L41 BME 5657, HW 3

This is completely my own work.

Collaboration:

* Received clarification on the Expectation Maximization algorithm from Lijun.

References:

* Stackoverflow and the matplotlib docs were sparingly used only as syntactical references for graphing

Aneesh Sachdeva

441472

L41 BME 5657

HW3

LDA vs PCA

The purpose of this experiment is to compare the advantages and disadvantages between Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) in classification tasks. It’s important to note that LDA inherently discriminates between classes while PCA does not: LDA measures the variance *between* classes while PCA measures the variance *within* a given dataset. In other words, PCA is an unsupervised task whereas LDA is supervised.

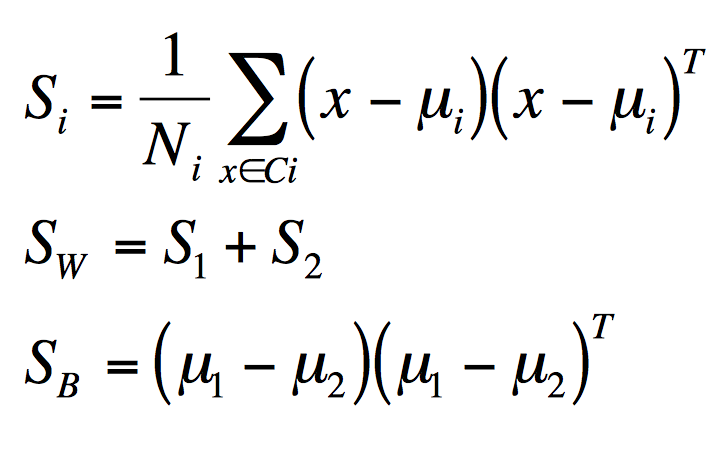


Figure 1: The Scatter Matrices of the Linear Discriminant Algorithm. S\_B is the scatter of the class means and S\_W is the cumulative scatter within each class.

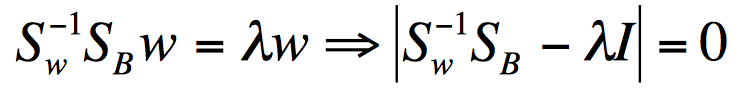


Figure 2: The LDA projection is obtained as the solution of the generalized eigenvalue problem.

The LDA algorithm (Figs 1, 2) is comprised of two main parts: measuring the interclass and intraclass variation, and computing the eigenvectors of the multiplication of those two measures. The intraclass variation (Fig 1) is measured by SW, which takes the average variance within each class and sums the matrices. The interclass variation (Fig 1), SB, is measured by squaring the difference of the two class means (their variation). The weights of the LDA vector are then computed by taking the strongest eigenvector of SW-1SB (Fig 2). The obtained weights can then be used to project data points from each class onto, thus reducing their dimensionality (Fig 3).

PCA is performed by taking the eigenvectors of the covariance matrix of the data. Dimensionality reduction can then be performed by projecting the data onto N eigenvectors, where N is the number of desired dimensions. See Appendix A for a deeper demonstration of PCA.

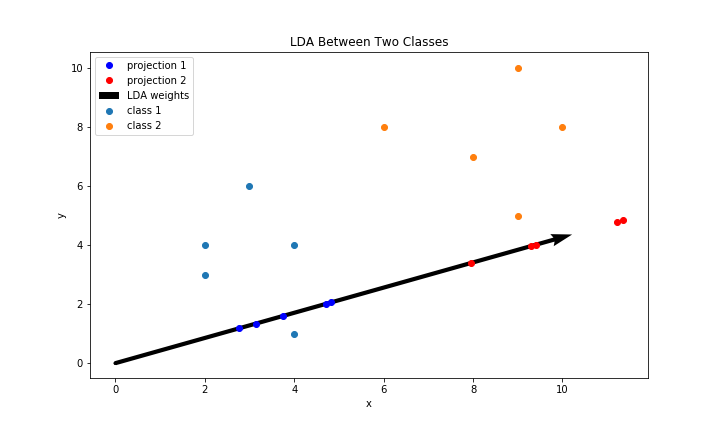


Figure 3a: The LDA projection of two classes of data. The points on the LDA vector are from the projections of the original data points. The separation of class 1 and 2 projections on the weight vector illustrates LDA's success in finding the most variance between the two classes.

