A Probability Model for Combining Ranks

Ofer Melnik, Yehuda Vardi, and Cun-Hui Zhang

Rutgers University, Piscataway NJ, USA
melnik@dimacs.rutgers.edu, {vardi, czhang}@stat.rutgers.edu

Abstract. Mixed Group Ranks is a parametric method for combining rank based classifiers that is effective for many-class problems. Its parametric structure combines qualities of voting methods with best rank approaches. In [1] the parameters of MGR were estimated using a logistic loss function. In this paper we describe how MGR can be cast as a probability model. In particular we show that using an exponential probability model, an algorithm for efficient maximum likelihood estimation of its parameters can be devised. While casting MGR as an exponential probability model offers provable asymptotic properties (consistency), the interpretability of probabilities allows for flexiblity and natural integration of MGR mixture models.

1 MGR as a Score Function

Many rank combination approaches can be cast as the problem of assigning scores to classes based on the ranks they receive from multiple consituent classifiers. Once assigned, then classes can be ordered based on their scores, generating a combined ranking.

We use the following notation. There are K target classes $t_1, \ldots, t_K$ and a collection of J component classifier algorithms $a_1, \ldots, a_J$. For any particular query, the output of algorithm a_j is $r^{(j)} \equiv \left(r^{(j)}(1), \ldots, r^{(j)}(K)\right)$, with $r^{(j)}(k)$ being the rank assigned by algorithm a_j to class t_k. A score function, for each class maps the rankings to a scalar

$$S(\theta) \equiv f_\theta \left(r^{(1)}, \ldots, r^{(J)}\right)$$

where θ is a class in $t_1, \ldots, t_K$. As score functions are ultimately used to generate new rankings, they have the important property of being invarient to monotonic transformations.

$$R^*(k) \leq R^*(k') \Leftrightarrow S(k) \geq S(k') \Leftrightarrow g(S(k)) \geq g(S(k')) \tag{1}$$

where g is monotonically increasing, and R^* denotes the combined ranking derived from the class scores.

Some example score functions are the Borda count, Linear Score and the Best Rank [2]. The Borda count is a voting method, which assigns the (negative) sum of ranks as a score, $S_{Borda}(\theta) = -\sum_{j=1}^{J} r^{(j)}(\theta)$. The Linear Score

N.C. Oza et al. (Eds.): MCS 2005, LNCS 3541, pp. 64–73, 2005.

generalizes the Borda count by assigning a weight to each classifier, $S_{Linear}(\theta) = -\sum_{j=1}^{J} w_j r^{(j)}(\theta)$. The Best Rank score selects the best rank a class receives as its score, $S_{Best}(\theta) = -\min_{j \in 1 \ldots J} r^{(j)}(\theta)$.

In [1] we proposed the Mixed Group Ranks (MGR) score function which generalizes the Best Rank and Linear scores. It is a weighted linear sum of the minimum rank of all subsets of classifiers

$$S_{MGR}(\theta) = -\sum_{A \subseteq \{1 \ldots J\}} w_A \min_{j \in A} r^{(j)}(\theta)$$

The MGR score function combines the democratic voting aspect of the Linear and Borda Scores with the emphasis on confident rankings of the Best Rank score. In [1] we describe the general category of score functions that embody these characteristics; Score functions that are both monotonic and quasiconvex (in the ranks assigned to a class) prefer lower ranks to bigger ranks and assign greater influence to smaller ranks. With non-negative weights, $w_A \geq 0, \forall A$, MGR is both monotonic and quasiconvex, embodying these score properties.

2 Probabilistic Framework for Rank Combination

In this section we reformulate combination of rank classifiers as a problem of estimating maximum likelihood (ML) and Bayes rules.

Unlike a black-box score function approach which does not model the process used to generate the constituent rankings, in the probabilistic setting we consider the rankings generated by the constituent classifiers as coming from a stochastic process, where

$$p_\theta\left(r^{(1)}, \ldots, r^{(J)}\right) \equiv p\left(r^{(1)}, \ldots, r^{(J)} \,|\theta\right) \tag{2}$$

is the conditional joint distribution of $R^{(j)} \equiv \left(R^{(j)}(1), \ldots, R^{(j)}(K)\right)$, $j \leq J$, as J random vectors, with $R^{(j)}(k)$ being the rank assigned by algorithm a_j to class t_k, when the actual class is t_θ, $\theta \in 1 \ldots K$. This formulation is the key to a probabilistic combination approach, considering each combination of ranks as having a distinct conditional probability that implicitly captures the biases and interactions in the rank combinations.

