95-828 Machine Learning for Problem Solving: Homework 2

H. John Heinz III College
Carnegie Mellon University

- There are six questions on this assignment. Three *conceptual* (short answers) and three *applied* (implementation) that involve coding. For the applied questions, you may write your code either in an R Markdown file with a .rmd extension or in an IPython notebook with a .ipynb extension. Include only final results in your write-up and turn in a hard copy in class. Specifically, do not print your codes in your write-up *unless otherwise stated* in a question. Turn in your codes electronically via Canvas. If you have multiple files to upload, please zip all your files, and name your .zip file with your andrew ID, e.g. amaurya.zip
- The questions involve the following ML concepts:
  - Logistic Regression
  - Decision Trees
  - Principal Component Analysis (PCA)
  - Model Evaluation and Model Selection
- Deliverables for R users: Submit a zip file to Canvas, which includes 1) a R markdown file and 2) a knitted PDF file, and submit a hard copy of the PDF file in class. In your submitted PDF file and the hard copy, do NOT include your codes unless otherwise stated in a question. To suppress code while creating the PDF submission, you can use `echo=False` in each code block in R.
- Deliverables for Python users: Submit a zip file to Canvas, which includes 1) an IPython notebook file and 2) an exported HTML file (File ⇒ Download as ⇒ HTML), and submit a printout of the HTML file in class. In your submitted HTML file and the hard copy, do NOT include your codes unless otherwise stated in a question. To suppress code while creating the report for submission, you can copy the code available at this link in a cell in your IPython notebook and execute it.
- The assignment (both hardcopy and online submission) is due at **4:30 PM** on **Mar 12, 2017**. Since this is no class on this day, please deposit your homework in a collection box placed at HBH 3034 (PhD TA room). The box will be available from March 11 4:30pm until March 12 4:30pm. If you are taking late day(s), please submit your homework zip file on Canvas to mark the time of submission and then submit the hard copy next time in class. Please note down the number of late days you used as well as the total number of late days remaining on top of the first page of your submission.
- Do not forget to put both your name and andrew ID on *every* page of your submission.
- If you have any questions about the HW, please use Piazza or come to office hours and recitations.
- You may *discuss* the questions with fellow students, *however* you must always write your own solutions and must acknowledge whom you discussed the question with. Exchange of information and ideas can only be from brain-to-brain and not from paper-to-paper or computer-to-computer. Do not copy from other sources, share your work with others, or search for solutions on the web. Plagiarism will be penalized according to university rules.

1 Conceptual: Logistic Regression [10 points]

In logistic regression, our goal is to learn a set of parameters by maximizing the conditional log likelihood of the data. Assume you are given a dataset with n training examples and p features.

1. **[2 points]** The logistic regression probability is given by

$$Pr(y = 1|\mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

Show that $Pr(y = 1|x; \mathbf{w}) = Pr(y = 0|x; -\mathbf{w})$

2. **[2 points]** Your friend has come up with a "generalization" of logistic regression in which the probabilities of the two classes are specified by two separate weight vectors i.e.

$$Pr(y = 1|\mathbf{x}; \mathbf{w}_0, \mathbf{w}_1) = \frac{e^{\mathbf{w}_1^T \mathbf{x}}}{e^{\mathbf{w}_0^T \mathbf{x}} + e^{\mathbf{w}_1^T \mathbf{x}}}$$

and

$$Pr(y = 0|\mathbf{x}; \mathbf{w}_0, \mathbf{w}_1) = \frac{e^{\mathbf{w}_0^T \mathbf{x}}}{e^{\mathbf{w}_0^T \mathbf{x}} + e^{\mathbf{w}_1^T \mathbf{x}}}$$

Show that this is just logistic regression with

$$Pr(y = 1|\mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

where $\mathbf{w} = \mathbf{w}_0 - \mathbf{w}_1$

3. **[2 points]** Write down a formula for the conditional log likelihood of the training data in terms of the the class labels $y^{(i)}$, the features $x_1^{(i)}, \ldots, x_p^{(i)}$, and the parameters $w_0, w_1, \ldots, w_p$, where the superscript (i) denotes the sample index. Let us denote this formula with f. This will be your objective function for gradient ascent.

