JMLR: Workshop and Conference Proceedings 63:318–333, 2016 ACML 2016

Learning Distance Metrics for Multi-Label Classification

Henry Gouk HGRG1@STUDENTS.WAIKATO.AC.NZ
Bernhard Pfahringer BERNHARD@WAIKATO.AC.NZ
Department of Computer Science, University of Waikato, Hamilton, New Zealand

Michael Cree CREE@WAIKATO.AC.NZ
School of Engineering, University of Waikato, Hamilton, New Zealand

Editors: Robert J. Durrant and Kee-Eung Kim

Abstract

Distance metric learning is a well studied problem in the field of machine learning, where it is typically used to improve the accuracy of instance based learning techniques. In this paper we propose a distance metric learning algorithm that is specialised for multi-label classification tasks, rather than the multiclass setting considered by most work in this area. The method trains an embedder that can transform instances into a feature space where squared Euclidean distance provides an estimate of the Jaccard distance between the corresponding label vectors. In addition to a linear Mahalanobis style metric, we also present a nonlinear extension that provides a substantial boost in performance. We show that this technique significantly improves upon current approaches for instance based multi-label classification, and also enables interesting data visualisations.

Keywords: Distance Metric Learning, Multi-Label Classification, Instance Based Learning.

1. Introduction

Multiclass classification is one of the most ubiquitous tasks found in the field of machine learning, so it should come as no surprise that the majority of methods for learning distance metrics are designed to be applied to multiclass data. Many of these techniques involve optimising a loss function that considers pairwise equality constraints between similar instances. In this context, two instances are considered similar if they both belong to the same class. However, we are interested in the case of multi-label classification, where each instance can be assigned multiple labels. In the multi-label setting the rule previously given for creating equality constraints is somewhat vague. Should two instances be considered similar only if they have exactly the same set of labels? Or should similarity be indicated by non-empty overlap between the label sets of the two instances? Should a threshold be selected for the number of shared labels required for two instances to be considered similar? In this work we investigate a method for learning distance metrics that avoids labelling pairs as equal or not equal, and instead assigns a real valued similarity score.

Of course, a distance metric alone is not capable of acting as a classifier—it must be paired with a classification algorithm that can exploit knowledge of the fine-grained similarity between instances. The k Nearest Neighbours (k-NN) classification rule is one such algorithm. A particularly interesting benefit of using k-NN over other algorithms is how efficiently it can be applied to problems where there are a large number of classes. If a binary

classification method is to be used on a dataset with a large number of classes, a meta-classification scheme must be employed to train an ensemble of binary classifiers capable of performing multiclass classification. Even methods that are able to perform multiclass classification natively often have a run-time that is dependent on the number of classes in the dataset—multinomial logistic regression is a good example of this. Thus, the ability to improve the accuracy of k-NN classifiers is one of the primary ways to advance how well classification problems with many classes can be solved.

In this work we propose a loss function for training metrics with the same functional form as the well known Mahalanobis distance. Rather than using equality constraints, as is typical for Mahalanobis style metrics Davis et al. (2007); Weinberger et al. (2005), we use the Jaccard distance between the label vectors to provide a more fine-grained estimate of similarity. Furthermore, we show that by adapting a new framework presented by Gouk et al. (2015), we are able to learn nonlinear metrics from multi-label data. In addition to improving the performance of k-NN classification, it is demonstrated that these nonlinear metrics are able to produce useful visualisations of multi-label data.

We first summarise related work in Section 2. Following that, we describe a method for learning linear multi-label distance metrics in Section 3, followed by a generalisation to nonlinear metrics in Section 4. In Section 5 we demonstrate that the linear metrics work well on high dimensional data and classification accuracy is substantially improved in general when using the nonlinear method. Finally, we summarise the contributions of our work and conclude in Section 6.

2. Related Work

Many distance metric learning algorithms are based on the Mahalanobis distance formula given in Equation 1, where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are feature vectors and $\mathbf{M}$ is the matrix of model parameters found during the training process.[1]

$$D(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \mathbf{M} (\boldsymbol{x}_i - \boldsymbol{x}_j)} \tag{1}$$

Prior work addressing metric learning for improving the performance of k-NN classifiers has almost exclusively focused on the popular multiclass classification setting. Some popular methods include Large Margin Nearest Neighbours (Weinberger et al., 2005), which is learned by optimising a loss function that considers triples of instances: a seed instance, an instance similar to the seed, and an instance dissimilar to the seed. The objective aims to make the distance between the similar instances smaller than the distance between the dissimilar instances. Another technique that fits into the same paradigm is Information Theoretic Metric Learning (Davis et al., 2007). The objective for this approach aims to minimise—subject to instance equality constraints—the Kullback-Leibler divergence between a prior Gaussian distribution and the Gaussian distribution with an inverse covariance matrix parameterised by $\mathbf{M}$. Our work is similar to these approaches in the sense that we propose a metric with the same functional form, except rather than using binary equality constraints, we use real valued ground truth measures of similarity derived from the label

1. In the original formulation of Mahalanobis distance, $\mathbf{M}$ is the inverse covariance matrix of the training data.

sets of the training instances when optimising the parameter matrix. Most similar to the linear metric learning algorithm that we porpose is Large Margin k-NN (Liu and Tsang, 2015). To our knowledge, the work of Liu and Tsang (2015) is the only other paper to directly address the problem of metric learning for multi-label data, however several other works exists when there are additional constraints (Jin et al., 2009; Guillaumin et al., 2010).

The conventional k-NN classification scheme is inherently a multiclass method, and not immediately capable of performing multi-label classification. A popular adaptation of k-NN to the multi-label problem is the Multi-label k-Nearest Neighbours (MLkNN) algorithm (Zhang and Zhou, 2007). This method finds the k nearest neighbours of a test instance and uses a maximum likelihood technique to predict the labels.

An alternative to adapting the classification algorithm, as in the case of MLkNN, is to reformulate the problem being solved. So called Problem Transformation methods do just that. The most basic is the Binary Relevance (BR) scheme, which simply trains a binary classifier for each label in the dataset, hence ignoring any information that can be gleaned from considering label correlations. Another popular method that fits into this framework is the Ensemble of Classifier Chains (ECC) method proposed by Read et al. (2011). Each node in the classifier chain predicts the presence of a single label using a binary classifier, however each prediction is then appended to the feature vector before continuing along the chain to the next binary classifier. The order that the labels are predicted is generated randomly during the classifier chain training process. Creating ensembles of these chains via bagging results in a classifier that is able to take advantage of correlations between labels when making predictions.

