Intro

Exercise sheet for Philipp Hanslovsky and Robert Walecki. Added source code for classifier, calculate distance with loops, calculate distance with vectorization as well as standard output of matlab after the answers to the questions.

1b)

Digit 2932:	b)
Digit 814:	c)
Number of digits with label 5 in training dataset:	542

Table 1: Exploring the data

2a)

see source code distance matrix calculated with loops

2b)

see source code distance matrix calculated using vectorization

Computation time	
Loop...	165.3483s
Vector...	3.2498s

Table 2: Computation time using for loops and vectorization respectively, also compare standard output

2c)

see source code classifier as well as table 3 for classification rates for k=5

2d)

For even values of k the classification rate (true positive) is not as good as for nearby odd values of k due to possible ties in counting the lables five/not-five. Therefore k should always be odd valued. For larger k the true positives and positives are the same, however the number of false negatives increases. This is due to the clustering in featurespace. Also see standard output

k	5
classified as 5	86
true positive	83
false positive	3
classified as not 5	914
true negative	910
false negative	4

Table 3: Classification Rates, also compare standard output

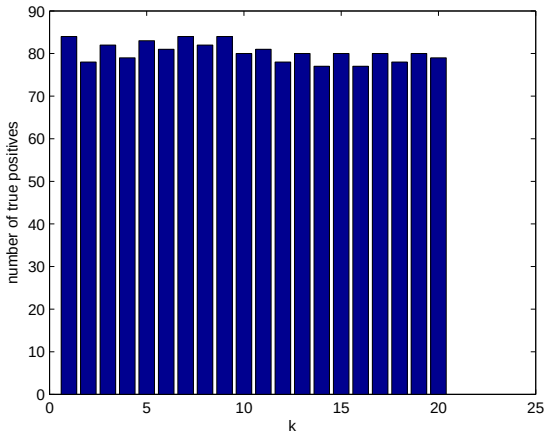

Figure 1: x - axis: k, y - axis: true positive

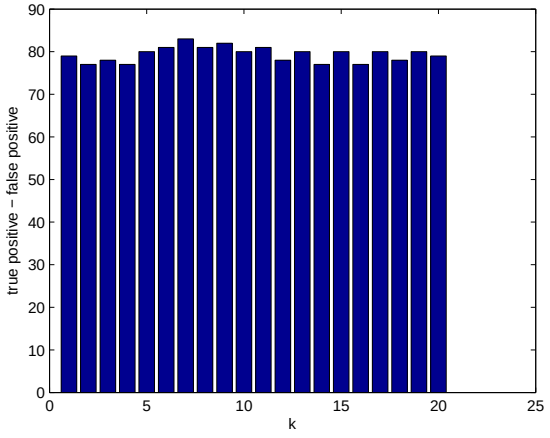

Figure 2: x - axis: k, y -axis: true positive - false positive

3

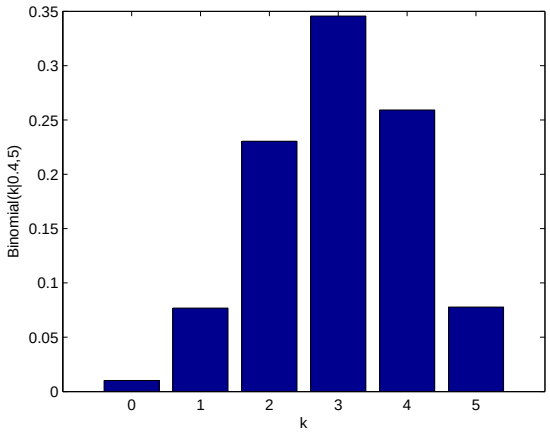

Figure 3: Probability Mass Function for k = 5

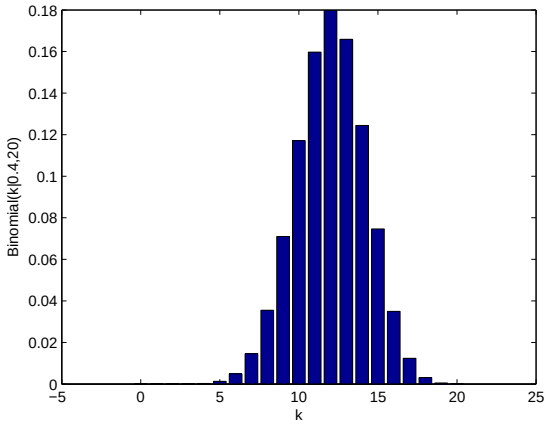

Figure 4: Probability Mass Function for k = 20

Assumg there is no clustering in feature space, k = 20 yields better results. However, there's clustering and therefore choosing too many neighbours can reduce classification rate. In any case an odd value for k should be chosen to avoid ties.

ex01.m

```
clear; close all; clc

fprintf('loading data\n');
load('mnist-digits.mat');

% number of sample points
n1 = size(training, 1);
n2 = size(test, 1);


fprintf(['\nProblem 1b)\n' ...
         'Digit #2932: b)\n' ...
         'Digit # 814: c)\n' ...
         '# of digits with label 5 in training dataset: 542\n\n']);



fprintf('compute distance matrix using loops\n');
%time it

tic
dist_loop=calc_dist_mat_loop_a_b(training, test);
time_loop=toc;

fprintf('Program paused. Press enter to continue.\n');
pause

fprintf('compute distance matrix using vectors\n');
%time it
tic
dist_squ=calc_dist_mat_squ_a_b(training, test);
time_squ=toc;

fprintf(['Computation time\n' ...
         'Loop...    ' num2str(time_loop) 's\n' ...
         'Vector...  ' num2str(time_squ) 's\n\n']);

% check if both functions return the same result
assert(all(all(dist_loop == dist_squ)));
fprintf('Program paused. Press enter to continue.\n');
pause
```

```
fprintf('use distance matrix to compute kNN for various values of k\n\n');
%%%% your code here %%%%


% store maximum number of true positives and the corresponding k
true_positive_max   = 0;
true_positive_max_k = 0;
false_positive_min  = n2;


% get number of actual digits with label 5 in test
actual_fives = size(test_label(test_label == 5), 1);


% sort neighbours by distance
[dist_sort,indices]=sort(dist_squ);


% repeat training_label so each sample of test can
% look up the labels of k nearest neighbours in the
% training dataset
training_label_rep = repmat(training_label, 1, n2);

% set everything != 5 to zero
training_label_rep(training_label_rep ~= 5) = 0;

% histo for true positive and true positve - false positive
true_positive_histo = zeros(20, 1);
true_positive_false_positive_histo = zeros(20, 1);

