UiT

NORGES
ARKTISKE
UNIVERSITET

EXAMINATION PAPER

Home Exam in:	**Fys-3012 Pattern Recognition**
Hand-out:	**Monday 19th Oct. 2015 at 0900.**
Hand-in:	**Monday 2nd Nov. 2015 (1500 at latest)**

The exam contains 10 pages including this cover page and appendix.

Contact persons:

Robert Jenssen

Email: robert.jenssen@uit.no

Sigurd Løkse

Email sigurd.lokse@uit.no

Introduction

Machine learning and pattern recognition power much of today's technology and apps (think of e.g. Shazam), and a flurry of basic and applied research in data science and algorithms are carried out in academia and in the ICT-companies such as Google, IBM etc.

We will in this home exam develop such technology based on algorithms where the input to the analysis chain is data in the form of music and in the form of images representing bits.

Two things in particular will be important in addition to be able to program and execute our algorithms, namely

- That we explain mathematically how the methods we use work;
- That we discuss pros and cons.

We have to document our work, and to write a report.

In this home exam, you are free to write your report and to investigate (program, execute and explain) those machine learning and pattern recognition methods you want to, including data transformation and dimensionality reduction methods. The more comprehensive report, the better, of course. When it comes to classification, you should however at least as a minimum at some point investigate the following methods:

- Least (sum of) squares (LS) classifier;
- Two-layer perceptron (2LP);
- Bayes classifier assuming normal data with equal covariance structure;
- Support vector machine (SVM).

It is always better to provide analysis based on your own code in order to understand both how methods work, but also what may cause methods to fail.

Useful SVM Matlab software is provided in Fronter since this method requires optimization knowledge beyond Fys-3012:

1. `SMO2.m`
2. `svcplot_book.m (needs cmap.mat)`
3. `calkKernel.m`

Recall that for the hyperplane $\mathbf{w}^\top \mathbf{x} + b$, SMO2 outputs $-b$.

Data sets are available in Fronter in the folder "DATA" stored in Matlab-format. See Appendix A for a comment on the data sets.

Note: Students normally program in Matlab, and the SVM code is Matlab. You are however free to use whatever programming language you want to. In any case, code must be incorporated in your text or appended. The code should be commented in such a way that any person with programming knowledge

should be able to understand how the program works. For those of you using Python, you may use *SciPy* to load the Matlab data files. See

`http://docs.scipy.org/doc/scipy/reference/tutorial/io.html`

for more info.

Good luck!

Figure 1: Music recommender systems and genre classification systems are hot! (figure from djworx.com).

Music

You want to be rich by starting your own tech company, developing your own music genre classification system (Problem 1) and recommender system (Problem 2) app for sale. Performance is of course key. You have to write a report to your company board with your findings.

Importantly, you want to classify songs into genres and recommend music based on the content of the songs, and not on rule-based systems based on meta data. You turn to Echonest

```
http://developer.echonest.com
```

where you can use their *Developer center* to generate *acoustic attributes*, i.e. *music features*

```
http://developer.echonest.com/acoustic-attributes.html
```

representing a particular song. These attributes compute for instance a score for the *tempo* of the song, the *loudness*, the *danceability*, etc. Other attributes compute statistical moments (expectations) over the *timbre* and *pitch*.

You ask your employee, Sigurd Løkse, to generate the features comprising the music data sets from Echonest. He also programs a Matlab interface to Spo-

tify that you may use to play songs in the data set and to retrieve information about a particular song.

Interface to Spotify
The Spotify interface *scatter_song(X, y, song_info, genres)* creates a 2- or 3-dimensional scatterplot of the song data. Your input variable X must therefore constitute a 2- or 3-dimensional data set. Clicking on a data point in the scatterplot starts the Spotify web player and plays back the corresponding music track.

