```
%SOFTSVM    Learns an approximately separating hyperplane for the provided data.
% [w, b, xi] = softsvm( X, l, gamma )
%
% Input:
% X : D x N matrix of data points
% l : N x 1 vector with class labels (+/- 1)
% gamma : scalar slack variable penalty
%
% Output:
% w : D x 1 vector normal to the separating hyperplane
% b : scalar offset
% xi : N x 1 vector of slack variables
%
% classify data using sign( X'*w + b )

function [w, b, xi, a] = softsvm( X, l, gamma )

[D,N] = size(X);

% construct H, f, A, b, and lb

H = spdiags([zeros(N,1); ones(D,1); 0], 0, N+D+1, N+D+1);

f = sparse([gamma*ones(N,1); zeros(D+1,1)]);

A = -sparse([speye(N), l.*X', l.*1]);

b = -ones(N,1);

lb = sparse([zeros(N,1); -Inf(D+1,1)]);

%lambda is KKT multipliers a and mu (notation from PRML textbook)
[x,~,~,~,lambda] = quadprog(H, f, A, b, [], [], lb);

% distribute components of x into w, b, and xi:
w = x(N+(1:D));
b = x(end);
xi = x(1:N);

%extract KKT multiplers for data (x_n support vector <=> a_n > 0)
a = lambda.ineqlin;
end
```

Contents

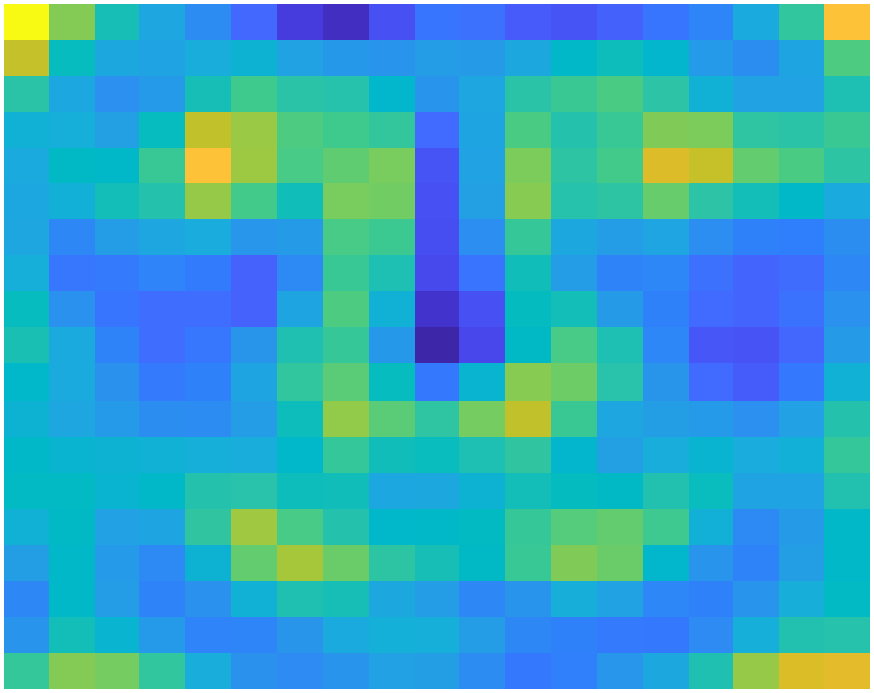
3.2.1 load CBCL and train SVM classifier

```
load 'cbcl.mat'

gamma = 0.005;
[w, b, xi, a] = softsvm(X,L,gamma);
N = length(xi);
```

```
Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.
```

