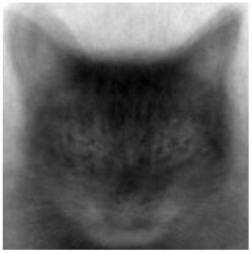
```
% Andrew Garwood
% Written HW 10
clear all; close all; clc;

load('training_images.mat')
[m, n] = size(training_data);
% This will create a 93 × 40, 000 matrix named training_data. The first
% 53 rows are pictures of Boris and the last 40 are pictures of Nandor.

%%

% 1a
h = 200;
w = 200;
% Both cats look nice
% You can look at the picture in row k with the code
% img = reshape(training_data(90, :), [h, w])
% imshow(img, [])

% In my opinion, Nandor is a cooler name, so I chose Nandor. My favorite
% image from the set is image 90. Nandor looks surprised or something.
```

```
% 1b, find average of all images
avg_img = mean(training_data); % 1 x 40,000 double
% figure()
% imshow(reshape(avg_img, [h, w]), [])
```

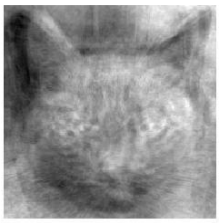


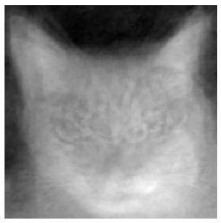
X = training_data - ones(m, 1) * avg_img; % 93x40k double - 93x40k double

% 1c reduced svd
[U, S, V] = svd(X, 'econ');

scores = X*V; % 93x93 double

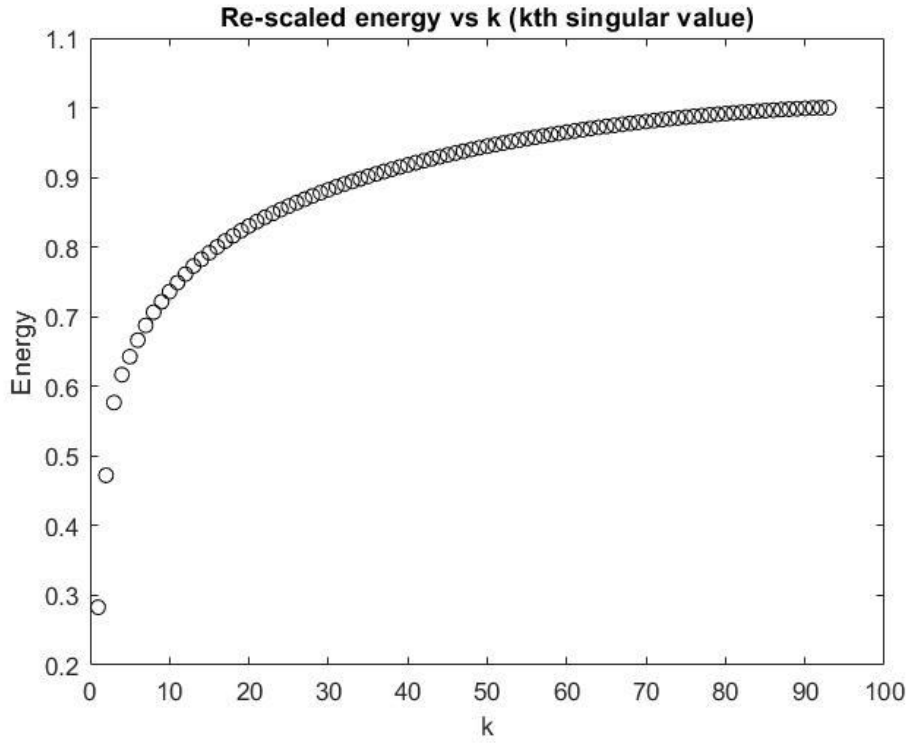
% 1d display first 2 eigenfaces using imshow
% figure()
% for k = 1:2
% img = reshape(V(:, k), [h, w]);
% subplot(1, 2, k)
% imshow(img, [])
% end
%%