The likelihood function of the probability, $L(\theta) \equiv p_\theta\left(R^{(1)}, \ldots, R^{(J)}\right)$, can form the basis for statistical combination procedures. In particular, the *ideal* Maximum Likelihood combination method would assign combined ranks, $R^*(k)$, to each class t_k that satisfy

$$R^*(k) \leq R^*(k') \Leftrightarrow L(k) \geq L(k') \tag{3}$$

according to the likelihood function $L(\cdot)$. This rule estimates θ, the index of the true class by $\widehat{\theta} \equiv \arg\max_{1 \leq \theta \leq K} L(\theta)$. It is an idealized algorithm in the sense that it requires full knowledge of the probability densities in (2), which are never available in practice.

The ideal ML rule becomes the Bayes rule when q is a random query with the uniform distribution $P\{q \sim t_k\} = 1/K$. In general, if the class has a prior distribution $g(k) = P\{q \sim t_k\}$, then the ideal Bayes algorithm would rank the classes according to the posterior distribution $p\left(\theta|R^{(1)}, \ldots, R^{(J)}\right) \propto g(\theta)L(\theta)$. We observe that the ideal Bayes rule, R^*, generates optimal rankings for all monotone loss functions in terms of the rank of the true identity. In other words, if $P_g\{q \sim t_k\} = g(k)$ and $\xi(k)$ is any rank combination function of the outputs of the component classifiers, then $E_g h(R^*(\theta)) \leq E_g h(\xi(\theta))$ for all nondecreasing loss functions h, where E_g is the expectation under P_g and θ is the index of the true identity of the random query q, i.e. $q \sim t_\theta$.

3 MGR as an Exponential Probability Model

Implementing the ideal Bayes rules requires full knowledge of the probability vectors p_θ in (2) and the prior probabilities for all possible values of θ, the classes. In reality p_θ is unkown and has to be estimated. This poses a serious dimensionality problem as the estimation (training) data is usually of smaller order than the K^J dimensionality of the domain of p_θ. This suggests that directly applying non-parametric frequency estimation for the probability vectors p_θ might be ineffective, (we will return to this question in a later section on mixture models), whereas a parametric model would not suffer from this dimensionality curse. In this paper we develop a parametric probability model based on the MGR structure.

Remember that score functions are invarient to monotonic transformations (see equation (1)). In particular, applying a montonically increasing function to the output of MGR does not change how it ranks. The problem is to find a monotonic transformation of MGR functions which form a mathematically and computationally tractable family of parameteric distributions. A solution we develop is the following exponential family with MGR scores as basis functions:

$$p_\theta(x) = h(s(x)) = \frac{\exp(s(x))}{C} = \frac{1}{C}\exp\left(-\sum_{A\subseteq\{1\ldots J\}} w_A min\{x_j : j \in A\}\right) \quad (4)$$

where C is an appropriate normalization constant.

The advantage of using an exponential family are the well behaved properties of such distributions. The likelihood of a distribution function in the exponential family is convex and therefore has the property of having at most one maxima, where this maxima is the unique MLE [3]. Moreover estimates of the MLE have the property of being typically asymptotically efficient w.r.t the quantity of estimation data.

3.1 Calculating the Normalization Factor

In the case of combining 2 classifiers, surprisingly there is an analytical solution for the model parameters (which is beyond the scope of this paper). In the

general case estimation of the parameters of (4) requires a way of computing the constant, C. Having C gives an explicit formula for (4) and allows the parameters to be estimated using convex programming techniques [4] (cite). We provide here a derivation and procedure for calculating C.

Let $\mathcal{A}_J = \{A \,|A \subseteq \{1, \ldots J\}\,,\, A \neq \emptyset\}$ be the set of non-empty subsets, and let $\overrightarrow{\beta} = \{\beta_1, \beta_2, \beta_{12}, \beta_3, \beta_{13}, \ldots\} = \{\beta_A \,|\forall A \in \mathcal{A}_J\}$ be the set of all coefficients. The probability distribution (4) can be written as

$$f(x_1, \ldots, x_J) = \frac{\exp\left(-\sum_{A\in\mathcal{A}_J} \beta_A \min_{j\in A} x_j\right)}{T_J\left(\overrightarrow{\beta}\right)} \tag{5}$$

where

$$T_J\left(\overrightarrow{\beta}\right) = \sum_{y_1=1}^{\infty}\sum_{y_2=1}^{\infty}\cdots\sum_{y_J=1}^{\infty} \exp\left(-\sum_{A\in\mathcal{A}_J} \beta_A \min_{j\in A} y_j\right) \tag{6}$$