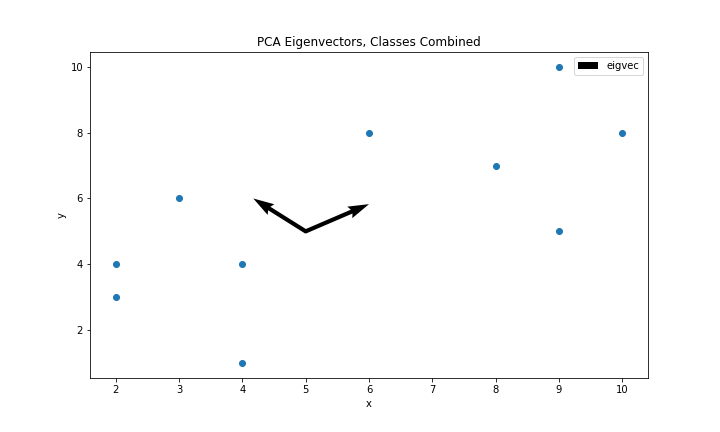


Figure 3b: The PCA eigenvectors of class 1 and 2 combined from Fig 3a. The north-east vector has an eigenvalue of 12.90 and the north-west vector has an eigenvalue of 2.35.

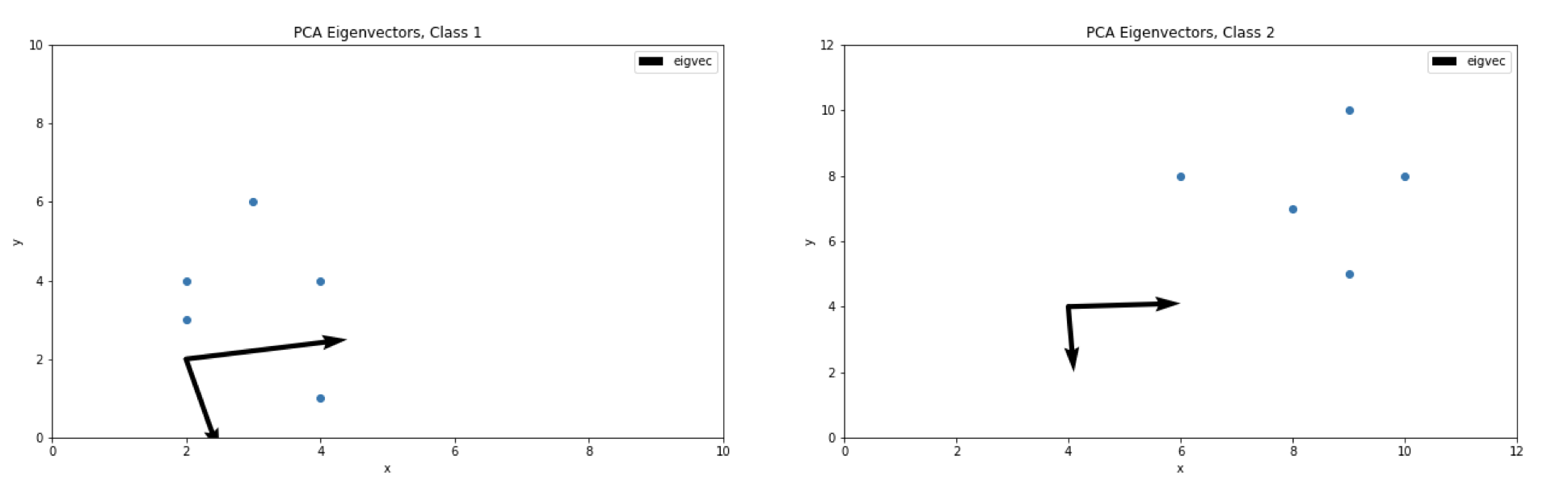


Figure 3c: PCA of each class. (Right) the largest eigenvector is in the +X direction with an eigenvalue of 2.72. (Left) the largest eigenvector is in the +X direction with an eigenvalue of 2.64.

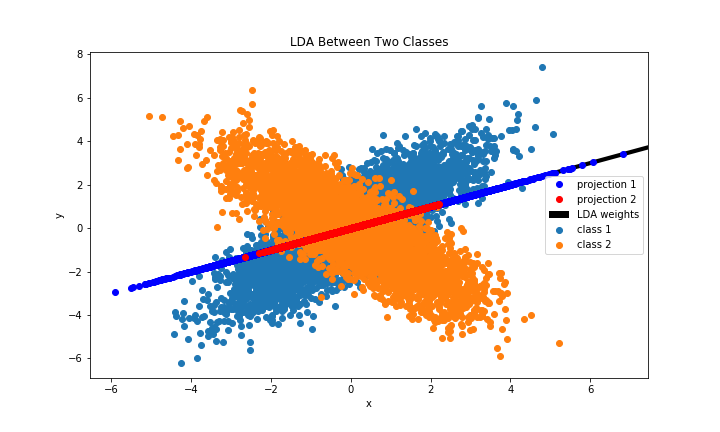


Figure 4a: When classes of data overlap in distribution and variance, LDA can fail in distinguishing between classes. This failure is observed in the overlap of each class's projections in the middle of the LDA weight vector.

