the data set, the measured relevance of the monotonic feature increases by about 10%, while the measured relevance of the non-monotonic feature decreases by over 25%.

Non-monotonic features are very common in practice. Many features have an acceptable range of values for a class, while extreme values either above or below this range indicate another class. In a two-class classification problem, this creates a version of Figure 1 where the y-axis is discretized to two values, but the feature is nonetheless non-monotonic.

Figure 1: The synthetic non-monotonic feature (top) and the synthetic monotonic feature (bottom) used. The horizontal axes indicate the feature value, while the vertical axes shows the target function value.
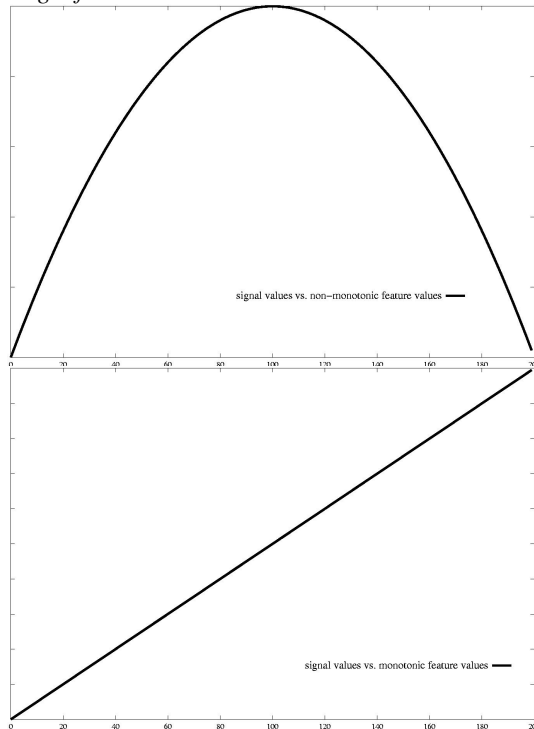

ReliefF's bias (which is shared by the original Relief) can be traced to the relative roles of

categories and the distance measure. ReliefF first groups samples according to their category. It then uses the distance measure to find the most similar samples of the same class ("hits") and the other classes ("misses").

In the case of a monotonic feature, the distance measure is almost irrelevant. The fact that two samples are from the same class bounds the possible difference in their feature values. Therefore monotonic features are given positive relevance weights, even when the distance function is unreliable. The difference between feature values of two samples from different classes, on the other hand, is bounded only by the range of the feature.

Figure 2: Plots of the change in weight for the synthetic features in Figure 1 as random features are added to the data set -- non-monotonic (top), monotonic (bottom). The horizontal axis represents the number of noise features in each sample, while the vertical axis indicates feature relevance as computed by ReliefF.
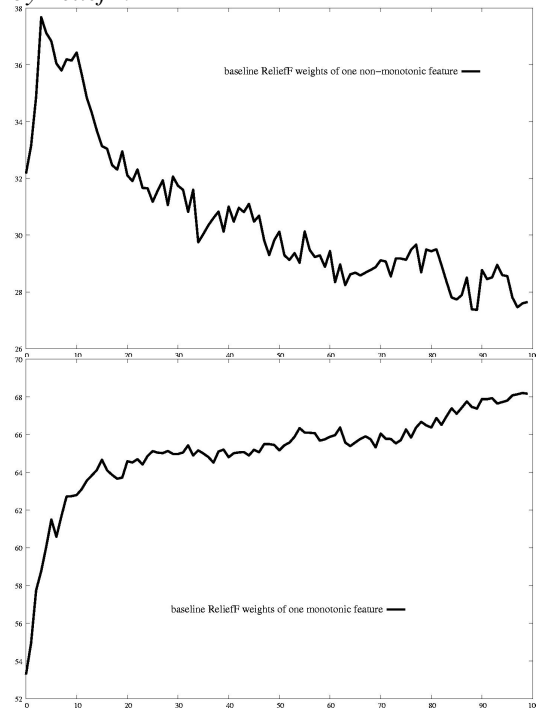
Non-monotonic features therefore rely heavily on the distance measure to select "similar" data samples. Otherwise instances with very large and very small feature values can be selected as "hits", lowering the measured relevance value. In fact, since "hits" are chosen at the level of samples (not features), it is not generally

