Information Discriminant Analysis (IDA)

Zoran Nenadic, D.Sc.

October 4, 2007

1 Introduction

This tutorial is an accompanying document to the computer code for *information discriminant analysis*. The details of the method can be found in [1] and the computer code is written in MATLAB^{TM}. The code consists of several basic functions:

(1) `negative_mu.m`

(2) `ida_feature_extraction_matrix.m`

(3) `orthonormalize.m`

`negative_mu.m` returns the value of the μ-measure whose maximization yields a feature extraction matrix, T^*. This function can return additional arguments, namely the gradient and the Hessian of μ with respect to the current feature extraction matrix, T. The minimization of μ is iterative in T, and knowing the gradient and the Hessian of μ enables feasible computations. The initial value of T is chosen by the user. For more info type `help negative_mu` in MATLAB^{TM}command prompt.

`ida_feature_extraction_matrix.m` returns the optimal feature extraction matrix, T^*, as well as the value of the μ-measure at the optimal feature subspace. `ida_feature_extraction_matrix.m` uses a built-in MATLAB^{TM}optimization function `fminunc.m`, therefore to run this function Optimization Toolbox may be necessary (see below for exceptions). Since all MATLAB^{TM}optimization routines are written as minimizations, the maximization of the μ-measure is achieved as the minimization of $-\mu$. Hence the name of the function above (`negative_mu`). The input arguments of `ida_feature_extraction_matrix.m` allow for various choices of initial condition, optimization tolerances, size of feature space, optimization method, etc. Type `ida_feature_extraction_matrix` in MATLAB^{TM}command prompt to learn more about this function.

`orthonormalize.m` is a function I wrote not being aware of MATLAB^{TM}function `orth.m`. It turns out that `orth(A')=orthonormalize(A)'`, where `A` is an arbitrary matrix. In addition, `orthonormalize.m` returns the largest singular value of `A`. This function is an auxiliary function and is used to orthonormalize the feature extraction matrix T.

The optimization in `ida_feature_extraction_matrix.m` can be implemented using the conjugate-gradient method. It runs very efficiently, and in general is faster than the trust-region method, used by `fminunc.m`. This is especially true for large-scale problems, where the feature extraction matrix, T, has a lot of elements. For this purpose two additional functions are needed:

(4) `conjugate_gradient.m`

(5) `linesearch.m`

These functions were written by Hans Bruun Nielsen, and the above links point to his web page. As far as I can tell, the functions are bug-free, except for one minor thing: I had to replace the variable named `alpha` in `conjugate_gradient.m` with `Alpha`. I think the code is several years old, and meanwhile `alpha.m` became a legitimate MATLAB™function. Therefore, using `alpha` will cause MATLAB™to call the function, and consequently report an error.

Using `conjugate_gradient.m` places some constraints on the way the objective function (in this case `negative_mu.m`) is called. In particular, the parameters of `negative_mu.m` have to be passed as a single argument. In addition, conjugate-gradient uses no Hessian, and so I decided to write a version of `negative_mu.m` that works with `conjugate_gradient.m`. The function is called:

(6) `negative_mu_cg.m`

This function is marginally different from its original version, but requires some manipulations of the feature extraction matrix, T, so I decided to write a separate function. Anyway, with these 6 functions, one should be able to implement IDA as a feature extraction technique. Final remark: running conjugate-gradient method does not require MATLAB™Optimization Toolbox.

2 Example

Application of these functions is illustrated on data set Satellite from the UCI machine learning repository. This data set consists of 6435 samples (each sample is 36-D). The first 4435 samples are used for training, with the remaining samples used for testing. The number of classes is 6. For example, running `[pc mu T et] = test_satellite(2,10,1,'lda','tr');` from MATLAB™command prompt returns the following:

- The classification accuracy `pc`, determined by the linear and quadratic Bayesian classifiers.
- The value of the μ-measure `mu` in the optimal feature space.
- The optimal feature extraction matrix `T`.
- The elapsed time `et`.