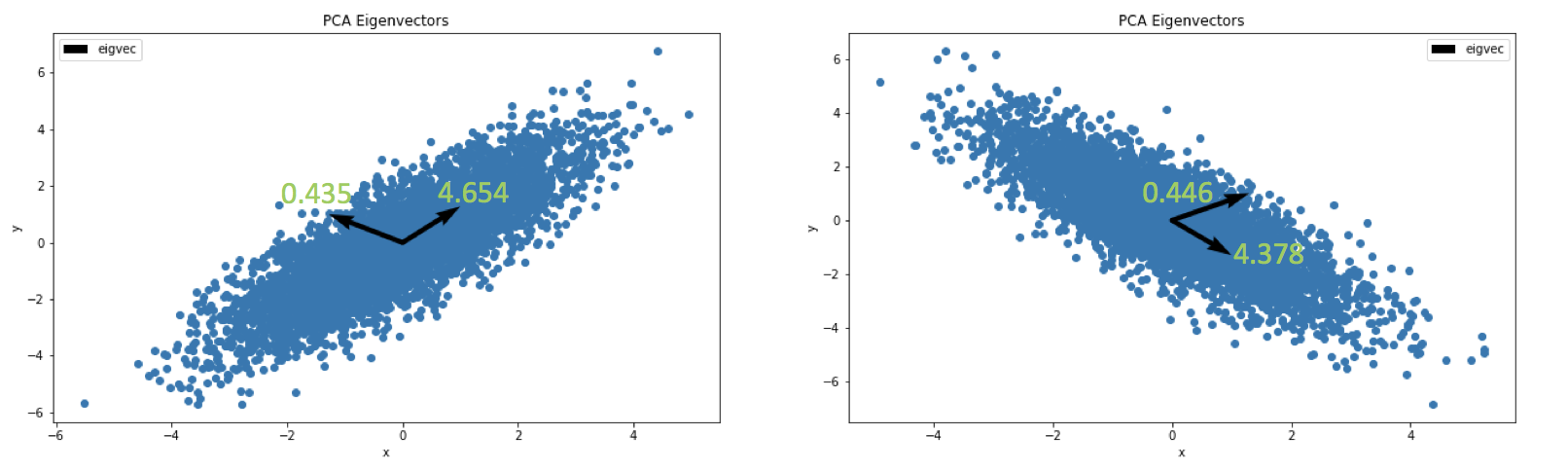


Figure 4b: (Left) two-dimensional Gaussian distribution with covariance matrix=[[2, 2], [2, 3]] and mean=[0, 0]. (Right) two-dimensional Gaussian distribution with covariance matrix=[[-2, 2], [-2, 3]] and mean=[0, 0]. The attached vectors are the eigenvectors of each distribution, included with their corresponding eigenvalues.

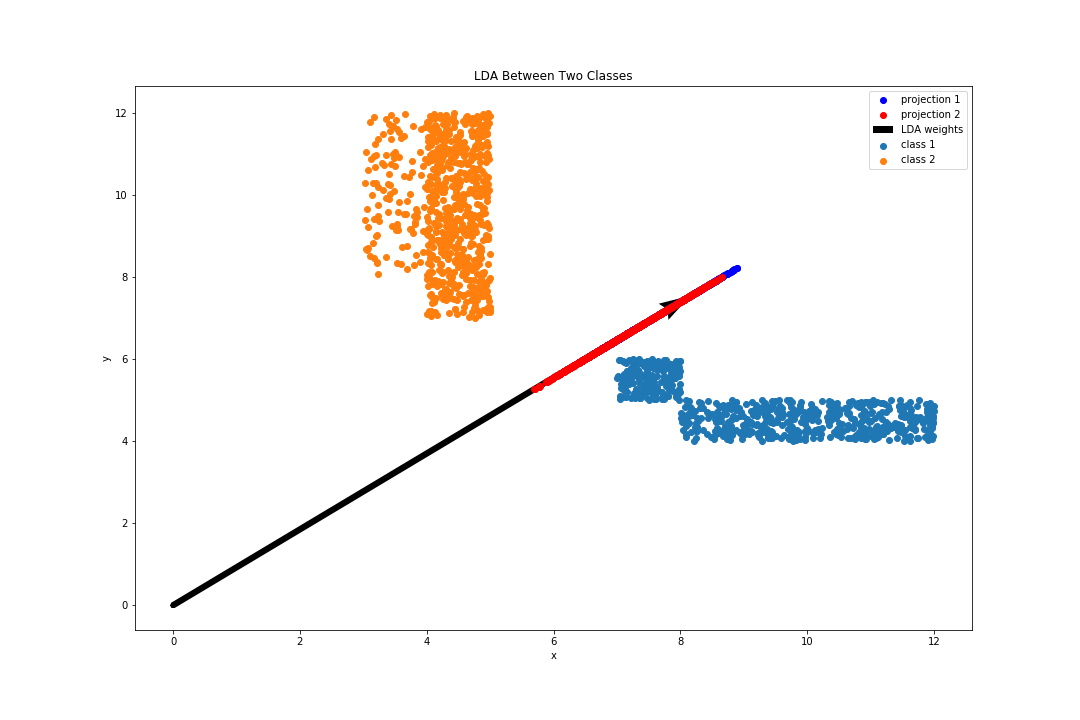


Figure 5: LDA on non-Gaussian (uniform) data. LDA fails to classify due to its Gaussian assumption.