for k=1:20




    % sum up the labels of the k nearest neighbours for
    % each sample of test
    % if the sum is larger than 5*k/2, sample is labelled 5

    % if k == 1 test_label_classifier is a vector
    % therefore summation would lead to a wrong result
```

```
    if k == 1
        test_label_classifier=training_label_rep(indices(1:k, :));
    else
        test_label_classifier=sum(training_label_rep(indices(1:k, :)));
    end

    indices_fives = test_label_classifier > 5*k/2;
    test_label_classifier = 5*indices_fives;

    % get number of digits labelled 5 and true positive
    % classifications for test
    size_fives = size(test_label_classifier(indices_fives), 2);
    indices_true = test_label_classifier(indices_fives) == (test_label(indices_fives'))';
    size_fives_true = size(test_label_classifier(indices_true), 2);
    size_fives_false = size_fives - size_fives_true;

    true_positive_histo(k) = size_fives_true;
    true_positive_false_positive_histo(k) = size_fives_true - size_fives_false;




    % print results to standard output
    fprintf(['\nConsidering ' int2str(k) ' nearest neighbours.\n' ...
             'Number of digits classified as 5...      ' ...
              int2str(size_fives) '\n' ...
             'True positive classifications...         ' ...
              int2str(size_fives_true) '\n' ...
             'Actual number of digits labelled 5...    ' ...
              int2str(actual_fives) '\n' ...
             'True positive/actual number...           ' ...
              num2str(size_fives_true/actual_fives) '\n\n'])


    % criterion 1: maximize true_positive
    % criterion 2: minimize false positives

    if size_fives_true >= true_positive_max && ...
      size_fives - size_fives_true < false_positive_min
        true_positive_max   = size_fives_true;
        false_positive_min     = size_fives - size_fives_true;
        true_positive_max_k = k;

    end

end
```

```
fprintf(['\n' ...
         'Positives...                                    ' ...
          int2str(false_positive_min + true_positive_max) '\n' ...
         'Max true positives...                           ' ...
          int2str(true_positive_max) '\n' ...
         'Number of neighbours taken into account...     ' ...
          int2str(true_positive_max_k) '\n' ...
         'True positive/actual number...                  ' ...
          num2str(true_positive_max/actual_fives) '\n\n'])


bar(true_positive_histo);
xlabel('k')
ylabel('number of true positives')
fprintf('\n\nProgram paused. Press enter to continue.\n');
pause

figure
bar(true_positive_false_positive_histo);
xlabel('k')
ylabel('true positive - false positive')


%%%%%%%%%%%%%%%%%%%%%%%%%

fprintf('\n\nProgram paused. Press enter to continue.\n');
pause


% bonus:
%%%% your code here %%%%

% binomial distributions for 5 and 20 neighbours respectively
hist_5  = factorial(5)./(factorial(0:5).*factorial(-(0:5) + 5)) ...
          .*0.6.^(0:5).*0.4.^(-(0:5) + 5);
hist_20 = factorial(20)./(factorial(0:20).*factorial(-(0:20) + 20)) ...
          .*0.6.^(0:20).*0.4.^(-(0:20) + 20);

figure
bar(0:5, hist_5)
xlabel('k')
ylabel('Binomial(k|0.4,5)')

fprintf('\n\nProgram paused. Press enter to continue.\n');
pause
```

```
figure
bar(0:20, hist_20)
xlabel('k')
ylabel('Binomial(k|0.4,20)')


fprintf('\n\nProgram paused. Press enter to continue.\n');
pause

p_predict_not5_k5  = sum(hist_5(1:3));
p_predict_not5_k20 = sum(hist_20(1:11));

fprintf(['probability of false prediction\n' ...
         'k =  5:      ' num2str(p_predict_not5_k5) '\n' ...
         'k = 20:      ' num2str(p_predict_not5_k20) '\n' ])


%%%%%%%%%%%%%%%%%%%%%%%%
```