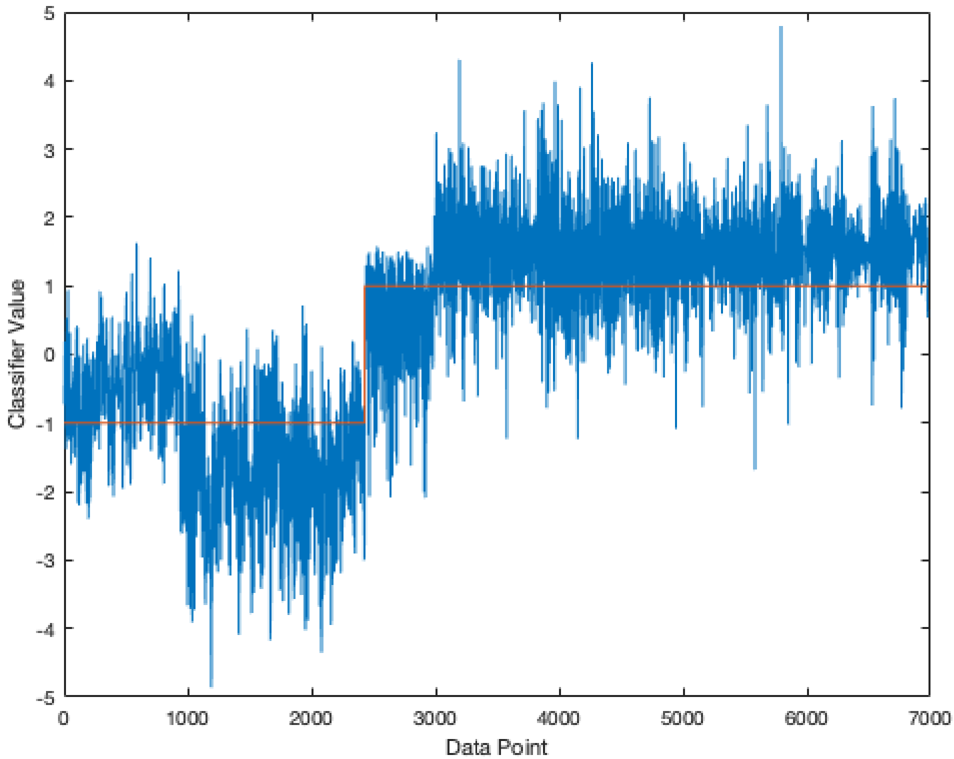
3.2.2 visualize the w orthogonal to decision boundary

```
%The interpretation is to recall that the decision boundary is orthogonal
%to w, meaning w captures the direction where the features changes the most
%between the 2 classes. Think if it like this: as you increase/decrease the
%linear combination of w in some data point, you will most quickly move from
%one class to the other, according to SVM prediction.

%Here, the picture of w captures some important features of face, including
%the eyes, the outline of nose, as well as the lips (smile?). This suggests
%that the largest decider of whether a data point in class A or class B
%(face/random image) is strength of eyes, nose, lips pattern.

%NOTE: data with tag +1 is objects, and -1 is faces

figure;
imagesc(reshape(w, dims));
axis off;
set(gcf,'Color','w');
```



3.3 Plot predictions y(x) = w'*X+b and labels L

```
%Observation: The extremes of this plot represents datapoints that have the
%highest energy y(x_n) in magnitude, i.e. data points which the classifier
%most strongly recognizes as one class or the other. We note that most of
%the extreme data points x_n have sign y(x_n) same as sign of corresponding
%label L_n (marked by orange line), which is good since it means SVM is
%correctly classifying pictures its most confident at.

%There are some misclassified points and we can identify those by the points
%crossing the decision boundary (y = 0 on plot) towards the sign opposite to
%the sign which the label (orange line) at that point belongs to. For example
%there is a decent chunk of misclassified points between x=2500 and x=3000,
%and a few between x = 0 and x = 1000.

figure
plot(w'*X+b); hold on;
plot(L); hold off;
xlabel('Data Point');
ylabel('Classifier Value');
```



3.3.2 misclassifaction rate (extra)

