Henri, Jennifer, & Josh – Group A

RUDIMENTARY FACIAL CLASSIFICATION WITH PRINCIPAL COMPONENT ANALYSIS AND SUPPORT VECTOR MACHINES

Outline

1. Project goals
2. Data set
3. Principal Component Analysis and Support Vector Machines
4. Our method
5. Results of PCA, SVM, and classification
6. Areas of improvement
7. Conclusion

Project goals

Use principal component analysis and support vector machines to quickly and accurately classify a face

```
Result:
Laura Bush
```

(Not what we want)

Data set

1. We used the dataset Labeled Faces in the Wild from University of Massachusetts[2]
2. Total of 13293 images and 5752 people in the database.
3. We took 60 pictures from every person that had at least 60 pictures in the database (8 people)
4. Pictures needed to be in .jpg format and size 250x250
5. We deal with the grayscale and cropping of the pictures

Data set people


Ariel Sharon


Colin Powell

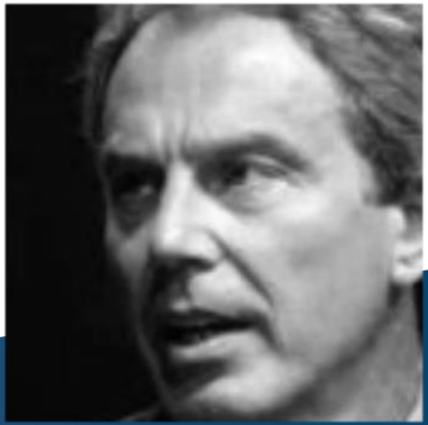

Donald Rumsfeld


George W Bush


Gerhard Schroeder


Hugo Chavez


Junichiro Koizumi


Tony Blair

What is Principal Component Analysis (PCA)?

- PCA is a technique used for dimension reduction with minimal information loss[1]
- Two of the biggest uses of PCA are clustering and dimension reduction
- In this project we use PCA to reduce the dimensions of the dataset to improve accuracy in facial classification

The math behind our PCA

- We're given 480 250x250 images (crop them to 125x125)
- We convert each image into a 125^2 dimensional vector

$$x_1, x_2, \dots, x_{480} \in \mathbb{R}^{125^2}$$

- Then we take the mean vector of all these images

$$\vec{\mu} = \frac{1}{480} \sum_{i=1}^{480} \vec{x}_i$$

The math behind our PCA

- Define the variance of any vector $\vec{u}$ as following

$$\mathrm{var}(\vec{u}) = \frac{1}{480}\sum_{i=1}^{480} |\vec{u}^*(\vec{x}_i - \vec{\mu})|^2$$

- We want to find the vector $\vec{u}$ that maximizes this variance to find the direction that captures the most variation in the dataset

The math behind our PCA

- Notice that the expression can be written

$$\mathrm{var}(\vec{u}) = \vec{u}^*(YY^*)\vec{u} \text{ and } Y = \frac{1}{\sqrt{480}}[\vec{x}_1 - \vec{\mu}|\cdots|\vec{x}_{480} - \vec{\mu}]$$

$$C = \frac{1}{480}\sum_{i=1}^{480}(\vec{x}_i - \vec{\mu})(\vec{x}_i - \vec{\mu})^*$$

- Where $C = YY^*$ is the sample covariance matrix and

$$\mathrm{var}(\vec{u}) = \vec{u}^* C \vec{u}$$

The math behind our PCA

- When we are trying to maximize the variance, this is the same as trying to maximize the previous expression

$$\max_{\|\vec{u}\|=1} \vec{u}^* C \vec{u} = \lambda_{\max}(C) = \sigma_1(Y^*)^2$$

- And the maximizing $\vec{u}$ is the top left singular vector of Y, or the first eigenvector of C

The math behind our PCA

- Define the principal components of a data set X to be the eigenvectors of the sample covariance matrix
- Then we have the following eigendecomposition with the eigenvalues in decreasing order