4. **[2 points]** Compute the partial derivative of the objective function f with respect to w_0 and with respect to an arbitrary w_j, i.e. derive $\partial f / \partial w_0$ and $\partial f / \partial w_j$, where f is the objective that you provided above.

5. **[2 points]** Consider a one-dimensional logistic regression. The input feature is $X \in \mathbb{R}$, and the binary output is $Y \in \{0, 1\}$, and the coefficient corresponding to the single feature is w. Show that the MLE estimate of w is $+\infty$ or $-\infty$ if the datapoints are linearly separable and the logistic regression likelihood is *not* regularized with any shrinkage priors such as l_1 or l_2 penalty.

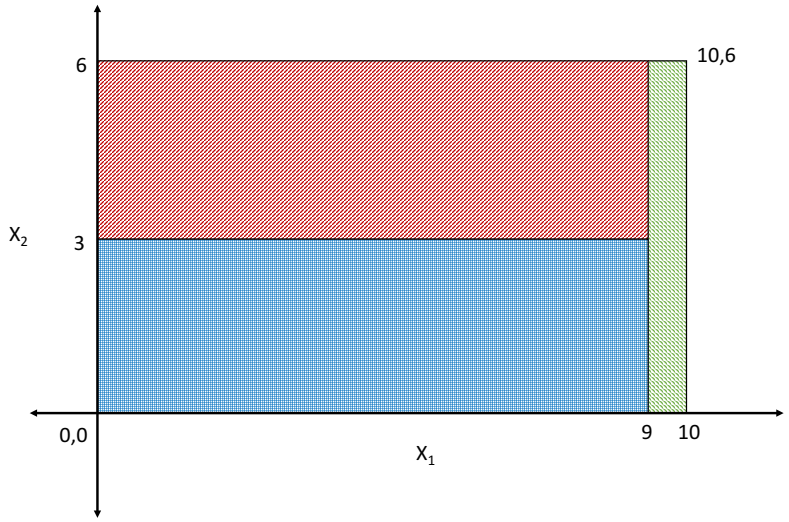
2 Conceptual: Decision Trees [25 points]

1. For each statement, answer if it is true or false. For the ones you answer 'False', briefly explain why it is false.

   (a) **[1 point]** Decision trees can obtain 100% train accuracy on any dataset.

   (b) **[1 point]** Each feature can be used for a node split in a decision tree only once.

2. A decision tree learned on a dataset is *sub-optimal* if another decision tree with lower complexity can provide the same accuracy on the dataset. The complexity of a decision tree is the total number of feature splits in the tree. Note that since decision tree construction is a greedy procedure, it is possible to learn decision trees with suboptimal complexity i.e. more nodes/splits than required to achieve the same accuracy. Consider the following diagram which shows two numerical features X_1 and X_2:



Datapoints belong to one of three classes. The regions belonging to the three classes are indicated by 3 different colors (*R*ed, *G*reen, *B*lue) in the above diagram. Consider that a million datapoints are sampled uniformly from the domain of the features i.e. the rectangle from (0,0) to (10,6) to create a dataset. The label of each datapoint is from $\{R, G, B\}$ depending on which of the three colored rectangles it falls into.

(a) **[2 points]** Draw the decision tree that would be learned using greedy node splits based on criteria such as Information Gain. How many total node splits does it have?

(b) **[2 points]** Draw a decision tree with minimum number of node splits by manually examining the dataset above. How many total node splits does it have?

(c) **[1 points]** Is the first tree suboptimal compared to the second tree? Why (not)?

3. Your friend has implemented a decision tree construction algorithm, and asked you to review it. In the implementation, for each possible node split, she calculates the Information Gain for all possible features to split on, and splits on the feature that provides the maximum Information Gain provided that the resulting Information Gain is positive.

   (a) **[2 points]** Provide a simple test dataset which demonstrates an algorithmic flaw in your friend's algorithm design logic.

   (b) **[2 points]** What is the flaw? (*Hint:* Think about the termination condition.)

   (c) **[2 points]** What is the correction for the flaw?

4. Suppose we want to evaluate a node split on a feature that takes on one of V values during the process of creating a binary decision tree.

   (a) **[1 points]** How many possible ways of splitting the node do you need to evaluate to find the optimal binary split if the feature is categorical?

   (b) **[1 points]** How does the answer change if the feature is ordinal?