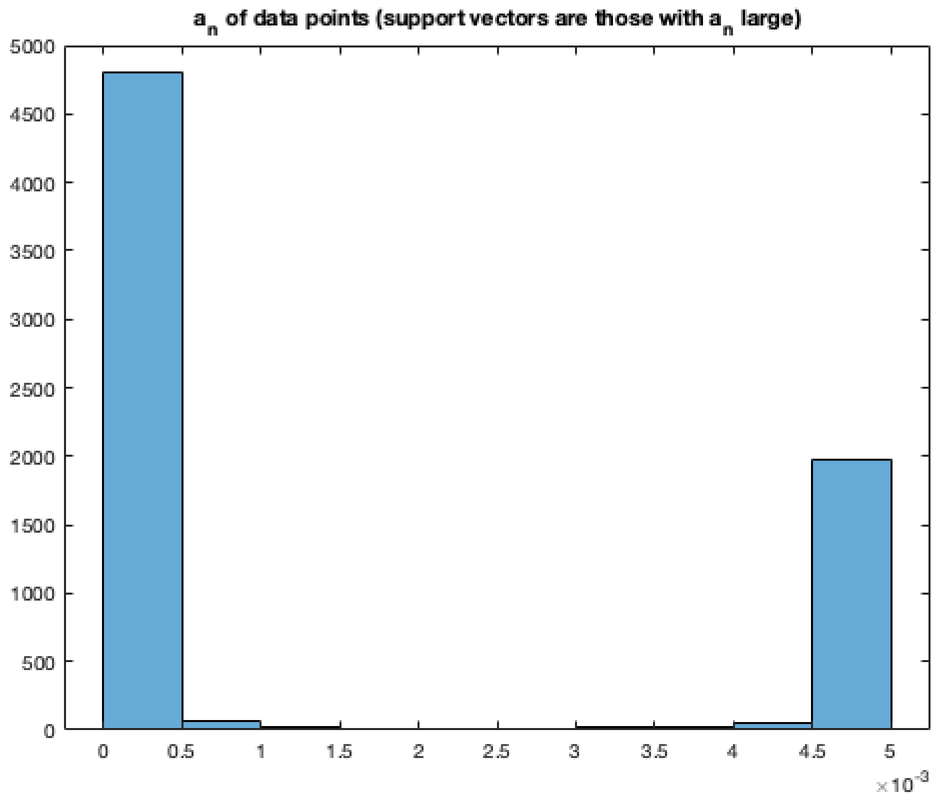
```
%95% accuracy is pretty good
STR = L.*(X'*w+b);
nC = length(STR(STR > 0));
err = (N-nC)/N;
fprintf('misclassification rate is: %d \n', err);
```

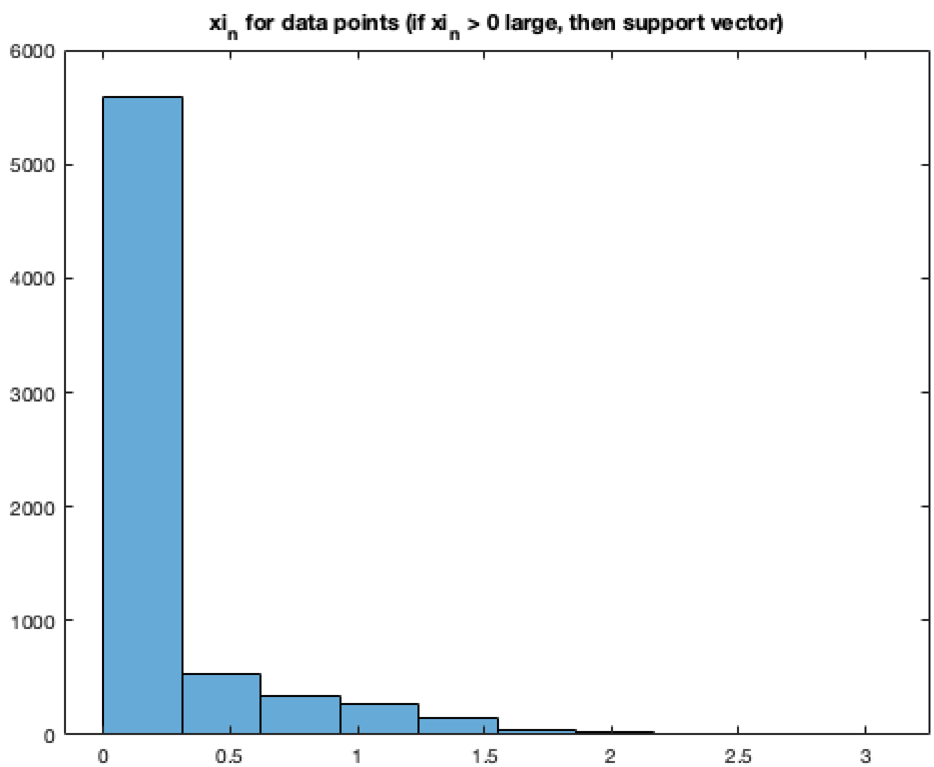
```
misclassification rate is: 6.206106e-02
```

