UiT

THE ARCTIC UNIVERSITY OF NORWAY

EXAMINATION PAPER

Home exam

Home exam in:	**FYS-3012/FYS-8012 Pattern Recognition**
Hand-out:	**Wednesday October 16, 2019, 14:00**
Hand-in:	**Friday November 1, 2019, 12:00**

The exam contains 10 pages including this cover page

Contact person:	**Stian Normann Anfinsen**
Email:	**stian.normann.anfinsen@uit.no**

UiT

THE ARCTIC UNIVERSITY OF NORWAY

Before You Start

Module examination

This is a take-home exam for FYS-3012/FYS-8012 Pattern Recognition. Assessment of this assignment counts about 25% towards the final grade in the course. Note that access to the final examination requires submission of this take-home exam. A report which is clearly does not constitute an academic attempt will be deemed as not submitted.

Portfolio instructions

Make sure the report explains your reasoning and does not merely provide equations without further explanation. Computer code should be commented in such a way that any person with programming knowledge should be able to understand how the program works. Like your report, the code must be your own individual work.

As there is a lot of code available online on the web, make sure that your report and code clearly show that you understand what you are doing.

Hand-in format

Please submit *one* single `.zip` file that contains two folders, one called `doc` that contains your report, and another one called `src` containing the code. The file name of the `.zip` file should follow the format `homeexam_candidate`*XX*`.pdf` (replace *XX* with your candidate number obtained from StudWeb) for anonymity.

Follow the hand-in instruction in WiseFlow and make sure to submit before the WiseFlow room closes.

UNIVERSITY OF TROMSØ

Problem 1

In this problem you will design spam filters using the following classifiers:

- Least (sum of) squares classifier
- Bayes classifier using Parzen windowing
- Support vector machine (SVM)

In order to investigate these methods, a training and a test data set of collected real-life e-mails with labels is provided in the following files:

`Xtr_spam.mat`	`ytr_spam.mat`
`Xte_spam.mat`	`yte_spam.mat`

You can download these from the FYS-3012 Canvas room: https://uit.instructure.com/courses/14666/files/folder/Data/SpamData

Label "1" corresponds to spam and label "-1" corresponds to normal e-mails. See Appendix A for more information. For the SVM classifier you may use a kernel width of $\sigma = 4.5$ and parameter $C = 1.7$ (but you are also free to experiment with these values if you wish). For the Parzen window-based method, you may use a kernel width of $\sigma = 0.8$.

Note: Program code must be incorporated in your text or appended. The code should be commented to explain the logic of the program. You may use a programming language of your choice.

(1a) Report the classification error for each method and comment the results.

(1b) Report a *confusion matrix* for each method and comment the results.

(1c) Do you have any recommendations?

Problem 2

In this exercise you will compare Fisher discriminant analysis (FDA) with its kernelised version, known as kernel FDA. The FDA and kernel FDA algorithms are commonly used for feature extraction with labelled data. You will here apply them as classifiers to try to discriminate two classes in an artificially generated data set shown in Figure 1.

