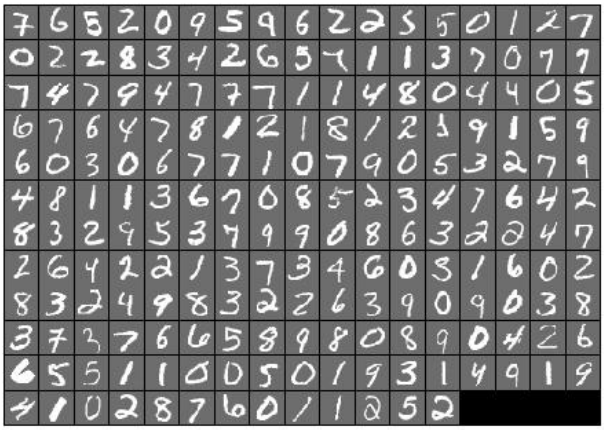
Alex Rosengarten
University of California, San Diego

March 14, 2015 (Pi Day!)

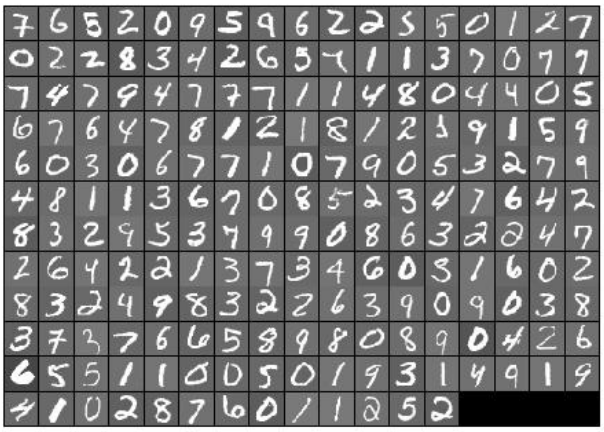
HOMEWORK 4 — LaTeX

Problem 1. PCA Tutorial

(a) Image of randomly selected network and mean subtracted network.

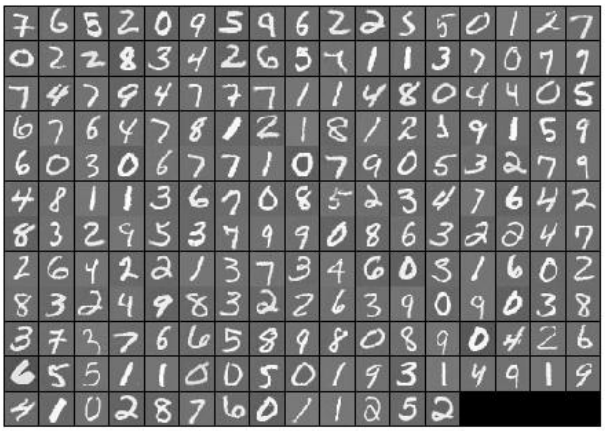


Raw image:

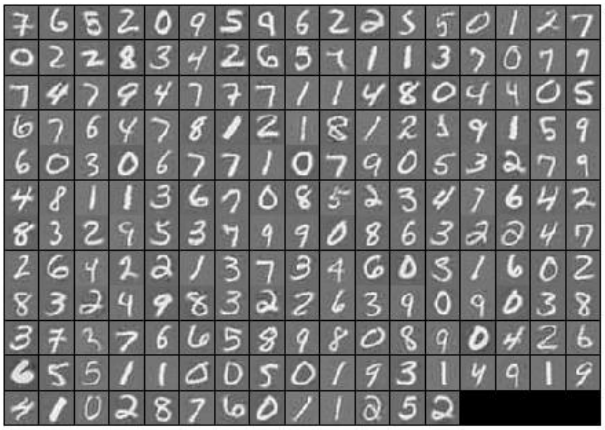


Zero Mean image:

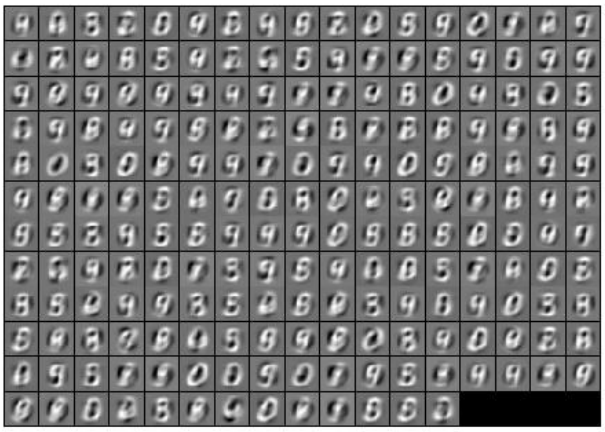
(b) (i) 99% variance retained. Corresponding k: 325



(ii) 90% variance retained. Corresponding k: 86

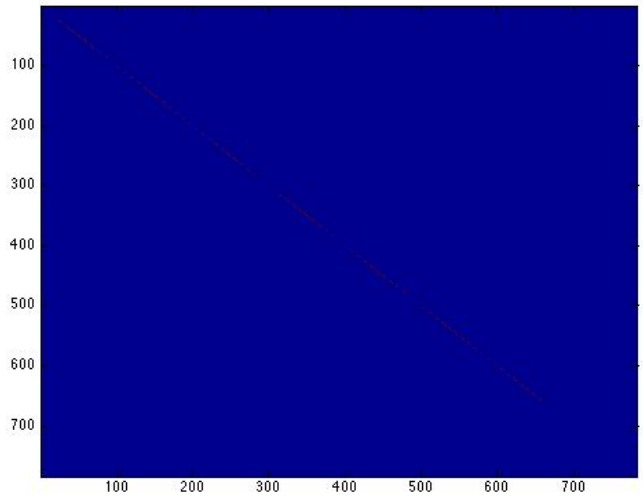


(iii) 30% variance retained. Corresponding k: 5

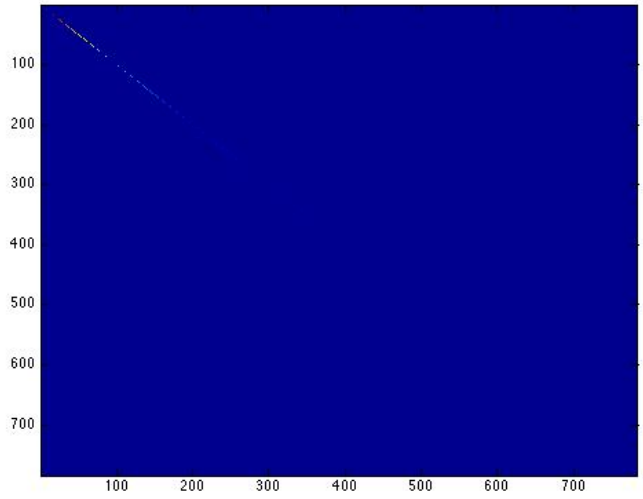


Explain why the appearance of the images changes in the three cases: Most pixels in any given image are highly correlated to the pixels surrounding it, so there is a lot of redundancy of information. PCA will allow us to projects data on to a lower dimensional subspace with minimal information lost. When we choose to retain 99% of the variance, we get a much more precise, sharper image of our characters. At 90% variance, we don't see much difference, but there is a slight blurring of characters. The last image, at 30% variance, is very blurry. Characters are not distinguishable from each other. The reason the images change is because information is lost when not all the principal components are used. When we use more principle components, we retain more information with which we can use to reconstruct the original image. When we use fewer principle components for our projection and rotation, we lose vital information.