3.4 Support vectors

```
%In quadprog optimization I also stored the lagrange (KKT) multipliers in
%variable a, which I can now examine. Support vectors are those with nonzero
%a_n, but due to numerical reasons non-support vectors may have very small
%non-zero a_n, which we can ignore. (they have minimal effect to prediction
%anyways since y(x) = sum[a_n*t_n*K(x,xn)]+b)

%An interesting thing to note is most support vectors have a_n = 0.005, and
%theoretically its expected since in the KKT condition we can derive the
%restriction a_n between 0 and gamma. a_n = gamma (0.005)  also means mu_n = 0
%and so constraint xi_n ≥ 0 is inactive. This means it's quite likely that
%some number of support vectors have some slack xi_n > 0.

figure;
histogram(a);
title('a_n of data points (support vectors are those with a_n large)');

%The relation between whether a_n is support vector and if xi_n=0 is clear.
%If xi_n > 0 is not small, then a_n mut be a support vector since if a_n ~ 0
```

```
%then mu_n > 0 so xi_n ~ 0. Again no exact equality for numerical reasons.

%But on the other hand it is not easy to answer the question of how likely
%are points s.t. xi_n ~ 0 and a_n > 0 (i.e. with small slack but are still
%support vectors)?

%So without cheating and looking at the KKT multiplier a, we can definitely
%say that the points with xi_n > 0 must be support vectors, and non-support
%vectors can still have xi_n nonzero, just very small.

figure;
histogram(xi,10);
title('xi_n for data points (if xi_n > 0 large, then support vector)');

%Comparing the histograms we note ~2500 points have a_n = 0.005, but only
%~1500 points have xi_n > 0.3, meaning there should be quite a number of
%support vectors with very small xi_n (lie on margin).
```