possible to select k "hits" such that the target is near all of the hit samples in every non-monotonic feature. Therefore non-monotonic features get lower weights. Moreover, as more and more noise features are added to the data set, the distance measure degrades and more and more "hits" are selected from the far side of the distribution, until the selection of hits becomes essentially random. This causes the relevance weights of non-monotonic features to drop before eventually leveling off.

There is another way to interpret the bias. Because ReliefF divides the data samples by category first, it favors features that are invertible functions, in the sense that the feature value can be predicted from the category, rather than the other way around. Non-monotonic features are not invertible functions.

4 Reducing the Bias: Iterative Relief

ReliefF's preference for monotonic features is properly viewed as part of its semantics: non-monotonic features place samples of the same class in different parts of feature space, thereby lowering the effectiveness of k-NN classifier relative to monotonic features. Non-monotonic features therefore should be less relevant than comparable monotonic ones.

The relevance of a feature (monotonic or otherwise) should not be a function of the number of noisy features in the data set, however. Purely noise features should be assigned a weight of zero, and have no effect on any other feature. The fact that monotonic features get larger weights as noise features are added and non-monotonic features get lower weights is therefore a bias.

To eliminate this bias, we need to "fix" the distance measure so that it is effective at selecting similar hits and misses, even in the presence of confusing noise. The obvious way to do this is to use ReliefF's own estimates of feature relevance to weight the distance measure, so that relevant features are used to compute similarity and irrelevant features are discarded.

This suggests an iterative version of Relief. Feature relevance is estimated according to:

$$W_{f_{i+1}} += \frac{W_{f_i}\left(\sum_{m \in misses}\left|f_{t \arg et} - f_m\right| - \sum_{h \in hits}\left|f_{t \arg et} - f_h\right|\right)}{|misses| \cdot |hits| \cdot Range(f)}$$

where i is the iteration counter. In principle, the relevance estimates produced by the first iteration will still be biased, but the estimates produced on the second iteration will be less biased, and the third iteration will be less biased still. This continues until the relevance weights converge to their true, unbiased values. Some details: on the first iteration, all features are weighted equally. Negative feature weights are statistical anomalies when they occur, and are rounded up to zero. The algorithm iterates until the weights converge.

This iterative algorithm is outlined in Figure 3. Unfortunately, it proves to be unstable using the distance metric above; the relevance weights do not always converge. The problem is that a sample either is or is not included in the set of hits or misses; there is no sense of soft inclusion. As a result, changing the feature weights causes a discrete change in the memberships of the hit and miss sets on the next iteration, which may in fact reverse the original weight change. As a result, we get a high dimensional "teeter-totter" effect.

```
initialize iterations to 0;
initialize all distance weights to 1.0;
do {
    iterations++;
    reset all feature weights to 0.0;
    for every sample {
        find the weighted L1 distance from that
            sample to all the other samples;
        identify its nearest hits and misses;
        update feature weights using the
            difference of probabilities;
    }
    update distance weights by feature weights;
} while(( iterations < max ) and ( changed ));
```

Figure 3: Iterative Relief.

