

1. (Rotation) For a symmetric matrix  $A$ , can there be a nonzero vector  $x$  such that  $Ax$  is nonzero and orthogonal to  $x$ ? Either prove that this is impossible, or explain under what condition on the eigenvalues of  $A$  such a vector exists.

Let  $x \in V, x \neq 0$ , an inner product space, by the spectral theorem there exists an orthonormal basis of  $V$ , consisting of eigenvectors of  $A$ , let  $u_1, \dots, u_n$  be the eigenbasis of  $A$ , and  $\lambda_1, \dots, \lambda_n$  the eigenvalues for each of these eigenvectors.  $x \in \text{span}\{u_1, \dots, u_n\} \Rightarrow x = \sum_{i=1,n} \alpha_i u_i, \alpha_i \neq 0$ .  $x^T(Ax) = (\sum_{i=1,n} \alpha_i u_i)(\sum_{j=1,n} \alpha_j Au_j) = (\sum_{i=1,n} \alpha_i u_i)(\sum_{j=1,n} \alpha_j \lambda_j u_j) = \sum_{i=1,n} \alpha_i^2 \lambda_i$  since  $u_i^T u_j = 0$  for  $i \neq j$  and  $u_i^T u_i = 1$ .  $Ax$  is orthogonal to  $x$ :  $x^T(Ax) = 0 \Rightarrow \sum_{i=1,n} \alpha_i^2 \lambda_i = 0$ .

2. (Matrix decomposition) The trace can be used to define an inner product between matrices:

$$\langle A, B \rangle := \text{tr}(A^T B), \quad A, B \in \mathbb{R}^{m \times n}, \quad (1)$$

where the corresponding norm is the Frobenius norm  $\|A\|_F := \langle A, A \rangle$ .

- (a) Express the inner product in terms of vectorized matrices and use the result to prove that this is a valid inner product.  $(AB)_{ij} = (\sum_k A_{ik} B_{kj})_{ij}$ , and  $(A^T B)_{ij} = (\sum_k A_{ki} B_{kj})_{ij}$ .  $\text{tr}(A) = \sum_i A_{ii} \Rightarrow \text{tr}(A^T B) = \sum_i \sum_k A_{ki} B_{ki} = \sum_i \sum_j A_{ij} B_{ij} = \text{vec}(A)^T \text{vec}(B) = \langle \text{vec}(A), \text{vec}(B) \rangle$ . The trace is then the inner product between vectors in  $\mathbb{R}^{mn}$  thus is a valid inner product.
- (b) Prove that for any  $A, B \in \mathbb{R}^{m \times n}$ ,  $\text{tr}(A^T B) = \text{tr}(B A^T)$ .  $\text{tr}(B A^T) = \sum_i \sum_k B_{ik} A_{ik} = \sum_i \sum_j A_{ij} B_{ij} = \text{tr}(A^T B)$ .
- (c) Let  $u_1, \dots, u_n$  be the eigenvectors of a symmetric matrix  $A$ . Compute the inner product between the rank-1 matrices  $u_i u_i^T$  and  $u_j u_j^T$  for  $i \neq j$ , and also the norm of  $u_i u_i^T$  for  $i = 1, \dots, n$ . For  $i \neq j$ ,  $\langle u_i u_i^T, u_j u_j^T \rangle = \text{tr}(u_i u_i^T u_j u_j^T) = \text{tr}(u_i 0 u_j^T) = 0$ , since  $u_i, u_j$  are two eigenvectors of a symmetric matrix therefore orthogonal. if  $i = j$  then  $\langle u_i u_i^T, u_i u_i^T \rangle = \text{tr}(u_i u_i^T u_i u_i^T) = \text{tr}(u_i^T I u_i) = \text{tr}(u_i^T u_i) = 1$  if the eigenvectors are also orthonormal.
- (d) What is the projection of  $A$  onto  $u_i u_i^T$ ? If  $A$  is a symmetric matrix, by the spectral theorem,  $A = U D U^T$  where  $D$  is the diagonal matrix having  $\lambda_i, i = 1, \dots, n$  the eigenvalues of  $A$  on the diagonal. Then  $A = \sum_i \lambda_i u_i u_i^T$ , where  $u_1, \dots, u_n$  are the eigenvectors of  $A$ . The projection of  $A$  onto  $u_i u_i^T$  is  $\langle A, u_i u_i^T \rangle$  thus

