Applying Maximum Mean Discrepancy for obtaining Entity Occurrence Ratios

Aman Madaan

May 8, 2014

1 Introduction

The final exercise was application of Maximum mean discrepancy for learning entity occurrence ratios. We continue with the same dataset, pruned to remove all the entities for which there are only single instances, leaving us with 212 different entities. The results indicate that svm is better than mmd for estimating the class ratios. We discuss the results, possible reasons for error and suggest ways of improvement.

1.1 Features

As discussed, the features for struct learn take the following form

$$E_{true}\ E_1 : W_1a, W_1b, W_1c ... E_n : W_na, W_nb, W_nc$$

where E_{true} is the true entity E_i (Hand labeled) and the weights are as defined before.

We now flatten the features using the maximum entity id as follows :

$$E_{true}\ E_1 + 0 * MEID : W_{1a}E_1 + 1 * MEID : W_{1b}E_1 + 2 * MEID : W_{1c} ... E_n + 0 * i : W_{na}E_n + 1 * i : W_{nb}E_n + 2 * i : W_{nc}$$

where MEID is the highest entity id.

1.2 Data

Out of the pruned data points (1294), we deleted instances for which only one mention per class was available. Upon deletion of such points, we were left with 966 rows.

1.2.1 Sampling

Subset.py from the libsvm tools was used for randomly dividing the training data into training and test sets.

1.3 Training Data

- 807 mentions
- 212 different (entities) classes
- average 3.8 mentions per entity
- median 2 mentions per entity

1.3.1 Test Data

- 159 mentions
- 109 different (entities) classes
- average 1.5 mentions per entity
- median 1 mentions per entity

For this dataset, it was ensured that for every class in the test set, we have at least one instance in the training set.

2 Code

This section briefly outlines the functionality of the different packages in mmdned. The MirrorDescentOptimizer by Prof. Sunita Sarawagi was used for solving the objective.

2.1 Gradient

- MakeMatrices : Calculates matrices A and b required to calculate objective and the gradient
- NedGradientComputer : Calculates the gradient. An instance of NedGradientComputer gradient compute has to be supplied to the optimizer during its instantiation.

2.2 Kernel

Implementation of the Gaussian Kernel to handle the sparse feature representation.

2.3 Neddata

Framework for representation and handling of the sparse features.

- DataSetInfo : Stores standard facts about the dataset (Number of classes, class proportions etc)
- Feature : Represents a feature, consisting of an id and a value
- NedRow : A row consists of a true label followed by a list of features
- NedDataReader : Returns a list of NedRows with DataSetInfo filled in

2.4 Finding Parameters

We use the gaussian kernel to determine similarity between two instances.

$$K(x1, x2) = exp^{-\gamma * ||x1 - x2||^2} \tag{1}$$

Where $x1$ and $x2$ are sparse representations of the features. The only parameter to be supplied is gamma. We find the gamma by using 5 fold cross validation on the training data.

3 The case of impossible ratios

Consider the case where we have 10 unlabeled instances. In such a setting, we *cannot* have class ratios with one of the elements set to 0.99 (9.9 elements belonging to a class is not possible!). In fact, for the current case, any class proportion in which any of the elements takes a value which when multiplied by 10 does not yield an integer is impossible.

Although it is easy to see that not every class proportion is possible given the unlabeled dataset, yet the mmd formulation does not consider this fact right now. This may lead to errors, since we are looking for the solution in the complete R^n space where we *must* be looking in only a proper subset of it. Intuitively, we need to add the following constraint to the mmd objective.

> Let $|U|$ be the number of elements in the unlabeled test set. For any element θ_i of then (unknown) class proportion vector, $\theta_i * |U|$ should be an integer.

4 Modified optimization objective

The current mmd objective can be seen as an approximation to the following optimization objective.

Given an unlabeled dataset U, find the values of n_i such that

$$\Sigma_i^c n_i = N \tag{2}$$

where N is the total number of unlabeled instances, c is the number of classes and n_i is the number of instances belonging to the i^{th} class. Clearly, each of the n_i needs to be an integer. Dividing the objective above by N on both sides and rewriting n_i/N as θ_i gives us the current optimization objective. The problem

is that once we do that, there is no way to restrict θ_i to only those values that are legal, i.e. fractions n_i/N where $n_i <= N$.

As an extension to the current work, we can use an IP solver to obtain the required values.

5 Rounding the fractions to obtain the "correct' ratios

A hack to convert the ratios obtained via mmd to one of the possible ratios is as follows. We first multiply each element of the ratio vector by the number of unlabeled instances. Each entry is then rounded and finally normalized to obtained the new ratio.