To illustrate the differences between LDA and PCA, numerous datasets of varying size and covariance were generated and analyzed. One set of data illustrates LDA and PCA used on two low volume, non-overlapping classes of data, and the other set illustrates LDA and PCA on two high volume, partially-overlapping classes of data. For the first set (Fig 3), LDA correctly identifies the vector of highest variation between the two classes. When points in both classes are projected onto this vector, they remain non-overlapping, but now in a lower dimension. The projection of a 2D point onto the weight vector results in a 1D value, which is then scaled back onto the weight vector (in 2D space) just for the sake of visualization (Fig 3a). When PCA is applied to the union of the two classes (Fig 3b), the eigenvector associated with the largest eigenvalue (12.9) ends up being near identical to the LDA weight vector. Mathematically, this is plausible if the maximum variation *within* each class is similar to the maximum variation *between*. This is confirmed by the PCA’s of each class (Fig 3c); each class’s dominant eigenvector is similar to each other and the LDA weights.

Although for the first set of data there happens to be a correspondence between LDA and PCA, it should be noted that this is due to unique characteristics of the data rather than the functional objectives of the algorithm. In Figure 3, both data sets already have similar vectors of maximum variance and no overlap, so it’s intuitive that LDA, which finds the maximum variance between both classes, produces a vector that is similar. However, PCA is inherently *unsupervised* (does not consider classes), whereas LDA is *supervised* (recognizes classes). Thus, it cannot always be the case that the PCA’s of each class agree with LDA for any given dataset. An example of this is provided in Figure 4. The dataset is constructed with two classes of high volume (5000 points), overlapping data, and sampled from two different respective Gaussian distributions (see figure caption for details). Both distributions are mirror images of each other, so their axes of maximum variation should also be mirror images. This is confirmed in Figure 4b, the PCA of each class produces eigenvectors that are of near equal magnitude but are mirror images (note that eigenvectors are not unique for a given task, they should be regarded as a plane). It is clear that their vectors of maximum variation are not equal, and thus no correspondence with LDA should be expected. Figure 4a shows the results of LDA on the Gaussian data sets. Due to the significant overlap in distributions LDA isn’t as effective as it could be in distinguishing the classes. However, both classes do fit LDA’s Gaussian and i.i.d assumptions, so it is possible that the LDA projection could be used as an intermediate in data processing.

Where both LDA and PCA fail, however, is when dealing with non-Gaussian or non i.i.d data. A fundemental assumption of LDA and PCA is that all the variables in the data follow a Gaussian distribution. This is how the algorithms are able to use covariance matrices to measure scatter between and within the classes. When this assumption is broken, the math can no longer reliably find maximum variance within the data. This is demonstrated in Figure 5: Two uniform (non-Gaussian), non-overlapping distributions are generated and LDA is applied. By analyzing the projections of the data onto the LDA weight vector and observing the significant overlap, it is clear that LDA does not discriminate between the two classes.

When the Gaussian assumptions of LDA is violated, other methods may be more suitable for classification. For example, a clustering algorithm (e.g. K-means clustering), would do significantly better on classifying the two distinct groups of data (Fig. 5) than LDA currently does. Also, another disadvantage of LDA is its compute requirements: on large data sets LDA in its most traditional form may not be suitable due to the immense computational efforts required to compute the inverse of the scatter matrix, SW (Fig. 2). These results also demonstrate that PCA should not be used as a classifier, but rather as a method of dimensionality reduction. On high dimensional data sets, PCA could be used to reduce dimensions and then LDA could be used to classify.

Expectation Maximization

The purpose of this experiment is to determine the probabilities of obtaining heads for two independent coins, in a dataset where the flips are known but the coin identities are not. This will be solved by using the Expectation Maximization algorithm, which is derived from Bayes Theorem (Fig. 6). The dataset contains N number of trials, where each trial is the result of a randomly (equal probability) selected coin flipped 10 times. Various N’s were tested and convergence was observed for each.

The algorithm for expectation maximization involves two key steps: the estimation step (E-step) and the maximization step (M-step). In the E-step, the posterior probabilities of both coins will be estimated for each trial using the equation in Figure 4. By assuming a binomial distribution for the heads and tails of the coin flips and using our current estimation of the prior, , the probability of the observed trial given a coin, can be calculated (Fig. 8). The likelihoods (Fig. 7) of Coin A and B are then used to “assign credit” to our estimated distributions of Theta A and B, respectively. After each trial has been stepped through, we then perform the M-step. In the M-step, we update our belief of each Theta by calculating its probability from our likelihood estimations in the E-step. This cycle is repeated until our estimations of Theta converge.

