Exploring the Space of Coding Matrix Classifiers for Hierarchical Multiclass Text Categorization

Janez Brank

Introduction

- We’ll be dealing with (single-label) multi-class classification problems
  - Each instance is assigned to one (out of *k*) classes
- One way to handle a multi-class classification problem is to transform it into several binary (i.e. 2-class) problems
  - For each new binary problem, we have to define what is the positive and what is the negative class
    - We define the positive class as the union of one or more classes from the original problem
    - Likewise for the negative class
    - Some of the original classes might remain unused in this particular binary problem
  - We train an ensemble of binary classifiers, one for each of these new problems
  - Combine their predictions through some sort of voting to get a prediction for the original multi-class problem

Coding Matrices

- The relationship between the k classes of the original multi-class problem and the m new binary problems can be concisely described by a $k \times m$ coding matrix
  - One row for each original class
  - One column for each new binary problem
  - Entries are +1, -1, 0, meaning that the original class is used as positive / negative / unused in that particular binary problem
- Typical approaches for defining multi-class problems:
  - One vs. others: k problems, the i'th problem uses class i as positive and other classes as negative
  - One vs. one: $k(k + 1)/2$ problems, one for each pair of classes (i, j), using i as positive, j as negative, others unused
  - Exhaustive – one column for each partition of the original k classes into positive and negative
  - Error-correcting output codes
- Coding matrices are a generalization of all these
  - The space of all possible coding matrices is exponentially large in k and m
  - And gives rise to a corresponding space of classifiers for the original problem
  - We're interested in knowing more about this space of classifiers, about their performance and its relationship to the properties of the coding matrices

