

## Chapter 3

# AstraZeneca: Classification of cancer using 2D-electrophoresis

### 3.1 Introduction

Anders Holst and Thorsteinn Rögnauldsson

The AstraZeneca task consisted in discriminating between three ovarian tumor types: Benign, borderline, and malignant. The discrimination is based on the spots found on a 2D electrophoresis gel using the PDQuest software. Each spot on a 2D gel corresponds to a polypeptide (proteinlike structure) and the hypothesis was that different types of ovarian cancers were indicated by different proteins (or modified proteins).

The PDQuest software was used, by AstraZeneca, to find a set of potentially interesting spots in the set of gels and the intensity for each spot was estimated for each gel (some spots had zero intensity meaning that they were absent from that particular gel).

Our task was to take this set of intensities for the gels and apply classification methods to see how well the cancer types can be categorized and separated.

The challenge here is the very few available data samples (40) combined with the very high number of inputs (1553). Even if all inputs are random, it would suffice with 39 dimensions to find separating planes for the three classes in the 40 samples. This calls for a method inherently robust to overtraining. One approach is to reduce the dimensionality of the data space with PCA or PLS. This has also been done in another study [Alaiya *et al.*, 2000], and fairly good results were achieved when the three first principal components were used.

## 3.2 The model used at SICS

Anders Holst

### 3.2.1 Choice of method

Another approach is to use the *naive Bayesian classifier*. This is what was done at SICS. In calculating what each of the inputs means by itself for the data samples, there are only four probabilities (for each) to estimate from the 40 samples, which is a dramatic improvement to the situation. Still, among 1553 inputs, some of them will by pure chance seem to be good indicators for the classes on the training set. The hope is that by then combining the evidence from all inputs, there will be as many accidentally bad as accidentally good indicators, canceling each other out, which will thus make the possibly real indicators for the classes stand out from the noise.

The 1553 inputs, representing the proteins at different locations in the gel, is extracted at AstraZeneca by an image manipulating program, which must decide which spots represent real protein traces and which are only noise in the gel image. Since this preprocessing step is in itself quite complicated, there was an idea that instead of feeding these extracted 1553 values into the naive Bayesian classifier, the pixels of the image could be used directly. There would be a lot of noise, and many proteins would cover several pixels, but provided that the same protein location in the different gels are located at approximately the same pixels in the image, the noise would again cancel out and only the pixels corresponding to real and relevant protein locations would contribute to the classification. However, the gels are soft and quite heavily deformed, and therefore the image manipulation must involve a warping of the images to fit the proteins into place. Unfortunately, we had access only to the raw images, and not the warped ones. We tried hard for some time to warp the pictures ourselves, but the gels are too deformed for a simple linear or almost linear warping to suffice. Since AstraZeneca already knows how to warp the images, we figured that this was not the problem we should spend our time to solve, so we gave up this approach and used the pre-extracted values for the 1553 proteins instead.

### 3.2.2 Preprocessing and experimental setup

For each protein location and gel there is a number  $i$  the data that tells how strong the spot is at that location. The maximum value of the strength varies a lot between different proteins. A value of -1 means that no spot was detected. We assumed that the important difference is between the cases of no spot or anything at all, rather than the exact strength of the existing spots. (The alternative assumption, not used at SICS, was that -1 should be interpreted as missing data, and that the relevant information was in the strengths of the detected spots.) Thus we changed -1 in the data to zero, put a threshold at some strength above which the value was changed to one, and below to zero. Since different proteins had very different maximum strengths of their spots, using thresholds at some ratio of the maximum strength of each protein was also tried. Six proteins that had value -1 in all samples were removed.

To be able to compare the results with those from AstraZeneca, the same division in training and testing data was made. This meant that 22 patterns were used for training and 18 for test. Runs were also done with cross-validation, where one of the 40 patterns were removed for testing each time and the other 39 used for training.

The naive Bayesian classifier was trained using Bayesian estimation of the probabilities, and the alpha value of Equation (2.83) set to the reciprocal of the number of training patterns in each case (39 when crossvalidation is used, 22 otherwise). The class with the highest probability is selected as the answer. Ties are counted as failures.