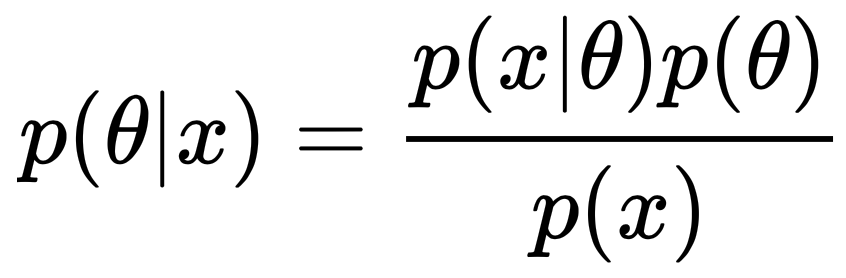


Figure 6: Baye's Theorem, equation for calculating the posterior probability, p(theta|x). This is calculated by multiplying the prior probability, p(x|theta)/p(x), by the likelihood p(theta).

Figure 7: Derived from Baye's Theorem, this is the likelihood equation for measuring the likelihood of Coin A (and can reverse variables for B) given the heads observed in a given trial. This is used in the E-step of the algorithm. Note that all three terms of the equation will have the same binomial coefficient, so it can be omitted from the calculation.

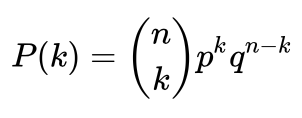


Figure 8: The probability of getting heads (within each trial) follows a Binomial Distribution, as measured by this equation. Note, the binomial coefficient can be omitted from the code due to the division made in the E-step, Fig 7.

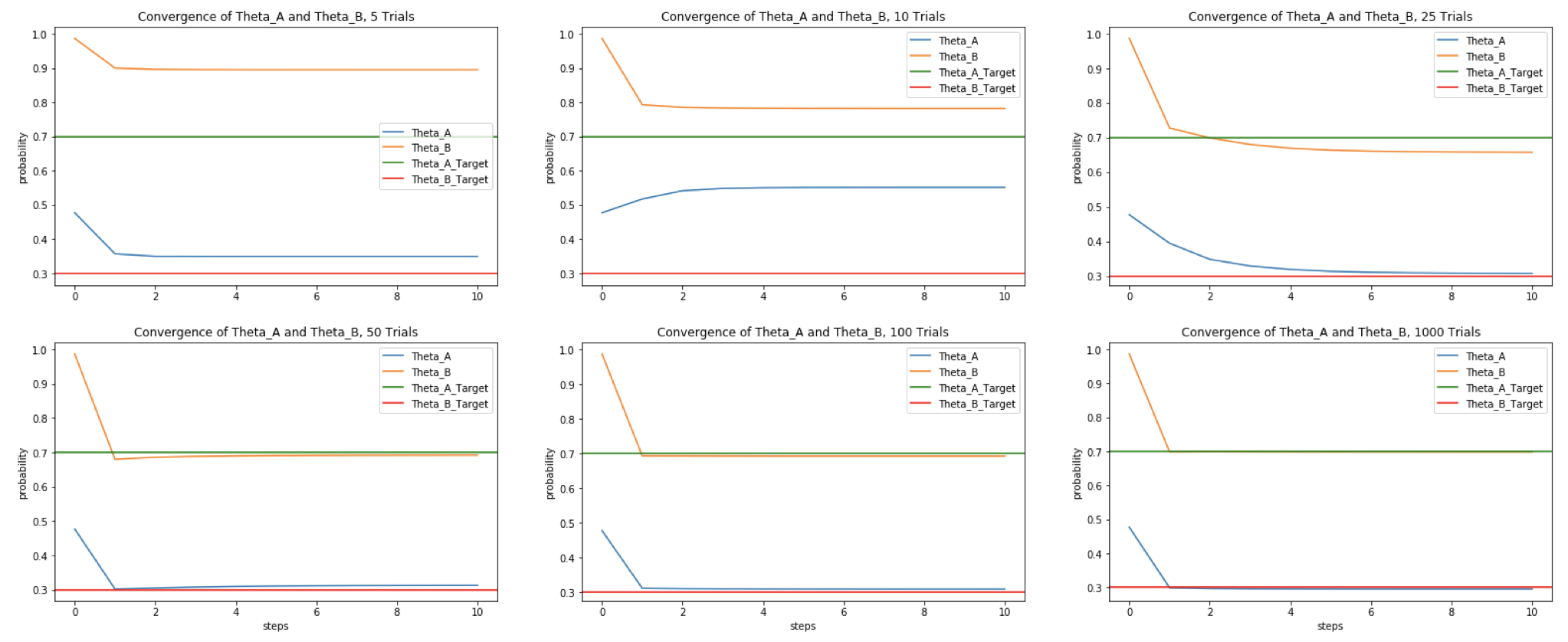


Figure 9: Convergence of Thetas A and B given true probabilities of 0.7 and 0.3 respectively. Each subplot is convergence given N number of observed trials.