   (c) **[1 points]** How about if the feature is numerical i.e. takes on real values from $\mathbb{R}$? Note that

the numerical feature can take on infinite possible values, but takes on V unique values in the training data.

5. **[2 points]** Describe how to use the Chi-Square test to prune a tree. You should specify: 1) what does the chi-square test quantify? 2) what is the hyper-parameter and its meaning? and 3) what is the pruning process?

6. **[1 point]** If you let a decision tree grow until it's impossible to split (all records in data subset have the same output or the same set of input attributes) and use it as the final prediction model without any pruning, what could be a potential problem of the resulting model? Explain briefly.

7. Consider the problem of predicting whether the university will be closed on a particular day. We will assume that the factors which determine if the school is 'Closed' are whether there is a Snowstorm, whether it is an official Holiday and whether or not it is the Weekend. Suppose we have the training examples as shown in Table 1 below. Answer the following two questions.

Snowstorm	Holiday	Weekend	Closed
T	T	F	F
T	T	F	T
F	T	F	F
T	T	F	F
F	F	F	F
F	F	F	T
T	F	F	T
F	F	F	T

Table 1: Training examples for decision tree

   (a) **[2 points]** What would be the effect of the Weekend attribute on the decision tree if it were made the root? Explain in terms of Information Gain.

   (b) **[2 points]** If we cannot make Weekend the root node, which attribute should be made the root node of the decision tree? Explain your reasoning and show your calculations. (You may use $\log_2\ 0.75 = -0.4$ and $\log_2\ 0.25 = -2$)

   (c) **[2 points]** Draw the complete Decision Tree using Information Gain as the node splitting criterion.

3 Conceptual: Model Evaluation and Model Selection [10 points]

Consider a cross-validation setup with a train set `Train` and a test set `Test`. For finding a hyperparameter using cross-validation, we split `Train` into K folds. We then iterate through the folds and possible values of the hyperparameter h. For the i^{th} fold and h_j i.e. j^{th} value of the hyperparameter, we train on the remaining $K-1$ folds and calculate the accuracy on the i^{th} fold. Let $M(i, h_j)$ be the calculated accuracy.

1. For each statement, answer if it is true or false. For the ones you answer 'False', briefly explain why it is false.

   (a) **[0.5 point]** The best value of the hyperparameter is the average of best hyperparameter values from each fold i.e. $\frac{1}{K}\sum_i \text{argmax}_{h_j}\, M(i, h_j)$.

   (b) **[0.5 point]** The best value of the hyperparameter is the one that maximizes the evaluation metric averaged across all folds i.e. $\text{argmax}_{h_j}\, \frac{1}{K}\sum_i M(i, h_j)$

(c) **[1 point]** If there are no hyperparameters to tune, the train accuracy is a reliable estimate of the test accuracy.

(d) **[1 point]** If there are no hyperparameters to tune, the cross-validation accuracy is a reliable estimate of the test accuracy.

(e) **[1 point]** If we tune a hyperparameter using cross-validation to maximize accuracy, the cross-validation accuracy is typically lower than train accuracy.

(f) **[1 point]** If we tune a hyperparameter using cross-validation to maximize accuracy, the cross-validation accuracy is typically lower than test accuracy.

(g) **[1 point]** Numerical hyperparameters i.e. hyperparameters which can take on values from the infinite set $\mathbb{R}$ *cannot* be optimized using cross-validation.

2. **[2 points]** Describe briefly in 1-2 sentences any changes you will make to the above described cross-validation procedure if you wish to maximize a metric other than accuracy; such as F1-score, Area under ROC, etc.

3. **[2 points]** Your friend is building an image classification system using decision trees. She performed cross-validation to choose the maximum allowed depth D of her tree from $\{1, 2, ..., 50\}$. She then plotted the train, cross-validation, and test error against $D = \{1, 2, ..., 50\}$, and saw that the test error hadn't plateaued and continued to decrease even around $D = 50$. Realizing that trying out hyperparameter values $D > 50$ might help reduce the prediction error even further, she repeated the experiments but chose D from $\{1, 2, ..., 100\}$, plotting the three types of errors (train, cross-validation, test) against D again. She then chose $D^* = 78$ as the optimal hyperparameter value that minimizes the cross-validation error, and reported $TestError(D^*)$ as the performance of the decision tree algorithm. Is this procedure correct for reporting test error of decision tree on the machine learning task? If not, mention and explain any precautions/corrections to the procedure your friend should have taken or been aware of during their experimentation.