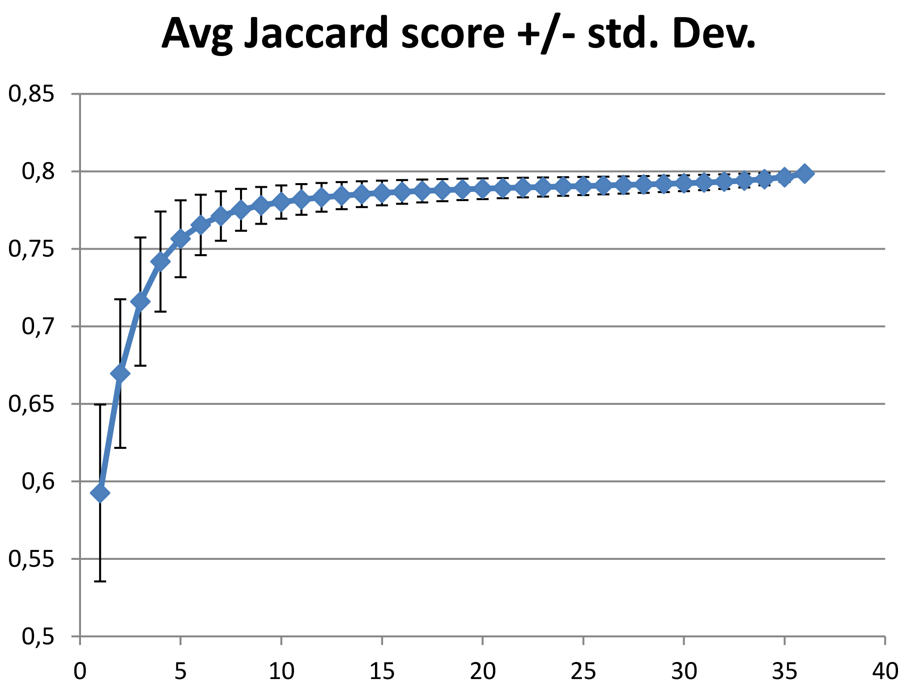
Exploring the Space of Coding Matrices

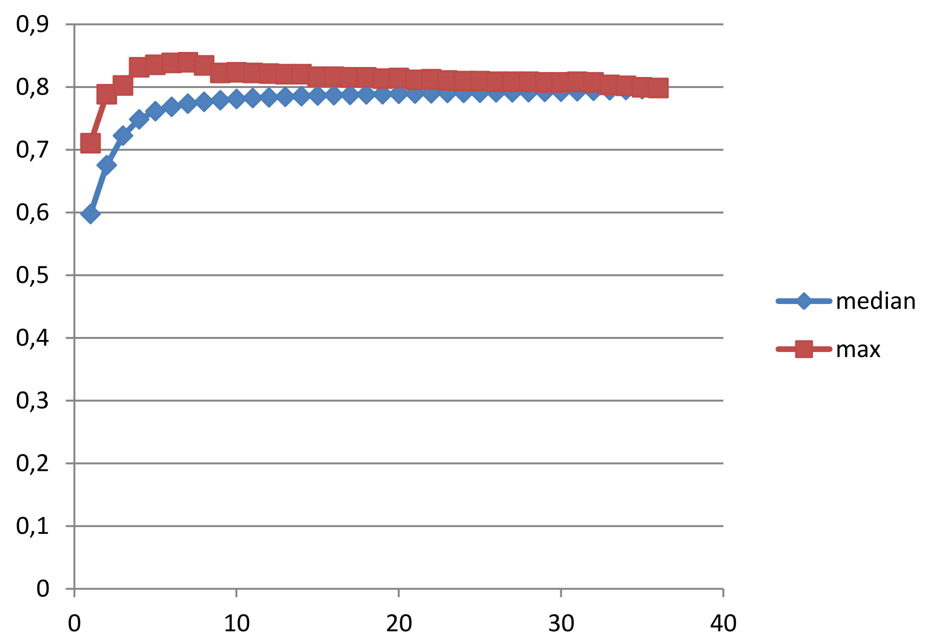
- We'll take a small multi-class problem
  - 7 classes, arranged into a 3-level hierarchy
  - This imposes an additional constraint, if a class is positive/negative, its descendants (subclasses) must also be positive/negative
  - Thus there are not 3^7 ways to fill up a column of the matrix, but fewer
  - Additionally, a column should contain at least one +1 and at least one -1
  - This leaves us with 36 possible states of a column
  - If we don't want to have multiple identical columns in the matrix, there are only (36 choose *m*) = 36!/(*m*! (36 – *m*)!) possible matrices of *m* columns
  - For small *m*, or *m* close to 36, we can examine all possible matrices; for intermediate *m* this is intractable so we examined a random sample of 10^6 matrices for each *m*
  - We evaluate the classification performance of each matrix by the *average Jaccard score* of its predictions
    - Since the classes are in a hierarchy, not all misclassifications are equally wrong
    - So this measure gives a higher score to predictions where the predicted class was close to the correct one in the hierarchy

m	1	2	3	4	5	6
$\binom{36}{m}$	36	630	7 140	58 905	376 992	1 947 792

Performance as a function of *m*

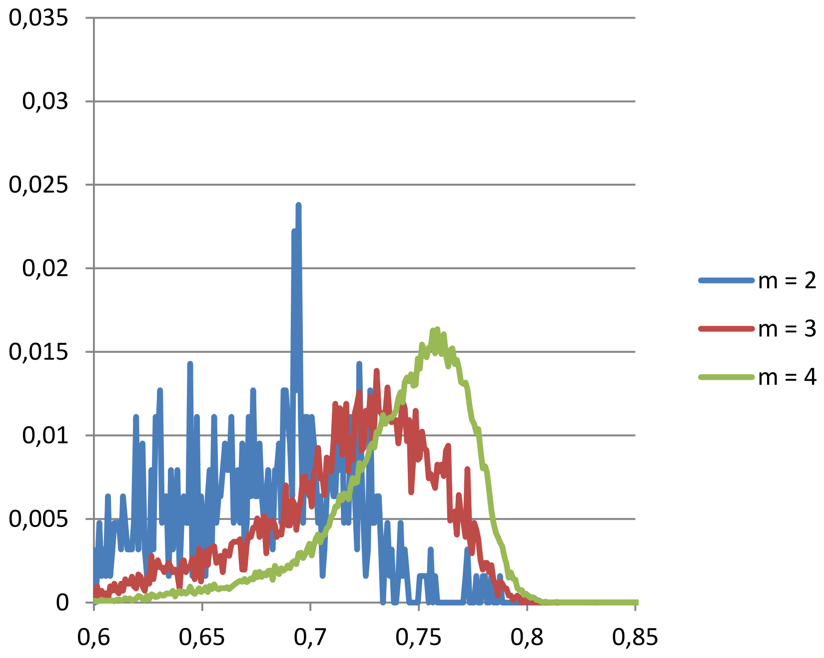
- What is the average / median / best performance over all *m*-column matrices, as a function of *m*?