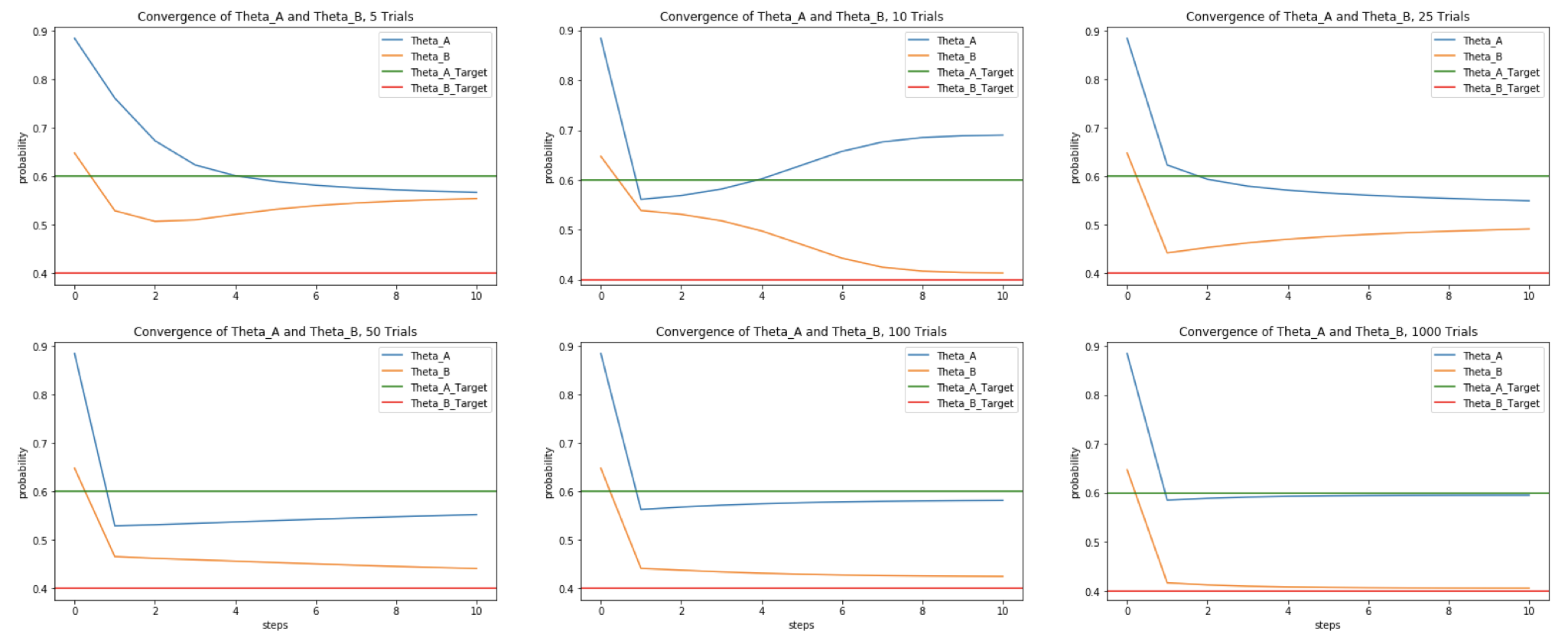


Figure 10: Convergence of Thetas A and B given true probabilities of 0.6 and 0.4 respectively. Each subplot is convergence given N number of observed trials.