The solution is to modify how Relief weights are computed. Instead of including the nearest k samples in the hit and miss sets, we include all samples within a radius r of feature space, and weight the samples (as opposed to the features) by their distance from the target sample. The update rule for the weighting function becomes:

$$\sum_{f \in F} w[f] += \left(\left(\frac{\sum_{i=0}^{k-1} \left(|m_i[f] - s[f]| * Max\left(0, 1 - \left(\frac{d_{m_i}^2}{r^2}\right)\right)\right)}{\sum_{i=0}^{k-1} Max\left(0, 1 - \left(\frac{d_{m_i}^2}{r^2}\right)\right)} \right) - \left(\frac{\sum_{i=0}^{k-1} \left(|h_i[f] - s[f]| * Max\left(0, 1 - \left(\frac{d_{h_i}^2}{r^2}\right)\right)\right)}{\sum_{i=0}^{k-1} Max\left(0, 1 - \left(\frac{d_{h_i}^2}{r^2}\right)\right)} \right) \right)$$

This weighting function combined with the algorithm in Figure 3 is what we call *Iterative Relief*. Iterative Relief no longer explicitly calculates hit and miss sets; instead, a target sample is compared to every other sample of the same class, and every sample of other classes. When samples are outside the radius *r*, their contribution to the relevance weight is zero.

The radius must be large enough to guarantee each sample a minimum of nearest hits and misses. This minimum number replaces the absolute number of nearest hits and nearest misses as a user-set parameter. Also, since their contribution to the feature weights is averaged, the actual number of hits need not equal the actual number of misses.

Figure 4 shows the weights computed by Iterative Relief for the synthetic features in Figure 1 as we add random features to the data set. The top figure shows the non-monotonic feature, while the bottom shows the monotonic feature. The x-axis represents the number of noise features in each sample, while the y-axis indicates the relevance of the synthetic feature.

Although the *shape* of the graph for the non-monotonic feature resembles the corresponding one for baseline ReliefF, the *scale* is radically different. The weight of the non-monotonic feature drops by 0.003 as up to a hundred random features are added to the data set. The weight of the monotonic feature changes by 0.001. This compares to the corresponding ranges of 10.5 (non-monotonic) and 15 (monotonic) for baseline ReliefF.

5   Results on Real Data

Results on synthetic data are fine, but results on real data are always more convincing. The problem is that we have no ground truth relevance measurements for real data sets. We therefore evaluate feature-weighting algorithms indirectly, according to the accuracy of a nearest neighbor classifier operating on the N most relevant features, as measured by the weighting algorithm.
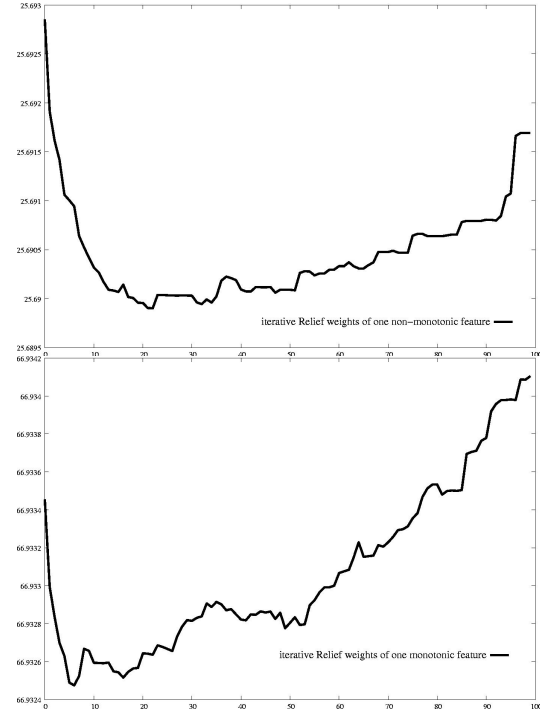

Figure 4. Weights computed by Iterative Relief for synthetic features. The x axis is the number of random features added to the data set..

Since it is fairly rare to use k-NN to classify non-compressed images, we add one more level of indirection to the test. We evaluate feature-weighting algorithms according to a k-NN classifier that operates in a PCA subspace computed over the selected features.