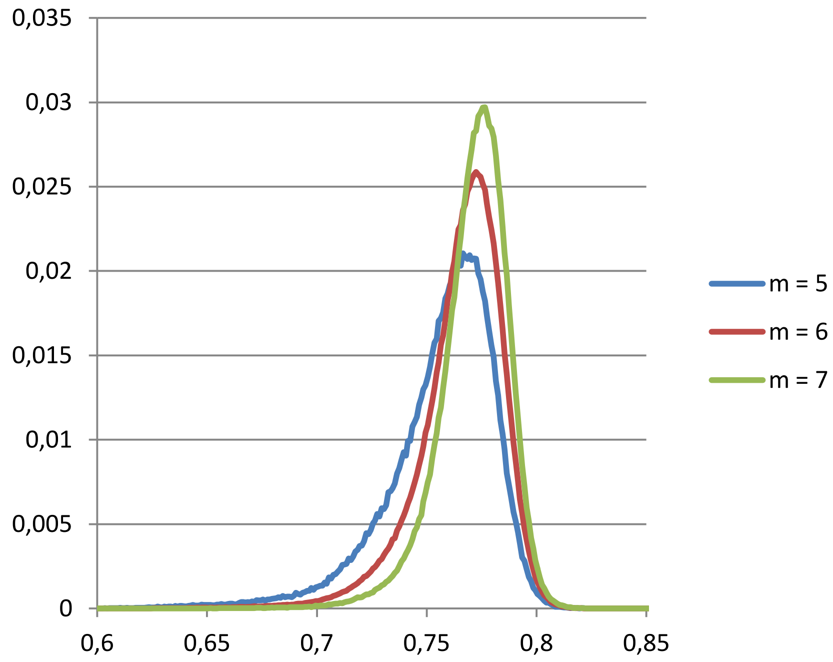




Distribution of matrix scores

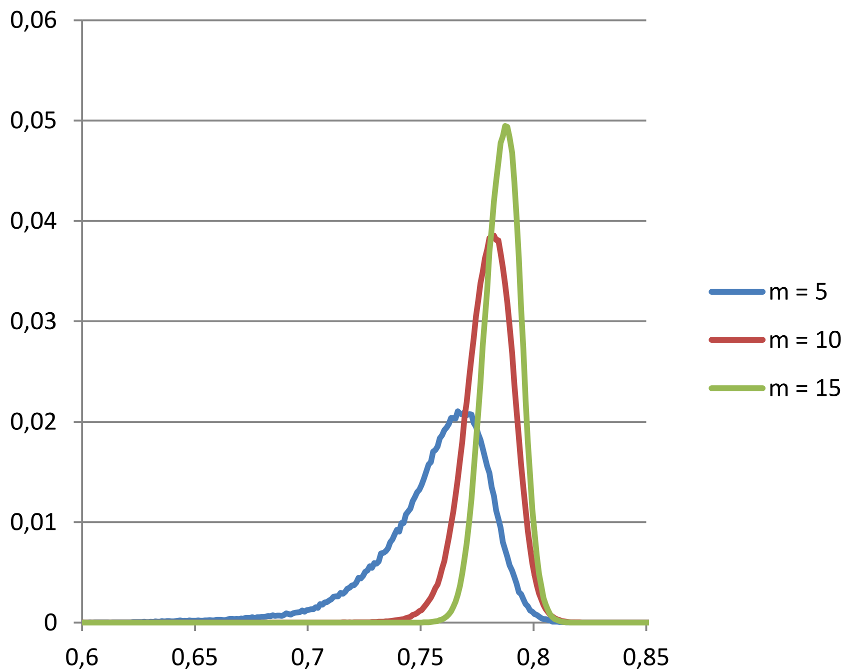
- Suppose we choose a *m*-column matrix randomly and observe its performance score
  - This score can be thought of as a random variable
  - What is its distribution (depending on *m*)?