$$\begin{aligned} \langle A, u_i u_i^T \rangle &= \left\langle \sum_{j=1}^n \lambda_j u_j u_j^T, u_i u_i^T \right\rangle \\ &= \sum_{j=1}^n \langle \lambda_j u_j u_j^T, u_i u_i^T \rangle \\ &= \sum_{j=1}^n \lambda_j \langle u_j u_j^T, u_i u_i^T \rangle \\ &= \lambda_i \langle u_i u_i^T, u_i u_i^T \rangle \\ &= \lambda_i \end{aligned}$$

Where we applied linearity of the inner product for equations 2 and 3 and reuse the results of the inner product between eigenvectors from the previous question (assuming we chose eigenvectors orthonormal).

- (e) Provide a geometric interpretation of the matrix  $A' := A - \lambda_1 u_1 u_1^T$ , which we defined in the proof of the spectral theorem, based on your previous answers. From the previous question the orthogonal projection of  $A$  in  $u_i u_i^T$  is  $\lambda_i u_i u_i^T$  so  $A' = \sum_i \lambda_i u_i u_i^T, i \neq 1$  has row or column subspaces contained in  $(u_1)^\perp$ .

3. (Quadratic forms) Let  $A \in \mathbb{R}^{n \times n}$  be a symmetric matrix, and let  $f(x) := x^T A x$  be the corresponding quadratic form. We consider the 1D function  $g_v(t) = f(tv)$  obtained by restricting the quadratic form to lie in the direction of a vector  $v$  with unit  $\ell_2$  norm.
- (a) Is  $g_v(t)$  a polynomial? If so, what kind?  $g_v(t) = f(tv) = (tv)^T A (tv) = t^2 v^T A v = v^T A v t^2$ ,  $v^T A v$  is a scalar, and  $g_v(t)$  is a second-order polynomial in  $t$ .
  - (b) What is the curvature (i.e. the second derivative) of  $g_v(t) = f(tv)$  at an arbitrary point  $t$ ?  $g'_v(t) = 2v^T A v t$  and the curvature is  $g''_v(t) = 2v^T A v$
  - (c) What are the directions of maximum and minimum curvature of the quadratic form? What are the corresponding curvatures equal to? By the spectral theorem,  $A = U \mathbf{diag}(\lambda) U^T$  where **diag** is the diagonal matrix with on the diagonal:  $\lambda_1 \geq \lambda_2 \geq \dots \lambda_n$ , which are the eigenvalues and  $u_1, \dots, u_n$  the corresponding eigenvectors. The largest eigenvalue is  $\lambda_1 = \max_{\|v\|_2=1} v^T A v$  with eigenvector  $u_1 = \arg \max_{\|v\|_2=1} v^T A v$ , and the smaller eigenvalue is given by  $\lambda_n = \min_{\|v\|_2=1} v^T A v$ ,  $u_n = \arg \min_{\|v\|_2=1} v^T A v$ . Thus the maximum curvature is given by the largest eigenvalue  $\lambda_1$  and is in the direction of the corresponding eigenvector  $u_1$ . The smallest curvature is given by the smallest eigenvalue  $\lambda_n$  and is in the direction of the corresponding eigenvector  $u_n$ .

4. (Projected gradient ascent) Projected gradient descent is a method designed to find the maximum of a differentiable function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  in a constraint set  $\mathcal{S}$ . Let  $\mathcal{P}_{\mathcal{S}}$  denote the projection onto  $\mathcal{S}$ , i.e.

$$\mathcal{P}_{\mathcal{S}}(x) := \arg \min_{y \in \mathcal{S}} \|x - y\|_2^2. \quad (2)$$

The  $k$ th update of projected gradient ascent equals

$$x^{[k]} := \mathcal{P}_{\mathcal{S}}(x^{[k-1]} + \alpha \nabla f(x^{[k-1]})), \quad k = 1, 2, \dots, \quad (3)$$

where  $\alpha$  is a positive constant and  $x^{[0]}$  is an arbitrary initial point.