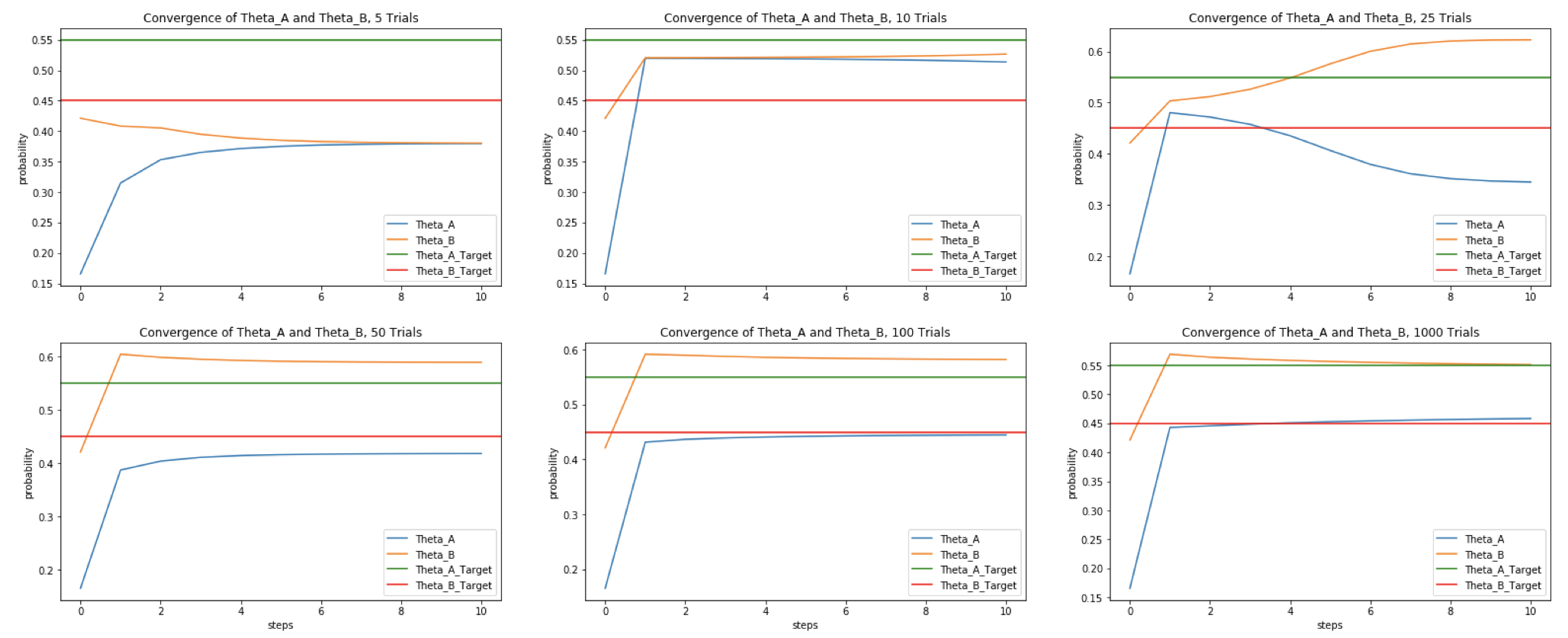


Figure 11: Convergence of Thetas A and B given true probabilities of 0.55 and 0.45 respectively. Each subplot is convergence given N number of observed trials.

A key factor in allowing Theta A and Theta B to converge on their true probabilities was the number of observed trials provided to the algorithm (Figs 9, 10, 11). As the difference between Theta A and Theta B decreases (their true probabilities become more similar), the number of trials required for proper convergence increases. When Theta A and Theta B were set to 0.55 and 0.45 respectively, it took until N=1000 trials before proper convergence was observed (Fig. 11). This makes sense both intuitively and mathematically. Intuitively, if the two unidentified coins have similar probabilities, it’s going to require more flips of each coin before we observe p(heads) converging for both coins. Mathematically, the strength of our Bayesian updates is entirely dependent on the abundance and quality of the data. With limited data (fewer trials), our estimated prior will not be near strong enough to obtain convergence, especially if the discrimination task is more difficult. In other words, the prior will attempt to capture the random variations in our limited data set. More data will reduce magnitude of the randomness. The effect of a “poor prior” can be observed in significant deviations of Theta A and Theta B from their true values. The N=10 run in Figure 10 and the N=25 run in Figure 11 both clearly show this behavior.

In conclusion, when given an adequate dataset the Expectation Maximization algorithm can be efficient in determining unique and unlabeled distributions. However, the more similar the true distributions, the more difficult it will to obtain convergence. As the strength of our priors depend on the information observed, the discrimination task can be improved by providing more data and running more consecutive steps until convergence is observed.

