Principal Component Analysis (PA1 Problem Discussion)

Michael Liu

January 13, 2020

Intuition of using PCA for image data

Suppose we have a set of M images $[\Phi_1, \Phi_2, \ldots, \Phi_M]$, and each image has a fixed size of $N \times N$, as an input feature vector, a single image will have a dimension of $1 \times N^2$ (flatten version of that image). This means you need N^2 input nodes to train a classifier like Logistic/Softmax Regression, and it is computationally expensive to do so. This is why we want to use PCA as dimension reduction for this programming assignment.

Later on in the course, you will find out that neural network can also be viewed similarly as a dimension reduction technique

Definition/Derivation of PCA

PCA uses *Principal Components* to perform dimension reduction.

Principal Components have both **direction** and **magnitude**. The direction represents across which principal axes the data is mostly spread out or has most variance and the magnitude signifies the amount of variance that *Principal Component* captures of the data when projected onto that axis.

The principal axes can further be derived as the dominated eigen-vectors $(v_1, v_2, \ldots, v_k)$ associated with the largest k eigen-values for the covariance matrix Σ of your training data.

$$\text{PC}_k = \lambda_k v_k$$

Visualization of PCA

The projection process in PCA can be viewed as a change of basis.

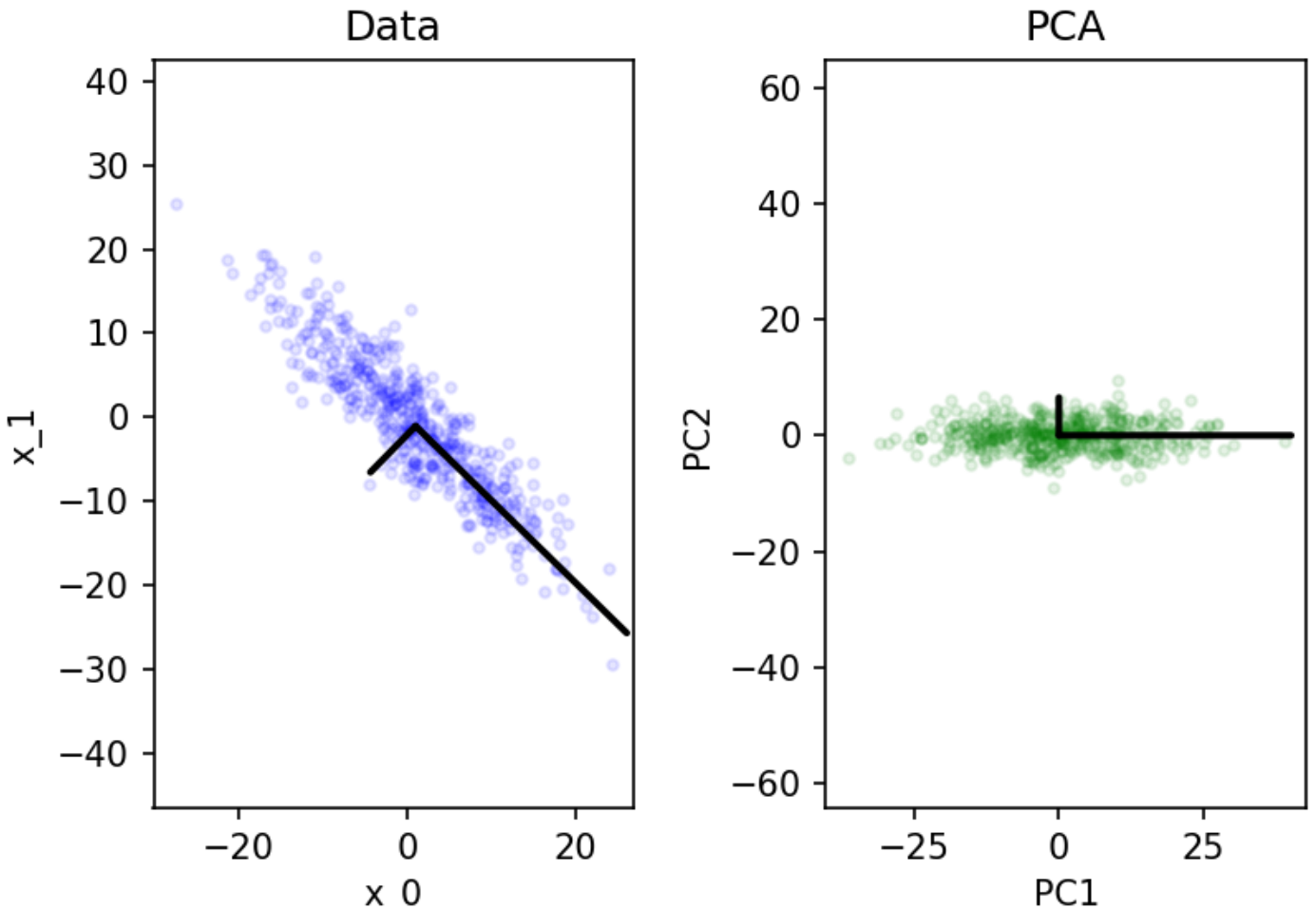


Figure: The projection from XY axes space to principal axes space

Find Principal Components for Training Images

Suppose we have M number of training images $[\Gamma_1, \Gamma_2, \ldots, \Gamma_k]$ with a total dimension of $(M \times N^2)$, we first center all the images by subtracting the average image:

$$\Phi_i = \Gamma_i - \frac{1}{M} \sum_{i}^{M} \Gamma_i$$

Then we compute the covariance matrix C:

$$C = \frac{1}{M} \sum_{i}^{M} \Phi_i \Phi_i^T = \frac{1}{M} \Phi \Phi^T$$