- (a) Use the same arguments we used to prove Lemmas 5.1 and 5.2 in the notes on PCA to derive the projection of a vector  $x$  onto the unit sphere in  $n$  dimensions. Let define  $f(x) = \|x - y\|_2^2, y \in \mathcal{S}$ , the directional derivative cannot be different than zero  $f'_v(x) = \langle \nabla_x f, v \rangle = 0$  for any  $v$  such that  $x + \epsilon v$  is on the sphere  $\mathcal{S}$ . Let  $g(x) = x^T x, \|y\|_2 = 1$ ,  $g$  describes points on the surface of the unit sphere.  $x + \epsilon v$  is in the tangent plane of  $g$  at  $x$  if  $\nabla g(x)^T v = 0$ , and for  $\epsilon \approx 0$ ,  $g(x + \epsilon v) \approx g(x)$ . We are then looking for global minimizer points (global because  $f$  is convex), where the level curves of  $f$  are tangent to the curve  $g$ , or where the gradients are colinear.  $\nabla_x f(x) = \nabla_x (x^T x - 2x^T y + y^T y) = 2(x - y)$  and  $\nabla_x g(x) = 2x$ , thus the projection of  $x$  on  $\mathcal{S}$ ,  $y_p$ , verifies  $x - y_p = \lambda x$  or  $y_p = (1 - \lambda)x$ . for any vector  $y \in \mathcal{S}$ , we have  $y = (1 - \lambda)x + x_{\perp}$  where  $x_{\perp}$  is in the hyperplane orthogonal to  $x$ . We want to show that the projection point is the closest to  $x$ . By Pythagoras' theorem,  $\|y\|_2^2 = (1 - \lambda)^2 \|x\|^2 + \|x_{\perp}\|^2$  and:

$$\begin{aligned} \|y - x\|_2^2 &= \|y\|_2^2 - 2y^T x + \|x\|_2^2 \\ y^T x &= ((1 - \lambda)x^T + x_{\perp}^T)x \\ &= (1 - \lambda)x^T x \Rightarrow \\ \|y - x\|_2^2 &= \|y\|_2^2 - 2(1 - \lambda)\|x\|_2^2 + \|x\|_2^2 \\ &= (1 - \lambda)^2 \|x\|^2 + \|x_{\perp}\|^2 - 2(1 - \lambda)\|x\|_2^2 + \|x\|_2^2 \\ &= \lambda^2 \|x\|_2^2 + \|x_{\perp}\|^2 \\ &> \|x - y_p\|_2^2 \end{aligned}$$

Thus  $\arg \min_{y \in \mathcal{S}} \|x - y\|_2^2 = \arg \min_{\lambda x \in \mathcal{S}} (1 - \lambda)^2 \|x\|_2^2, \lambda x \in \mathcal{S}$ . If  $x \in \mathcal{S}$  then  $\lambda = 1$ , if  $x \notin \mathcal{S}$  and  $\lambda x \in \mathcal{S} \Rightarrow \|\lambda x\|_2 = 1 \Rightarrow \lambda = \frac{1}{\|x\|_2}$ , thus  $\lambda = \min(1, \frac{1}{\|x\|_2})$ , that is  $\mathcal{P}_{\mathcal{S}}(x) = \min(x, \frac{x}{\|x\|_2})$ .

- (b) Derive an algorithm based on projected gradient ascent to find the maximum eigenvalue of a symmetric matrix  $A \in \mathbb{R}^{n \times n}$ . Let  $f(x) = x^T A x$ , the largest eigenvalue can be found by solving the optimization problem  $\lambda_1 = \max_{\|x\|_2=1} x^T A x$  or equivalently  $\lambda_1 = \min_{\|x\|_2=1} -f(x)$ . We have  $\nabla f(x) = 2Ax$ , by assumption and using the previous result, the algorithm to find the largest eigenvalue of a symmetric matrix  $A \in \mathbb{R}^{n \times n}$  is:

$$\begin{aligned}
x'^{[k-1]} &= x^{[k-1]} + \alpha \nabla f(x^{[k-1]}) \\
&= x^{[k-1]} - 2\alpha A x^{[k-1]} \\
x^{[k]} &= \frac{x'^{[k-1]}}{\|x'^{[k-1]}\|_2} \\
&= \frac{(I - 2\alpha A)x^{[k-1]}}{\|(I - 2\alpha A)x^{[k-1]}\|_2} \quad k = 1, \dots
\end{aligned}$$

where  $\alpha$  is a positive constant and  $x^{[0]}$  is an arbitrary initial point. We run this algorithm until either the number maximum of iterations is reached or  $|x^{[k]} - x^{[k-1]}| < \epsilon$ .