If we write $y_* = \min(y_1 \ldots y_J)$ then we can rewrite (6) as

$$T_J\left(\overrightarrow{\beta}\right) = \sum_{B\in\mathcal{A}_J} \sum_{\substack{y_1\cdots y_J \\ B = \{j\,|y_j = y_*\}}} \exp\left(-\sum_{A\in\mathcal{A}_J} \beta_A \min_{j\in A} y_j\right), \tag{7}$$

where the inner sum runs over all vectors of $(y_1 \cdots y_J)$ where the y_j , $j \in B$, are the minimal elements, i.e. $y_j = y_*$. Thus, the outer sum picks which $y's$ will be the minimal elements, and the inner sum goes over all values of the $y's$ where that holds.

Defining $\beta_* = \sum_{A\in\mathcal{A}_J} \beta_A = 1^T\overrightarrow{\beta}$ and using the notation $\overrightarrow{\beta}^C$ as the restriction of $\overrightarrow{\beta}$ on the subset C, the inner sum of (7) can be rewritten as

$$\begin{aligned}
&\sum_{\substack{y_1\cdots y_J \\ B = \{j\,|y_j = y_*\}}} \exp\left(-\sum_{A\in\mathcal{A}_J} \beta_A \min_{j\in A} y_j\right) \\
&= \sum_{\substack{y_1\cdots y_J \\ B = \{j\,|y_j = y_*\}}} \exp\left(-\beta_* y_* - \sum_{A\in\mathcal{A}_J} \beta_A \min_{j\in A}(y_j - y_*)\right) \\
&= \sum_{y_*=1}^{\infty} \sum_{\substack{y_j > y_* \\ j\in B^c}} \exp(-\beta_* y_*) \exp\left(-\sum_{\substack{A\in\mathcal{A}_J \\ A\subset B^c}} \beta_A \min_{j\in A}(y_j - y_*)\right)
\end{aligned}$$

Algorithm 1 A Dynamic-Programming algorithm for computing $T_j\left(\overrightarrow{\beta}\right)$

$T(B)$ corresponds to $T_{|B|}\left(\overrightarrow{\beta}^B\right)$
for $i = 1 \ldots J$ **do**
  for all
  $B \in \mathcal{A}_J$, $|B| = i$ **do**
    $T(B) \leftarrow 1$
    for all
    $C \subset B$, $C \neq \emptyset$ **do**
      $(*)$ $T(B) \leftarrow T(B) + T(C)$
    end for
    $T(B) \leftarrow \frac{e^{-\overrightarrow{\beta}^B_*}}{1-e^{-\overrightarrow{\beta}^B_*}} T(B) + T(C)$
  end for
end for

$$= \sum_{y_*=1}^{\infty} \exp(-\beta_* y_*) \sum_{\substack{y_j - y_* = 1 \\ j \in B^c}}^{\infty} \exp\left(- \sum_{\substack{A \in \mathcal{A}_J \\ A \subset B^c}} \beta_A \min_{j \in A} (y_j - y_*) \right)$$

$$= \sum_{y_*=1}^{\infty} \exp(-\beta_* y_*) T_{|B^c|}\left(\overrightarrow{\beta}^{B^c}\right) \tag{8}$$

Thus equation (7) can be written using the recursive relationship

$$T_J\left(\overrightarrow{\beta}\right) = \sum_{y_*=1}^{\infty} \exp(-\beta_* y_*) \sum_{B \in \mathcal{A}_J} T_{|B^c|}\left(\overrightarrow{\beta}^{B^c}\right)$$
$$= \frac{e^{-\beta_*}}{1 - e^{-\beta_*}} \sum_{B \in \mathcal{A}_J} T_{|B^c|}\left(\overrightarrow{\beta}^{B^c}\right)$$

with $T_0\left(\overrightarrow{\beta}^{\emptyset}\right) = 1$. This relationship says that $T_j\left(\overrightarrow{\beta}\right)$ can be calculated by recursively summing over all subsets. This type of structure lends itself to a dynamic programming type algorithm [5] that caches values of smaller subgroups.