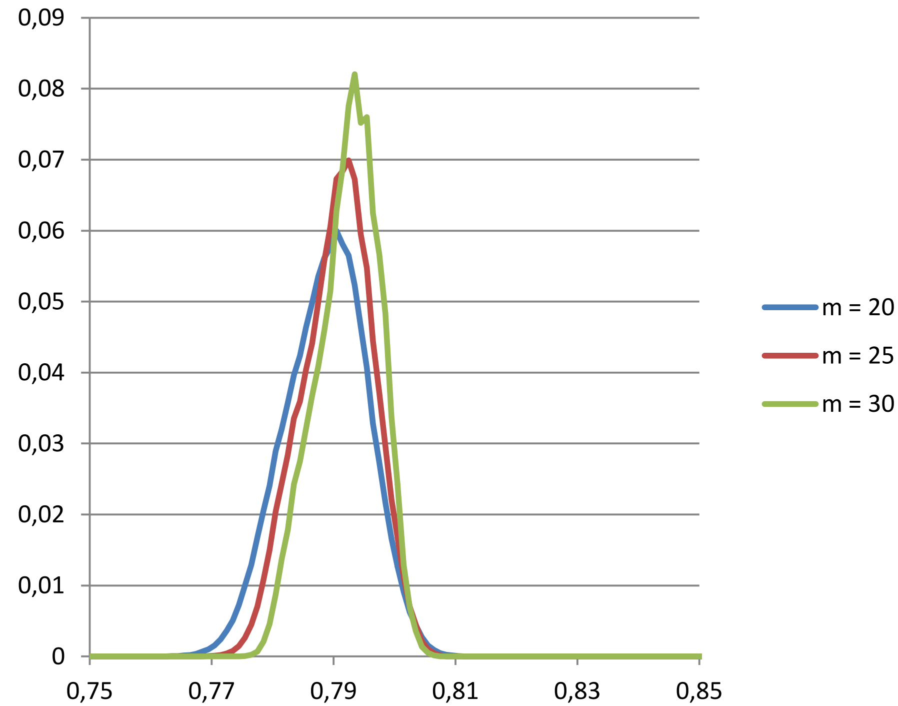




Distribution of matrix scores

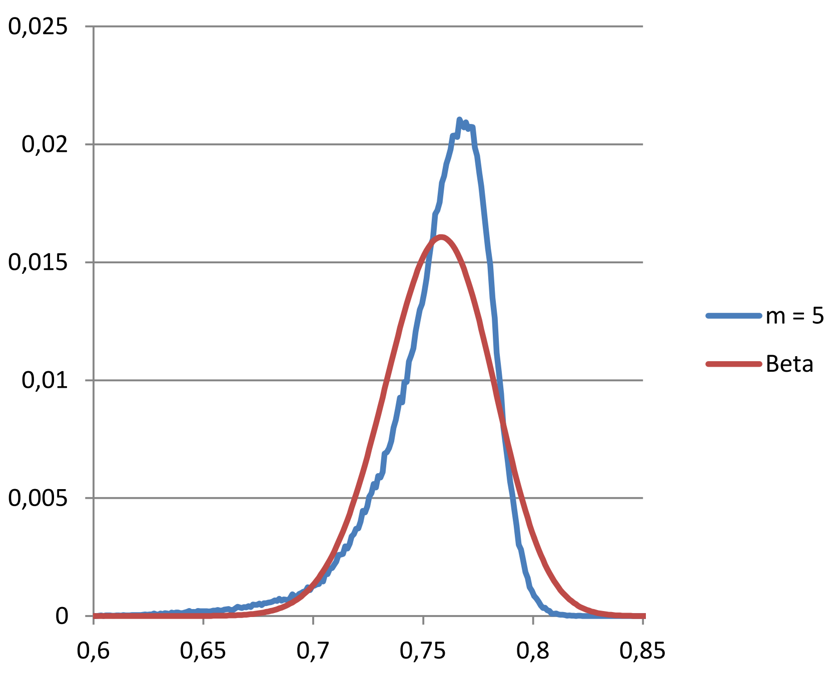
- Suppose we choose a *m*-column matrix randomly and observe its performance score
  - This score can be thought of as a random variable
  - What is its distribution (depending on *m*)?