Our domain consisted of 100 registered, frontal cat face images and 100 registered, frontal dog face images, some of which are shown in Figure 5. Each image is 64 by 64 pixels. The feature-weighting algorithm is given 4,096 features (pixels) to weight. In Figure 6, we plot recognition accuracy of k-NN in the PCA subspace as a function of *N*, the number of features selected, as *N* goes from 1 to 1,000. Since there are a total of 200 images (100 cats, 100 dogs), we performed each test six times, each time using 160 images for training and 40 for testing. As a baseline, PCA applied to all 4,096 pixels had an average recognition accuracy of 81.25%.

Figure 5: Examples of cat and dog images.

Figure 6 shows the results of applying PCA/k-NN to the top N features (pixels) as selected by ReliefF (in gray) and Iterative Relief (in black). The recognition accuracy for both techniques peaks between 450 and 850 features. Presumably, fewer than 850 pixels are relevant to the task of discriminating cats from dogs, and the other approximately 3,200 pixels only add noise. This explains why both techniques produce significantly better results than those obtained by applying PCA/k-NN to the full set of pixels. It is also clear that Iterative Relief does a better job than ReliefF (an increase in recognition rate of about 7.5%), at least for this data set, presumably because of the large number of irrelevant pixels. Figure 7 shows the top 615 pixels, as selected by ReliefF (left) and by Iterative Relief (right).
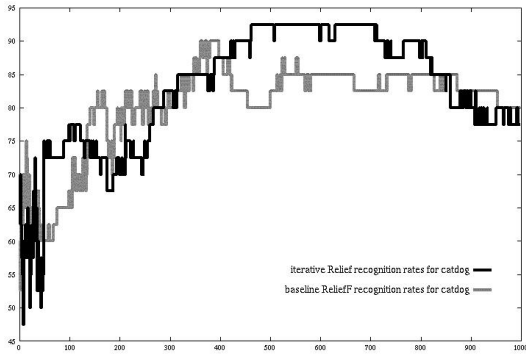

Figure 6: Recognition rates resulting from applying PCA/k-NN to the N most relevant features (pixels) selected by ReliefF (in red) and by Iterative Relief (in green). The horizontal axis is N, the number of pixels; the vertical axis is the recognition rate.

Figure 7: The 615 most relevant pixels (the top 15%), as selected by ReliefF (left) and by Iterative Relief (right).

6 Conclusions

Relief-style algorithms weight features so as to maximize the performance of k-NN classifiers. Unfortunately, the most common Relief algorithm, ReliefF, is biased. It increases the relevance weights of monotonic features and decreases the relevance weights of non-monotonic features in the presence of completely irrelevant (noise) features.

This paper presents a new variant of Relief, Iterative Relief, which is unbiased in this respect, and demonstrates that Iterative Relief outperforms ReliefF, at least on the task of cat and dog discrimination using PCA and k-NN.

7 Bibliography

Bins, J. (2000). Feature Selection of Huge Feature Sets in the Context of Computer Vision. Computer Science. Fort Collins, CO, Colorado State University: 156.

Bins, J. and B. A. Draper (2001). Feature Selection from Huge Feature Sets. International Conference on Computer Vision, Vancouver, IEEE CS Press.

Caruana, R. and D. Freitag (1994). Greedy Attribute Selection. International Conference on Machine Learning, Morgan Kaufman.

Ishiguro, H., R. Sato, et al. (1996). Robot Oriented State Space Construction. IEEE/RSJ International Conference on Intelligent Robots and Systems, Osaka.

Kira, K. and L. A. Rendell (1992). A Practical Approach to Feature Selection. 9th International Workshop on Machine Intelligence, Aberdeen, Scotland, Morgan-Kaufman.

Kononenko, I. (1994). Estimation Attributes: Analysis and Extensions of RELIEF. European Conference on Machine Learning, Catana, Italy, Springer-Verlag.

Robnik-Šikonja, M. and I. Kononenko (2001). Comprehensible Interpretation of Relief's Estimates. International Conference on Machine Learning, Williamstown, MA, Morgan Kaufman.