4 Applied: Principal Components Analysis [15 points]

In this question, we explore PCA on two datasets – a dataset of human faces and a dataset of Bag-of-Words (BoW) features from a text corpus.

4.1 Eigenfaces

You are provided a dataset of human faces in a compressed file `CroppedYale.zip` alongwith this assignment. Extracting the zipped file should create a folder named `CroppedYale` with multiple folders named `yaleB<X><Y>` within it. Each of these folders is associated with a single person and contains multiple pictures of that person. There are 38 unique human faces with upto 64 images per person in the dataset.

Load the dataset in the language of your choice in grayscale format, convert each image into a vector in row-major format, and stack up these vectors horizontally into a matrix $\mathbf{D}$. If the width and height of each image are W and H respectively and the total number of images is N, the size of $\mathbf{D}$ is $N \times (W \cdot H)$. The label for each datapoint is the person's identity, stored into an N-dimensional vector $\mathbf{y}$ where $y_i \in \{1, .., 38\}$.

1. **[2 points]** Find the "mean" face. Average the rows of $\mathbf{D}$, reshape the resulting $W \cdot H$ vector `mean` into a W-width, H-height image, and display this mean image.

2. **[2 points]** Demean the data matrix i.e. subtract `mean` from each row to ensure that the average value of any column of $\mathbf{D}$ is 0. Run PCA on the transformed data matrix $\mathbf{D}$. Provide the screeplot i.e. a plot

of the proportion of variance explained versus the principal component number. Do you see an elbow in the scree plot which can help you as a heuristic on choosing the number of principal components to retain for dimensionality reduction?

3. **[3 points]** Plot the first 10 principal components, again by reshaping each $W \cdot H$-dimensional principal component into a W-width, H-height image. These principal components are called *eigenfaces* [1]. Let's call $\mathbf{E}$ as the matrix which stores the principal components as rows.

4. **[3 points]** Project the datapoints onto the principal components by matrix multiplication $\mathbf{P} = \mathbf{D} \cdot \mathbf{E}^T$. Notice that projections of a datapoint onto the principal components are obtained by taking the dot product of the datapoint with each of the principal components. The matrix multiplication just expresses this in a compact and computationally efficient format. The i^{th} row of $\mathbf{P}$ consists of principal component projections for the i^{th} datapoint i.e. principal component scores that can be used to reconstruct the i^{th} datapoint using the principal components in $\mathbf{E}$. The j^{th} column of $\mathbf{P}$ captures the projection of datapoints onto the j^{th} principal component. Plot separate histograms for each of the first 10 PCA projections i.e. the first 10 columns of $\mathbf{P}$.

4.2 PCA on Text Corpus

In NLP, words such as "the", "an", "of", etc. that occur commonly in any English document are called *stopwords*. Because such words occur throughout a text dataset (often called a corpus), they typically have high corpus-wide counts. As a result, their corresponding BoW features are often dominant in the first principal component of the dataset because they often explain a lot of the variance in the data. In this question, we will discover stopwords from the top principal component of a dataset of text articles, each coming from either the serious European magazine *The Economist*, or from the not-so-serious American magazine *The Onion*. The data is provided to you as `'onion_vs_economist.zip'` on Canvas, containing five files which can be loaded using appropriate commands in your programming language to yield the following variables:

- `Vocabulary` is a V dimensional list that that contains every word appearing in the documents. When we refer to the j^{th} word, we mean `Vocabulary(j)`.
- `XTrain` is a $n \times V$ dimensional matrix describing the n documents used for training your Naive Bayes classifier. The entry `XTrain(i,j)` is 1 if word j appears in the i^{th} training document and 0 otherwise.
- `yTrain` is a $n \times 1$ dimensional matrix containing the class labels for the training documents. `yTrain(i,1)` is 0 if the i^{th} document belongs to The Economist and 1 if it belongs to The Onion.
- Finally, `XTest` and `yTest` are the same as `XTrain` and `yTrain`, except instead of having n rows, they have m rows. This is the data you will test your classifier on and it should not be used for training.

