The WORST Algorithm

Weight Optimizing Random Subspace Technique with 2D-LDA

Garrett Bingham

University of North Carolina Wilmington

July 14, 2017

Table of contents

Random Subspace Method

Intro to RSM

In ensemble learning one tries to obtain a set of diverse classifiers. In bagging, for example, individual classifiers are trained on random subsets of the data so that they produce different models that can be later combined.

Instead of selecting a random subset of the training data, the random subspace method selects random features for each classifier. This means that highly predictive features will not dominate the decisions of individual classifiers.

Previous Work

Nam Nguyen, Wanquan Liu, and Svetha Venkatesh. *Random subspace two-dimensional pca for face recognition*. In *Advances in Multimedia Information Processing – PCM 2007: 8th Pacific Rim Conference on Multimedia, Hong Kong, China, December 11-14, 2007. Proceedings*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pages 655–664.

While RSM is generally used with decision trees, Nguyen et. al. have applied it to 2D-PCA, obtaining 93.5% accuracy on the ORL dataset. They chose the random subspace method because it allows a model to consider the information from all eigenvectors without overfitting on the training data.

My Contributions

- I perform face recognition on a subset of MORPH-II. The majority of papers published on the topic of 2D-PCA and its variants apply face recognition to ORL and Yale datasets. MORPH-II is larger, longitudinal, and suffers from more extreme variations in lighting and pose.
- Instead of using the eigenvectors from 2D-PCA, I use those from 2D-LDA. 2D-LDA has been shown to have more discriminative power in face recognition than 2D-PCA.
- I utilize entropy measure in order to seek parameters that result in objectively diverse classifiers.
- I develop a weighting scheme to increase the overall accuracy of the model in the final decision step.
- I consider multiple distance metrics and evaluate their performance.

Experiments

MORPH-II Dataset

Experiments on "easy" datasets like ORL are not very informative because most classifiers are able to get well over 90% accuracy without significant tuning. This makes it difficult to judge which classifiers are the most effective.

Because of this, I evaluate my algorithm on MORPH-II, a much more difficult dataset. All images have been rotated so that the eyes are level. Each image was converted to grayscale and cropped to 60×70 pixels to reduce the effect of hair and the background. (Thanks Ben and Rachel!)

MORPH-II is longitudinal and suffers from high variability in illumination. To help adjust for this, all images are histogram equalized with a built-in python function.

Experiment Design

I choose 50 people from MORPH-II. 5 images per person are selected at random for training and 5 for testing. I compute the eigenvectors for 2D-PCA and 2D-LDA, and discard those that explain less than 1% of the variability (or discriminability in the case of 2D-LDA) because they are likely just random noise.

Of the remaining eigenvectors, I choose 10 at random to build a classifier. I build 50 such classifiers, estimate their performance with a clustering similarity measure, and then use a weighted voting scheme to combine their results into a final decision. Each classifier uses KNN ($k = 5$) to make its individual decision. Different distance metrics are considered.

I consider the left, right, and bilateral versions of 2D-PCA and 2D-LDA. If not specified, I am using Bilateral 2D-LDA.

Entropy Measure

If our set of classifiers is diverse, each classifier will make different mistakes. That way, we have a better chance of being able to correct those mistakes when we combine results into the final decision.

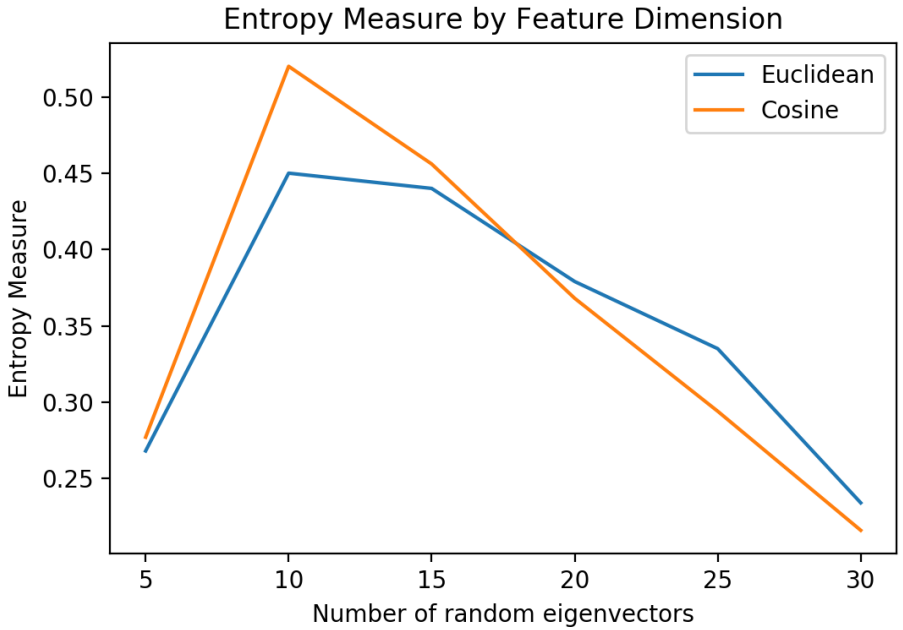
Entropy Measure assumes this is optimal when half of the classifiers are correct for a given test image. Define ζ_i as the number of classifiers out of T that misclassifies the ith image. Then entropy is defined as