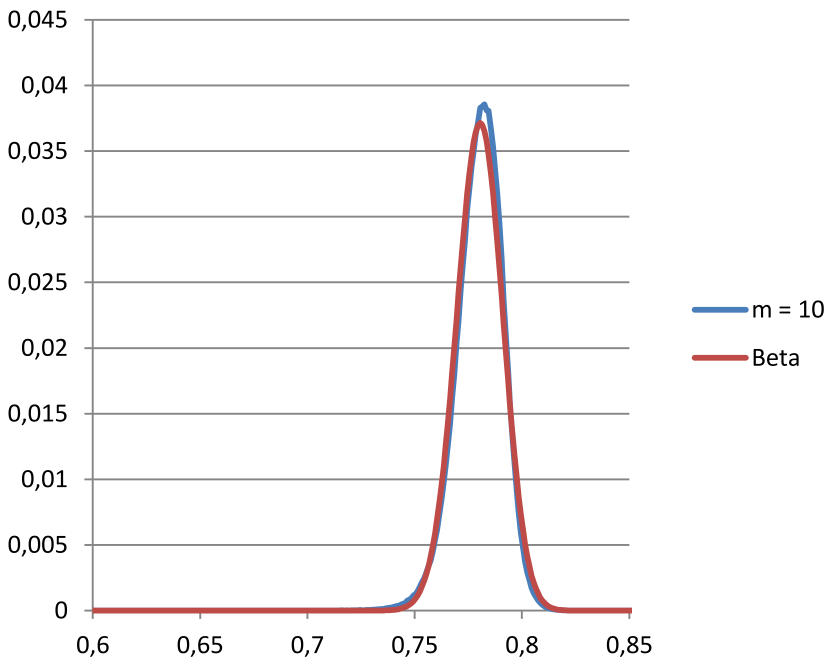




Fitting a beta distribution

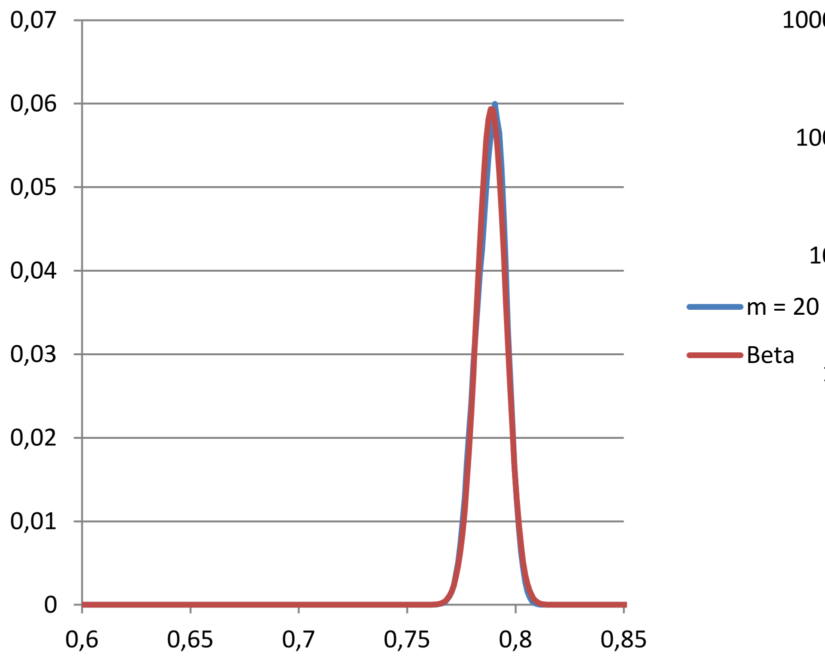
- The shape of these empirical distributions can be roughly modelled by a beta distribution
  - We estimate its parameters α_m and β_m using the method of moments
  - The fit is better for higher values of *m*