We will use only the variables `XTrain` and `Vocabulary` in this question. However, all of the above variables from this dataset will also be used in a later section.

1. **[4 points]** Find the top principal component of `XTrain`. Note that the length of the principal compnent is equal to the number of words in `Vocabulary`. Provide a list of the top-30 words from `Vocabulary` sorted in decreasing order of the *absolute* value of their coefficient in the top principal component.

2. **[1 points]** How many of the top-30 words in the list are stopwords i.e. occur in the `stopwords.txt` file provided with the data?

5 Applied: Logistic Regression [18 points]

In this question you will implement a Logistic Regression classifier for a text classification problem. You will be given a collection of text articles, each coming from either the serious European magazine *The Economist*, or from the not-so-serious American magazine *The Onion*. The goal is to learn a classifier that can distinguish between articles from each magazine.

Download the data `'onion_vs_economist.zip'` from Canvas (same dataset that you used earlier for PCA on a text corpus). We have pre-processed the articles so that they are easier to use in your experiments. We extracted the set of all words that occur in any of the articles, known as the vocabulary. The dataset is provided in the form of five files which can be loaded using appropriate commands in your programming language to yield the following variables:

- `Vocabulary` is a V dimensional list that that contains every word appearing in the documents. When we refer to the j^{th} word, we mean `Vocabulary(j)`.
- `XTrain` is a $n \times V$ dimensional matrix describing the n documents used for training your Naive Bayes classifier. The entry `XTrain(i,j)` is 1 if word j appears in the i^{th} training document and 0 otherwise.
- `yTrain` is a $n \times 1$ dimensional matrix containing the class labels for the training documents. `yTrain(i,1)` is 0 if the i^{th} document belongs to The Economist and 1 if it belongs to The Onion.
- Finally, `XTest` and `yTest` are the same as `XTrain` and `yTrain`, except instead of having n rows, they have m rows. This is the data you will test your classifier on and it should not be used for training.

You may want to first take a look at the data by summarizing it (i.e. through summary statistics). You are free to use any packages and their built-in routines in Python/R for this question. Provide answers to the following questions in your submitted report:

1. **[2 points]** Fit a logistic regression classifier to the provided data (with no regularization). State the test misclassification rate. Also, produce a confusion matrix for the classifier.
2. **[2 points]** Plot the precision-recall curve and the ROC curve for the model using test predictions and groundtruth.
3. **[3 points]** Plot a wordcloud where the font size for each word is the magnitude of its logistic regression coefficient. Show words with a positive coefficient in green, and words with a negative coefficient in red. Include the wordcloud in your report. What differences do you see between prominent green and red words?
4. **[3 points]** In this question, we will use l_1 regularized logistic regression. Use cross-validation on the training data to tune the regularization hyperparameter of l_1 regularization. Plot the regularization hyperparameter λ versus the cross-validation error. What is the optimal hyperparameter λ^*_{cv} chosen through CV?
5. **[3 points]** Train an l_1 regularized logistic regression model on all of training data using the optimal hyperparameter λ^*_{cv} chosen through CV. Use the resulting model to predict on the test set. State the test misclassification rate. Is the test error lower than that for the unregularized model? Also, produce a confusion matrix for the classifier.
6. **[2 points]** Plot the precision-recall curve and the ROC curve for the regularized model using test predictions and groundtruth.
7. **[3 points]** Plot a wordcloud where the font size for each word is the magnitude of its logistic regression coefficient in the regularized model. Show words with a positive coefficient in green, and words with a negative coefficient in red. Include the wordcloud in your report. What differences do you see between this wordcloud and the one from the unregularized model?

Note: There are good packages both in Python and R which you can use to produce custom wordclouds for the questions that need them.

6 Applied: Decision Trees [22 points]

In this question, we will use a combination of PCA-based dimensionality reduction and decision trees to build a face classification system. We are now ready to perform face classification using the original features $\mathbf{D}$ as well as the features obtained from PCA-based dimensionality reduction $\mathbf{P}$ in an earlier question. The i^{th} row in both matrices corresponds to the i^{th} datapoint. The label for each datapoint is the person's identity, stored into an N-dimensional vector $\mathbf{y}$ where $y_i \in \{1, .., 38\}$.