$$E = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{T - \lceil T/2 \rceil} \min\{\zeta_i, T - \zeta_i\} \tag{1}$$

where $E \in [0, 1]$. Low values indicate similar classifiers, and high values indicate diverse classifiers.

Entropy Experiments

Figure: Choosing 10 eigenvectors results in high entropy in both scenarios



If we choose too few eigenvectors, each classifier is too weak and makes a large number of errors, resulting in low entropy. On the other hand, if we choose too many eigenvectors, then the classifiers will make near identical predictions, again resulting in low entropy.

Entropy Experiments

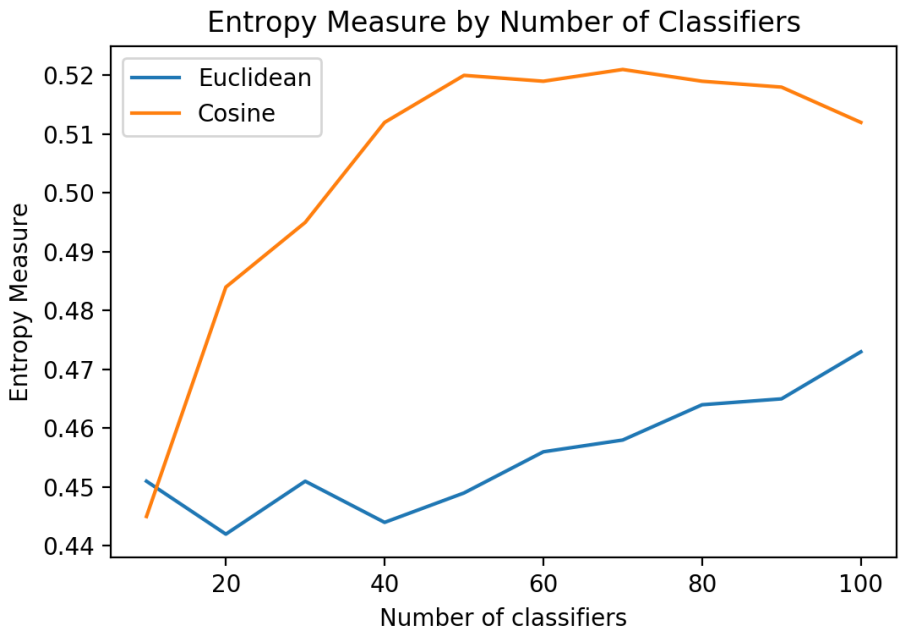


Figure: Entropy levels off

Not training enough classifiers results in low entropy, because we haven't sampled the eigenvectors thoroughly.

Changing these parameters affects the entropy, but the accuracy remains more or less stable. This implies that although we aren't affecting the the unweighted accuracy, we may be able to increase overall accuracy with a weighted voting scheme.

Majority Voting

In majority voting, each classifier gets one vote. We select the class with the most votes, regardless of whether the percentage of votes is above 50%.

Define the decision of the tth classifier as $d_{t,j} \in \{0, 1\}$, where $t = 1, \ldots, T$, and $j = 1, \ldots, C$. T is the number of classifiers and C the number of classes. If the tth classifier chooses class J, then $d_{t,J} = 1$, and otherwise 0. We choose class J if

$$\sum_{t=1}^{T} d_{t,J} = \max_{j=1}^{C} \sum_{t=1}^{T} d_{t,j} \tag{2}$$

Weighted Majority Voting

If we know that some classifiers are more accurate than others, then we can weight their decisions so that more credible classifiers have a higher influence on the final decision. Assign weight w_t to the tth classifier. We choose class J if

$$\sum_{t=1}^{T} w_t d_{t,J} = \max_{j=1}^{C} \sum_{t=1}^{T} w_t d_{t,j} \tag{3}$$

Evaluating the Classifiers

Each classifier is a random sample of eigenvectors that projects all training and testing images to a new space. To get an idea of how well a classifier will perform, we can evaluate its performance on the training data. We will expect a classifier to perform well if projected images of the same person are tightly clustered together, while images of different people are well-separated.