The input arguments represent:

- The size of the feature space `m = 2` (passed to (2)).
- The number of optimization runs `Nruns = 10` (passed to (2)).
- Plot indicator `PlotFlag = 1` to enable the plotting of features.
- Initial condition (initial choice of feature extraction matrix) is set to linear discriminant analysis (LDA) matrix (passed to (2)).
- Choice of optimization method is set to trust-region method (functions (2) and (1) will be called).

In case you care, here is the function `test_satellite.m` posted. Note that the function is specifically tailored to Satellite data. Also note that it is assumed that a file `Satimage.mat` resides in the same directory as `test_satellite.m`. This file should contain a variable `Data` in the form of 6435×37 matrix. The last column of this matrix contains class label indicators (integers), e.g. $\{0, 1 \cdots, 5\}$. Type `help test_satellite` or take a look at the source code for further information on this function. Final note: running `test_satellite.m` will call `classify.m` which is a function from MATLAB$^{\mathrm{TM}}$Statistics Toolbox. If the toolbox is not installed, these functions (linear and quadratic classifier) can be easily written.

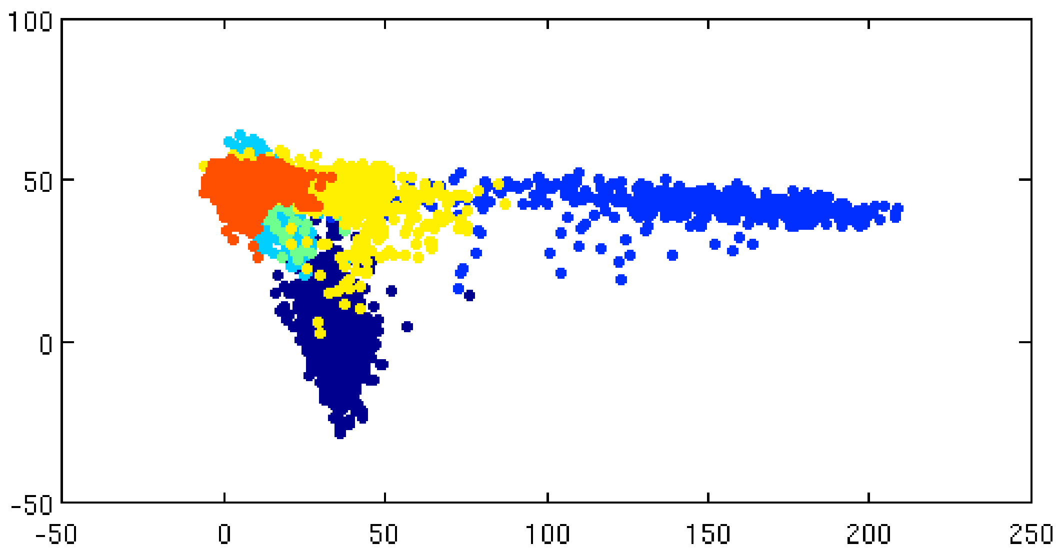

Figure 1: 2-D features corresponding to the training samples of the Satellite data. Colors indicate class memberships.

Fig. 1 shows the resulting 2-D feature plot. The performances (percent correct) of the linear and quadratic classifiers are 65.90% and 72.45%, respectively. Table 1 shows results for various dimensions of the feature space. IDA is initialized with two methods: LDA and CHE (Chernoff method of Loog and Duin [2]). For $m > 5$, LDA does not yield a feature extraction matrix, therefore a random matrix was used instead. These results are separated from LDA-initialized results by a horizonal line (see Table 1). The boxed values represent the best results for each classifier-method combination.

