Machine Learning Assignment-1

DARSHAN AGARWAL

201225189

Problem Setting

- Image annotation is one of the major research areas in the field of computer vision. It aims at **predicting a set of textual labels** for an image that describe its semantics.
- It is used in **image retrieval systems** to organize and locate images of interest from a database.
- With increase in the visual data there is a high demand of correlating two images semantically.
- This method can be regarded as a type of **multiclass image classification** with a very large number of classes as large as the vocabulary size.

Problem Setting

In this assignment, we try to solve the problem of annotating images with multiple labels using three different approaches, namely :

1. Joint Equal Contribution (JEC)
2. Tag propagation
3. 2 Pass K-Nearest Neighbour (2PKNN)

Problem

Given an unseen image the task is to assign multiple labels relevant to that image.

Dataset

We have used Corel5k dataset for the assignment. It has 5000 manually labelled images. We have used 4500 images as training data and 499 images as testing data for all the experiments.

Features Used

There are two types of features we are using :

1. Feat1 : Standard features like color, texture, shape. HSV, RGB, LAB, Densehue, Gist, etc.
2. Feat2 : By combining all the basic features, a single feature vector is computed. This is used for computing a single pairwise distance matrix.

Nearest Neighbour Techniques

- Nearest Neighbour techniques have been found to achieve good performance in the image annotation task. These are based on simple idea that “**similar images share similar labels**”.
- To predict the image for a given label we first **identify a set of its similar images** and **propagate the labels** using different methods.

Joint Equal Contribution (JEC):

In this method, to find the similar images we take **equal contribution of all the features** and use the **distance between the images** as the measure for similarity. After finding the k nearest neighbors of the test image it transfers the labels by following the below steps:

- Assign all the labels of the nearest image.
- If still more labels are to be assigned to the test image, sort the labels of the remaining neighboring images on 2 criteria, one on the basis of **co-occurrence frequency** with the labels of the nearest image and second on the basis of **local frequency** of the labels

Tag Propagation (Tagprop):

This method predicts tags by taking a **weighted combination of the tag absence/presence** among neighbors. The weights for neighbors are either determined based on the neighbor rank or its distance, and set automatically by maximizing the likelihood of annotations in a set of training images.

$$P(y_{iw} = +1) = \sum_j C_{ij} P(y_{iw} = +1 \mid j)$$

Distance Based Weights

$$C_{ij} = \frac{\exp(-dw(i,j)}{\sum_{j'} \exp(-dw(i,j'))}$$

$$d_w(i,j) = w^T d_{ij}$$

d_{ij} is a vector of base distances b/w image i and j
w contains the positive coefficients of the linear distance combination.

Tag Propagation using SD

Only one base distance is combined. In this case, w, the weight vector is a scalar, i.e., we are taking equal contributions of all the base distances.

Tag Propagation using sigma SD

Sometimes, frequent tags skew the results, suppressing the effect of rare tags. To overcome this, we define a sigmoid type of function that enhances of rare tags and suppresses the results of very frequent tags.

Tag Propagation using ML

The weight vector w, is not a scalar, hence the number of base distances.

Tag Propagation using sigma ML

It also takes into account the frequency of tags using the sigmoid function as described above.