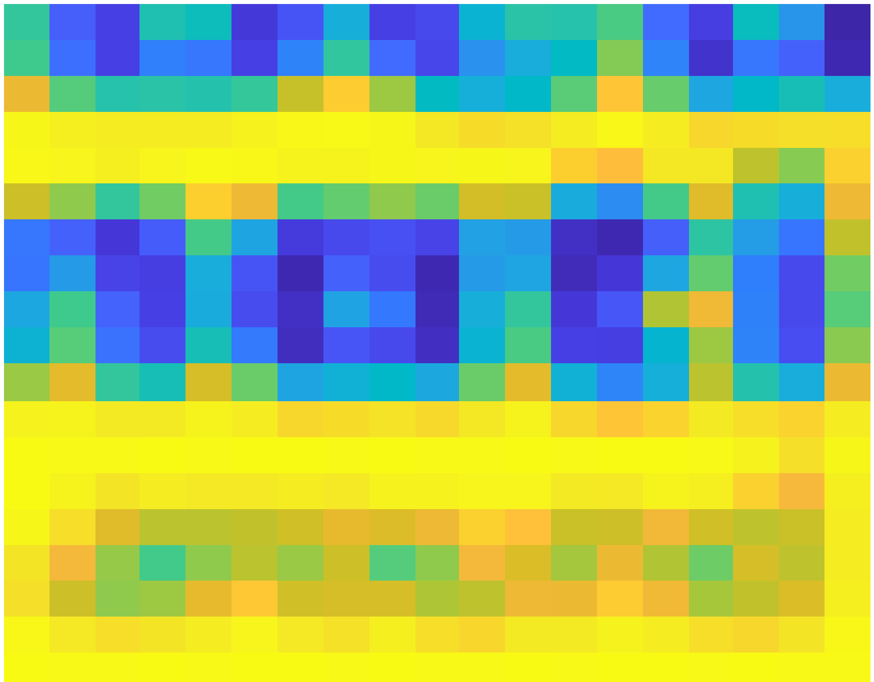


3.5 data with strongest & worst classification prediction

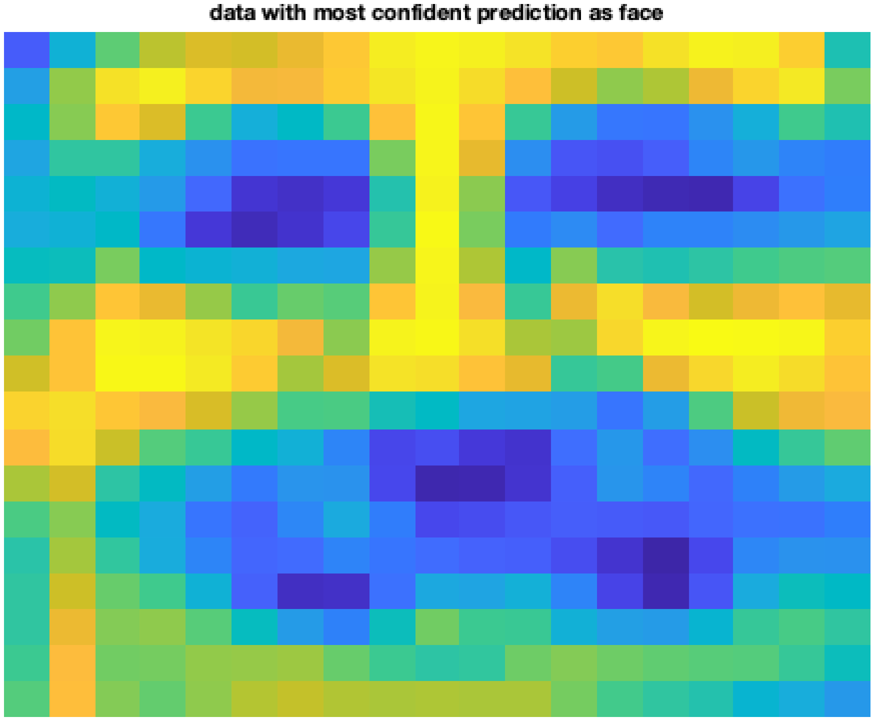
```
%we first pick 2 points with most extreme y(x_n) (one most positive and the
%other most negative)
```

```
%point with largest y(x_n) > 0 (predict most strongly as random object)
[~,i] = max(w'*X+b);
figure;
imagesc(reshape(X(:,i), dims));
axis off;
set(gcf,'Color','w');
title('data with most confident prediction as random object');

%point with smallest y(x_n) < 0 (predict most strongly as face)
[~,i] = min(w'*X+b);
figure;
imagesc(reshape(X(:,i), dims));
axis off;
set(gcf,'Color','w');
title('data with most confident prediction as face');


%then pick 2 data pts that are support vector, specifically largest xi_n,
%s.t. label = 1 for one and label = -1 for other. These are points which
%SVM did the worst performance on.

%random object (a support vector) with maximum xi_n misclassified as face
[~,i] = max(xi .* (L == 1));
figure;
imagesc(reshape(X(:,i), dims));
axis off;
set(gcf,'Color','w');
title('random object misclassified most strongly as face');

%face (a support vector) with maximum xi_n misclassified as random object

[~,i] = max(xi .* (L == -1));
figure;
imagesc(reshape(X(:,i), dims));
axis off;
set(gcf,'Color','w');
title('face misclassified most strongly as random object');

%What we observed is sort of expected. The data point with y(x_n) largest
%looks very like a random object, and data point with y(x_n) smallest look
%like a face with very strong facial features.

%The support vectors with largest xi_n also made sense. The one misclassified
%as random object but actually is a face did not have strong facial features
%based on colouring and the one misclassified as face but actually is random
%object had dark dents like eyes which caused confusion.
```