This boils down to the following matlab operation :

$$r' = round(r * |U|)/sum(round(r * |U|)) \tag{3}$$

Where r' are the rounded proportion vector, r is the original proportion vector and $|U|$ is the number of instances in the unlabeled dataset.

6 Results

We compare the ratio vectors obtained from the different methods. To quantify the difference between two ratio vectors, we use the L1 norm.

$$l1(x1, x2) = \Sigma_{i=0}^{n} |x1_i - x2_i| \tag{4}$$

Method	L1 Norm
SVM	0.20482
STRUCT LEARN	0.22892
MMD	0.5561

We use 3 different plots to visualize the difference between true and obtained ratios.

- **Dispersion Plot** Let k be the total number of elements in the ratio vector. Let *true* be the true ratio vector and let *model* be the obtained ratio vector. Define

$$I = [1, 2, ..., k] \tag{5}$$

$$model'_i = \frac{model_i}{true_i} * i \ \forall i \in I \tag{6}$$

  We then plot $model'$ *vs* I. In the ideal case, the plot would be a straight line. Diversion from the straight line thus becomes an indicator of the degree of error of the ratios obtained.

- **Cumulative Frequency Plot** Plot of cumulative sum of the obtained ratios and the original ratios. In the ideal case, the shape of the 2 curves should match.

- **Error Plot** We plot the error of the original ratios and the obtained ratios. In the ideal case, all the points should fall on error = 0 line.