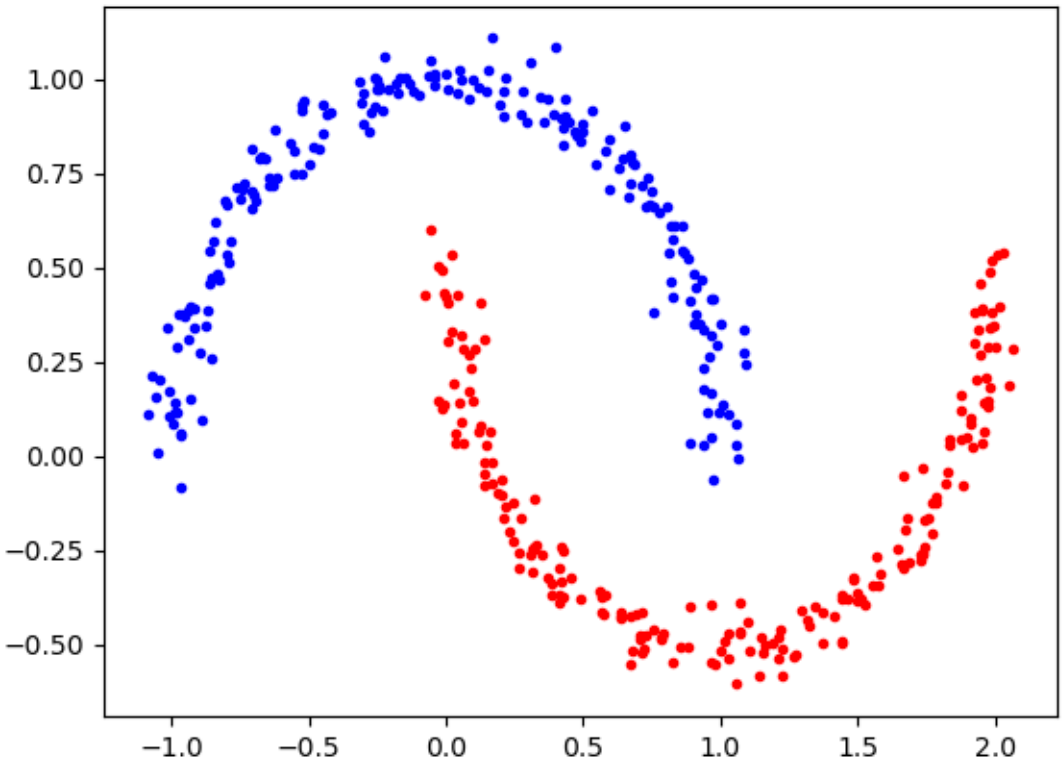

Figure 1: *Data set used in the comparison of of FDA and kernel FDA. The data set consists of two moon shaped classes that are not linearly separable.*

The data set can be downloaded from Canvas: https://uit.instructure.com/files/657308/download?download_frd=1

Thus, we can assume that we have a training data sample $\{\mathbf{x}_i, y_i\}_{i=1}^N$ of N data vectors $\mathbf{x_i}$ and labels y_i drawn from classes $m = \{1, \ldots, M\}$, where $M = 2$ in the two-class problem. We have N_1 training data points from class 1 and N_2 training data points from class 2, such that $N = N_1 + N_2$.

Fisher's linear discriminant is given by the weight vector $\mathbf{w}$ which maximises the cost function:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}, \tag{1}$$

where in the two-class case we can define the between-class scatter matrix as

$$\mathbf{S}_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \tag{2}$$

with class-wise sample means $\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{i=1}^{N_1}$ and $\boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{i=1}^{N_2}$, while the within-class scatter matrix is defined as

$$\mathbf{S}_W = \sum_{m=1}^{2} \sum_{i=1}^{N_m} (\mathbf{x} - \boldsymbol{\mu}_m)(\mathbf{x} - \boldsymbol{\mu}_m)^T . \tag{3}$$

The problem at hand is clearly not linearly separable. We decide that we need a nonlinear discriminant, and therefore want to find a nonlinear mapping $\boldsymbol{\Phi}$ to a Hilbert space $\mathcal{H}$, $\mathbf{x} \mapsto \boldsymbol{\Phi}(\mathbf{x})$, in which we can perform ordinary FDA. This can be obtained by applying a weight vector $\tilde{\mathbf{w}}$ to $\boldsymbol{\Phi}(\mathbf{x})$ in $\mathcal{H}$, which is found by maximising the new cost function

$$J^{\Phi}(\tilde{\mathbf{w}}) = \frac{\tilde{\mathbf{w}}^T \mathbf{S}_B^{\Phi} \tilde{\mathbf{w}}}{\tilde{\mathbf{w}}^T \mathbf{S}_W^{\Phi} \tilde{\mathbf{w}}} . \tag{4}$$