Table 1: Performances (% correct) of IDA, the method of Loog and Duin [2] (CHE) and LDA. The options in the parentheses are the number of runs (random restarts), the initialization method (Chernoff,IDA), and the optimization method (trust-region, conjugate-gradient).

m	IDA, (10,'che','tr')		IDA, (10,'lda','tr')		IDA, (10,'lda','cg')		CHE		LDA	
	(L)	(Q)	(L)	(Q)	(L)	(Q)	(L)	(Q)	(L)	(Q)
1	59.20	67.30	59.20	67.30	59.20	67.30	71.45	73.45	52.35	56.50
2	65.90	72.45	65.90	72.45	65.90	72.45	80.75	81.10	75.95	78.35
3	82.15	84.75	82.15	84.75	82.15	84.75	82.00	84.55	82.30	84.10
4	82.30	85.20	82.30	85.15	82.30	85.15	82.20	84.25	82.75	**84.70**
5	82.25	83.65	82.25	83.65	82.25	83.65	82.25	84.10	**82.85**	84.50
6	81.80	83.40	81.65	83.25	81.80	83.40	82.05	83.50		
7	82.00	84.15	81.65	84.20	81.65	84.15	82.45	84.25		
8	82.30	83.85	81.55	83.60	81.80	84.60	82.55	84.00		
9	82.65	84.20	82.15	84.15	82.25	84.15	82.70	84.05		
10	82.75	84.15	82.50	84.25	82.60	84.50	82.95	84.35		
11	82.85	83.75	82.15	84.10	82.55	83.95	82.75	84.30		
12	82.80	83.95	82.65	84.25	82.40	84.30	83.00	84.50		
13	82.75	84.10	82.55	84.35	82.60	84.75	82.80	84.35		
14	82.80	83.95	82.70	84.55	82.75	84.45	82.75	84.60		
15	82.80	84.50	82.50	84.85	82.80	84.35	82.80	84.90		
16	82.85	84.50	82.40	84.95	82.90	84.50	82.75	84.55		
17	83.10	84.70	83.05	84.65	83.10	84.70	82.90	84.85		
18	83.20	84.95	83.05	84.85	83.20	84.95	83.00	85.15		
19	**83.30**	85.10	**83.20**	85.05	**83.30**	85.10	83.00	85.10		
20	83.10	84.95	82.90	84.95	82.85	85.00	82.85	**85.25**		
21	82.85	85.10	82.90	85.15	82.85	85.10	82.65	84.95		
22	83.00	85.20	82.75	85.20	83.00	85.20	82.75	85.05		
23	83.05	85.00	83.05	85.15	83.05	85.00	82.85	85.05		
24	82.95	85.00	82.75	84.95	82.95	85.00	82.75	84.90		
25	82.75	84.85	82.90	85.00	82.75	84.85	82.80	84.95		
26	82.90	84.70	82.90	84.65	82.90	84.65	82.80	84.90		
27	82.85	84.75	83.00	85.00	83.10	84.95	**83.05**	84.85		
28	82.80	85.00	82.90	84.95	82.80	85.00	82.80	84.80		
29	82.95	85.20	82.85	85.15	82.85	84.95	82.90	84.85		
30	82.90	85.00	82.90	85.05	82.90	85.00	82.90	85.15		
31	82.65	**85.25**	82.65	**85.35**	82.65	**85.25**	82.95	85.10		
32	82.85	85.05	82.85	85.05	82.85	85.05	82.95	84.90		
33	82.90	84.90	82.90	84.90	82.90	84.90	82.80	84.85		
34	82.70	84.85	82.70	84.85	82.70	84.85	82.70	84.90		
35	82.80	84.85	82.80	84.85	82.80	84.85	82.85	84.90		

References

[1] Z. Nenadic, Information discriminant analysis: Feature extraction with an information-theoretic objective, *IEEE T. Pattern Anal.*, vol. 29 (8), pp. 1394-1407, 2007.

[2] M. Loog and R.P.W. Duin, Linear Dimensionality Reduction via a Heteroscedastic Extension of LDA: The Chernoff Criterion, *IEEE T. Pattern Anal.*, vol. 26, pp. 732-739, 2004.