Nonlinear metric learning is also considered in this work. We take inspiration from Gouk et al. (2015) to transform our linear metric learning method into a nonlinear metric learner that performs significantly better. When employing the framework of Gouk et al. (2015), our nonlinear technique most resembles the nonlinear label compression method of Wicker et al. (2016), in the sense that our approach involves learning a nonlinear mapping from label space to a real valued vector space. As such, our nonlinear method can also be considered a label compression scheme, even though it is an extension of a more conventional distance metric. The benefit of our approach is that Euclidean distance can be applied to the compressed labels to provide a good indication of instance similarity.

3. Learning Linear Metrics

We start by defining a loss function that can be applied to metrics of the form given in Equation 1. To do this, we must first decide what this loss function should accomplish. Unlike multiclass classification, there is no single appropriate definition for accuracy when performing multi-label classification. Instead, evaluation measures such as the Jaccard index between two label sets, Y_i and Y_j, given in Equation 2, between the ground truth label set and the predicted label set are used. It is important to note that because there is no single definitive measure for multi-label classification performance, several should be reported—as is customary in the literature (Read et al., 2011; Zhang and Zhou, 2007; Cheng and Hüllermeier, 2009).

$$J(Y_i, Y_j) = \frac{|Y_i \cap Y_j|}{|Y_i \cup Y_j|} \tag{2}$$

Interestingly, one can transform the Jaccard index into a distance metric, called the Jaccard distance, fairly trivially:

$$D_J(Y_i, Y_j) = 1 - J(Y_i, Y_j) \tag{3}$$

Rather than using binary constraints to determine how similar two instances are when training a learned distance metric, we propose the use of the Jaccard distance between the label sets of training instances. This provides a fine-grained measure of similarity between the examples in the training data. This information is used to train a linear distance metric that can estimate the Jaccard distance between two unknown label sets by considering only the features associated with the label sets.

More formally, suppose we have a training set, $\mathcal{X}$, with elements of the form $(\boldsymbol{x}_i, Y_i)$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is the d-dimensional column vector of features for instance i and Y_i is the label set. We define a set, $\mathcal{Z} = \mathcal{X} \times \mathcal{X}$, which contains all possible pairings of instances in $\mathcal{X}$. The parameters for a Mahalanobis style distance metric are learned by solving the following optimisation problem:

$$\mathbf{M}^* = \arg\min_{\mathbf{M}} \frac{1}{|\mathcal{Z}|} \sum_{((\boldsymbol{x}_i, Y_i), (\boldsymbol{x}_j, Y_j)) \in \mathcal{Z}} L(\boldsymbol{x}_i, \boldsymbol{x}_j, Y_i, Y_j) \tag{4}$$

$$L(\boldsymbol{x}_i, \boldsymbol{x}_j, Y_i, Y_j) = \begin{cases} (D_J(Y_i, Y_j) - D(\boldsymbol{x}_i, \boldsymbol{x}_j)^2)^2, & \text{if } D_J(Y_i, Y_j) < 1 \\ -\min(1, D(\boldsymbol{x}_i, \boldsymbol{x}_j)^2), & \text{otherwise} \end{cases} \tag{5}$$

where $D(\cdot, \cdot)$ is a Mahalanobis style metric parameterised by the positive-semidefinite matrix $\mathbf{M}$, and $D_J(\cdot, \cdot)$ is the Jaccard distance. This function has the effect of training a clamped Mahalanobis style metric that is capable of estimating the Jaccard distance between the unknown label sets of two data points. The min function prevents metric from being uneccesarily penalised when large values are predicted for completely dissimilar instances, as opposed to simply trying to predict a Jaccard distance of one.

In order to avoid solving a constrained optimisation problem, we reparametrise the Mahalanobis distance metric as $\mathbf{M} = \mathbf{G}^T\mathbf{G}$. In this formulation $\mathbf{G}$ is a matrix with the same number of rows as $\mathbf{M}$, but the number of columns is specified by a new hyper-parameter, t. That is, $\mathbf{M}$ is $d \times d$ and $\mathbf{G}$ is $t \times d$. This guarantees that $\mathbf{M}$ is positive-semidefinite, and thus $D(\cdot, \cdot)$ behaves as a proper distance metric. We can now equivalently express the canonical formula for Mahalanobis metrics as:

$$D(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\mathbf{G} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j))^T (\mathbf{G} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j))} \tag{6}$$

One can think of this alternative formulation as embedding the instances into a t dimensional vector space and then applying Euclidean distance to this new representation. These embeddings can then be stored in a data structure, such as a k-d tree, capable of performing efficient nearest neighbour searches. This removes the need for matrix multiplications when

comparing instances during k-NN classification. After performing this reparametrisation, we now solve for the optimal value of $\mathbf{G}$, rather than the optimal value of $\mathbf{M}$.

Because $\mathcal{Z}$ is of size $|\mathcal{X}|^2$, moderate growth in the training set size will lead to a great increase of $|\mathcal{Z}|$, causing exact optimisation methods to become very slow. The conditional expression and $\min(\cdot,\cdot)$ function also present problems, as this means our objective is non-smooth and also results in a very large plateau on the error surface—something exact optimisation methods tend to struggle with. It is for these reasons that we resort to the use of the Adam optimisation algorithm (Kingma and Ba, 2014), a variant of stochastic gradient descent (SGD). An additional advantage of using Adam is that it was designed for training deep neural networks, where local minima are abundant. Our objective is not convex with respect to $\mathbf{G}$, so our choice of optimiser has a great impact on the final accuracy of the learned metrics.