INPUTS:

- $\mathbf{X} = \{x_{ij}\}_{N\times p}$: Input data. N is the number of songs and p is the dimensionality of the data.
- $\mathbf{y} = \{y_i\}_{N\times 1}, \quad y_i \in \{1, 2, \ldots, 7\}$: Vector of labels used for coloring.
- *song_info*: $N \times 3$ cell array containing artist, song name and Spotify URL.
- *genres*: Cell array containing the different music genres. *(OPTIONAL)*

Note that you can retrieve information, such as artist, about a song from the cell array *song_info* which you can use to play a song, for example using Spotify or some other service, without necessarily using Sigurd's interface script.

Problem 1: Music genre classification system

You will in this problem develop music genre classification systems. It is very important in the music industry to be able to categorize a song into genres based on its content, or perhaps even to categorize a song into several genres at the same time. You will in *(a)* and *(b)* work on two different music data sets. Common for both is that the training data is composed of music from two different categories, hence we are talking about two-class problems.

You have to present your results, and preferably you should describe your results also by listening to the songs. For example, you may listen to misclassified songs and try to explain or speculate why these were misclassified, and to investigate differences between classifiers. You may for example listen to and describe the most "typical" song assigned to any of the two classes by finding the song closest to the mean of the obtained categories. Basically, try to be creative when writing the report.

(a) In Fronter you will find the file `songdata_small.mat`. The file contains training data, test data, labels, etc. These you will use to develop your music genre classification systems. The songs are here described by four Echonest features, namely *tempo*, *loudness*, and two different statistical moments.

(b) Now you will use many more features, namely altogether 101 music features from Echonest, `songdata_big.mat` in Fronter. Now repeat your genre classification experiments based on this expanded data set.

Problem 2: Music recommender system

In a music recommender system, you will have to output a ranked list of songs with respect to a particular (seed) song, i.e. the song you base your recommendations on. The top ranked song with respect to the seed is the one you would recommend to the user. But you could also recommend a playlist of songs, and in that case you could for example present your top 10 ranked songs with respect to the seed. For a user, it would also be of interest to be presented with a visualization of the songs in the form of a transformation of the features into a 2-dimensional or 3-dimensional space, and in that space illustrate (plot) the relationship to the seed. Your recommender system is *unsupervised*.

- Base your recommender system on the data set `songdata_full.mat` available in Fronter. Again, please listen to songs when presenting results, and illuminate differences between methods. Please provide different ways to visualize song and results (here for example PCA and/or Laplacian eigenmaps can be used for the recommender engine and also for the visualization). Concretely, please at some point in your discussion recommend songs with respect to seed song number 82 in the data set (think of this as a particular user's personal favorite song).

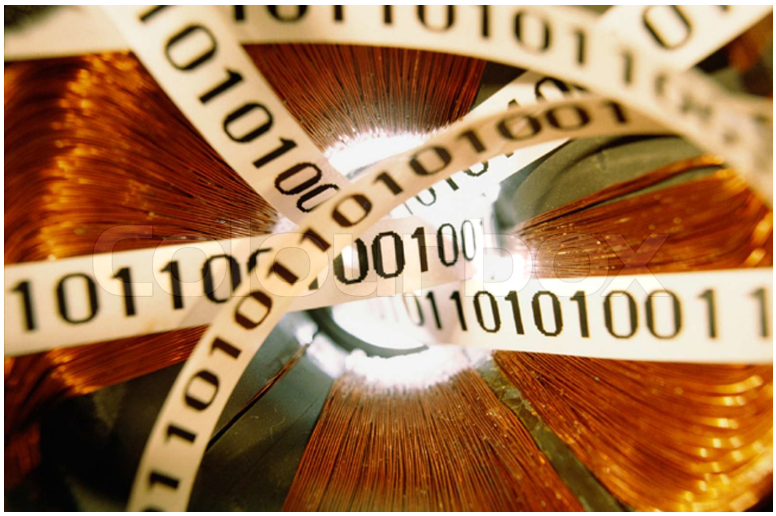

Figure 2: We will develop a bit-based communication system using handwritten digit recognition machines (figure from colourbox.com).