Let the ground truth values of all training images be one clustering. For a given classifier, let its predictions on the training images be another clustering. Then we can use a clustering similarity measure to get an idea of how well a classifier will perform on testing data.

Rand Index

Given a set of n images A, and two clusterings of A to compare, T the truth values for the images, and P the predicted values for the images, define the following:

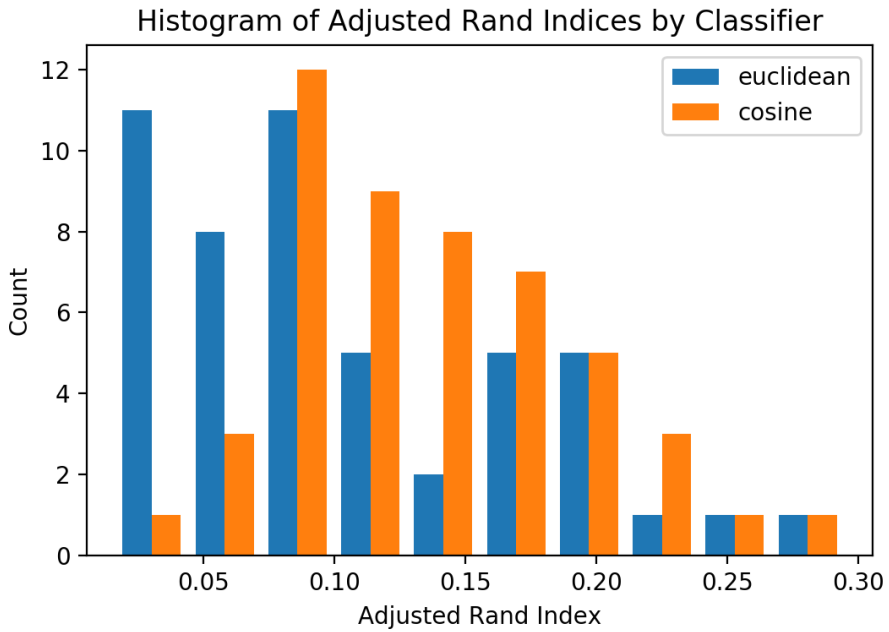
- a: the number of pairs of images of A that are in the same cluster in T and the same cluster in P
- b: the number of pairs of images of A that are in different clusters in T and different clusters in P
- c: the number of pairs of images of A that are in the same cluster in T and in different clusters in P
- d: the number of pairs of images of A that are in different clusters in T and the same cluster in P

The Rand Index, R, is

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}} \tag{4}$$

Adjusted Rand Index

While the Rand index ranges from 0 to 1, the adjusted Rand index (ARI) is a corrected-for-chance version and ranges from -1 to 1. Higher values indicate two clusterings that are very similar, whereas negative values indicate two dissimilar clusterings.



We want to give classifiers that have a higher ARI more weight in the final decision.

It appears that using the cosine metric results in a higher ARI than euclidean.