Adam requires gradient information to optimise functions, however because our objective is nonsmooth we must settle for having a means of computing subgradients. This is accomplished by rewriting our reparametrised objective function in a form more amenable to differentiation, as in Equation 7, and subsequently differentiating all parts of the per instance loss function (Equation 8) with respect to the elements of $\mathbf{G}$.

$$\mathbf{G}^* = \arg\min_{\mathbf{G}} \frac{1}{|\mathcal{Z}|} \sum_{((\boldsymbol{x}_i,Y_i),(\boldsymbol{x}_j,Y_j))\in\mathcal{Z}} L(\boldsymbol{x}_i,\boldsymbol{x}_j,Y_i,Y_j) \tag{7}$$

$$L(\boldsymbol{x}_i,\boldsymbol{x}_j,Y_i,Y_j) = \begin{cases} (D_J(Y_i,Y_j) - D(\boldsymbol{x}_i,\boldsymbol{x}_j)^2)^2, & \text{if } D_J(Y_i,Y_j) < 1 \\ -D(\boldsymbol{x}_i,\boldsymbol{x}_j)^2, & \text{if } D_J(Y_i,Y_j) = 1 \text{ and } D(\boldsymbol{x}_i,\boldsymbol{x}_j)^2 < 1 \\ -1, & \text{otherwise} \end{cases} \tag{8}$$

Taking the partial derivative of Equation 8 with respect to $\mathbf{G}$ results in:

$$\nabla_{\mathbf{G}} L = \mathbf{G}(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \times 2 \times \begin{cases} 2(D(\boldsymbol{x}_i,\boldsymbol{x}_j)^2 - D_J(Y_i,Y_j)), & \text{if } D_J(Y_i,Y_j) < 1 \\ -1, & \text{if } D_J(Y_i,Y_j) = 1 \text{ and } D(\boldsymbol{x}_i,\boldsymbol{x}_j)^2 < 1 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

This equation is used by Adam to compute the required gradients. Algorithm 1 describes the procedure used to optimise the linear metrics, including parameter initialisation, optimisation, and training pair generation. We use the parameters suggested by Kingma and Ba (2014) for Adam: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The convergence tolerance, e, is the number of iterations through the dataset that will be tolerated without observing an improvement in the loss function. We set this value to 5 in all of our experiments.

4. Extension to Nonlinear Models

As mentioned in Section 3, the formulation of Mahalanobis distance given in Equation 6 can be thought of as a transformation of two instances into a t dimensional vector space,

Algorithm 1: Optimisation procedure for finding $\mathbf{G}$.

Input: Training data $\mathcal{X}$, training data dimensionality d, embedding dimensionality t, convergence tolerance e

Output: Parameter matrix $\mathbf{G}$

$\mathsf{stop} \leftarrow 0$;
$\mathsf{bestloss} \leftarrow \infty$;
$\mathsf{loss} \leftarrow 0$;
foreach $i, j \in \{1, ..., T\} \times \{1, ..., D\}$ **do** $\mathbf{G}_{ij} \leftarrow \texttt{RandomSample}(\mathcal{N}(0, \frac{1}{D+T}))$;
while $\mathsf{stop} < e$ **do**
  foreach $(\boldsymbol{x}_i, Y_i) \in \mathcal{X}$ **do**
    $(\boldsymbol{x}_j, Y_j) \leftarrow \texttt{UniformRandomElement}(\mathcal{X})$;
    $\mathsf{loss} \leftarrow \mathsf{loss} + L(\boldsymbol{x}_i, \boldsymbol{x}_j, Y_i, Y_j)$;
    $\mathbf{G} \leftarrow \texttt{Adam}(\nabla_{\mathbf{G}} L, \mathbf{G}, \boldsymbol{x}_i, \boldsymbol{x}_j, Y_i, Y_j)$;
  end
  if $\mathsf{loss} < \mathsf{bestloss}$ **then**
    $\mathsf{bestloss} \leftarrow \mathsf{loss}$;
    $\mathsf{stop} \leftarrow 0$;
  end
  else $\mathsf{stop} \leftarrow \mathsf{stop} + 1$;
end

followed by a comparison using Euclidean distance. In this section we describe how the linear transformation can be swapped out for a nonlinear embedding model.

Gouk et al. (2015) recently presented a framework that allows algorithms that create nonlinear regression models to be used as components in distance metrics. The method is composed of two phases: learning the targets vectors that encode the ideal representation for each instance, and then learning the embedding function. The target vectors are found by minimising any distance metric learning loss function that involves embedding instances into a vector space where some predefined metric (such as Euclidean distance) is a useful indicator of similarity. Equation 8 is a perfect example of the kind of loss function intended to be used by this method.

In this first phase, rather than actually learning a function that can perform the embedding, the method simply computes what the ideal embedding for each instance is for the supplied loss function. This method also requires the hyper-parameter, t, that determines the size of these embeddings. The first phase can be concisely described as the following optimisation problem:

$$E^* = \arg\min_E \frac{1}{|\mathcal{Z}|} \sum_{((\boldsymbol{x}_i,Y_i),(\boldsymbol{x}_j,Y_j))\in\mathcal{Z}} L(\boldsymbol{x}_i,\boldsymbol{x}_j,Y_i,Y_j) \tag{10}$$

$$L(\boldsymbol{x}_i,\boldsymbol{x}_j,Y_i,Y_j) = \begin{cases} (D_J(Y_i,Y_j) - D(\boldsymbol{x}_i,\boldsymbol{x}_j)^2)^2, & \text{if } D_J(Y_i,Y_j) < 1 \\ -\min(1, D(\boldsymbol{x}_i,\boldsymbol{x}_j)^2), & \text{otherwise} \end{cases} \tag{11}$$

$$D(\boldsymbol{x}_i,\boldsymbol{x}_j) = \|\boldsymbol{e}_i - \boldsymbol{e}_j\|_2 \tag{12}$$

In Equation 12 $\boldsymbol{e}_i, \boldsymbol{e}_j \in E$ are the t-dimensional target vectors associated with $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$. We use the Adam optimiser to find the set, E^*, of locally optimal target vectors. However, for this problem we must take the derivative with respect to the target vectors we are attempting to learn:

$$\left(\frac{\partial L}{\partial \boldsymbol{e}_i}, \frac{\partial L}{\partial \boldsymbol{e}_j}\right) = (\boldsymbol{e}_i - \boldsymbol{e}_j, \boldsymbol{e}_j - \boldsymbol{e}_i) \times 2 \times \begin{cases} 2(D(\boldsymbol{x}_i,\boldsymbol{x}_j)^2 - D_J(Y_i,Y_j)), & \text{if } D_J(Y_i,Y_j) < 1 \\ -1, & \text{if } D_J(Y_i,Y_j) = 1 \text{ and } D(\boldsymbol{x}_i,\boldsymbol{x}_j)^2 < 1 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

The optimisation is performed by adapting Algorithm 1 to use the appropriate derivatives and simply perform 10,000 epochs of training. We found that, for this problem, our simple stopping criterion of monitoring the value of the loss function was not a good heuristic for determining when the optimisation was complete.