- (c) Let us express the iterations in the basis of eigenvectors of  $A$ :  $x^{[k]} := \sum_{i=1}^n \beta_i^{[k]} u_i$ . Compute the ratio between the coefficient corresponding to the largest eigenvalue and the rest  $\frac{\beta_1^{[k]}}{\beta_i^{[k]}}$  as a function of  $k$ ,  $\alpha$ , and  $\beta_1^{[0]}, \dots, \beta_n^{[0]}$  (and also the eigenvalues). Under what conditions on  $\alpha$  and the initial point does the algorithm converge to the eigenvector  $u_1$  corresponding to the largest eigenvalue? What happens if  $\alpha$  is extremely large (i.e. when  $\alpha \rightarrow \infty$ )?

Let  $x^{[0]} = \sum_{i=1}^n \beta_i^{[0]} u_i$ , and  $\lambda_1, \dots, \lambda_n$  the eigenvalues of  $A$ , from the previous question, we have:

$$\begin{aligned}
x^{[k]} &= \frac{(I - 2\alpha A)x^{[k-1]}}{\|(I - 2\alpha A)x^{[k-1]}\|_2} \\
&= \frac{(I - 2\alpha A)^k x^{[0]}}{\|(I - 2\alpha A)^k x^{[0]}\|_2} \\
&= \frac{(I - 2\alpha A)^k \sum_{i=1}^n \beta_i^{[0]} u_i}{\|(I - 2\alpha A)^k \sum_{i=1}^n \beta_i^{[0]} u_i\|_2} \\
&= \frac{\sum_{i=1}^n \beta_i^{[0]} (1 - 2\alpha \lambda_i)^k u_i}{\|\sum_{i=1}^n \beta_i^{[0]} (1 - 2\alpha \lambda_i)^k u_i\|_2} \\
&= \frac{\sum_{i=1}^n \beta_i^{[0]} (1 - 2\alpha \lambda_i)^k u_i}{(\sum_{i=1}^n (\beta_i^{[0]})^2 (1 - 2\alpha \lambda_i)^{2k})^{\frac{1}{2}}}
\end{aligned}$$

This give us:

$$\begin{aligned}
u_j^T x^{[k]} &= \frac{\beta_j^{[0]} (1 - 2\alpha \lambda_j)^k}{(\sum_{i=1}^n (\beta_i^{[0]})^2 (1 - 2\alpha \lambda_i)^{2k})^{\frac{1}{2}}} \\
u_1^T x^{[k]} &= \frac{\beta_1^{[0]} (1 - 2\alpha \lambda_1)^k}{(\sum_{i=1}^n (\beta_i^{[0]})^2 (1 - 2\alpha \lambda_i)^{2k})^{\frac{1}{2}}}
\end{aligned}$$

We want  $|u_1^T x^{[k]}| \rightarrow 1$ . By the spectral theorem,  $\lambda_n \leq \dots \leq \lambda_i \leq \dots \leq \lambda_1 \Rightarrow (1 -$

$2\alpha\lambda_n)^{2k} \geq \dots \geq (1 - 2\alpha\lambda_i)^{2k} \dots \geq (1 - 2\alpha\lambda_1)^{2k}$ , so we have

$$\left| \left( \frac{1 - 2\alpha\lambda_1}{1 - 2\alpha\lambda_n} \right)^k \frac{\beta_1^{[0]}}{(\sum_{i=1}^n (\beta_i^{[0]})^2)^{\frac{1}{2}}} \right| \leq \left| \frac{\beta_1^{[0]} (1 - 2\alpha\lambda_1)^k}{(\sum_{i=1}^n (\beta_i^{[0]})^2 (1 - 2\alpha\lambda_i)^{2k})^{\frac{1}{2}}} \right| \leq \left| \left( \frac{1 - 2\alpha\lambda_1}{1 - 2\alpha\lambda_1} \right)^k \frac{\beta_1^{[0]}}{(\sum_{i=1}^n (\beta_i^{[0]})^2)^{\frac{1}{2}}} \right|$$