The new weight vector $\tilde{\boldsymbol{w}}$ has the same dimension as $\boldsymbol{\Phi}(\mathbf{x})$ (this dimension may in general be infinite), and the between-class and within-class scattering matrices are defined as

$$\mathbf{S}_B^{\Phi} = (\boldsymbol{\mu}_1^{\Phi} - \boldsymbol{\mu}_2^{\Phi})(\boldsymbol{\mu}_1^{\Phi} - \boldsymbol{\mu}_2^{\Phi})^T \tag{5}$$

and

$$\mathbf{S}_W^{\Phi} = \sum_{m=1}^{2} \sum_{i=1}^{N_m} (\boldsymbol{\Phi}(\mathbf{x}_i) - \boldsymbol{\mu}_m^{\Phi})(\boldsymbol{\Phi}(\mathbf{x}_i) - \boldsymbol{\mu}_m^{\Phi})^T , \tag{6}$$

with

$$\boldsymbol{\mu}_m^{\Phi} = \frac{1}{N_m} \sum_{i=1}^{N_m} \boldsymbol{\Phi}(\mathbf{x_i}) \tag{7}$$

as the class-wise sample means in $\mathcal{H}$.

As with other kernel methods, such as kernel PCA and kernelised SVM, we never have to find the nonlinear transformation $\boldsymbol{\Phi}(\mathbf{x})$ or compute it explicitly. We also do not have to worry about the dimensionality of $\mathcal{H}$. Instead, it can be shown that maximisation of (4) is equivalent to maximising

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T \mathbf{N} \boldsymbol{\alpha}} , \tag{8}$$

and that projections of new data points $\mathbf{x}_t$ in kernel space can be computed as

$$\tilde{\mathbf{w}}^T \boldsymbol{\Phi}(\mathbf{x}_t) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}_t) . \tag{9}$$

That is, the projection in kernel space is found as a weighted sum of inner products between the training data points, $\{\mathbf{x}_i\}_{i=1}^N$, and the new data point, $\mathbf{x}_t$. The inner products in kernel space $\mathcal{H}$ is computed with a kernel function of choice as $k(\mathbf{x}_i, \mathbf{x}_t)$. These are weighted with the α_i obtained as elements of the $\boldsymbol{\alpha}$ which maximises (4). (Details in the derivation are omitted, but can be found in the original paper by Mika et al.: http://courses.cs.tamu.edu/rgutier/csce666_f13/mika1999kernelLDA.pdf).

In order to solve (4), we need to determine the $N \times N$ matrices $\mathbf{M}$ and $\mathbf{N}$. The matrix $\mathbf{M}$ is defined as

$$\mathbf{M} = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T \tag{10}$$

where element j of vectors $\mathbf{m}_1$ and $\mathbf{m}_2$ is given, respectively, as

$$\mathbf{m}_1(j) = \frac{1}{N_1} \sum_{i=1}^{N_1} k(\mathbf{x}_j, \mathbf{x}_i^{(1)}) \tag{11}$$

and

$$\mathbf{m}_2(j) = \frac{1}{N_2} \sum_{i=1}^{N_2} k(\mathbf{x}_j, \mathbf{x}_i^{(2)}) \,, \tag{12}$$

where $\{\mathbf{x}_i^{(1)}\}_{i=1}^{N_1}$ and $\{\mathbf{x}_i^{(2)}\}_{i=1}^{N_2}$ are the data points representing class 1 and 2, respectively.

The matrix $\mathbf{N}$ is defined as

$$\mathbf{N} = \mathbf{K}_1(\mathbf{I}_{N_1} - \mathbf{1}_{N_1})\mathbf{K}_1^T + \mathbf{K}_2(\mathbf{I}_{N_2} - \mathbf{1}_{N_2})\mathbf{K}_2^T \,. \tag{13}$$

Here, $\mathbf{K}_1$ is an $N \times N_1$ kernel matrix containing the kernel space inner products between the full set of N training data points and the set of N_1 training data points from class 1. That is, element $[K_1]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, where $\mathbf{x}_i$ can be any data point whereas $\mathbf{x}_j$ belongs to class 1. Kernel matrix $\mathbf{K}_2$ is defined accordingly, pairing the full set of data points with the data points in class 2. We also use the matrices $\mathbf{I}$, $\mathbf{1}_{N_1}$ and $\mathbf{1}_{N_2}$, where $\mathbf{I}_N$ is the $N \times N$ identity matrix, $\mathbf{1}_{N_1}$ is an $N \times N_1$ matrix with all elements set to $1/N_1$, and $\mathbf{1}_{N_2}$ is an $N \times N_2$ matrix with all elements set to $1/N_2$.