Once the target vectors are obtained, one must train a model that can perform multi-target regression—the equivalent of multi-label classification where one wishes to estimate a vector of real valued response variables. In their experiments Gouk et al. (2015) used only deep neural networks on image datasets, however they acknowledge that any multi-target regression method can be used. In this work we investigate how well this framework performs when the multi-target regression model is a set of random forests trained for regression. That is, we train a separate random forest to make predictions for each component in the target vector space. This can be seen as a binary relevance method for regression. Figure 1 outlines the procedure for constructing distance metrics using this nonlinear approach.

After the metric has been learned it can be applied to novel instances in a similar manner to other embedding based metrics. Firstly, two instances are embedded into a t-dimensional vector space by the random forests. Secondly, these embeddings are compared using squared Euclidean distance, resulting in an estimate of the Jaccard distance between the two unknown label sets.

5. Experiments

In our experiments we evaluate our metric learning approaches using two use cases for custom embedding based distance metrics: the improvement of k-NN type classifiers, and the utility for data visualisation. We take advantage of several freely available datasets that are

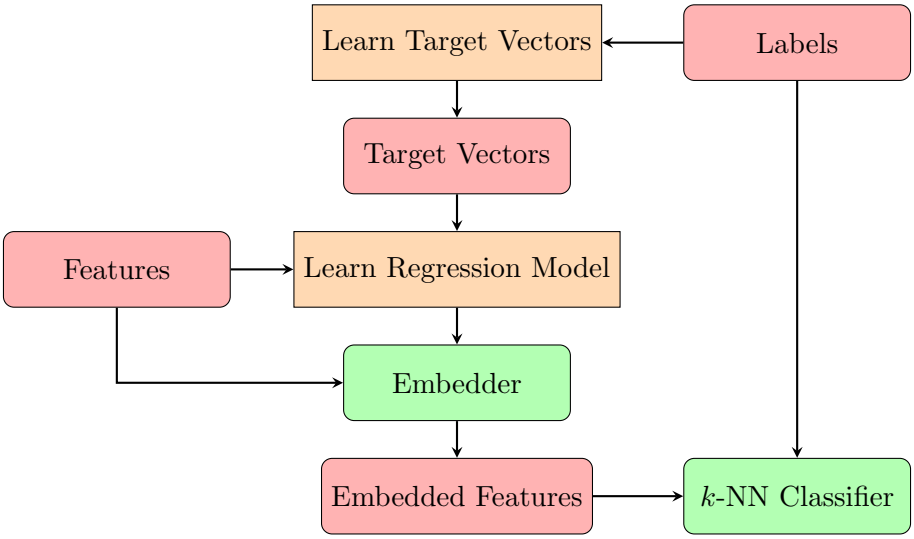


Figure 1: This diagram provides an overview of how the nonlinear metrics are constructed. The first step in the process is to learn the target vectors for which squared Euclidean distance is an accurate estimate of the Jaccard distance over the original label sets. Once the target vectors have been learned they, along with the original features, are used to create a multi-target regression model. In this work we use a separate random forest for each target vector component. Once the multi-target regression model has been learned, it can be used to embed the features into the same space as the target vectors. Now the embedded features and the original labels can be used to construct a k-NN style classifier that has been adapted to work with multi-label data.

Table 1: A summary of the datasets that we use in our experiments. The label cardinality is the average number of labels assigned to each example.

Dataset	Instances	Features	Labels	Cardinality
scene	2,407	294	6	1.074
yeast	2,417	103	14	4.237
enron	1,702	1,001	53	3.378
emotions	592	72	6	1.869
genbase	662	1,186	27	3.392
medical	978	1,449	45	1.245

summarised in Table 1. The algorithms were implemented in the MEKA framework (Read et al., 2016) and the source code has been released publicly.[2]

2. http://github.com/henrygouk/meka-metric-learning

5.1. Classification

The motivation behind distance metric learning is to improve the performance of k-NN classifiers that, in our case, have been adapted to work with multi-label data. As such, we compare several standard k-NN multi-label classification schemes that use only Euclidean distance with variants of each method that incorporate metrics learned using the techniques presented herein. In particular we consider the binary relevance method and ensembles of classifier chains (both with k-NN base learners) as implemented by the MEKA framework. We also compare against the MLkNN scheme implemented in the MULAN framework (Tsoumakas et al., 2011). We set $k = 10$ for all models, and for all the models taking advantage of the nonlinear metric learning procedure we set $t = 16$. In the case of the linear models we set $t = 32$. All ECC models use an ensemble size of 50.

As there is no single evaluation metric that is suitable for multi-label classification, we report results using the Jaccard index, the log loss averaged over each label, the hamming loss, and the F1 score averaged over each example.

We first investigate how well the linear metric learning algorithm performs. The results for these experiments are summarised in Table 2. On average, performance is competitive with two of the three baselines. MLkNN is the clear winner according to all evaluation metrics, however the performance of all the linear metric learning approaches is significantly better on enron and medical, the high dimensional natural language datasets.

Results for the models taking advantage of learned nonlinear distance metrics can be found in Table 3. In this experiment we also evaluate the MANIAC method (Wicker et al., 2016), which is similar to our technique in the sense that a nonlinear transformation of the labels is used in an attempt to improve predictive accuracy. When considering only the average rank of each method for each metric, one can see that the all the models taking advantage of the Jaccard embedding have superior performance to all models that simply use Euclidean distance. We observe that the performance of MANIAC is positively correlated with the number of labels in the dataset, which is congruent with conclusions made by Wicker et al. (2016). In contrast, our nonlinear method performs well irrespective of the number of labels in the dataset. Investigating further, the BR models that use the Jaccard embedding perform the best on three of the four metrics, and on the other metric they are ranked second best. We see this as strong evidence that the nonlinear Jaccard embedding method coupled with the BR problem transformation is a good choice for instance based learning on multi-label data.