loop

```
function d = calc_dist_mat_loop_a_b(x1, x2)
% computes squared distance matrix between two sets of points with p dimensions
% and n1 and n2 points, respectively.
% Points are stored in an (n1 by p)-matrix x1 and
% (n2 by p)-matrix x2
% Output is a (n1 by n2) matrix of squared distances
x1 = double(x1);
x2 = double(x2);

n1 = size(x1, 1);
n2 = size(x2, 1);
p  = size(x1, 2);

d = zeros(n1,n2);

%%%% your code here %%%%



for index1 = 1:n1
    for index2 = 1:n2
        for featureIndex = 1:p
            d(index1, index2) = d(index1, index2) + (x1(index1, featureIndex) - x2(index2, f
        end
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%
```

squ

```
function d = calc_dist_mat_squ_a_b(x1, x2)
% computes squared distance matrix between two sets of points with p dimensions
% and n1 and n2 points, respectively.
% Points are stored in an (n1 by p)-matrix x1 and
% (n2 by p)-matrix x2
% Output is a (n1 by n2) matrix of squared distances
x1 = double(x1);
x2 = double(x2);

n1 = size(x1, 1);
n2 = size(x2, 1);

d = zeros(n1, n2);

%%%% your code here %%%%



d = repmat(diag(x1*x1'),1,n2) + repmat(diag(x2*x2')',n1,1) - 2*x1*x2';



%%%%%%%%%%%%%%%%%%%%%%%%%
```