Binary Digit Recognition

In numerous situations, it is of interest to be able to program computers to recognize handwritten text. You probably have software to do this on your computer, and such systems are based on pattern recognition and machine learning. It took 20 years of intense research to be able to create reasonably good systems to do this (and it is not long since such systems were first operative), many based on methods taught in Fys-3012. The research is still intensely ongoing, for instance with respect to so-called deep learning.

We will study a "revolutionary" ASCII code (see table in Appendix B) bit-based communication system. This means that each character is represented by a 7-bit code. Hence, if a person writes for example: 1000001 by hand, this is the letter "A". The recognition system only needs to recognize two different handwritten symbols! This is the great advantage of the system, from a methods viewpoint. The system will therefore be *binary*, with only two classes "1" and "0". Note that all digital communication systems are binary, however bits are represented in different ways. All such systems will in any case need classifiers at the receiving end, just like our system in this problem.

We have available two different training sets. One is a set of 10 (16×16) handwritten ones and zeros in

`Xtr_digits.mat` `ytr_digits.mat`

Each digit (image) is represented as a 256-dimensional vector. See Appendix C for more information on this real data set, obtained from the US postal service. A plot of the 10 training digits at our disposal is shown in Fig. 3.

The other training set is larger, and can be found in

`Xtr_digits_larger.mat` `ytr_digits_larger.mat`

Figure 3: Plot of the training set for the "1" and "0" digits.

In Fronter, there is a Matlab-function

`ascii.m`

which takes as input a vector of labels $\in \{-1, 1\}$, separated by spaces, and outputs the *text* corresponding to the labels using the ASCII code. Only lower case letters are currently implemented, in addition to "!" and "." If a 7-bit string doesn't correspond to any symbol "a-z" or "!" and ".", the symbol "#" is output. The message you will try to decode in order to test our binary handwritten digit recognition system, is available in

`Xte_digits_2015.mat`

Problem 3: Binary digits in bits-based communication

(*) Design binary digit recognition systems and run `ascii.m` on classification results. You may need to find good parameters by experimenting a little bit, or by other means. You are free to use all methods you would like to use, and to test out data transformation and dimensionality reductions methods. Enlighten the world on any differences etc. with respect to the two different training sets.

Appendix A: Data Format

It seems that for some people Matlab data stored in Fronter is downloaded in the format struct. For example, if you use the command "whos" and you see something like this:

```
>> whos Xtest
  Name       Size              Bytes  Class     Attributes

  Xtest      1x1             1935536  struct
```

then "Xtest" is a struct variable. You can check the content of the variable just by typing it's name:

```
>> Xtest

Xtest =

    Xtest: [256x945 double]
```

So here, the struct variable "Xtest" contains the data in double format, and you need to write in this particular case "Xtest.Xtest" in the Matlab command window to access the actual data. Please ask if you experience problems, or if you find find this confusing.

Appendix B: ASCII Table

b_7 →				0	0	0	0	1	1	1	1
b_6 →				0	0	1	1	0	0	1	1
b_5 →				0	1	0	1	0	1	0	1
Bits	b_4 ↓	b_3 ↓	b_2 ↓	b_1 ↓	Row ↓ / Column →	0	1	2	3	4	5	6	7
	0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
	0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
	0	0	1	0	2	STX	DC2	"	2	B	R	b	r
	0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
	0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
	0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
	0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
	0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
	1	0	0	0	8	BS	CAN	(	8	H	X	h	x
	1	0	0	1	9	HT	EM	)	9	I	Y	i	y
	1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
	1	0	1	1	11	VT	ESC	+	;	K	[	k	{
	1	1	0	0	12	FF	FC	,	<	L	\	l	|
	1	1	0	1	13	CR	GS	-	=	M	]	m	}
	1	1	1	0	14	SO	RS	.	>	N	^	n	~
	1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Figure 4: ASCII Table. The letter "A", for example, has a 7-bit code 1000001.

Appendix C: About the Digits Data

Taken from `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`

```
Normalized handwritten digits, automatically
scanned from envelopes by the U.S. Postal Service. The original
scanned digits are binary and of different sizes and orientations; the
images  here have been deslanted and size normalized, resulting
in 16 x 16 grayscale images (Le Cun et al., 1990).
```