5.2. Target Vector Size

The size of the target vectors, controlled by t, has a huge impact on the final performance of a k-NN based multi-label classification scheme. The dimensionality of the vector space the features are embedded into must be large enough to accurately capture all the interactions (correlations, dependencies, etc) between the different labels. However, setting t to be too large could make the optimisation problem unnecessarily harder and the process of training the multi-target regression model more time consuming. Thus, we investigate how t impacts the performance of k-NN trained under the BR scheme when using our nonlinear metric learning method. The results of these experiments are summarised in Figure 2. It can be seen that the performance of each algorithm on each dataset converges towards a constant

Table 2: Results for the instance based multi-label classification methods when using no learned distance metric and when the linear metric learning method presented herein is used. Methods prefixed with LJE indicate the use of our linear metric learning method.

(a) F1 score (macro averaged over examples, higher is better)

Datasets	BR	MLkNN	ECC	LJE-BR	LJE-MLkNN	LJE-ECC
yeast	**0.660 (6)**	0.652 (4)	0.656 (5)	0.641 (2)	0.640 (1)	0.649 (3)
enron	0.364 (2)	0.456 (3)	0.339 (1)	0.507 (4)	**0.533 (6)**	0.518 (5)
medical	0.610 (2)	0.662 (3)	0.583 (1)	0.686 (4)	0.689 (5)	**0.696 (6)**
emotions	0.648 (5)	0.638 (4)	**0.657 (6)**	0.611 (2)	0.608 (1)	0.624 (3)
genbase	0.945 (5)	**0.961 (6)**	0.914 (2)	0.919 (3)	0.890 (1)	0.941 (4)
scene	0.642 (3)	**0.663 (6)**	0.634 (1)	0.652 (5)	0.647 (4)	0.641 (2)
Avg. Rank	3.833	**4.333**	2.667	3.333	3.000	3.833

(b) Hamming loss (lower is better)

Datasets	BR	MLkNN	ECC	LJE-BR	LJE-MLkNN	LJE-ECC
yeast	0.217 (5)	0.202 (2)	**0.201 (1)**	0.229 (6)	0.210 (4)	0.206 (3)
enron	0.079 (6)	0.067 (3)	0.071 (5)	0.068 (4)	**0.059 (1)**	0.063 (2)
medical	0.020 (5)	0.018 (3)	0.022 (6)	0.018 (4)	**0.017 (1)**	0.017 (2)
emotions	0.209 (3)	0.207 (2)	**0.200 (1)**	0.233 (6)	0.226 (5)	0.217 (4)
genbase	0.007 (2)	**0.006 (1)**	0.011 (4)	0.043 (6)	0.014 (5)	0.008 (3)
scene	0.136 (6)	**0.119 (1)**	0.131 (4)	0.134 (5)	0.126 (2)	0.131 (3)
Avg. Rank	4.500	**2.000**	3.500	5.167	3.000	2.833

(c) Jaccard index (higher is better)

Datasets	BR	MLkNN	ECC	LJE-BR	LJE-MLkNN	LJE-ECC
yeast	0.551 (5)	0.546 (4)	**0.556 (6)**	0.529 (1)	0.531 (2)	0.546 (3)
enron	0.260 (2)	0.324 (3)	0.260 (1)	0.375 (4)	**0.405 (6)**	0.399 (5)
medical	0.562 (2)	0.616 (3)	0.536 (1)	0.634 (4)	0.645 (5)	**0.653 (6)**
emotions	0.559 (5)	0.551 (4)	**0.575 (6)**	0.523 (2)	0.518 (1)	0.542 (3)
genbase	0.930 (5)	**0.946 (6)**	0.894 (2)	0.902 (3)	0.841 (1)	0.922 (4)
scene	0.606 (1)	**0.631 (6)**	0.611 (2)	0.616 (3)	0.620 (5)	0.619 (4)
Avg. Rank	3.333	**4.333**	3.000	2.833	3.333	4.167

(d) Log loss (averaged over labels, lower is better)

Datasets	BR	MLkNN	ECC	LJE-BR	LJE-MLkNN	LJE-ECC
yeast	**0.422 (1)**	0.431 (2)	0.435 (3)	0.443 (4)	0.446 (5)	0.458 (6)
enron	0.179 (5)	0.157 (3)	0.226 (6)	0.151 (2)	**0.149 (1)**	0.160 (4)
medical	0.049 (5)	0.046 (3)	0.057 (6)	0.045 (2)	**0.044 (1)**	0.048 (4)
emotions	0.399 (3)	0.416 (4)	**0.372 (1)**	0.428 (5)	0.444 (6)	0.391 (2)
genbase	0.015 (3)	0.014 (2)	0.023 (6)	**0.013 (1)**	0.017 (4)	0.018 (5)
scene	0.245 (6)	0.232 (3)	**0.227 (1)**	0.239 (4)	0.240 (5)	0.228 (2)
Avg. Rank	3.833	**2.833**	3.833	3.000	3.667	3.833

Table 3: Results for the multi-label instance based learning algorithms with and without the nonlinear metric learning methods presented in Section 4. Methods prefixed with NJE indicate the use of our nonlinear metric learning method.

(a) F1 score (macro averaged over examples, higher is better)

Datasets	MANIAC	BR	MLkNN	ECC	NJE-BR	NJE-MLkNN	NJE-ECC
yeast	0.567 (1)	0.660 (4)	0.652 (2)	0.656 (3)	**0.667 (7)**	0.663 (5)	0.664 (6)
enron	0.441 (3)	0.364 (2)	0.456 (4)	0.339 (1)	0.534 (5)	**0.540 (7)**	0.535 (6)
medical	**0.745 (7)**	0.610 (2)	0.662 (3)	0.583 (1)	0.736 (6)	0.693 (4)	0.728 (5)
emotions	0.472 (1)	0.648 (3)	0.638 (2)	0.657 (5)	**0.673 (7)**	0.668 (6)	0.656 (4)
genbase	**0.985 (7)**	0.945 (4)	0.961 (6)	0.914 (2)	0.914 (3)	0.849 (1)	0.960 (5)
scene	0.435 (1)	0.642 (3)	0.663 (4)	0.634 (2)	**0.727 (7)**	0.689 (5)	0.721 (6)
Avg. Rank	3.333	3.000	3.500	2.333	**5.833**	4.667	5.333

(b) Hamming loss (lower is better)