When solving for the $\boldsymbol{\alpha}$ which maximises $J(\boldsymbol{\alpha}$, $\mathbf{N}$ must be regularised and replaced by

$$\mathbf{N}_\epsilon = \mathbf{N} + \epsilon\mathbf{I} \tag{14}$$

in order to make it invertible (full rank) and avoid numerical issues. The constant ϵ should be kept as small as possible, but must be large enough to make $\mathbf{N}_\epsilon$ positive definite (all eigenvalues must be positive).

(2a) Implement FDA and use it to classify the data set. Split the data set into training data and test data. State the parameters in your experiment and provide the classification error. Plot the data set together with the decision boundary in input space.

(2b) Compare the direction of the weight vector you obtain with FDA with the direction of the first principal component from linear PCA, which should also be plotted with the data. Explain the difference.

(2c) Give a general description of the kernel trick and try to relate it to the specifics of the kernel FDA algorithm. (The derivation of kernel FDA is admittably more difficult than the derivation of kernelised SVM and kernel PCA, so it is acceptable if some analogies are left unexplained.)

(2d) Explain why maximisation of the cost function in (8) leads to an eigendecomposition problem where the solution for $\boldsymbol{\alpha}$ is given by the eigenvector associated with the largest eigenvalue of $\mathbf{N}^{-1}\mathbf{M}$ (or, to avoid numerical issues, $\mathbf{N}_{\epsilon}^{-1}\mathbf{M}$).

(2e) Implement kernel FDA with a kernel function of your choice and use it to classify both data sets. You have to find suitable parameters for the kernel function. State the parameters and design choices for your experiment and provide the final classification error. Plot the kernel FDA projections of the data sets together with the decision boundary. Comment on the results and compare with FDA.

Appendix A: About the spam data

Taken from https://web.stanford.edu/~hastie/ElemStatLearn/

1. Title: SPAM E-mail Database

2. Sources:
   (a) Creators: Mark Hopkins, Erik Reeber, George Forman, Jaap Suermondt
   Hewlett-Packard Labs, 1501 Page Mill Rd., Palo Alto, CA 94304
   (b) Donor: George Forman (gforman at nospam hpl.hp.com) 650-857-7835
   (c) Generated: June-July 1999

3. Past Usage:
   (a) Hewlett-Packard Internal-only Technical Report. External forthcoming.
   (b) Determine whether a given email is spam or not.
   (c) ~7% misclassification error.
   False positives (marking good mail as spam) are very undesirable.
   If we insist on zero false positives in the training/testing set,
   20-25% of the spam passed through the filter.

4. Relevant Information:
   The "spam" concept is diverse: advertisements for products/web
   sites, make money fast schemes, chain letters, pornography...
Our collection of spam e-mails came from our postmaster and
individuals who had filed spam. Our collection of non-spam
e-mails came from filed work and personal e-mails, and hence
the word 'george' and the area code '650' are indicators of
non-spam. These are useful when constructing a personalized
spam filter. One would either have to blind such non-spam
indicators or get a very wide collection of non-spam to
generate a general purpose spam filter.

   For background on spam:
   Cranor, Lorrie F., LaMacchia, Brian A. Spam!
   Communications of the ACM, 41(8):74-83, 1998.

5. Number of Instances: 4601 (1813 Spam = 39.4%)

6. Number of Attributes: 58 (57 continuous, 1 nominal class label)

7. Attribute Information:
The last column of 'spambase.data' denotes whether the e-mail was
considered spam (1) or not (0), i.e. unsolicited commercial e-mail.
Most of the attributes indicate whether a particular word or
character was frequently occuring in the e-mail. The run-length
attributes (55-57) measure the length of sequences of consecutive
capital letters. For the statistical measures of each attribute,
see the end of this file. Here are the definitions of the attributes:

48 continuous real [0,100] attributes of type word_freq_WORD
= percentage of words in the e-mail that match WORD,

i.e. 100 * (number of times the WORD appears in the e-mail) /
total number of words in e-mail. A "word" in this case is any
string of alphanumeric characters bounded by non-alphanumeric
characters or end-of-string.