$$\left| \frac{1 - 2\alpha\lambda_1}{1 - 2\alpha\lambda_n} \right|^k \left| \frac{\beta_1^{[0]}}{(\sum_{i=1}^n (\beta_i^{[0]})^2)^{\frac{1}{2}}} \right| \leq \left| \frac{\beta_1^{[0]} (1 - 2\alpha\lambda_1)^k}{(\sum_{i=1}^n (\beta_i^{[0]})^2 (1 - 2\alpha\lambda_i)^{2k})^{\frac{1}{2}}} \right| \leq \left| \frac{\beta_1^{[0]}}{(\sum_{i=1}^n (\beta_i^{[0]})^2)^{\frac{1}{2}}} \right|$$

One condition for convergence of the previous inequality will be  $2\alpha|\lambda_1| - 1 < |1 - 2\alpha\lambda_1| < 1$  so  $|1 - 2\alpha\lambda_1|^k \rightarrow 0$  which gives us  $0 < \alpha < \frac{1}{|\lambda_1|}$  and for upper bound we need  $|\beta_1^{[0]}| = |(\sum_{i=1}^n (\beta_i^{[0]})^2)^{\frac{1}{2}}|$ , then  $u_1^T x^{[k]} \rightarrow +1$  or  $-1$ , and  $x^{[k]T} A x^{[k]} \rightarrow +\lambda_1$  or  $-\lambda_1$ .

- (d) Implement the algorithm derived in part (b). Support code is provided in `main.py` within `Q4.zip`. Observe what happens for different sizes of  $\alpha$ . Report the plots generated by the script. As we increase  $\alpha$ , the algorithm converges faster to the maximum eigenvalue and corresponding eigenvector, for  $\alpha = 0.1 < 1$ , there is no convergence. We also observe that the initial point plays a role to the rate of convergence:

```
import os

import matplotlib.pyplot as plt
import numpy as np

def unit_vector(vector):
    """ Returns the unit vector of the vector. """
    return vector / np.linalg.norm(vector)

def angle_between(v1, v2):
    """ Returns the angle in radians between vectors 'v1' and 'v2': """
    v1_u = unit_vector(v1)
    v2_u = unit_vector(v2)
    return np.arccos(np.clip(np.dot(v1_u, v2_u), -1.0, 1.0))

def calc_true_error(x1, x2):
    """ eigenvecs could converge to u or -u - both are valid eigvecs
    The function should output the L2 norm of (x1 - x2)
    If x1 = u and x2 = -u, we still want the function to output 0 error
    return 1 - abs(np.cos(angle_between(x1, x2)))
    # return np.linalg.norm(np.abs(x1) - np.abs(x2))

def eigen_iteration(A, x0, alpha, max_iter=50, thresh=1e-5):
```

```

'''A - nxn symmetric matrix
x0 - np.array of dimension n which is the starting point
alpha - learning rate parameter
max_iter - number of iterations to perform
thresh - threshold for stopping iteration

stopping criteria: can stop when |lambda[k] - lambda[k-1]| <= t

return:
relative_error_eigvec: array with ||x[k] - x[k-1]||_2
true_error_eigvec: array with ||x[k] - u_1||_2 where u_1 is the first eigenvector
relative_error_eigval: array with |lambda[k] - lambda[k-1]|
true_error_eigval: array with |lambda[k] - lambda_1|

x[k] is your estimated max eigenvector at iteration k and lambda[k] is the estimated max eigenvalue at iteration k
lambda_1 is the max eigenvalue of A and u_1 is the corresponding eigenvector'''

assert ((A.transpose() == A).all()) # asserting A is symmetric
assert (A.shape[0] == len(x0))

w, v = np.linalg.eigh(A)
true_lam = w[w.size - 1] # fill in your code to find max eigenvalue
true_u1 = v[:, v.shape[1] - 1] # np array with the first eigenvector
relative_errors_eigvec = list()
true_errors_eigvec = list()
relative_errors_eigval = list()
true_errors_eigval = list()
curr_eigvec = x0.copy()
iteration = 1
while True:
    next_eigv = curr_eigvec + alpha * np.matmul(-2 * A, curr_eigvec)
    next_eigv = unit_vector(next_eigv)

    rel_eigvec_error = np.linalg.norm(next_eigv - curr_eigvec)
    relative_errors_eigvec.append(rel_eigvec_error)
    true_eigvec_error = calc_true_error(true_u1, next_eigv)
    true_errors_eigvec.append(true_eigvec_error)

    eigval_prev = curr_eigvec.T.dot(np.matmul(A, curr_eigvec))
    eigval_next = next_eigv.T.dot(np.matmul(A, next_eigv))
    rel_eigval_error = abs(eigval_next - eigval_prev)
    relative_errors_eigval.append(rel_eigval_error)
    true_eigval_error = abs(true_lam - eigval_next)
    true_errors_eigval.append(true_eigval_error)