Datasets	MANIAC	BR	MLkNN	ECC	NJE-BR	NJE-MLkNN	NJE-ECC
yeast	0.216 (6)	0.217 (7)	0.202 (5)	0.201 (4)	0.196 (2)	0.197 (3)	**0.195 (1)**
enron	**0.051 (1)**	0.079 (7)	0.067 (5)	0.071 (6)	0.060 (4)	0.058 (2)	0.059 (3)
medical	**0.013 (1)**	0.020 (6)	0.018 (5)	0.022 (7)	0.015 (2)	0.017 (4)	0.016 (3)
emotions	0.334 (7)	0.209 (6)	0.207 (5)	0.200 (4)	0.188 (2)	**0.186 (1)**	0.191 (3)
genbase	**0.002 (1)**	0.007 (4)	0.006 (2)	0.011 (5)	0.065 (7)	0.019 (6)	0.006 (3)
scene	0.255 (7)	0.136 (6)	0.119 (4)	0.131 (5)	**0.102 (1)**	0.113 (3)	0.102 (2)
Avg. Rank	3.833	6.000	4.333	5.167	3.000	3.167	**2.500**

(c) Jaccard index (higher is better)

Datasets	MANIAC	BR	MLkNN	ECC	NJE-BR	NJE-MLkNN	NJE-ECC
yeast	0.450 (1)	0.551 (3)	0.546 (2)	0.556 (4)	**0.565 (7)**	0.561 (5)	0.564 (6)
enron	0.339 (4)	0.260 (2)	0.324 (3)	0.260 (1)	0.409 (5)	0.413 (6)	**0.414 (7)**
medical	**0.704 (7)**	0.562 (2)	0.616 (3)	0.536 (1)	0.698 (6)	0.656 (4)	0.691 (5)
emotions	0.385 (1)	0.559 (3)	0.551 (2)	0.575 (4)	**0.593 (7)**	0.589 (6)	0.576 (5)
genbase	**0.979 (7)**	0.930 (4)	0.946 (6)	0.894 (2)	0.898 (3)	0.775 (1)	0.942 (5)
scene	0.366 (1)	0.606 (2)	0.631 (4)	0.611 (3)	**0.708 (7)**	0.668 (5)	0.704 (6)
Avg. Rank	3.500	2.667	3.333	2.500	**5.833**	4.500	5.667

(d) Log loss (averaged over labels, lower is better)

Datasets	MANIAC	BR	MLkNN	ECC	NJE-BR	NJE-MLkNN	NJE-ECC
yeast	0.552 (7)	**0.422 (1)**	0.431 (2)	0.435 (3)	0.483 (4)	0.502 (5)	0.503 (6)
enron	0.194 (6)	0.179 (5)	**0.157 (1)**	0.226 (7)	0.165 (2)	0.169 (3)	0.174 (4)
medical	0.047 (4)	0.049 (5)	0.046 (2)	0.057 (7)	**0.045 (1)**	0.047 (3)	0.050 (6)
emotions	0.597 (7)	0.399 (5)	0.416 (6)	0.372 (4)	0.360 (3)	**0.358 (1)**	0.358 (2)
genbase	**0.006 (1)**	0.015 (5)	0.014 (3)	0.023 (7)	0.012 (2)	0.015 (4)	0.016 (6)
scene	0.456 (7)	0.245 (6)	0.232 (5)	0.227 (4)	0.183 (2)	0.196 (3)	**0.182 (1)**
Avg. Rank	5.333	4.500	3.167	5.333	**2.333**	3.167	4.167

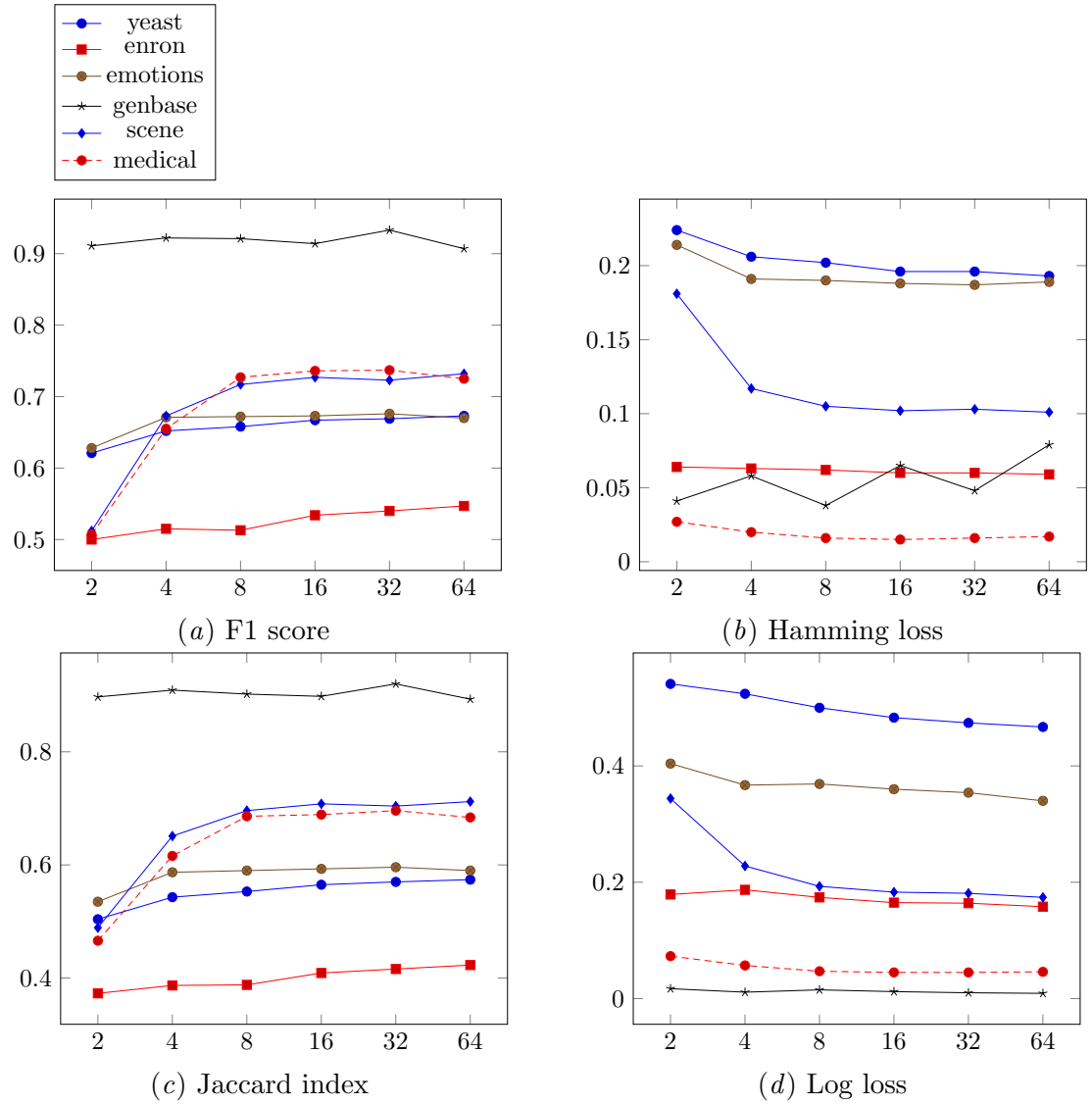


Figure 2: A demonstration of the impact that the target vector size has on the performance of the nonlinear models, as measured by several standard multi-label evaluation measures. The same legend applies to all plots.