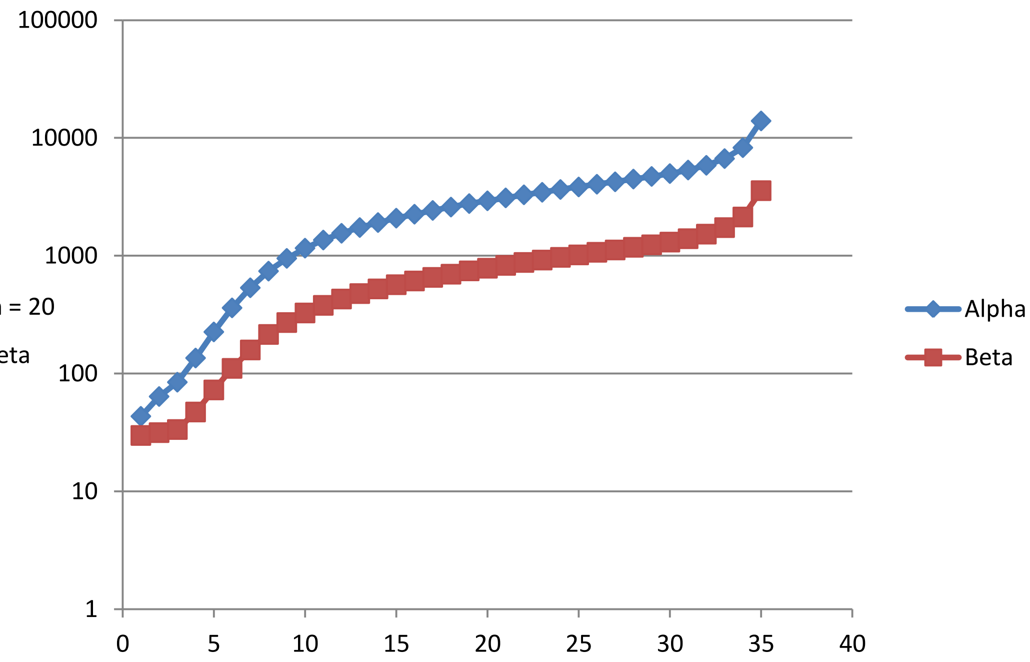




Fitting a beta distribution

- The chart on the right shows α_m and β_m as a function of *m*
  - We can see ranges [3, 7] and [10, 30] where both parameters are roughly exponential in *m*