$$C = U \Lambda U^*$$

- And the columns of *U* are the principal components

The math behind our PCA

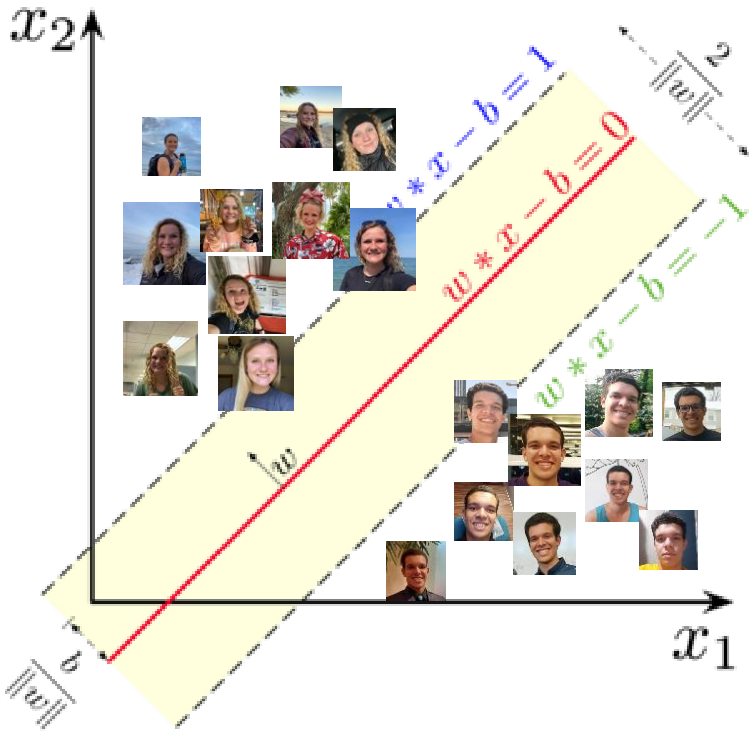
- In most datasets, only a few eigenvalues of C are large, so we can write each data point as a linear combination of only a few principal components with *very little information lost*
- This allows us to reduce the dimensions of the data significantly and avoid problems like collinearity when training our model

The math behind our PCA

- First *r* principal components: multiply dataset by the 1st *r* columns of the matrix *U* to get the data in PCA form (reduced dimensions)
- Then use that reduced dimension data to train our SVM model
- Alternatively, you could use the function `prcomp()` in R and it does all that work for you

What is Support Vector Machines (SVM)?

- SVM is a supervised machine learning algorithm used for classification
- We use SVM to classify faces given images that have been transformed into lower dimensions with PCA
- For the math behind SVM, view Topic 10



SVM visualization example [3]

Method

- Filter Data

```
filter_data <- function(path, number_pics, new_path=NULL, test=TRUE)
{
  if(test)
  {

  }
  # Grab all folder paths under destination
  all_paths<-list.dirs(path = path, full.names = TRUE, recursive = FALSE)

  # Create empty list, containg number of pics inside that path
  sizes<-c()
  # Loop through all the paths
  for (i in 1:length(all_paths)) {
    # Find the number of images in that path, and store in our list
    sizes<-c(sizes,length(list.files(all_paths[i])))
  }

  # Create a new list with only the number that are bigger than the specified
  list_new_paths <- c()
  for (i in 1:length(all_paths)){
    if(sizes[i]>=number_pics)
      list_new_paths<-c(list_new_paths, all_paths[i])
  }

  # Create the new folders in the new location
  dir.create(new_path)
  file.copy(from=list_new_paths, to=new_path,
            overwrite = TRUE, recursive = TRUE,
            copy.mode = TRUE)
  return(list_new_paths)
}
```

- Create Matrix of Images

```
create_matrix <- function(folders,nphotos=60,n=100)
{

  X <- c()
    for (i in 1:length(folders))
    {
      setwd(paste0(reduced_dir, "/",folders[i]))
      photos <- dir(path = paste0(reduced_dir, "/",folders[i]),
                    pattern = NULL, all.files = FALSE,
                    full.names = FALSE, recursive = FALSE,
                    ignore.case = FALSE, include.dirs = FALSE, no.. = FALSE)
      for (j in 1:nphotos)
      {
        pic <- grayscale(load.image(photos[j]))
        pic <- resize(pic, n, n)
        pic <- matrix(pic,nrow=n)[-c(c(1:floor(0.25*n)),c(floor(0.75*n+1):n)),-c(c(1:floor(0.25*n)),c(floor(0.75*n+1):n))]
        vector <- matrix(pic, ncol=1)[,1]
        X <- cbind(X, vector)
      }
    }
  return(X)
}
```

Method

- PCA

```r
faces_pca <- prcomp(t(matrix_images), center = FALSE, scale. = FALSE)
```

- Check for importance

```r
importance <- as.data.frame(summary(faces_pca)$importance)
imp_val <- check_importance(importance)
```

- Create a response variable

```r
response_var <- c()
for (i in 1:length(folders))
{
  setwd(paste0(reduced_dir, "/",folders[i]))
  photos <- dir(path = paste0(reduced_dir, "/",folders[i]),
                pattern = NULL, all.files = FALSE,
                full.names = FALSE, recursive = FALSE,
                ignore.case = FALSE, include.dirs = FALSE, no.. = FALSE)
  response_var <- c(y, rep(i, each=60))
}
```