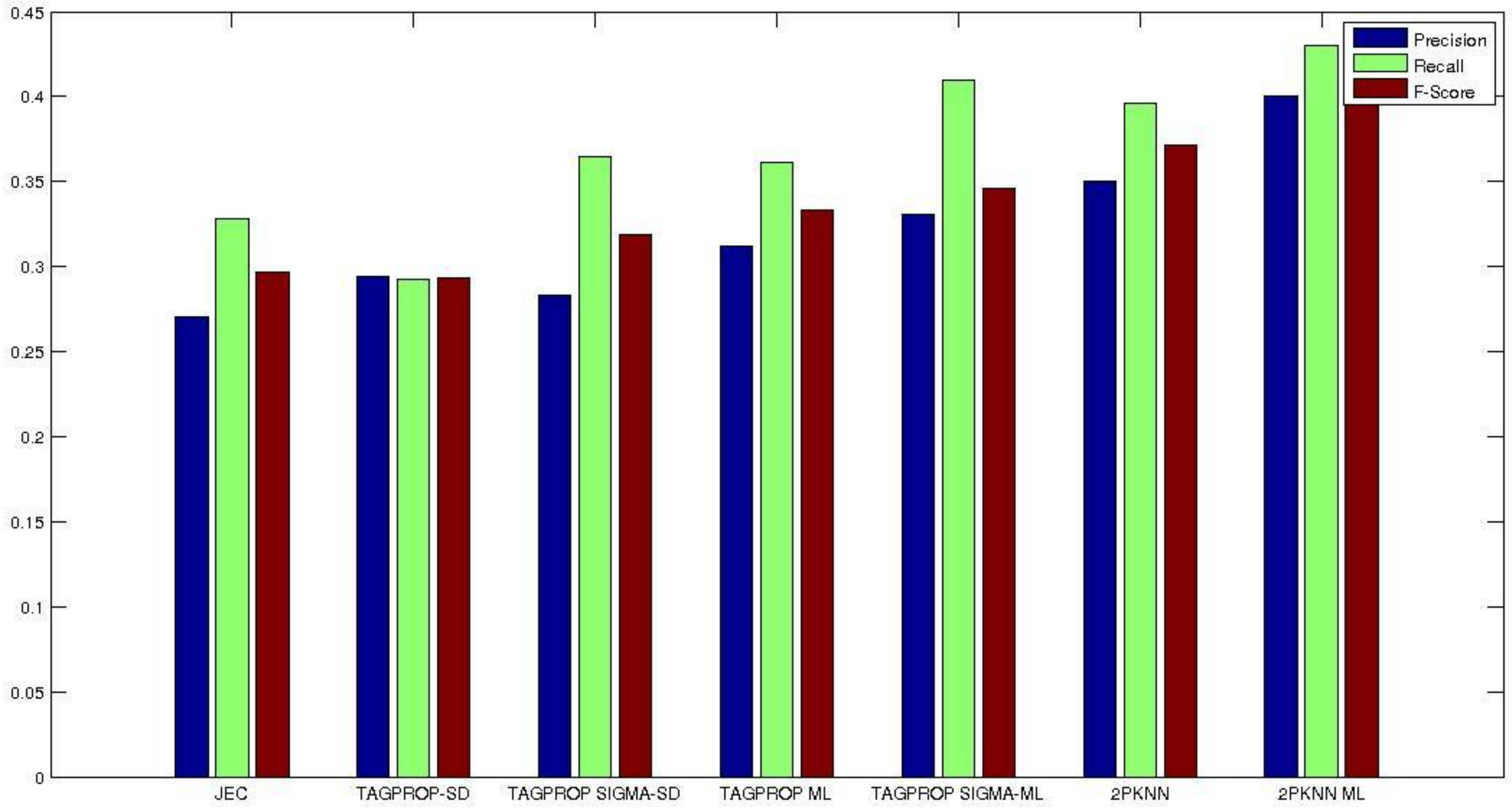
2PKNN (Two pass K-Nearest Neighbour)