(c) (i) Covariance matrix whitened without regularization



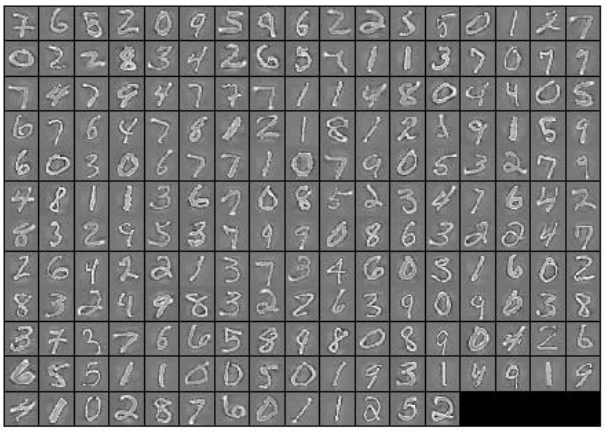
(ii) Covariance matrix whitened with regularization (Epsilon: 0.1)



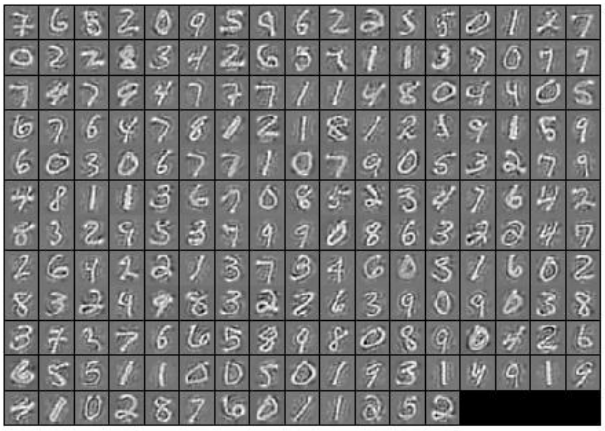
How did the regularization make a difference? Without regularization, the values along the diagonal of the covariance matrix tend near 1. The reason behind this is somewhat intuitive: the eigenvalues of the covariance matrices are measures of the variance of each principal component, so dividing the appropriate element of xRot by the square root of the variance – or the standard deviation – is a way of normalizing the image matrix, ensuring that each of our input features have unit variance. Regularization, on the other hand, makes the values along the diagonal decrease. This happens because epsilon is a constant which inflates the denominator of the regularization expression. The eigenvalues

and eigenvectors decrease proportionally to each other, so as the index i of λ_i increases, the denominator gets much bigger than the numerator.

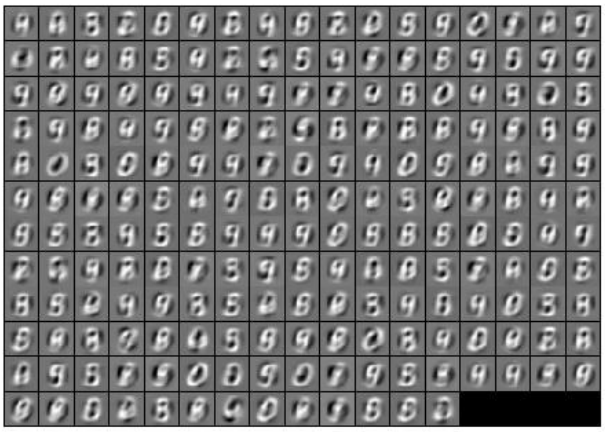
(d) Epsilon used: 0.1 (i) 99% variance retained. Corresponding k:



(ii) 90% variance retained. Corresponding k:



(iii) 30% variance retained. Corresponding k:



Did I observe enhanced edges? Why?

Yes, I definitely observe enhanced edges. Compare each of these figures to the three figures above (b). You'll notice that the edges are more enhanced.

ZCA whitening can be thought of as rotating X to the space of its principal components, dividing each principal component by the square root of the variance in that direction, then rotating back to pixel space. ZCA involves rotating the whitened PCA space while maintaining the property that the data to have a covariance near the identity I. ZCA rotates the matrix to be as close to the original data as possible.

The reason why the observed edges are enhanced, then, is due to the quality that each input feature has unit variance. This makes features less correlated with each other and makes all the features have the same variance.