The complexity of Algorithm 1 is dependent on the number of times that statement $(*)$ is executed. There are $\binom{J}{i}$ subsets of size i, each of which has $2^i - 2$ relevant subsets (not including itself and the empty set). Thus summing over subsets of different sizes we get

$$\sum_{i=2}^{J} \binom{J}{i} (2^i - 2) < (2+1)^J = 3^J$$

executions of statement $(*)$. Therefore calculating $T_j\left(\overrightarrow{\beta}\right)$ of the probability distribution (4) with n coefficients is $O\left(n^{1.59}\right)$, where $n = 2^J$ and $1.59 > \log(3)/\log(2)$.

3.2 Parameter Estimation

Given an efficient algorithm for calculating the MGR probability distribution function (5), the parameters of this function can be estimated using a maximum likelihood approach. The MLE solution is the set of parameters that maximize the log likelihood,

$$\log \prod_{i=1}^{n} \frac{\exp\left(-\sum_{A\in\mathcal{A}_J} \beta_A \min_{j\in A} x_{ji}\right)}{T_J\left(\overrightarrow{\beta}\right)} \tag{9}$$

where the product is over all sample data and subject to $\beta_A \geq 0$ for the monotnicity and quasiconvexity of the MGR function and $\beta_A > 0$ for $|A| = 1$ to maintain the feasibility of $T_J\left(\overrightarrow{\beta}\right)$.

This is a convex optimization problem, as the maximization criteria (9) is concave being the likelihood of an exponential family, and the constraints are linear. There are standard algorithms for optimizing nonlinear programming problems of this sort (e.g., primal-dual interior point algorithms [4]). In the experiments we present in this paper we used the MATLAB optimization toolbox for simplicity. Note that there are many alternative packages avaialble, see `http://www.ece.northwestern.edu/OTC/` for more information.

4 Mixture Models

A probability model for ranks offers a consistent mechanism for integration with other probability models. The advantage of using a parametric probability model, such as the probabilistic MGR is its ability to handle and generalize from relatively sparse data. However, it seems that data in combination datasets are typically not uniformally sparse. Since the classifiers that are combined already posses significant accuracy, we expect that many of the rank vectors for the correct class will contain ranks of 1. In effect, what we see in many datasets is that a small minority of rank vectors repeat while the majority do not. In table 1 we see this for the combination dataset generated by two different face recognition algorithms run on a face recognition dataset. This table shows for each rank vector (that appears more than once) the number of times it appears as the correct class in the combination training set. For the remaining rank vectors (not shown), 98 appear only once and the rest never appear as the correct class in the training set.

In presenting the MGR model we stated that the purpose of using a parametric model was to handle the datas' sparseness. While this is true in general, a

Table 1. The frequency of rank vectors in a combination training dataset that was generated applying the UMD and USC classifiers on the dup I dataset, shows a non-uniform distribution. The remaining 98 unlisted rank vectors appear only once in the dataset

Rank Vector	(1,1)	(1,2)	(2,1)	(3,1)	(1,3)	(2,2)	(1,6)
# appearances	*252*	*12*	*9*	*5*	*5*	*4*	*4*
Rank Vector	(1,4)	(4,1)	(1,5)	(17,1)	(33,1)	(1,14)	(1,18)
# appearances	*3*	*3*	*3*	*2*	*2*	*2*	*2*

small subset of rank vectors have denser coverage. This suggests that the probability of the rank vectors can be estimated more accurately by a nonparametric method, cell frequency estimation. Having the flexibility of a probabilistic interpretation of rank generation, allows us to combine these two different approaches. In particular, we can construct a mixture model that uses cell frequency estimation for the dense rank vectors while using an exponential MGR model for other rank vectors.

Let χ be the set of dense rank vectors. We define the mixture model as

$$p(x) = \begin{cases} P_\chi(x) & x \in \chi \\ \left(1 - \sum_{x\in\chi} P_\chi(x)\right) \dfrac{\exp\left(-\sum_{A\in\mathcal{A}_J} \beta_A \min_{j\in A} x_j\right)}{T_J\left(\overrightarrow{\beta}\right) - D} & x \notin \chi \end{cases} \tag{10}$$

where $D = \sum_{x\in\chi} \exp\left(-\sum_{A\in\mathcal{A}_J} \beta_A \min_{j\in A} x_j\right)$. This model is estimated as

$$\widehat{p}(x) = \begin{cases} \frac{\#\{x\in I\}}{|I|} & x \in \chi \\ \lambda \dfrac{\exp\left(-\sum_{A\in\mathcal{A}_J} \beta_A \min_{j\in A} x_j\right)}{T_J\left(\overrightarrow{\beta}\right) - D} & x \notin \chi \end{cases}$$

where $\lambda = 1 - |I|^{-1} \sum_{x\in\chi} \#\{x \in I\}$, $\#\{x \in I\}$ indicates the number of times that rank vector x appeared as the correct class in a combination training set I and $|I|$ is the number of correct class rank vectors in the combination training set. The β_A parameters are estimated as before, except that we have a different denominator term and we only estimate the parametric model using the rank vectors that are not in χ.