```

```

if rel_eigval_error <= thresh:
    print("Convergence in {} iterations, alpha:{},\
        init_point_norm={}".format(iteration, alpha, np.linalg.
    print("True ul:{}, computed ul:{}, rel_error:{}, true_ei
        .format(true_ul, next_eigv, rel_eigvec_error, true
    print("True max.eigenval:{}, computed max_eigval:{}, rel
        .format(true_lam, eigval_next, rel_eigval_error, t
        break
iteration += 1
if iteration >= max_iter:
    print("Maximum iteration exceeded!")
    print("True ul:{}, computed ul:{}, rel_error:{}, true_ei
        .format(true_ul, next_eigv, rel_eigvec_error, true
    print("True max.eigenval:{}, computed max_eigval:{}, rel
        .format(true_lam, eigval_next, rel_eigval_error, t
        break

curr_eigvec = next_eigv

## fill in code to do your projected gradient ascent
## append both the list with the errors

return relative_errors_eigvec, true_errors_eigvec, relative_errors_eigval

```

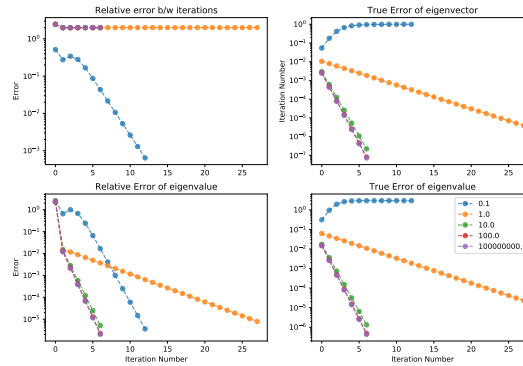


Figure 1: First matrix: relative errors on the left, absolute errors on the right for largest eigenvalue and corresponding eigenvector.



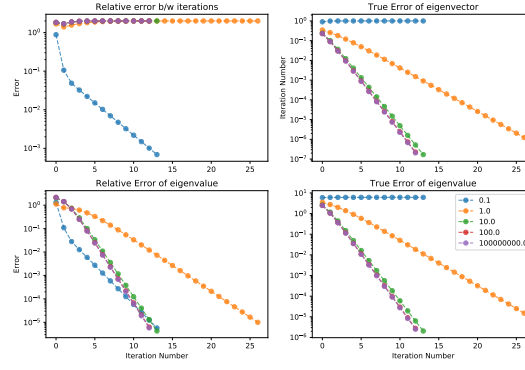


Figure 2: Second matrix: relative errors on the left, absolute errors on the right for largest eigenvalue and corresponding eigenvector.

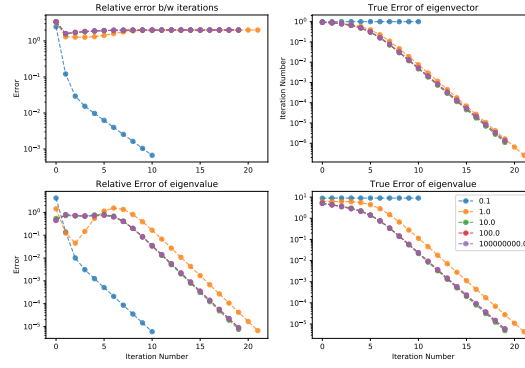


Figure 3: Third matrix: relative errors on the left, absolute errors on the right for largest eigenvalue and corresponding eigenvector.