value under all metrics as the target size grows larger. In most cases the best performing models are when $t \geq 16$, however for genbase this is not the case, and we believe the cause for this is that the embedder is overfitting due to the small number of instances and large number of attributes. Another potential concern is that because the loss function has multiple minima and we are only performing local optimisation, we may find low quality local minima. In practice we have found that the local minima that have been reached during optimisation are of similar quality across multiple runs.

5.3. Visualisation

An interesting aspect of our framework is the ability to reduce the dimensionality of the data to an arbitrary size, which gives one the ability to create two dimensional visualisations of multi-label data. Figure 3 uses the scene dataset to demonstrate how this can be an effective visualisation method. These images are the result of plotting the output of a nonlinear metric learned using the technique described in Section 4, with $t = 2$. An ensemble of random forests trained using the BR scheme was used to perform multi-target regression. A separate plot is used for each label and, possibly because the label cardinality of this dataset is quite low, it can be seen that each label is mapped primarily to one cluster. There are several small groups of instances that appear between clusters, and one can see that the instances in these groups have multiple labels and generally belong to the classes associated with both nearby clusters.

Figure 4 shows a similar visualisation for the emotions dataset. This dataset has a noticeably higher label cardinality, which has resulted in a more interesting visualisation. Each label is associated with multiple clusters, and each cluster is also associated with multiple labels, as one would expect from multi-label data. This requirement that certain clusters must be adjacent, as enforced by the correlations between labels, is something that is absent from distance metric learning for multiclass classification.

6. Conclusion

In this paper we have introduced linear and nonlinear distance metric learning methods aimed at improving the performance of k-NN applied to multi-label data. The linear metric learning approach was of little added benefit in the general case, however we observed a significant increase in performance when applying it to problems with high dimensional data. When the nonlinear extension is used performance is further improved. The nonlinear extension enables us to utilise random forests for performing a wider class of transformations, relative to the linear method, when computing distances between instances. In addition to improving classification performance, the embedding components of the nonlinear metrics are effective tools for multi-label data visualisation.

In the future we would like to explore other ways in which the proposed loss function can be applied to nonlinear models. For example, a deep convolutional neural network trained with this loss function could be used to perform web image tag prediction. We would also like to investigate the scalability of this approach to large datasets—likely taking advantage of an algorithm specialised for multi-target regression instead of using a problem transformation method.

References

Weiwei Cheng and Eyke Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009.

Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 209–216. ACM, 2007.