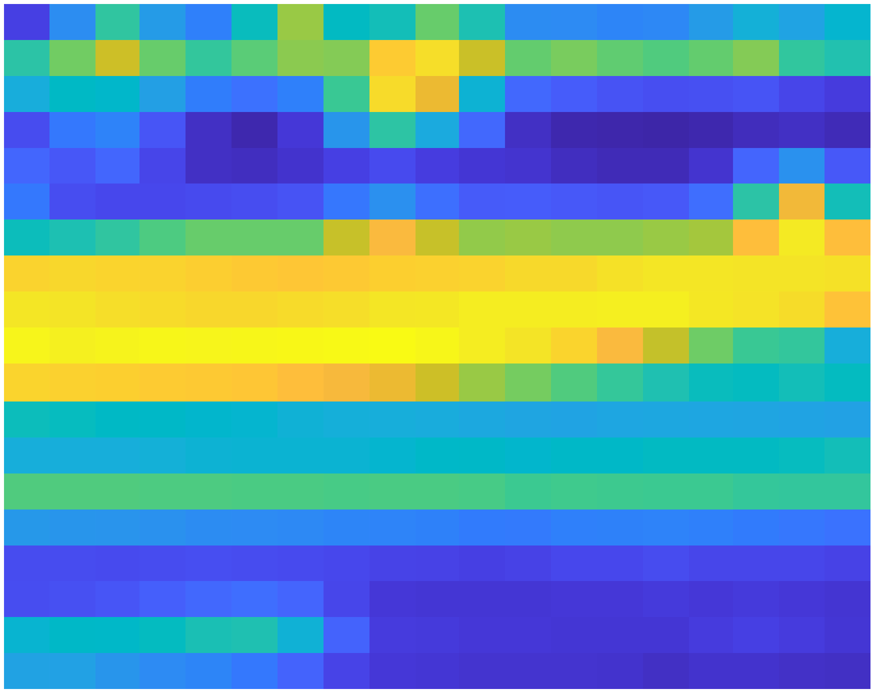
data with most confident prediction as random object



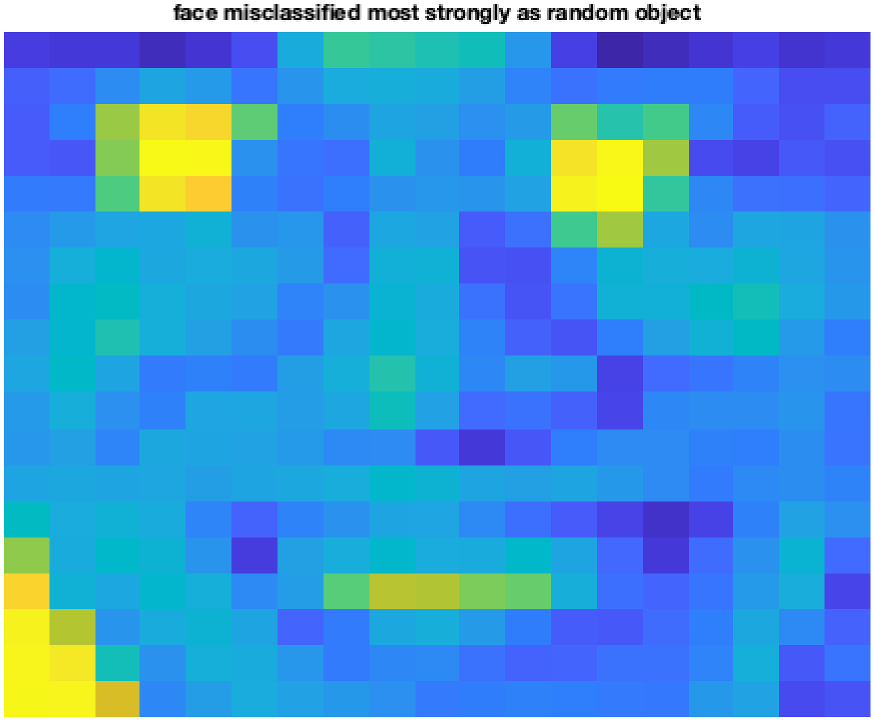
data with most confident prediction as face