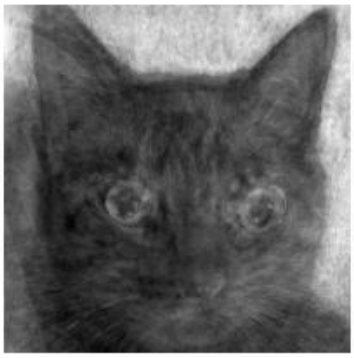
% 1e Calculate re-scaled energies
sigma = diag(S); % put sigma values from S into vector "sigmas"
% cumsum -> cumulative sum of values
E = cumsum(sigma.^2) / sum(sigma.^2); % 93x1 double

```
% plot them
% figure()
% plot(E, 'ko')
% title('Re-scaled energy vs k (kth singular value)')
% xlabel('k')
% ylabel('Energy')
```




```
%%
% 1f
% find smallest k st E(k) > .99

k = length(E);
for i = 1:length(E)
   if (E(i) > .99)
      k = i;
      break;
   end
end
% k = 79

% there are 93 singular values all of which are significantly greater than
% zero except the 93rd singular value (which is equal to
% 9.697858972507550e-12)
```

```
% Using megabyte calculation from hw10:
% original num of mb required to store data set
original_mb = 93 * (1 + m + n) * 8 / 1e6; % = 29.8299

% size for data set compressed to 79 singular values
comp_mb = 79 * (1 + m + n) * 8 / 1e6; % = 25.3394

% compressed mem required / original mem required =   0.8495
% => we end up only saving about 15 percent i.e. 15 percent less space/mb
% required to store the compressed version.
% I do not think this is a meaningful compression, especially if we
% arbitrarily increase the size of the data set. We would end up spending a
% lot of time for little benefit.
% however the SVD is still useful as we can use it to test new data
% and not take up too much space storing the svd


% 1g
% reconstruct favorite pictures using first 40 singular stuffs

% reconstruction of image 90, surprised Nandor
% figure()
% N = 40;
% Uk = U(:, 1:N);
% Sk = diag(sigma(1:N));
% scores_k = Uk * Sk;
% Vk = V(:, 1:N);
% reconstructed_Xk = scores_k * Vk';
% reconstructed_img = reconstructed_Xk(90, :) + avg_img;
% imshow(reshape(reconstructed_img, [h, w]), [])
```



```
%%
% Problem 2
load('testing_images.mat')
% -> testing_data 38x40,000
[m, n] = size(testing_data);

%%
% 2a
Y = testing_data - ones(m, 1) * avg_img;
scores_test = Y * V; % 38x93

% 2b
% identify cat in first test image. get dist between first row of
% scores_test and each of 93 rows of scores

% show first test image: -> Boris
% img = reshape(testing_data(1, :), [h, w])
% imshow(img, [])

score_test_1 = scores_test(1,:); % 1x93

min = norm(score_test_1 - scores(1,:));
scores_index = 1;

%   i = 1:length(scores_test(1,:)) .. likely a better way
for i = 2:93
   ith_norm = norm(score_test_1 - scores(i,:));
   if ith_norm < min
      min = ith_norm;
      scores_index = i;
   end
end
% -> scores index = 46, -> most like 46th image? -> Boris -> guessed
% correctly
% img = reshape(training_data(46, :), [h, w])
% imshow(img, [])

% 2c find percent guessed correctly for Boris
num_correct = 0;
for i = 1:22 % for all 22 test images of Boris

   % get ith row of scores_test
```

```
    ith_score_test = scores_test(i,:);

    % first norm = ith_score_test - first row of scores
    min = norm(ith_score_test - scores(1,:));
    scores_index = 1;
    for j = 2:93
        jth_norm = norm(ith_score_test - scores(j,:));
        if jth_norm < min
            min = jth_norm;
            scores_index = j;
        end
    end

    % check if we guessed correctly i.e. check if boris
    if scores_index <= 53
        num_correct = num_correct + 1;
    end
end
% -> num_corrrect = 22
% From the code above, i got 22/22 correct. This seems wrong
% hmm I guess 100 percent is okay. I mean it could be worse.
% just kidding, 100 percent is good

% 2d
% do the same for Nandor
num_correct = 0;
for i = 23:38 % for all 22 test images of Boris

    % get ith row of scores_test
    ith_score_test = scores_test(i,:);

    % first norm = ith_score_test - first row of scores
    min = norm(ith_score_test - scores(1,:));
    scores_index = 1;
    for j = 2:93
        jth_norm = norm(ith_score_test - scores(j,:));
        if jth_norm < min
            min = jth_norm;
            scores_index = j;
        end
    end

    % check if we guessed correctly i.e. check if nandor
    if scores_index > 53
```

```
        num_correct = num_correct + 1;
    end
end

% -> num_correct = 14 -> 14/16 correct -> 0.8750 success rate
% This is pretty good. More training and testing data would improve success
% rate -> more photos of Nandor please

% this class was fun, thank you reader, sorry for mess
```