4.1 Mixing Frequencies

The mixture model (10) requires the preselection of the rank vectors that will be estimated using cell frequencies. A natural question is which vectors to use. A heuristic criteria is to select those rank vectors that have sufficient data to warrant direct estimation. It is beneficial to examine this question empirically. Figure 1 shows how a variation in the composition of the χ set affects test

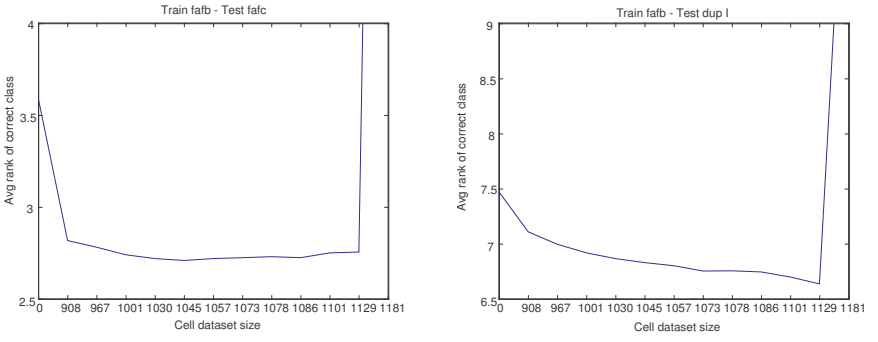

Fig. 1. These graphs show how varying the size of Cell datasets and thus controlling the proportions in the mixture model affects performance as emasured by the average rank of the correct class. They represent combination datasets generated using the ANM and USC classifiers on the the fafb dataset (described in section 5). The x-axis shows the number of rank vectors used in the cell frequency estimation. The first tick represents including the most frequent rank vectors for frequency estimation (1,1), the second tick represents including the first and second most frequent rank vectors and so on until all rank vectors are used for cell frequency estimation

performance. These graphs are typical of our experimental results. The x-axis represents different sizes of χ, the leftmost is a pure MGR exponential model, the rightmost represents a pure cell frequency model, while the intermediate positions are mixture models. For each mixture the average rank of the correct class (w.r.t. the test dataset) is shown, where ranks greater than 50 are truncated to diminish the noise effects of high ranks. As can be seen, estimating the denser rank vectors with cell frequencies improves performance almost consisitently. This implies that in selelcting the cutoff point for inclusion in χ we can allow sparse vectors to be included as long as the truely sparse rank vectors are estimated by probabilistic MGR.

5 Experimental Results

FERET [6] was a government sponsored program for the evaluation of face recognition algorithms. In this program commercial and academic algorithms were evaluated on their ability to differentiate between 1,196 individuals. The test consisted of different datasets of varying difficulty, for a total of 3,816 different images. The datasets in order of perceived difficulty are: the *fafb* dataset of 1,195 images which consists of pictures taken the same day with different facial expressions; the *fafc* dataset of 194 images that contains pictures taken with different cameras and lighting conditions; the *dup I* dataset of 488 images that has duplicate pictures taken within a year of the initial photo; and the most difficult, the *dup II* dataset of 234 images which contains duplicate pictures taken more than a year later. Note that in our experiments we separate the images of dup II

Table 2. Average rank in combining the USC, UMD and ANM face recognition algorithms with a cutoff at rank 50

test set	train set	Mix Model	Log MGR		test set	train set	Mix Model	Log MGR
dup i	dup ii	**8.58**	9.08		fafb	dup i	**1.21**	1.23
dup i	fafb	8.90	**8.44**		fafb	dup ii	**1.22**	1.26
dup i	fafc	**9.35**	9.37		fafb	fafc	1.18	**1.15**
dup ii	dup i	**10.59**	11.11		fafc	dup i	**2.31**	2.43
dup ii	fafb	**10.47**	10.92		fafc	dup ii	**1.86**	2.06
dup ii	fafc	**10.08**	10.68		fafc	fafb	2.12	**1.92**

from the dup I dataset, unlike the FERET study where dup II was also a subset of dup I.