Split the data ($\mathbf{D}$, $\mathbf{P}$, and $\mathbf{y}$) into 80% training data and 20% test data. Ensure that the same datapoints fall into train/test across $\mathbf{D}$, $\mathbf{P}$, and $\mathbf{y}$.

1. **[2 points]** Fit a decision tree model using the training data $\mathbf{D}$ and $\mathbf{y}$, and report the number of terminal nodes (leaves) and the depth of the tree. You should let the tree grow until it's impossible to split (all records in data subset have the same output or the same set of input attributes). Use the model to predict $\hat{\mathbf{y}}$ on test set, and evaluate your predictions against the true labels $\mathbf{y}$. State the test error.

   Note: You may use packages like `tree` or `rpart` in R, but please note that those packages may have default stopping criteria different from our requirement, and you need to disable them.

2. **[3 points]** Now you decide to prune the tree, because it is a bit too large. Fit a pruned tree where the maximum depth hyperparameter is chosen using 10-fold cross-validation (CV). Evaluate the pruned tree on test data. State the test error. Did the pruning help improve test error compared to the previous question? Plot the maximum depth hyperparameter versus the cross-validation error.

3. **[2 points]** Fit a decision tree model using the principal component training data $\mathbf{P}$ and $\mathbf{y}$, and report the number of terminal nodes (leaves) and the depth of the tree. You should let the tree grow until it's impossible to split (all records in data subset have the same output or the same set of input attributes). Use the model to predict $\hat{\mathbf{y}}$ on test set, and evaluate your predictions against the true labels $\mathbf{y}$. State the test error.

4. **[3 points]** Again fit a pruned tree where the maximum depth hyperparameter is chosen using 10-fold cross-validation (CV). Evaluate the pruned tree on test data. State the test error. Did the pruning help improve test error compared to the previous question? Plot the maximum depth hyperparameter versus the cross-validation error.

5. **[4 points]** Now instead of tuning just the maximum tree depth, perform 2D cross-validation on the training data $\mathbf{P}$ and $\mathbf{y}$ to tune two hyperparameters – (a) the maximum tree depth in decision tree training, and (b) the number of top principal components used as features. Plot a heatmap where X-axis is the maximum tree depth hyperparameter, Y-axis is the number of top principal component features used, and the heat value is the cross-validation error. What is the combination of hyperparameters that minimzes cross-validation error?

   Note: The principal component projections of datapoints stored in $\mathbf{P}$ are ordered by explained variance. To use the top-k principal components as features, pick the first k columns of $\mathbf{P}$ and use it as the features.

6. **[3 points]** Using the optimal combination of hyperparameters you have found, train a model using the entire training set. Use the model to predict $\hat{\mathbf{y}}$ on test set, and evaluate your predictions against the true labels $\mathbf{y}$. State the test error. Did the 2D hyperparameter search help improve test error compared to the 1D hyperparameter search for maximum tree depth?

7. **[5 points]** In small datasets such as the faces dataset here, it is preferable to be able to use all datapoints to calculate the test error to get a better estimate. This is done through nested cross-validation: Split the entire dataset into K folds, where $K-1$ folds are used for cross-validated training (explained in next paragraph) and the remaining fold is used as the test set. When we iterate through all the folds, we obtain a test error estimate for each fold. These are then used to compute a weighted average where each test error statistic is weighted by the number of test datapoints used to calculate it. This average is computed using all datapoints and is a much better (lower variance) estimate of true test error.

   Cross-validated training involves further splitting the training data into K' folds, using $K'-1$ folds for training and the remaining fold for getting validation error. Iterating through the folds provides K' estimates of validation error which can be combined to calculate the cross-validation error. This cross-validation error can be used for hyperparameter selection using data from all the K' folds. Once the optimal hyperparameters are selected, a final model is trained using that hyperparameter and data from all the K' folds. This model can now be used for testing on test data i.e. data not included in any of the K' folds.

   Code and perform nested cross-validation to tune the maximum depth hyperparameter of decision trees as well as the number of top principal components features chosen. Provide $K = 10$ and $K' = 10$ as inputs to your nested cross-validation routine. State the test error. Is it higher or lower than the test error obtained using a separately-held test dataset? **Please include your code in the PDF answers report you submit.**

References

[1] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.