Problem 2. Appendix

Listing 1: Main File

```
%%================================================================
%% Step 0a: Load data
%  Here we provide the code to load natural image data into x.
%  x will be a 784 * 600000 matrix, where the kth column x(:, k) corresponds to
%  the raw image data from the kth 12x12 image patch sampled.
%  You do not need to change the code below.

x = loadMNISTImages('/t10k-images-idx3-ubyte');
figure('name','Raw images');
randsel = randi(size(x,2),200,1); % A random selection of samples for visualization
display_network(x(:,randsel));

```

```
%%================================================================
%% Step 0b: Zero-mean the data (by row)
%  You can make use of the mean and repmat/bsxfun functions.

x_mean = mean(x, 1);
x_reg = x - repmat(x_mean, size(x,1),1);
% x_reg = bsxfun(@minus, x, repmat(x_mean, size(x,1), 1));
figure('name','Zero-mean image');
display_network(x_reg(:,randsel));
%%================================================================
%% Step 1a: Implement PCA to obtain xRot
%  Implement PCA to obtain xRot, the matrix in which the data is expressed
%  with respect to the eigenbasis of sigma, which is the matrix U.

sigma = cov(x_reg');
[U, S, V] = svd(sigma);
xRot = U'*x_reg;


%%================================================================
%% Step 1b: Check your implementation of PCA
%  The covariance matrix for the data expressed with respect to the basis U
%  should be a diagonal matrix with non-zero entries only along the main
%  diagonal. We will verify this here.
%  Write code to compute the covariance matrix, covar.
%  When visualised as an image, you should see a straight line across the
%  diagonal (non-zero entries) against a blue background (zero entries).

% xRot = bsxfun(@minus, xRot, repmat(mean(xRot,1), size(xRot,1), 1));
covar = cov(xRot');

% Visualise the covariance matrix. You should see a line across the
% diagonal against a blue background.
figure('name','Visualisation of covariance matrix');
imagesc(covar);

%%================================================================
%% Step 2: Find k, the number of components to retain
%  Write code to determine k, the number of components to retain in order
%  to retain at least 99% of the variance.


N = size(xRot,1);

var_thresh_99 = .99;
var_thresh_90 = .90;
```

```
59  var_thresh_30 = .3;
60
61  k_99 = 0;
62  k_90 = 0;
63  k_30 = 0;
64
65  gate30 = 1;
66  gate90 = 1;
67  gate99 = 1;
68  for k = 1:N
69      var_retained = sum(diag(S(:,1:k)))/sum(diag(S))
70      if(var_retained >= var_thresh_99)
71          if(gate99)
72              k_99 = k;
73              gate99=0;
74              break;
75          end
76
77      elseif(var_retained >= var_thresh_90)
78          if(gate90)
79              k_90 = k;
80              gate90 = 0;
81          end
82      elseif(var_retained >= var_thresh_30)
83          if(gate30)
84              k_30 = k
85              gate30 = 0;
86          end
87      end
88
89  end
90  [k_99, k_90, k_30]
91
92  %%=========================================================
93  %% Step 3: Implement PCA with dimension reduction
94  %  Now that you have found k, you can reduce the dimension of the data by
95  %  discarding the remaining dimensions. In this way, you can represent the
96  %  data in k dimensions instead of the original 144, which will save you
97  %  computational time when running learning algorithms on the reduced
98  %  representation.
99  %
100 %  Following the dimension reduction, invert the PCA transformation to produce
101 %  the matrix xHat, the dimension-reduced data with respect to the original basis.
102 %  Visualise the data and compare it to the raw data. You will observe that
103 %  there is little loss due to throwing away the principal components that
104 %  correspond to dimensions with low variation.