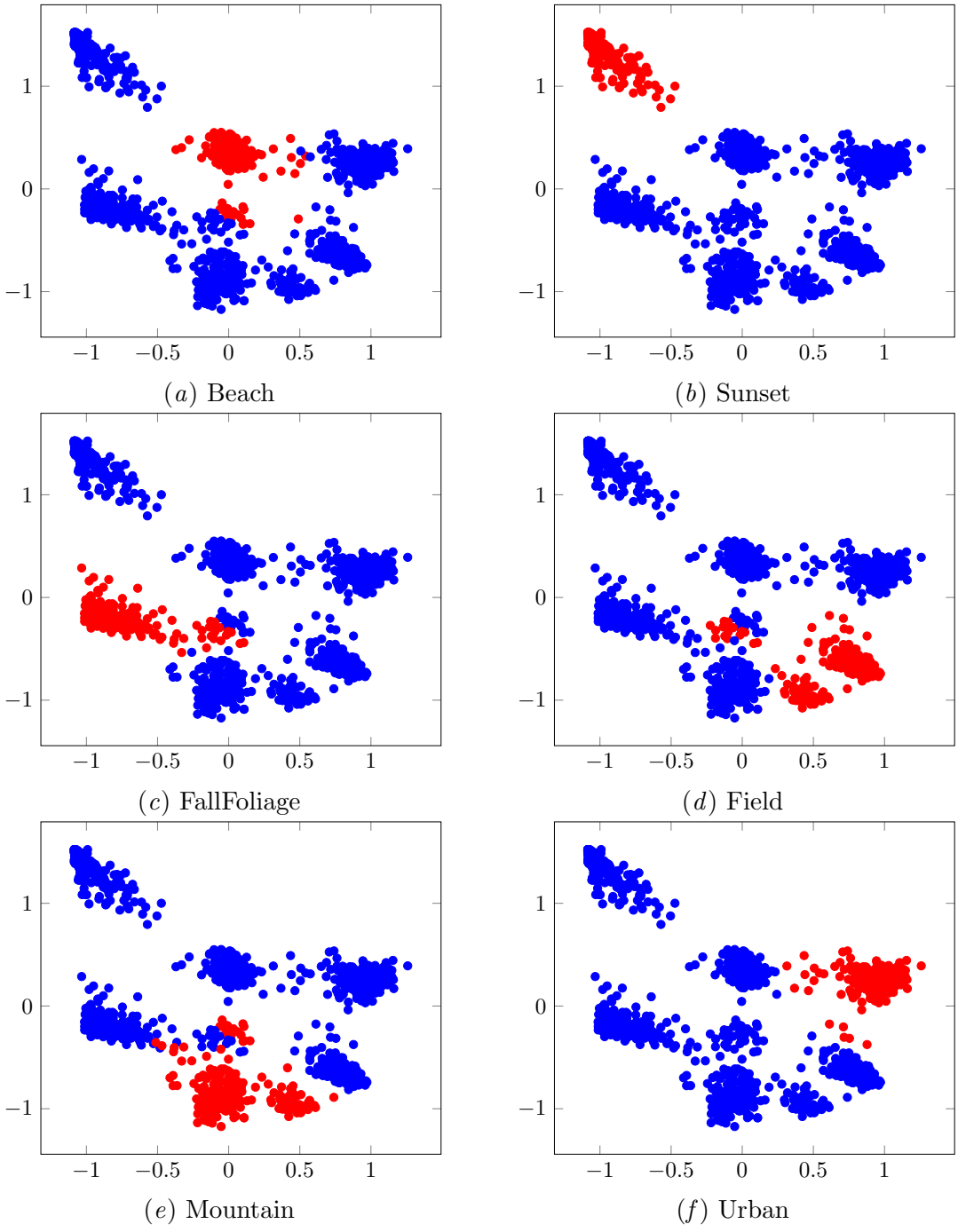
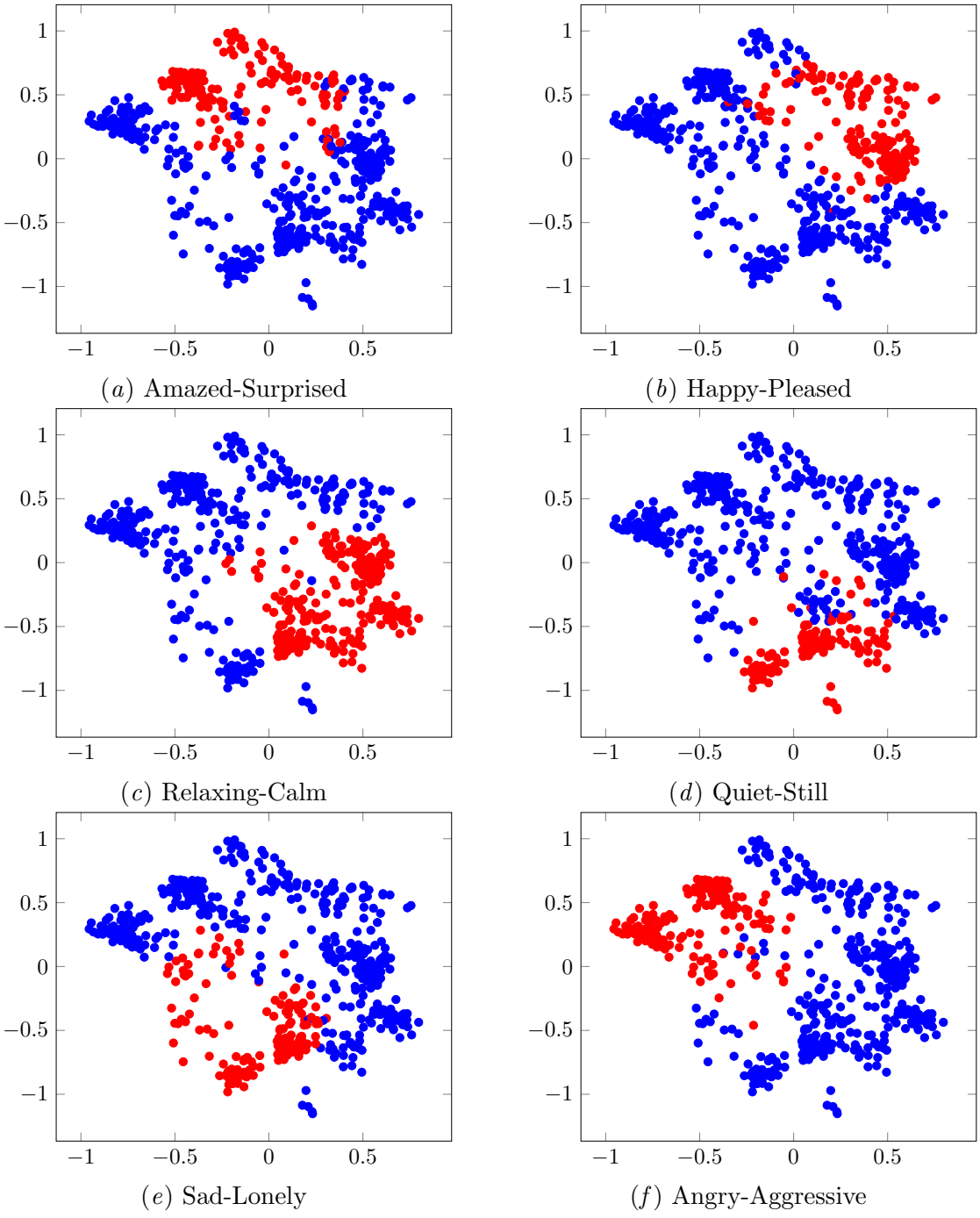


Figure 3: Visualisation of a random sample of the scene dataset. Each plot indicates the presence (red) or absence (blue) of a label for each instance.



(*a*) Amazed-Surprised (*b*) Happy-Pleased

(*c*) Relaxing-Calm (*d*) Quiet-Still

(*e*) Sad-Lonely (*f*) Angry-Aggressive

Figure 4: Visualisation of the emotions dataset. Each plot indicates the presence (red) or absence (blue) of a label for each instance.

Henry Gouk, Bernhard Pfahringer, and Michael Cree. Fast Metric Learning For Deep Neural Networks. *arXiv preprint arXiv:1511.06442*, 2015.

Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Multiple instance metric learning from automatically labeled bags of faces. In *European conference on Computer Vision*, pages 634–647. Springer, 2010.

Rong Jin, Shijun Wang, and Zhi-Hua Zhou. Learning a distance metric from multi-instance multi-label data. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 896–902. IEEE, 2009.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Weiwei Liu and Ivor W Tsang. Large margin metric learning for multi-label prediction. In *AAAI*, pages 2800–2806, 2015.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

Jesse Read, Peter Reutemann, Bernhard Pfahringer, and Geoff Holmes. MEKA: A Multi-label/Multi-target Extension to WEKA. *Journal of Machine Learning Research*, 17(21): 1–5, 2016.

Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *The Journal of Machine Learning Research*, 12:2411–2414, 2011.

Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 1473–1480, 2005.

Jörg Wicker, Andrey Tyukin, and Stefan Kramer. A Nonlinear Label Compression and Transformation Method for Multi-label Classification Using Autoencoders. In *Advances in Knowledge Discovery and Data Mining*, pages 328–340. Springer, 2016.

Min-Ling Zhang and Zhi-Hua Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.