6.1 Plots

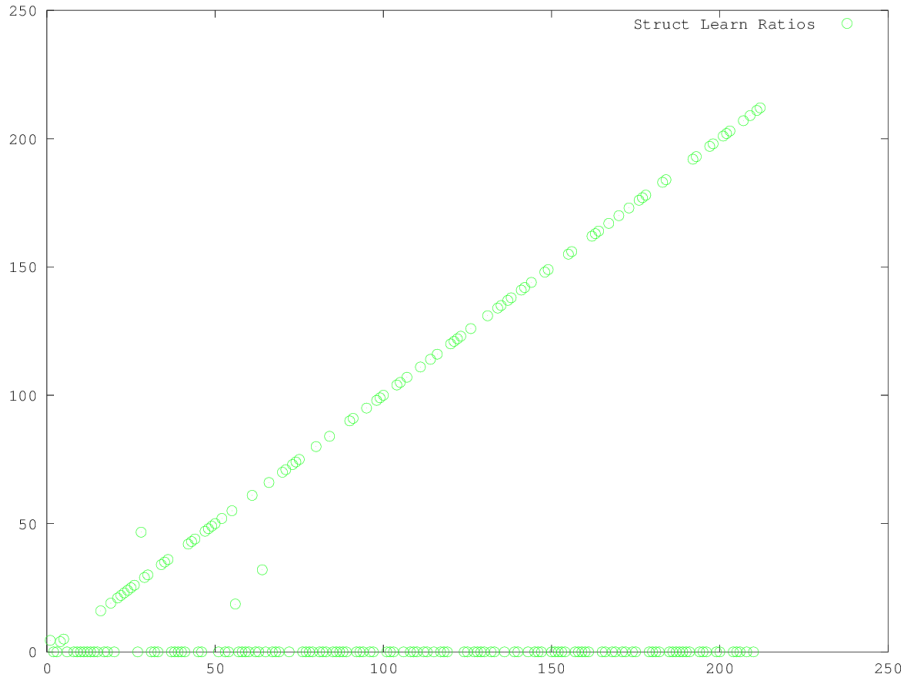

Figure 1: Ratios Obtained via Structured Learning

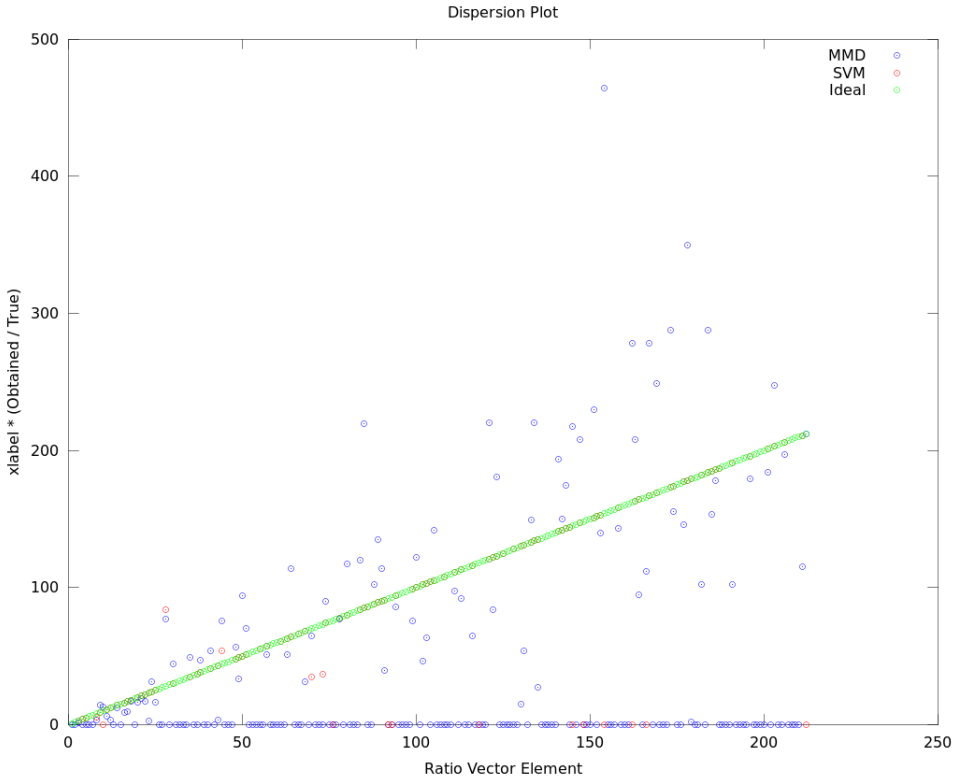

Figure 2: Digression : SVM vs MMD

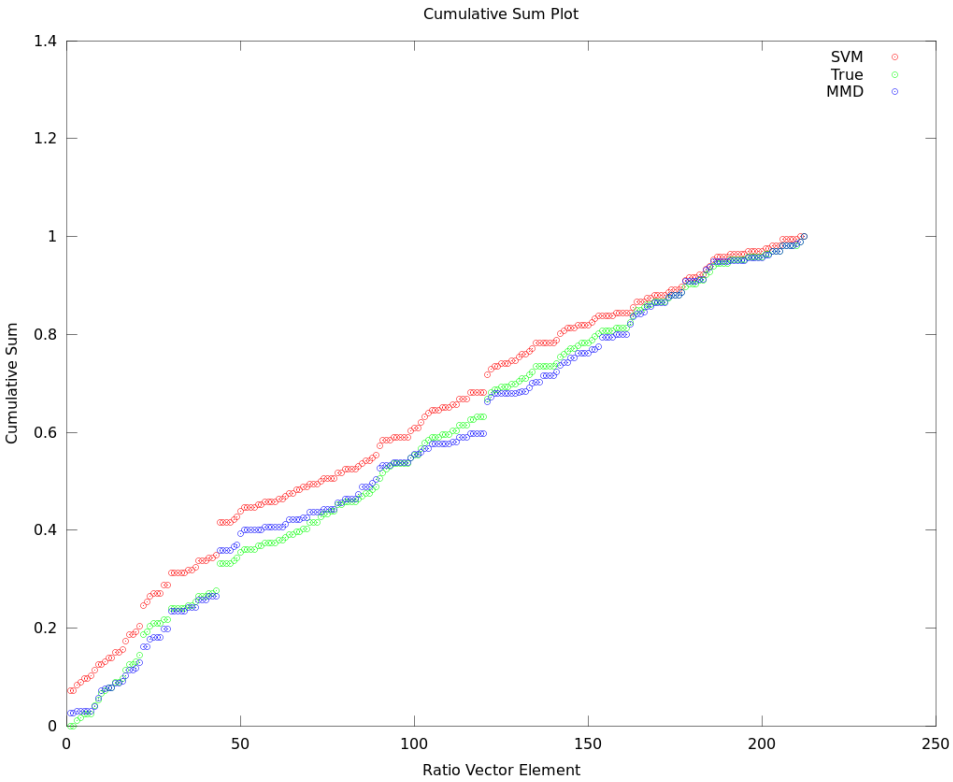

Figure 3: Cumulative Ratio

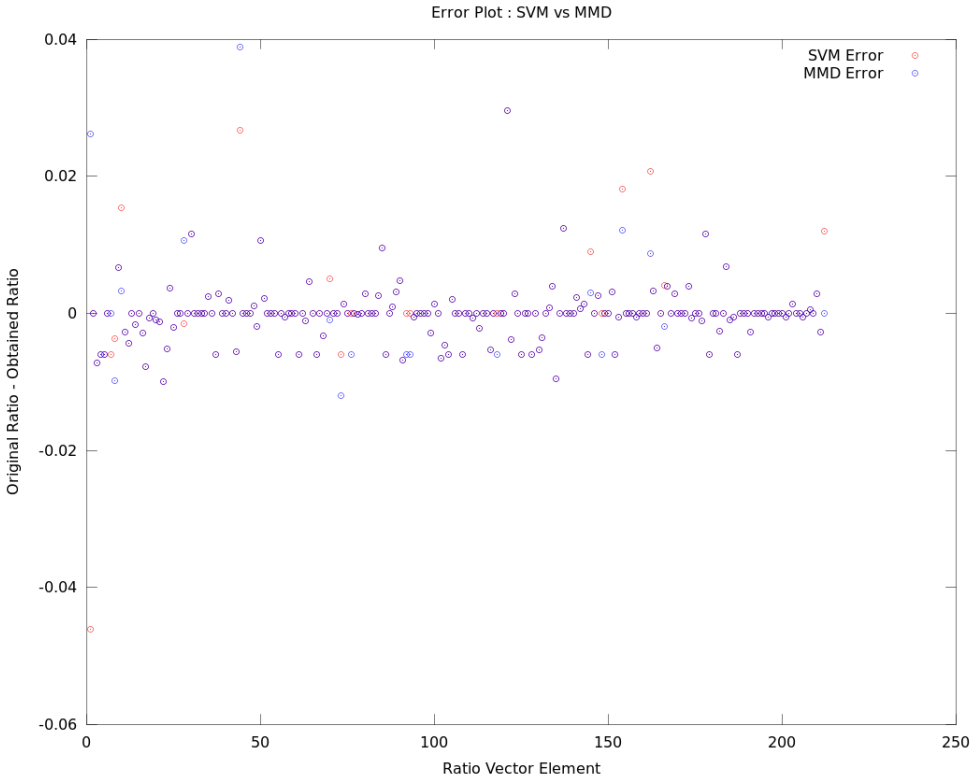

Figure 4: Per Element Error

7 Results for different sets

To test our hypothesis of the rounded ratios, we created 5 different datasets and trained svm and mmd on them. The results are as shown in table [7]. We note that although svm outperforms mmd approach, there is a consistent improvement in performance with rounding. L1 denotes the L1 error as defined above and PC denotes the per class error.

Run	SVM L1	SVM L1 PC	MMD L1	MMD L1 PC	MMD Rounded L1	MMD L1 PC
1	0.13	0.00061	0.577	0.00272	0.553	0.00261
2	0.144	0.00068	0.506	0.00239	0.47669	0.00225
3	0.1083	0.00051	0.54423	0..257	0.49288	0.00232
4	0.14458	0.00068	0.59434	0.00280	0.59036	0.00278
5	0.25301	0.00119	0.58582	0.00276	0.54217	0.00256

Please note that for these datasets, there can be unseen classes in the test set

8 Error Analysis

- **Kernel Selection** Experiments with multiclass datasets have earlier shown that mmd is highly sensitive to kernel selection. The earlier method of cross validation by fixing class proportions cannot work in the current setting because of lack of the training data. We thus use 5 fold cross validation. Improper γ learning can be a major source of error for the mmd methods.

- **Lack of training data** The training data had 4.5 mentions per entity on average. These instances may not be a good representative of an *average* mention for the candidate.

9 Conclusions

- **Restricting thetas** As mentioned, the search space for the θs can be restricted to a great extent by adding constraints to ensure that only possible values of the θs are considered while coming up with the best. This will complicate the existing mmd formulation, but our results have consistently indicated that the results will greatly benefit from such additional constraints.

- **Clustering Entities for disjoint occurrence statistics learning** Can we drive around the problem of dealing with large number of classes by just finding the ratios for each set of disambiguations? For example, can we find the distribution of all the different Michael Jordans on the web by collecting all the mentions that refer to one of these?

  Though the idea looks intuitive, there are multiple challenges involved. First of all, can we even create such disjoint subsets of mentions? Even if we can, how will we ensure that test set will have mentions that only have candidates from the set of Michael Jordans?

- **Cross validation** The existing method of obtaining γ via cross validation depends on varying the class proportions. This method cannot be applied to the case in hand because there aren't enough instances per class. To add to the problem, the number of classes is also very large.