6 continuous real [0,100] attributes of type char_freq_CHAR
= percentage of characters in the e-mail that match CHAR,
i.e. 100 * (number of CHAR occurences) / total characters in e-mail

1 continuous real [1,...] attribute of type capital_run_length_average
= average length of uninterrupted sequences of capital letters

1 continuous integer [1,...] attribute of type capital_run_length_longest
= length of longest uninterrupted sequence of capital letters

1 continuous integer [1,...] attribute of type capital_run_length_total
= sum of length of uninterrupted sequences of capital letters
= total number of capital letters in the e-mail

1 nominal {0,1} class attribute of type spam
= denotes whether the e-mail was considered spam (1) or not (0),
i.e. unsolicited commercial e-mail.

8. Missing Attribute Values: None

9. Class Distribution:
Spam 1813 (39.4%)
Non-Spam 2788 (60.6%)

Attribute Statistics:

	Min:	Max:	Average:	Std.Dev:	Coeff.Var_%:
1	0	4.54	0.10455	0.30536	292
2	0	14.28	0.21301	1.2906	606
3	0	5.1	0.28066	0.50414	180
4	0	42.81	0.065425	1.3952	2130
5	0	10	0.31222	0.67251	215
6	0	5.88	0.095901	0.27382	286
7	0	7.27	0.11421	0.39144	343
8	0	11.11	0.10529	0.40107	381
9	0	5.26	0.090067	0.27862	309
10	0	18.18	0.23941	0.64476	269
11	0	2.61	0.059824	0.20154	337
12	0	9.67	0.5417	0.8617	159
13	0	5.55	0.09393	0.30104	320
14	0	10	0.058626	0.33518	572
15	0	4.41	0.049205	0.25884	526
16	0	20	0.24885	0.82579	332
17	0	7.14	0.14259	0.44406	311
18	0	9.09	0.18474	0.53112	287
19	0	18.75	1.6621	1.7755	107

```
20 0    18.18  0.085577  0.50977  596
21 0    11.11  0.80976   1.2008   148
22 0    17.1   0.1212    1.0258   846
23 0    5.45   0.10165   0.35029  345
24 0    12.5   0.094269  0.44264  470
25 0    20.83  0.5495    1.6713   304
26 0    16.66  0.26538   0.88696  334
27 0    33.33  0.7673    3.3673   439
28 0    9.09   0.12484   0.53858  431
29 0    14.28  0.098915  0.59333  600
30 0    5.88   0.10285   0.45668  444
31 0    12.5   0.064753  0.40339  623
32 0    4.76   0.047048  0.32856  698
33 0    18.18  0.097229  0.55591  572
34 0    4.76   0.047835  0.32945  689
35 0    20     0.10541   0.53226  505
36 0    7.69   0.097477  0.40262  413
37 0    6.89   0.13695   0.42345  309
38 0    8.33   0.013201  0.22065  1670
39 0    11.11  0.078629  0.43467  553
40 0    4.76   0.064834  0.34992  540
41 0    7.14   0.043667  0.3612   827
42 0    14.28  0.13234   0.76682  579
43 0    3.57   0.046099  0.22381  486
44 0    20     0.079196  0.62198  785
45 0    21.42  0.30122   1.0117   336
46 0    22.05  0.17982   0.91112  507
47 0    2.17   0.0054445 0.076274 1400
48 0    10     0.031869  0.28573  897
49 0    4.385  0.038575  0.24347  631
50 0    9.752  0.13903   0.27036  194
51 0    4.081  0.016976  0.10939  644
52 0    32.478 0.26907   0.81567  303
53 0    6.003  0.075811  0.24588  324
54 0    19.829 0.044238  0.42934  971
55 1    1102.5 5.1915    31.729   611
56 1    9989   52.173    194.89   374
57 1    15841  283.29    606.35   214
58 0    1      0.39404   0.4887   124
```

This file: 'spambase.DOCUMENTATION' at the UCI Machine Learning Repository
http://www.ics.uci.edu/~mlearn/MLRepository.html