standard ouput

```
loading data

Problem 1b)
Digit #2932: b)
Digit # 814: c)
# of digits with label 5 in training dataset: 542

compute distance matrix using loops
Program paused. Press enter to continue.
compute distance matrix using vectors
Computation time
Loop...    165.3483s
Vector...  3.2498s

Program paused. Press enter to continue.
use distance matrix to compute kNN for various values of k


Considering 1 nearest neighbours.
Number of digits classified as 5...      89
True positive classifications...         84
Actual number of digits labelled 5...    90
True positive/actual number...           0.93333


Considering 2 nearest neighbours.
Number of digits classified as 5...      79
True positive classifications...         78
Actual number of digits labelled 5...    90
True positive/actual number...           0.86667


Considering 3 nearest neighbours.
Number of digits classified as 5...      86
True positive classifications...         82
Actual number of digits labelled 5...    90
True positive/actual number...           0.91111


Considering 4 nearest neighbours.
Number of digits classified as 5...      81
True positive classifications...         79
Actual number of digits labelled 5...    90
True positive/actual number...           0.87778
```

```
Considering 5 nearest neighbours.
Number of digits classified as 5...      86
True positive classifications...         83
Actual number of digits labelled 5...    90
True positive/actual number...           0.92222


Considering 6 nearest neighbours.
Number of digits classified as 5...      81
True positive classifications...         81
Actual number of digits labelled 5...    90
True positive/actual number...           0.9


Considering 7 nearest neighbours.
Number of digits classified as 5...      85
True positive classifications...         84
Actual number of digits labelled 5...    90
True positive/actual number...           0.93333


Considering 8 nearest neighbours.
Number of digits classified as 5...      83
True positive classifications...         82
Actual number of digits labelled 5...    90
True positive/actual number...           0.91111


Considering 9 nearest neighbours.
Number of digits classified as 5...      86
True positive classifications...         84
Actual number of digits labelled 5...    90
True positive/actual number...           0.93333


Considering 10 nearest neighbours.
Number of digits classified as 5...      80
True positive classifications...         80
Actual number of digits labelled 5...    90
True positive/actual number...           0.88889


Considering 11 nearest neighbours.
Number of digits classified as 5...      81
```

```
True positive classifications...         81
Actual number of digits labelled 5...    90
True positive/actual number...           0.9


Considering 12 nearest neighbours.
Number of digits classified as 5...      78
True positive classifications...         78
Actual number of digits labelled 5...    90
True positive/actual number...           0.86667


Considering 13 nearest neighbours.
Number of digits classified as 5...      80
True positive classifications...         80
Actual number of digits labelled 5...    90
True positive/actual number...           0.88889


Considering 14 nearest neighbours.
Number of digits classified as 5...      77
True positive classifications...         77
Actual number of digits labelled 5...    90
True positive/actual number...           0.85556


Considering 15 nearest neighbours.
Number of digits classified as 5...      80
True positive classifications...         80
Actual number of digits labelled 5...    90
True positive/actual number...           0.88889


Considering 16 nearest neighbours.
Number of digits classified as 5...      77
True positive classifications...         77
Actual number of digits labelled 5...    90
True positive/actual number...           0.85556


Considering 17 nearest neighbours.
Number of digits classified as 5...      80
True positive classifications...         80
Actual number of digits labelled 5...    90
True positive/actual number...           0.88889
```

```
Considering 18 nearest neighbours.
Number of digits classified as 5...      78
True positive classifications...         78
Actual number of digits labelled 5...    90
True positive/actual number...           0.86667


Considering 19 nearest neighbours.
Number of digits classified as 5...      80
True positive classifications...         80
Actual number of digits labelled 5...    90
True positive/actual number...           0.88889


Considering 20 nearest neighbours.
Number of digits classified as 5...      79
True positive classifications...         79
Actual number of digits labelled 5...    90
True positive/actual number...           0.87778


Positives...                                  85
Max true positives...                         84
Number of neighbours taken into account...    7
True positive/actual number...                0.93333



Program paused. Press enter to continue.


Program paused. Press enter to continue.


Program paused. Press enter to continue.


Program paused. Press enter to continue.
probability of false prediction
k =  5:      0.31744
k = 20:      0.24466
```