Method

- Split data

```
data_vec <- 1:length(mydata[,1])
train_vec <- data_vec[data_vec%%60 <= 48 & data_vec%%60 != 0]
val_vec <- data_vec[data_vec%%60 <= 54 & data_vec%%60 > 48]
test_vec <- data_vec[data_vec%%60 <= 60 & data_vec%%60 > 54 | data_vec%%60 == 0]

train <- mydata[train_vec,]
val <- mydata[val_vec,]
test <- mydata[test_vec,]
```

Method

- Train model

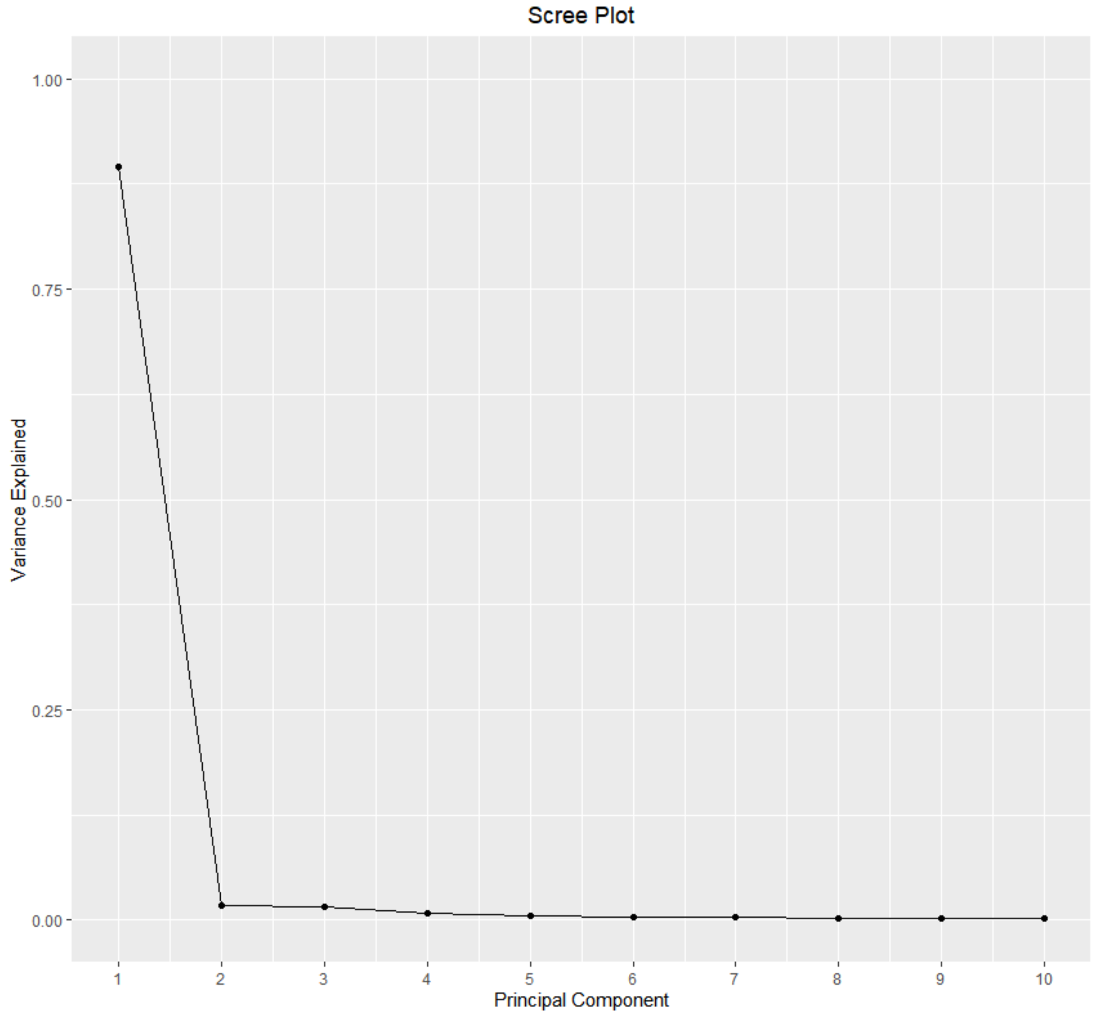
```
accuracy <- c()
max <- 0
b_gamma <- 0
gammas <- c(0.1,0.01,0.001,0.0001,0.00001,0.000001)

for (i in 1:length(gammas))
{
  classifier <- svm(y ~ ., data = train, gamma = gammas[i], kernel = "radial")
  prediction <- predict(classifier, newdata = val)
  m <- sum(prediction == val$y)/length(prediction)
  accuracy <- c(accuracy, m)
  if (m == max(accuracy))
  {
    max <- m
  }
  if (i == length(gammas))
  {
    print(paste0("Best gamma is ", gammas[i]))
    b_gamma <- gammas[i]
  }
}
```