where $\Phi = [\Phi_1, \Phi_2, \ldots, \Phi_M]$

This results in a covariance matrix C $(N^2 \times N^2)$, which is too large to effectively compute its top k eigen-vectors and eigen-values.

Turk & Pentland's Trick

If we have a matrix product $C = A^T A$ and we know the kth eigen-vector v_k and eigen-value λ_k of C, we can then efficiently compute the kth eigen-vector u_k and eigen-value λ_k of $C^T = AA^T$.

$$A^T A v_k = \lambda_k v_k$$

Left multiply both side with A,

$$AA^T A v_k = A\lambda_k v_k = \lambda_k A v_k$$

$$AA^T \hat{u_k} = \lambda_k \hat{u_k}$$

Therefore, with Turk & Pentland's Trick, we see that the eigen-vector for AA^T is $u_k = \frac{\hat{u_k}}{||\hat{u_k}||}$ (normalized to ensure unit-vector), and the eigen-value λ_k remains the same.

Putting it together

Previously, we have the matrix C as,

$$C = \frac{1}{M}\sum_{i}^{M}\Phi_i\Phi_i^T = \frac{1}{M}\Phi\Phi^T$$

Calculate the covariance matrix C^T as,

$$C^T = \frac{1}{M}\Phi^T\Phi = \frac{1}{M}\sum_{j}^{N^2}\Phi_{*}j\Phi_{*}j$$

where $*$ represents all the examples from the training set.
Calculate the eigen-vectors $\vec{v}$ $(M \times k)$ and the eigen-values $\vec{\lambda}$ $(1 \times k)$ of C^T $(M \times M)$,
Use Turk & Pentland's Trick to compute the eigen-vectors u_k for C

$$u_k = \frac{\Phi v_k}{||\Phi v_k||}$$

Projection onto Principal Components

Now, we have the principal axes u_k and the magnitude λ_k of our centered images Φ. Suppose we have a image Γ^* and we want to project it onto the principal axes, we first need to **Center** the image with the average image from training set

$$\Phi^* = \Gamma^* - \bar{\Gamma}$$

Project the centered image onto the principal axes

$$\hat{x} = \Phi^* \vec{u}$$

where $\vec{u}$ is the principal axes $(N^2 \times k)$, Φ^* is a single image $(1 \times N^2)$ and $\hat{x}$ is the reduced feature representation of the image $(1 \times k)$, living in the principal component space.