Appendix A

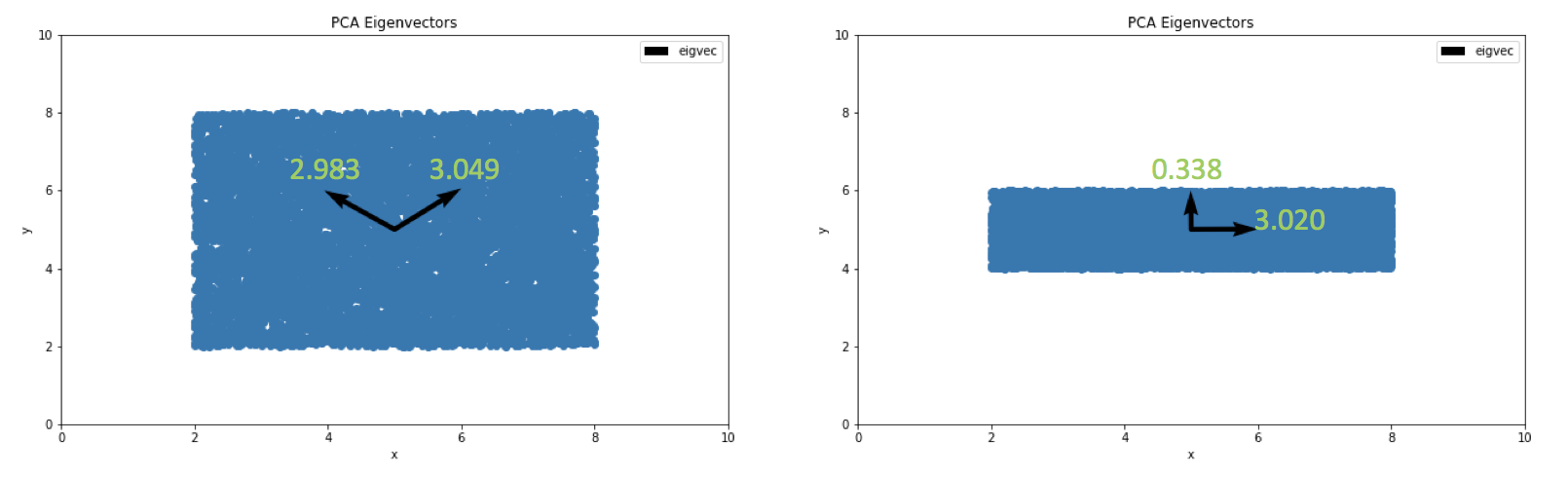


Figure 12: Demonstration of PCA finding the axis of most variation. This is reminiscent of the "cheese slice" example.

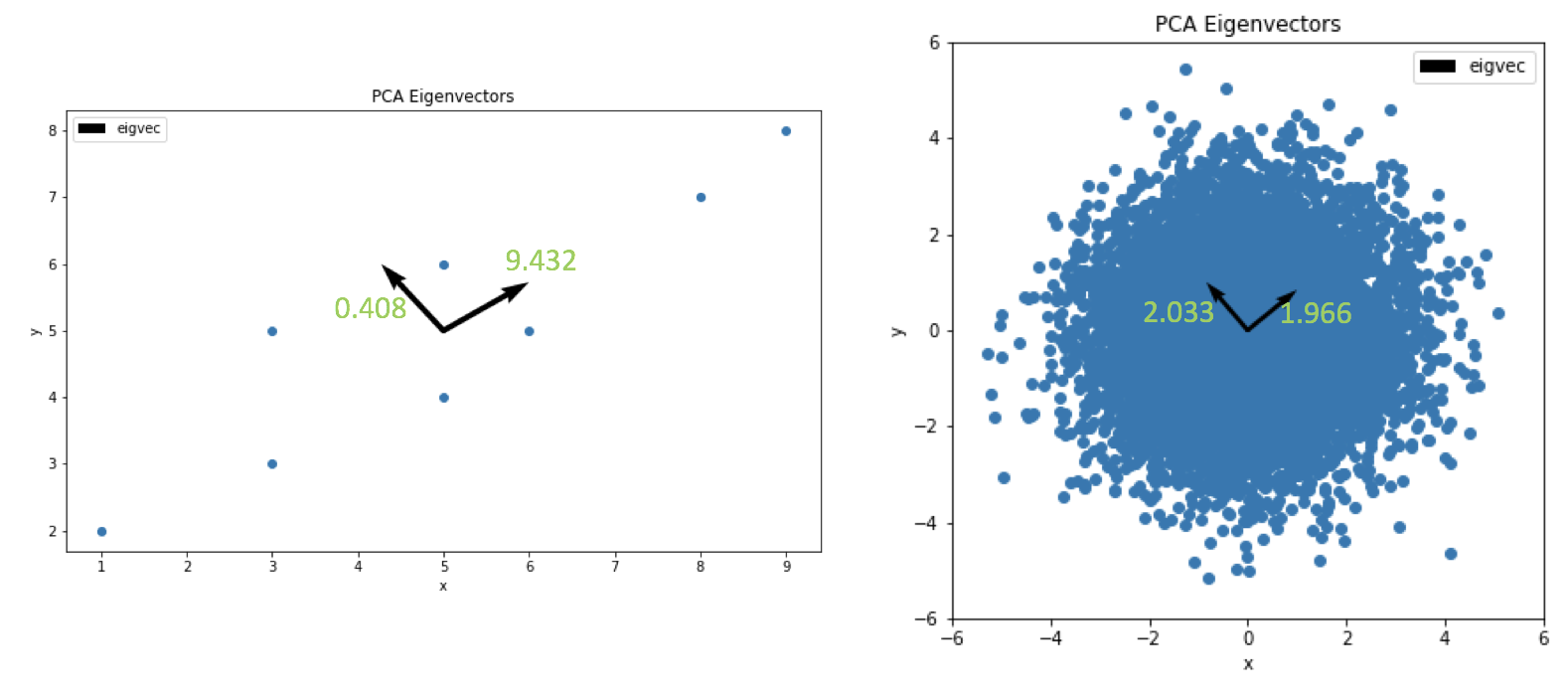


Figure 4: (Left) There is a clear vector of maximum variation which PCA finds. (Right) Each variable, X and Y, was sampled from an identical Gaussian distribution. Thus, there is no single vector of maximum variation. PCA produces eigenvectors of near equal corresponding eigenvalues.

These figures demonstrate PCA’s ability of quantifying the variation within a given dataset. When there is a vector of maximum variation within the data, PCA finds it. When there is not, PCA produces eigenvectors with near identical corresponding eigenvalues. The eigendecomposition is entirely unsupervised.