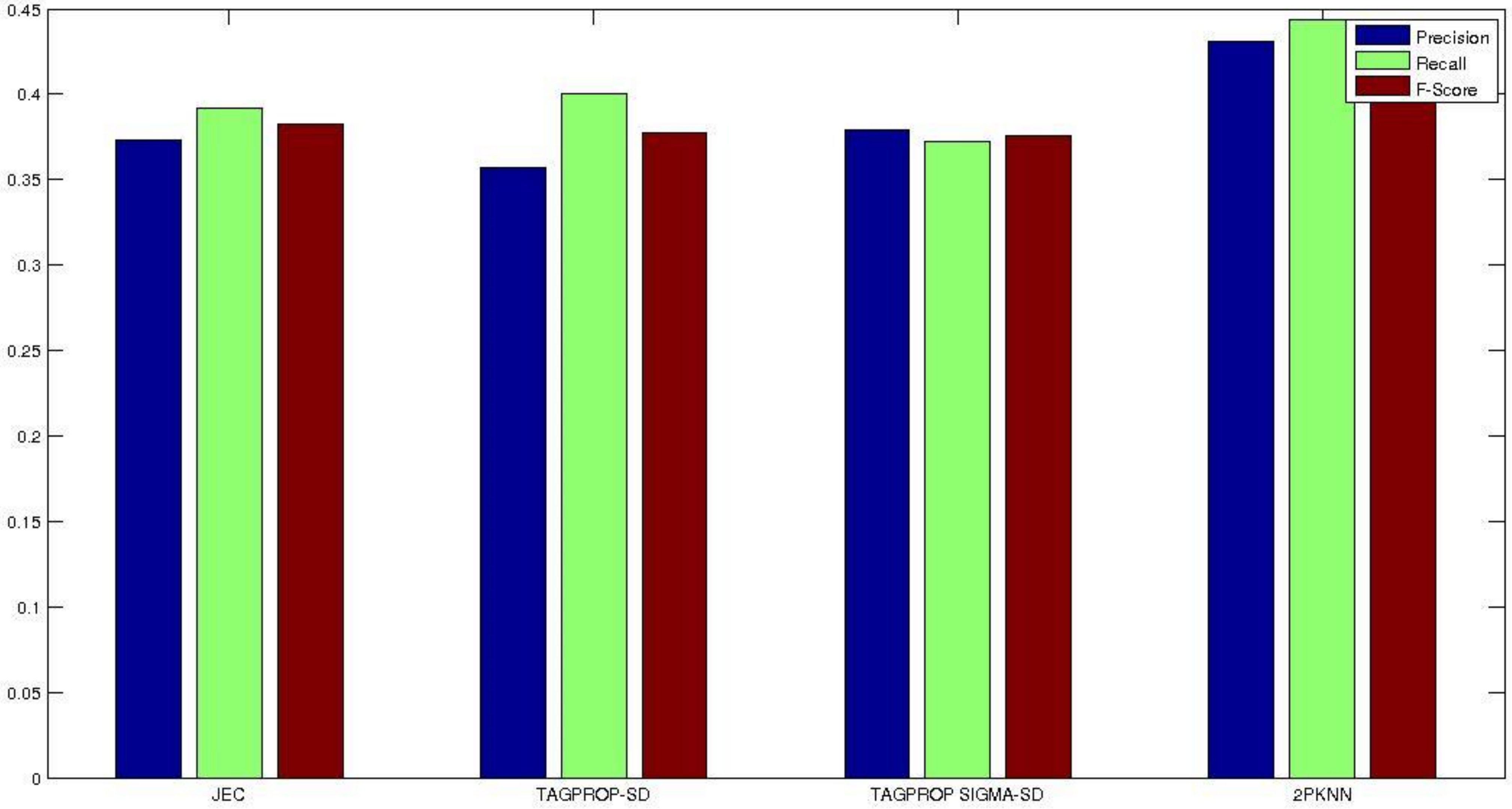
In this approach, instead of finding the nearest cluster, we filter out those results which are closer to the test vector. So, we find those R images which are closest in each cluster, and cluster those R images, and use them to classify the relevant tags.

Evaluation criterion

The evaluation criterion used

- Average per-label precision
- Average per-label recall
- Per-label F1 score
- Number of labels with positive recall


0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
Precision
Recall
F-Score
JEC
TAGPROP-SD
TAGPROP SIGMA-SD
TAGPROP ML
TAGPROP SIGMA-ML
2PKNN
2PKNN ML



0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
JEC
TAGPROP-SD
TAGPROP SIGMA-SD
2PKNN
Precision
Recall
F-Score

Role Of Validation

Validation dataset is defined to test the model in the training phase, in order to limit problems like overfitting, and to give an insight on how the model will generalize to an unknown dataset. The evaluation results of validation data is closer to the testing data when only 10% of the training data is used as validation data. We can observe that as the validation data increases the results move away from the test data and it overfits with the training data.

Per – Label Performance

- The label frequency plot of training set shows that there is significant amount of **class imbalance** i.e. lot of variation among frequency of classes.
- The issue of class imbalance is **successfully handled by 2pkNN** as visible from the frequency plot. The rare frequency labels occur in more number of test images.
- Feat-2 is less discriminative over lower frequency labels as compared to Feat-1.
- Increasing order of discrimination against lower frequency labels can be given as:
    - 2pknn-ML
    - 2pknn
    - TagProp – ML
    - TagProp
    - JEC

TagProp–sigma SD over JEC and 2pknn

Tagprop SD algorithm performs better than other two using ‘Feat2’. It is observed that in the given dataset the images need to be identified uniquely with labels rather and class imbalance seems on a lower side. Hence, we need to consider the weights of each label towards the same, but considering the uniqueness of the label at the same time which is handles by ‘sigmoid’ function in Tagprop. Thus, it performs better in such images.

2pknn over JEC and TagProp–sigma SD

2pknn is efficient in handling both class imbalance and weak labelling issues. JEC on the other hand, clearly suffers with class imbalance problem in the selected examples. TagProp suffers with weak labelling issues on selected examples.

Extension

Problem : Demonstrate how the annotation problem can be casted and solved using any one of (i) Decision Trees (ii) Neural Networks (iii) SVMs. Do explore thoroughly with the aim of obtaining superior results.

Multi label classification can be done using any of the above techniques. There can be two ways in which this problem can be solved :

1. Dataset transformation methods
2. Algorithm adaptation methods

Dataset transformation methods

- **Label Power set (LP)** : It is a straightforward method that considers each unique set of labels in a multi label training data as one class in the new transformed data. Therefore, the new transformed problem is a single label classification task. For a new instance, LP outputs the most probable class which actually is a set of classes in the original data.