To further help convergence of logistic regression we want to sample from a normalized input space, which means each input distribution needs to be 0 mean and unit standard deviation. This is where **the magnitude** of the PCs comes into the scene. To normalize the input vector, simply divide by the eigen-vectors.

$$x = \hat{x}/\vec{\lambda}$$

Inverse Projection from Reduced Feature Vector

Now, you know how to project your image to the principal axes, what do you think the inverse projection will be?

Hint: Don't forget to scale back **the magnitude** and add back **the average image**

Lastly...

Double check your equations with the ones from the Turk & Pentland's paper posted under "Resource" section.

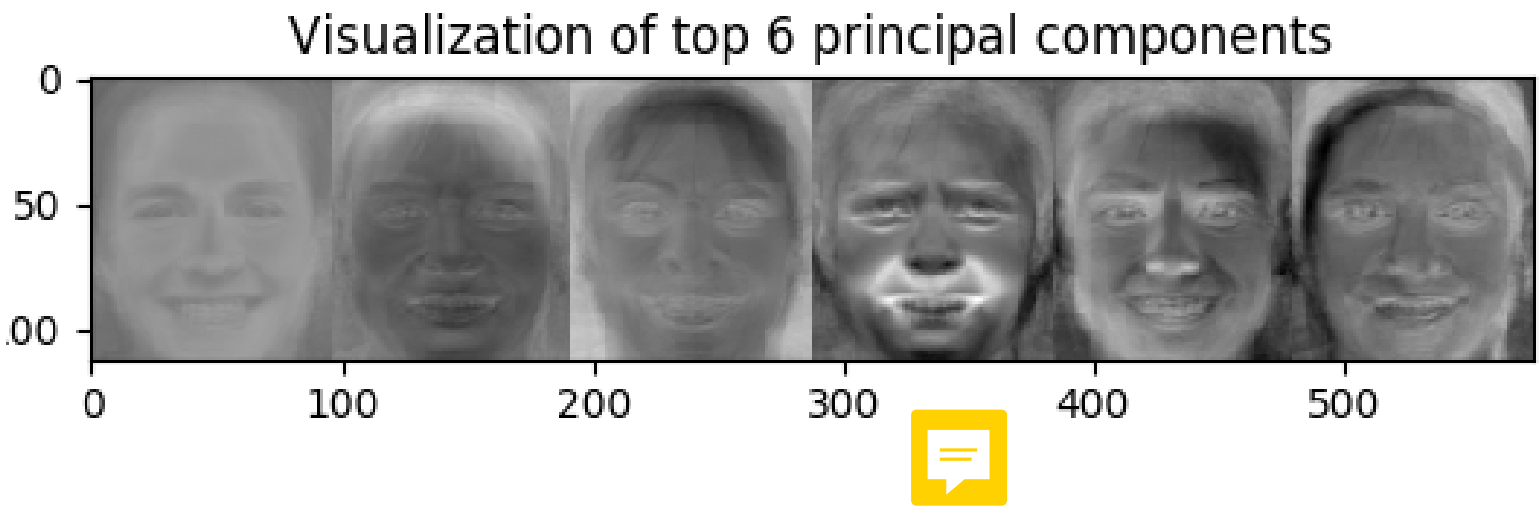
Perform Sanity Check at the bottom of the PA instruction to make sure your implementation is correct.

Perform PCA dimension reduction with a large k and inverse transform the feature back to the image dimension for reconstruction. Try to visualize the image to see how well your reconstruction is.

Note that PCA assumes (1) normal distribution of input space, (2) real continuous value of input space, and (3) a simple underlying manifold of the input space. However, if these properties (especially the third one) do not hold true, PCA will not serve as an effective dimension reduction technique. *you can see the importance of these assumptions by experimenting with* **Resized** *dataset and compare the result from using the* **Aligned** *dataset*

Some Visualization

Figure shows the top 6 PCs on Happiness and Anger training set (part of it). Note that the 4th PC highlights the mouth and eyebrow section and the 1th PC looks very close to a canonical face.



Other Resources

On Visualization: http://setosa.io/ev/principal-component-analysis/
On Derivation:
http://www.stat.columbia.edu/ fwood/Teaching/w4315/Fall2009/pca.pdf