random object misclassified most strongly as face



face misclassified most strongly as random object



3.6 Newsgroup data

```
load 'news.mat'
gamma = 0.005;
```

```
[w, b, xi, ~] = softsvm(X,L,gamma);
N = length(xi);
```

```
Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.
```

3.7.1 Important words, large in w

```
%w is a vector orthogonal to decision boundary, i.e. moving along w will
%cause largest change of the features of two classes. It also contains
%relatively symmetric negative/positive values

%So by sorting elements of w from small to large and extracting extreme w_n
%from both ends of sorted w, these extreme words frequency decrease/increase
%most rapidly as we move from articles of one class to the other.

%The elements in the middle after sort changes least, so they are probably
%words that are common to both types of articles
[~, idx] = sort(w);
fprintf('min value in w: %d \nmax value in w: %d \n', w(idx(1)), w(idx(end)));
```

(we are interested in the most positive/negative w_n since those are the feature words that most uniquely appear in one class and not the other)

```
%Observations: As expected, some words that are most distinct in encryption
%related articles are 'encryption', 'clipper', 'security', 'nsa', 'code' etc.
%and other words most distinct in space are 'orbit', 'launch', 'science','nasa',
%'moon', etc. but there's also some unexpected ones like 'steve', 'pat' which
%we cannot understand why are important features.

disp('--- strongest crypto ---');
disp(dict(idx(1:10),:));

disp('--- strongest space ---');
disp(dict(idx(end-9:end),:));
```

```
min value in w: -2.122646e-01
max value in w: 2.216006e-01
--- strongest crypto ---
clipper
encryption
key
chip
steve
security
code
your
na
nsa
--- strongest space ---
dc
nasa
sky
launch
prb
science
```

```
orbit
pat
moon
space
```

3.8.1 misclassification rate for gamma = 0.005

```
%even if gamma = 0.005, misclassification rate is < 1%, very good.
STR = L.*(X'*w+b);
nC = length(STR(STR > 0));
err = (N-nC)/N;
fprintf('misclassification rate is: %d \n', err);
```

```
misclassification rate is: 3.333333e-03
```

3.8.2 linearly separable? Use high gamma to test

```
load 'news.mat'
gamma = 100;
[w, b, xi, ~] = softsvm(X,L,gamma);
N = length(xi);

%Misclassification rate is 0! This is because large gamma penalizes any form
%of misclassification so high priority in correct classification and almost
%no extra margin maximization/generalization. As a result, minimization scheme
%prioritize finding decision boundary minimizing misclassification, which help
%reveal that data clouds is linearly separable.
STR = L.*(X'*w+b);
nC = length(STR(STR > 0));
err = (N-nC)/N;
fprintf('misclassification rate is: %d \n', err);

%Note, I tried doing this for cbcl dataset but failed to reduce misclassification
%rate to 0, meaning data not linearly separable there.
```

```
Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

misclassification rate is: 0
```