```

```
105
106 xTilde_99 = U(:,1:k_99)'*x_reg;
107 xHat_99 = U(:,1:k_99)*xTilde_99;
108
109 xTilde_90 = U(:,1:k_90)'*x_reg;
110 xHat_90 = U(:,1:k_90)*xTilde_90;
111
112 xTilde_30 = U(:,1:k_30)'*x_reg;
113 xHat_30 = U(:,1:k_30)*xTilde_30;
114 % Visualise the data, and compare it to the raw data
115 % You should observe that the raw and processed data are of comparable quality.
116 % For comparison, you may wish to generate a PCA reduced image which
117 % retains only 90% of the variance.
118
119 figure('name',['PCA processed images ',sprintf('(%d / %d dimensions)', k_99, size(x, 1
120 display_network(xHat_99(:,randsel));
121 figure('name',['PCA processed images ',sprintf('(%d / %d dimensions)', k_90, size(x, 1
122 display_network(xHat_90(:,randsel));
123 figure('name',['PCA processed images ',sprintf('(%d / %d dimensions)', k_30, size(x, 1
124 display_network(xHat_30(:,randsel));
125 figure('name','Raw images');
126 display_network(x(:,randsel));
127
128 %%=========================================================================
129 %% Step 4a: Implement PCA with whitening and regularisation
130 %  Implement PCA with whitening and regularisation to produce the matrix
131 %  xPCAWhite.
132
133 epsilon = 1e-1;
134
135 xPCAwhite_reg = diag(1./sqrt(diag(S) + epsilon)) * U' * x_reg;
136 xPCAwhite_noreg = diag(1./sqrt(diag(S))) * U' * x_reg;
137
138 %% Step 4b: Check your implementation of PCA whitening
139 %  Check your implementation of PCA whitening with and without regularisation.
140 %  PCA whitening without regularisation results a covariance matrix
141 %  that is equal to the identity matrix. PCA whitening with regularisation
142 %  results in a covariance matrix with diagonal entries starting close to
143 %  1 and gradually becoming smaller. We will verify these properties here.
144 %  Write code to compute the covariance matrix, covar.
145
146 covar1 = cov(xPCAwhite_reg');
147 covar2 = cov(xPCAwhite_noreg');
148 %  Without regularisation (set epsilon to 0 or close to 0),
149 %  when visualised as an image, you should see a red line across the
150 %  diagonal (one entries) against a blue background (zero entries).
```

```
% With regularisation, you should see a red line that slowly turns
% blue across the diagonal, corresponding to the one entries slowly
% becoming smaller.

% Visualise the covariance matrix. You should see a red line across the
% diagonal against a blue background.
figure('name','Visualisation of covariance matrix: Regularized PCA whitening');
imagesc(covar1);
figure('name','Visualisation of covariance matrix: PCA whitening w/o Regularization');
imagesc(covar2);

%%================================================================
%% Step 5: Implement ZCA whitening
%  Now implement ZCA whitening to produce the matrix xZCAWhite.
%  Visualise the data and compare it to the raw data. You should observe
%  that whitening results in, among other things, enhanced edges.

xZCAWhite = U * diag(1./sqrt(diag(S) + epsilon)) * U' * x_reg;

xZCAWhite_99 = U(:,1:k_99) * diag(1./sqrt(diag(S(:,1:k_99) + epsilon))) * U(:,1:k_99)';
xZCAWhite_90 = U(:,1:k_90) * diag(1./sqrt(diag(S(:,1:k_90) + epsilon))) * U(:,1:k_90)';
xZCAWhite_30 = U(:,1:k_30) * diag(1./sqrt(diag(S(:,1:k_30) + epsilon))) * U(:,1:k_30)';


% Visualise the data, and compare it to the raw data.
% You should observe that the whitened images have enhanced edges.
figure('name','ZCA whitened images');
display_network(xZCAWhite(:,randsel));
figure('name','ZCA whitened images: k_99');
display_network(xZCAWhite_99(:,randsel));
figure('name','ZCA whitened images: k_90');
display_network(xZCAWhite_90(:,randsel));
figure('name','ZCA whitened images: k_30');
display_network(xZCAWhite_30(:,randsel));
figure('name','Raw images');
display_network(x(:,randsel));
```

Submitted by Alex Rosengarten on March 14, 2015 (Pi Day!).