The FERET study evaluated 10 baseline and proprietary face recognition algorithms. The baseline algorithms consisted of a correlation based method and a number of eigenfaces (Principle Components) methods that differ in the internal metric they use. Of the 10 algorithms we selected three dominant algorithms. From the baseline algorithms we choose to use the *ANM* algorithm which uses a Mahalanobis distance variation on angular distances for eigenfaces [7]. Within the class of baseline algorithms this algorithm was strong. Moreover, in accuracy w.r.t. average rank of the correct class on the dup I dataset it demonstrated superior performance to all other algorithms. The other two algorithms we used were the University of Maryland's 1997 test submission (UMD) and the University of Southern California's 1997 test submission (USC). These algorithms clearly outperformed the other algorithms. UMD is based on a discriminant analysis of eigenfaces [8], and USC is an elastic bunch graph matching approach [9].

The outputs of these 3 face recognizers on the four FERET datasets, fafb, fafc, dup I and dup II were the data for the experiments. Thus, we never had access to the actual classifiers, only to data on how they ranked the different faces in these datasets. In each experiment one of the FERET datasets was selected as a training set and another dataset was selected for testing. This gave 12 experiments (not including training and testing on the same dataset) per group of face recognizers, where we get combinations of training on easy datasets and testing on hard datasets, training on hard and testing on easy datasets, and training and testing on hard datasets.

We compare the mixture model with the original MGR, estimated using a logistic error function. In [1], we already demonstrated that the original MGR is superior to traditional score function approaches in overall performance. In this mixture model rank vectors with more than 2% of the training data were estimated by cell frequency, while the sparser rank vectors were captured by the MGR probability model. For each combiner the average rank of the correct class (across the test dataset) is shown, where ranks greater than 50 are truncated to diminish the noise effects of high ranks. As these results show the mixture model is comparable and at times superior to the logistic estimation of MGR.

6 Conclusion

We present a new probabilistic framework for for the combination of rank generating classifiers. In this framework rank vectors are interpreted as coming from a conditional probability distribution given the correct class. The MGR model is translated into this framework by casting it as an exponential distribution. We give an algorithm for efficiently calculating probabilities from this distribution and use it to calculate maximum likelihood estimates of its parameters. One advantage of having a probability model is that we can naturally combine it with a direct cell frequency estimation model. Thus, we form a mixture combination model where denser vectors are estimated using cell frequencies and sparser ones with the MGR model. While, we would not expect the MLE method to outperform the logistic estimation the MGR model in all instances, we do see that the mixture model offers comparable and at times superior performance on combination of FERET algorithms.

In conclusion, in this paper we show how a probability approach to rank combination, offers the advantages of interpretability, asymptotic estimation consistency of parameters and flexibility in how it is applied.

References

1. Melnik, O., Vardi, Y., Zhang, C.H.: Mixed group ranks: Preference and confidence in classifier combination. IEEE Pattern Analysis and Machine Intelligence**26** (2004) 973–981
2. Ho, T.K., Hull, J.J., Srihari, S.N.: Combination of decisions by multiple classifiers. In Baird, H.S., Bunke, H., (Eds.), K.Y., eds.: Structured Document Image Analysis. Springer-Verlag, Heidelberg (1992) 188–202
3. Bickel, P., Doksum, K.: Mathematical Statistics: Basic Ideas and Selected Topics. Prentice Hall, Englewood Cliffs, NJ (1977)
4. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)
5. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton, NJ (1957)
6. Phillips, P., Moon, H., Rizvi, S., Rauss, P.: The feret evaluation methodology for face-recognition algorithms. IEEE Trans. on Pattern Analysis and Machine Intelligence **22** (2000)
7. Moon, H., Phillips, P.: Computational and performance aspects of pca-based face-recognition algorithms. Perception **30** (2001) 303–321
8. Zhao, W., Krishnaswamy, A., Chellappa, R., Swets, D., Weng, J.: Discriminant Analysis of Principal Components. In: Face Recognition: From Theory to Applications. Springer-Verlag, Berling (1998) 73–86
9. Wiskott, L., Fellous, J.M., Kruger, N., von der Masburg, C.: Face recognition by elastic bunch graph matching. IEEE Transactions on Pattern Analysis and Machine Intelligence **17** (1997) 775–779