- **Binary relevance (BR)**: It creates k datasets, each for one class label and trains the classifier on each of these datasets. Each of these datasets contains the same number of instances as the original data, but each dataset $D_{\lambda j}$, $1 \leq j \leq k$ positively labels instances that belong to class λj and negative otherwise.

Algorithm adaptation methods

- **Decision tree** based method Modify the entropy definition: Entropy $= - \square \{P(\lambda j)\log P(\lambda j) + (1 - P(\lambda j))\log(1 - P(\lambda j))\}$ where, $P(\lambda j)$ = probability of class λj . This allows to estimate the uncertainty in terms of number of bits in multi label setting. This modified method also allows multiple labels at the leaves.

- **BPMLL (Backpropagation multi label learning)** The error function for the very common neural network learning algorithm, backpropagation has been modified to account for multi label data. Multilayer perceptron is easy to extend for multi label data where one output node is maintained for each class label.

Algorithm adaptation methods

SVM based method

1. The first idea is to have an **extended dataset with k (= |L|) additional features** which are actually the predictions of each binary classifier at the first round. The k new binary classifiers are trained on this extended dataset. In this way the extended BR takes into account potential label dependencies.
2. The second idea, **ConfMat**, based on a **confusion matrix**, removes negative training examples of a complete label if it is very similar to the positive label.
3. The third idea is called **BandSVM**, removes very similar negative examples that are within a threshold distance from the learned decision hyperplane, and this helps building better models especially in the presence of overlapping classes.

Best Approach

After comparing the results of the above mentioned methods, it can be concluded that **LP and BR based random decision tree** approaches perform best overall. The following advantages drive this decision.

Advantages:

● Best training and testing time among all the algorithms.

● Good for cases when there are a large number of labels.

● Performs well in scene(or image) annotation task.

Random Decision Trees (RDT)

- RDT **constructs decision trees randomly and does not use label information much**. That nature makes RDT **fast and the computational cost independent of labels**.
- When constructing each tree, the algorithm **picks a remaining feature randomly** at each node expansion. Once a categorical feature is chosen, it is useless to pick it again on the same decision path as it will be the same throughout the remaining path. However, continuous features can be chosen lower down the path.
- **Classification is done at the leaf node level**. Each tree outputs a class probability distribution and the final output for each class is the average of all the values obtained from each decision tree.

LP-RDT

- Label Powerset considers each unique subset of labels that exists in the multi-label dataset as a single label. Let L be the set of all labels, L = {l1,l2,...,l|L|}. P(L) is the power set of L, and |P (L)| = pow(2,|L|) . Each element in P (L) stands for one possible combination of labels.
- During the procedure to construct random trees, except for a leaf node that summarizes class label distribution, there is no need to use the training examples class labels. Except on leaf node, the multiclass random tree building procedure is the same with binary classification tree.

BR-RDT

- Binary Relevance (or BR) **learns one binary classifier hk for each label k**.
- It builds **jLj datasets** by using all the instances with one label k each time.
- To classify an unlabeled instance, BR generates the multilabel prediction by summarizing the output of jLj classifiers.
- The **problem** of BR method is that it needs **as many classifiers as the number of labels**. When the number of labels is large, the training and test computation cost becomes significant.
- As a **random tree constructed is actually independent from class labels**, it can instead classify all labels, and an ensemble of random trees can be used for the problem with one to even hundreds of labels. The added trivial computation is only the label probability distribution counting on each leaf node.

Risk Analysis

- Learning risk of RDT (both BR and LP) is stable.
- The classified errors for training instances decrease monotonically with the increasing of the number of trees, the expected training risk of RDT will also decrease.
- Risk bound of RDT will not increase with the increasing of the number of trees and the risk doesn't exceed the average risk of all the trees.
- Under the optimistic situation, RDT can minimize the generalization error with the increasing of the number of trees.

Time Complexity

- Training complexity of LPRDT is O(mn[log(mn)]). n is the number of leaf nodes and m is the number of decision trees.
- While that of BRRDT is O(m(n log((n) + tn)).
- Note that the time complexities are independent of the number of labels.
- The test complexity of both LP and BR RDT is O(mnq), where m is the number of decision trees, n is the number of test instances and q is the average height of the decision trees.

Evaluation

The algorithm was tested on scene dataset which is a dataset for image annotation problem.

Metric	LP - RDT	BR - RDT
Accuracy	0.751	0.737
Recall	0.751	0.737
Precision	0.788	0.755