- Validate

```
classifier <- svm(y ~ ., data = train, gamma = b_gamma, kernel = "radial")
prediction <- predict(classifier, newdata = val)
```

Results of PCA

- There is a massive dropoff in explained variance as we increase the number of principal components
- Only 104 of the 480 PCs are needed to explain over 99% of the variance in the data!



Results of SVM

- At first our model struggled to classify accurately more than 40-50% of the time
- If we crop the pictures to the center, most of the images would be just the face and have less background to throw our model off

Results of SVM

- We split our 480 images into 80% training, 10% validation, and 10% testing
- Created multiple SVM models with different kernel parameters (with radial kernel type)
- Chose the model with the highest accuracy in the validation set as our model

Results of Model

Example of model prediction with PCA

Correct: George_W_Bush

Prediction: George_W_Bush

Correct: George_W_Bush

Prediction: Tony_Blair

Results of Model

- There was a massive difference in accuracy when using PCA compared to not using PCA

	Validation Accuracy	Testing Accuracy
With PCA	72.9%	70.8%
Without PCA	22.9%	14.6%

Results of validation classification

	folders <chr>	gender <chr>	Accuracy (%) <dbl>	Predicted individual as... <chr>	X2 <chr>	X3 <chr>	X4 <chr>
11	Colin_Powell	M	66.67	Donald_Rumsfeld	Tony_Blair	NA	NA
13	Donald_Rumsfeld	M	83.33	Colin_Powell	NA	NA	NA
15	George_W_Bush	M	66.67	Donald_Rumsfeld	Hugo_Chavez	NA	NA
16	Gerhard_Schroeder	M	33.33	Donald_Rumsfeld	George_W_Bush	Hugo_Chavez	Hugo_Chavez
23	Hugo_Chavez	M	66.67	George_W_Bush	Tony_Blair	NA	NA
62	Tony_Blair	M	66.67	George_W_Bush	George_W_Bush	NA	NA

- Total Accuracy of the algorithm for validation 72.92%
- The name that was predicted the most was George B with a count of 4
- The highest number of inaccurate predictions per person was: 4

Results of testing classification

	folders <chr>	gender <chr>	Accuracy (%) <dbl>	Predicted individual as... <chr>	X2 <chr>	X3 <chr>	X4 <chr>
6	Ariel_Sharon	M	83.33	Colin_Powell	NA	NA	NA
11	Colin_Powell	M	33.33	Ariel_Sharon	George_W_Bush	Gerhard_Schroeder	Junichiro_Koizumi
15	George_W_Bush	M	83.33	Tony_Blair	NA	NA	NA
16	Gerhard_Schroeder	M	50.00	Donald_Rumsfeld	Hugo_Chavez	Tony_Blair	NA
23	Hugo_Chavez	M	66.67	Junichiro_Koizumi	Tony_Blair	NA	NA
39	Junichiro_Koizumi	M	83.33	Ariel_Sharon	NA	NA	NA
62	Tony_Blair	M	66.67	Colin_Powell	Gerhard_Schroeder	NA	NA

- Total Accuracy of the algorithm for testing 70.83%
- The name that was predicted the most was Tony Blair with a count of 3
- The highest number of inaccurate predictions per person was: 4

Areas of Improvement

70%?

- NONE, WE ARE PERFECT 😎
- Only had 8 people with over 60 images
  - More people with a higher quantity of images might increase accuracy
- Crop to faces better
  - Our cropping took the center of the image, some faces weren't centered
- To test other predicting/classification methods
  - logistic regression, K-nearest neighbors, or convolutional neural networks

Conclusion

- Converted 480 grayscale images from 8 separate people into column vectors and created a matrix of images
- Ran PCA on this matrix and reduced our dimensions of the model from 480 to 104
- Used SVM to create a model classifying an image
- Our model had 70.8% accuracy in classifying faces

Correct: George_W_Bush

Prediction: George_W_Bush

(What we want)

References

1. https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202
2. http://vis-www.cs.umass.edu/lfw/
3. https://en.wikipedia.org/wiki/File:SVM_margin.png
4. GitHub

Libraries

- rcpp
- imager
- tictoc
- ggplot2
- magick
- e1071

Questions?