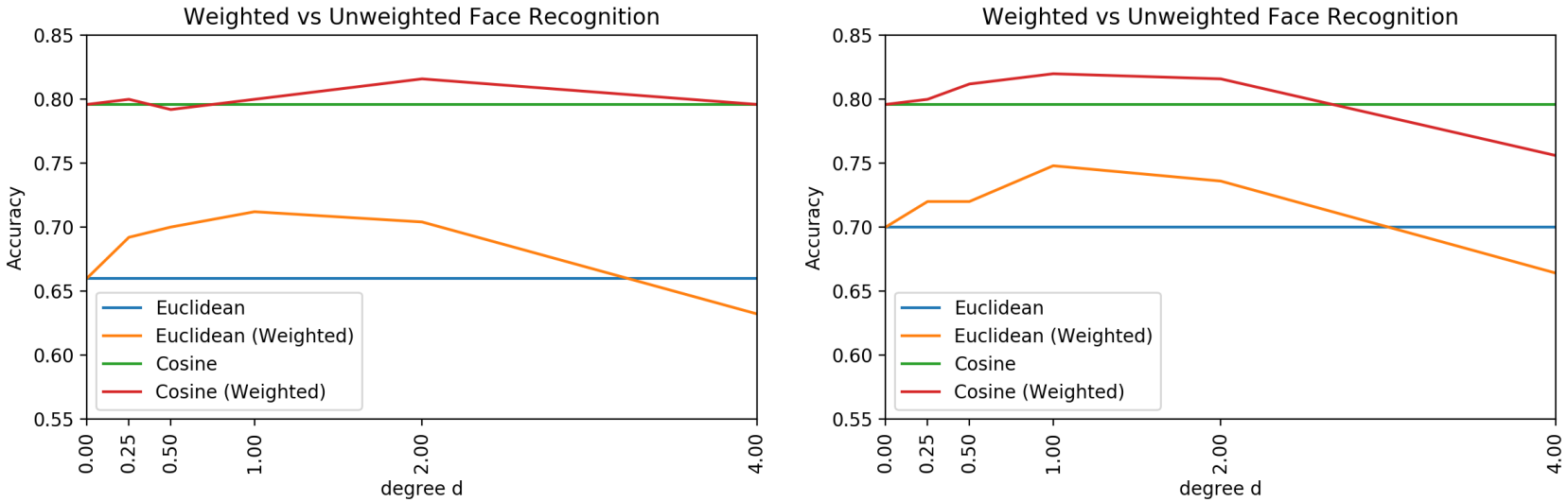
Weighting the Weights

To change how much more influence the classifiers with a high ARI have over the ones with a low ARI, we can try mapping the weights with a monotone function. Raising all weights to a common exponent d might do the trick, although other functions such as the logistic function could also be considered.

We expect that when d is low, the "strong" classifiers won't have much influence over the "weak" ones, and the accuracy won't increase by much. If d is too high, only the decision of a select few "strong" classifiers will matter, and their errors won't be corrected by the other classifiers.

Weights^d

Figure: The previous hypothesis is confirmed using two different random seeds



Choosing a moderate value for d gives an extra 2-5% in overall accuracy. The optimal value for d may be different in other scenarios (e.g. with datasets of different size and difficulty).

Conclusion

Data

Each of these experiments was run three times and the resulting accuracies averaged to obtain the following table.

	euclidean			cosine		
Algorithm	W	UW	O	W	UW	O
L2DLDA	.772	.767	.776	.808	.803	.784
R2DLDA	.734	.713	.740	.776	.804	.744
B2DLDA	.747	.695	.804	.792	.800	.776
L2DPCA	.641	.633	.568	.633	.628	.572
R2DPCA	.668	.668*	.656	.668	.668	.652
B2DPCA	.644	.640	.588	.649	.649	.588

Key: W: weighted, UW: unweighted, O: original algorithm (not RSM)

*The Nguyen et. al. framework does not perform well on MORPH-II.

Insights

- Cosine seems to consistently outperform euclidean. This could be a result of the high illumination variability in MORPH-II. Cosine is a measure of similarity, and disregards magnitude.
- The LDA-based algorithms achieve higher accuracies than their PCA counterparts. This was expected, and is due to the fact that the eigenvectors from LDA are based on maximizing discriminability instead of just explaining variability.
- The weighting scheme was based off of B2DLDA accuracies. Clearly it did not generalize well to some other algorithms.
- My contributions to the model proposed by Nguyen et. al. resulted in substantial increases in accuracy. I can achieve 97% accuracy on the ORL dataset, outperforming their 93.5%.

Future Work

- Further investigation into the differences of the left, right, and bilateral versions of 2D-PCA and 2D-LDA.
- Evaluating the model's performance on larger subsets of MORPH-II.
- Developing a more systematic and robust weighting scheme that can be applied regardless of algorithm or dataset.
- Exploring randomly sampling eigenvectors with replacement.

References I


Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, August 1998. ISSN: 0162-8828.


Nam Nguyen, Wanquan Liu, and Svetha Venkatesh. *Random subspace two-dimensional pca for face recognition*. In *Advances in Multimedia Information Processing – PCM 2007: 8th Pacific Rim Conference on Multimedia, Hong Kong, China, December 11-14, 2007. Proceedings*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pages 655–664.


S. Noushath, G. Hemantha Kumar, and P. Shivakumara. (2d)2lda: an efficient approach for face recognition. *Pattern Recognition*, 39(7):1396–1400, 2006. ISSN: 0031-3203.


R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006. ISSN: 1531-636X.

References II

 K. Ricanek and T. Tesafaye. Morph: a longitudinal image database of normal adult age-progression. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 341–345, April 2006. DOI: 10.1109/FGR.2006.78.

 Anbang Xu, Xin Jin, Yugang Jiang, and Ping Guo. Complete two-dimensional pca for face recognition. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 481–484, 2006.

 Jian Yang, D. Zhang, A. F. Frangi, and Jing-yu Yang. Two-dimensional pca: a new approach to appearance-based face representation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):131–137, January 2004. ISSN: 0162-8828.