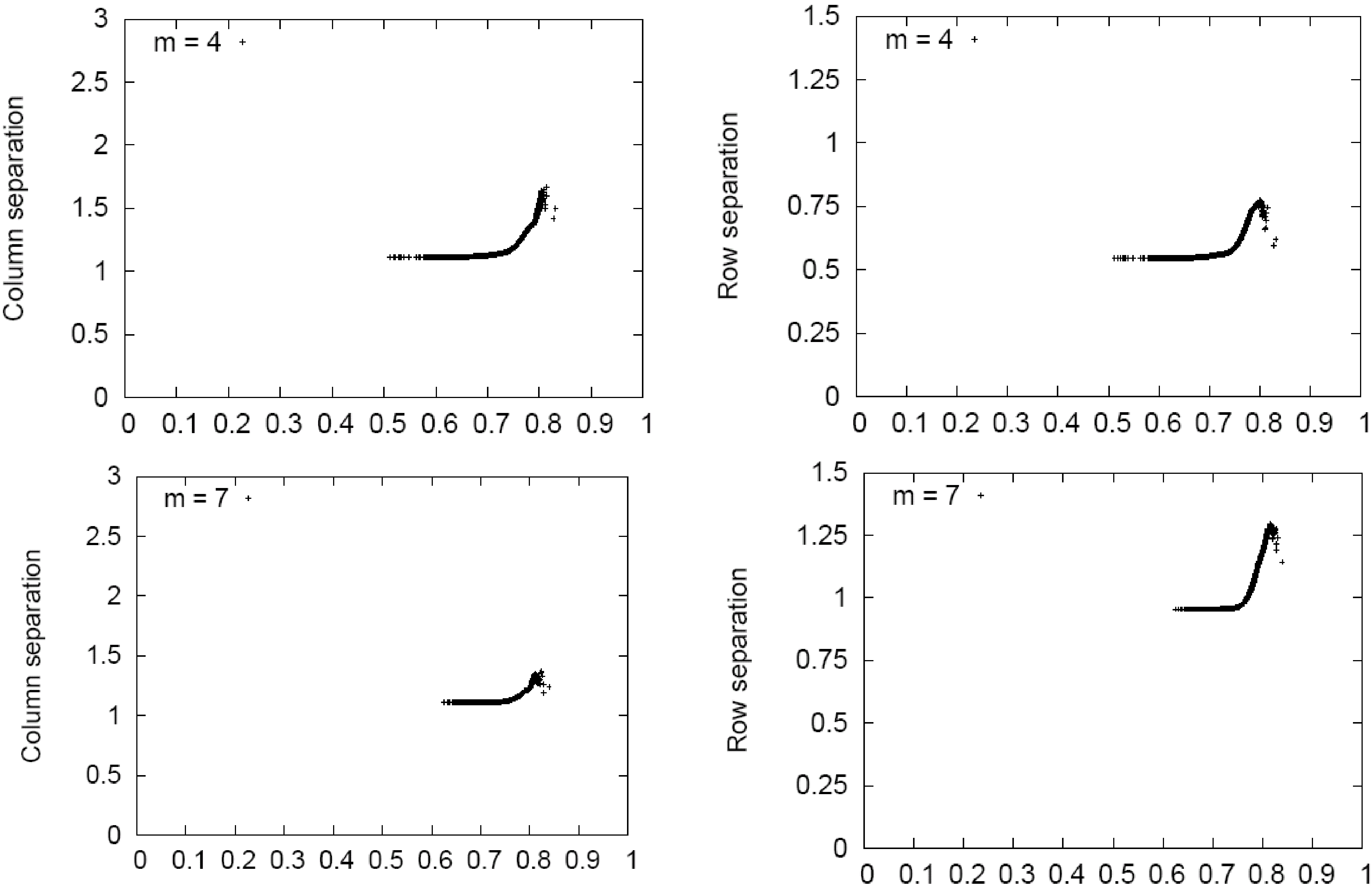




Matrix score vs. row/column separation

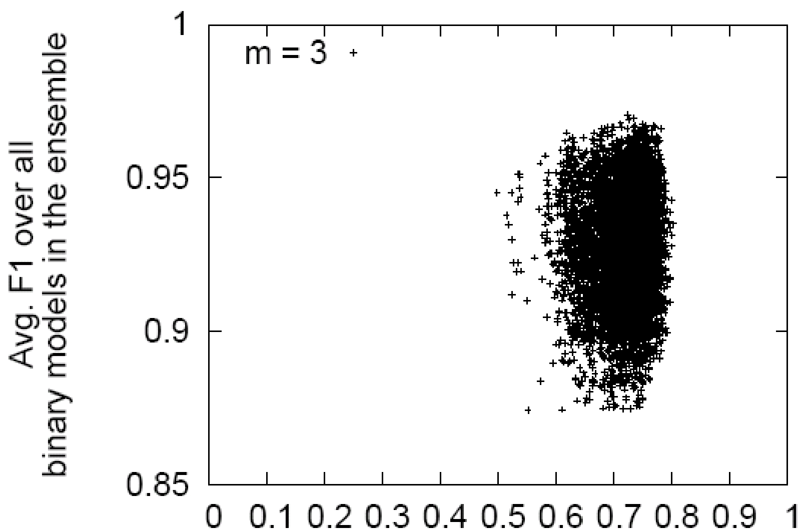
- It is considered desirable that the rows of the matrix aren't too similar to each other, and likewise the columns
  - Similar rows $\rightarrow$ expected predictions for classes corresponding to those rows are very similar and will be easily confused
  - Similar columns $\rightarrow$ resulting in similar binary classifiers, making mistakes at the same time, making decoding harder
  - Row (column) separation = average Hamming distance over all pairs of rows (columns)
  - What is the relation between the row (column) separation and the performance score of the matrix?

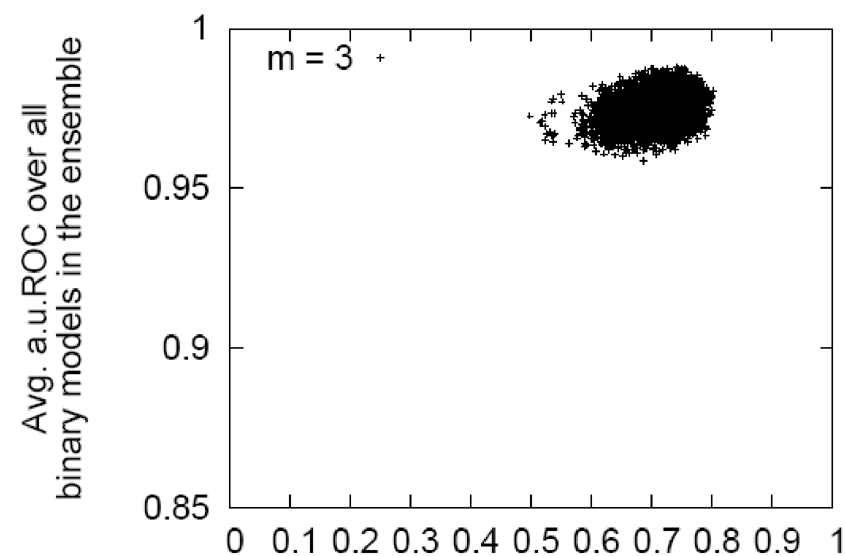
Matrix score vs. row/column separation



Binary classifier performance vs. Ensemble performance

- The matrix defines an ensemble of binary classifiers
  - Is the ensemble as a whole better if the individual binary classifiers are better?
  - We computed the F_1 and area under ROC measures for each individual binary classifier (relative to its own binary classification problem)
  - Is the average of this score (over all classifiers in the ensemble) correlated with the score of the entire ensemble?
  - Not really: R = 0.015 R = 0.25





Conclusions

- The best matrices are found at m = 7 or 8 columns
- At m = 4 we can still find almost equally good matrices, but they are much more rare
  - Ideally we want ensembles that perform well and have few classifiers
  - This result shows it might be worth looking for them, as they *do* exist
- At higher values of m, the best matrices aren't that good but good matrices are much easier to find
  - The more classifiers we can afford, the more we can afford to just pick a random matrix and use it
- The distribution of matrix scores is approximately like a beta distribution
  - With parameters α_m, β_m that are approximately exponential in m over wide ranges of m
- Row and column separation are indeed useful properties
  - Matrices with high row/column separation have good performance
  - *But* the matrices with the best performance are not the ones that maximize row/column separation
- The quality of individual binary classifiers in the ensemble is not correlated with the performance of the matrix as a whole
- Future work: test on more datasets, larger number of original classes (k), investigate other matrix properties (besides row/column separation)