### 3.2.3 Results

The naive Bayesian classifier used here doesn't have any further free parameters. The only parameter that is not fixed in this setting, is when to consider a protein to be present or not. Therefore several different thresholds of the spot strengths were tested. In table 3.1 the results of using the same fixed

	Train All	Crossvalidation	Trainingset	Testset
1	–	–	100% (22)	50% (9)
500	–	–	100% (22)	50% (9)
1000	–	–	100% (22)	55.6% (10)
1100	–	–	–	55.6% (10)
1200	–	–	–	61.1% (11)
1300	–	–	–	61.1% (11)
1400	100% (40)	72.5% (29)	100% (22)	61.1% (11)
1500	100% (40)	67.5% (27)	100% (22)	61.1% (11)
1600	100% (40)	67.5% (27)	100% (22)	55.6% (10)
1700	–	–	–	50% (9)
2000	–	–	100% (22)	55.6% (10)
3000	–	–	100% (22)	44.4% (8)

**Table 3.1:** Results with the naive Bayesian classifier for different thresholds of strengths.

	Train All	Crossvalidation	Trainingset	Testset
40%	–	–	–	55.6% (10)
50%	–	–	100% (22)	61.1% (11)
55%	–	–	–	61.1% (11)
60%	100% (40)	75% (30)	100% (22)	72.2% (13)
65%	–	–	–	66.7% (12)
70%	–	–	–	66.7% (12)
80%	–	–	–	61.1% (11)

**Table 3.2:** Results with the naive Bayesian classifier for different relative thresholds of the maximum strength for each protein.

thresholds for all proteins are used. In table 3.2 each protein has instead a threshold at some given ratio of its maximum spot strength. For each threshold, up to four different tests were performed. The first column in both tables shows the result of testing all 40 patterns on a model trained on the same patterns. The second column shows the results of leave-one-out crossvalidation, *i. e.* the model was tested 40 times with one pattern held out as a test pattern and the remaining 39 used for training. Finally the third and fourth columns shows the performance of a model trained on the 22 patterns dedicated for training, the third when tested on those same training patterns and the fourth when tested on the 18 test patterns.

Since the tests took some time to perform, all slots in the table were not run. Specifically since testing on the same set as was used for training (first and third columns) always seemed to give 100% correct, this didn't seem meaningful to run for the rest of the cases. Although interesting, the second column was the most heavy of them all to run, which is why it is only sparsely filled. The most important results here though are in the fourth column, which shows the generalization of the classifier on the test set. The result varies somewhat with the threshold, but a typical result is around 60% correct. The maximum is achieved with with relative thresholds of 60% of the protein strength, which gave 72%, or 13 out of 18 samples correctly classified.

From the naive Bayesian classifier it is possible to read out how much evidence each input contributed to the conclusion. Table 3.3 lists the indices of the proteins whose existence contributes the most information about each of the three classes of cancer.

### 3.2.4 Conclusions

The results show that the Naive Bayesian Classifier is very suitable for a problem like this. In spite of the very large number of inputs and few training data, it succeeds to classify the samples at least as good as the other methods tried. This is done using all the available input dimensions, *i. e.* no dimension reduction via variable selection or principal component analysis is needed.

Benign		Borderline		Malignant	
Index	Information	Index	Information	Index	Information
784	0.111756	184	0.0767320	619	0.118567
1321	0.0949511	1406	0.0767225	1321	0.103583
1266	0.0949511	522	0.0678651	1266	0.103583
802	0.0866867	1463	0.0677796	1124	0.0890701
619	0.0805614	1067	0.0677796	1518	0.0750026
452	0.0779572	921	0.0677796	1308	0.0750026
1276	0.0771985	1317	0.0677704	927	0.0750026
1311	0.0695426	922	0.0677704	239	0.0750026

**Table 3.3:** Indices for the proteins with the highest expected information gains to each of the three types of cancer.

One advantage with the naive Bayesian classifier is that it can easily be read out how much each input contributes to the classification, and thus to find out which inputs are most important for a certain class. This can be of great value when analyzing the domain.

### 3.3 The models used at Halmstad University

#### Thorsteinn Rögvaldsson

We decided to compare two different types of classifiers: Two simple Gaussian classifiers, which are Bayesian classifiers, and a multilayer perceptron, which is an “a posteriori” classifier.

It is described by Alaiya *et al.* [2000] how principal component regression and partial least squares regression are used on this data set. We have used the results in [Alaiya *et al.*, 2000] as benchmark for our methods.

#### 3.3.1 Data, preprocessing, and variable selection

The data consists of 40 vectors with spot intensities (*i. e.* 40 cases). Each vector has 1553 spot intensities. Of the 40 cases, 22 are used for training, and 18 are saved for out-of-sample testing after the model has been built. The true classification of the test cases was unknown throughout the modeling process, but the test case inputs were known (to check *e. g.* if there was a systematic difference between training and test cases).

We removed all spots that were not present in all the training cases. This was potentially dangerous, since it may well be that the most interesting difference would be the absence or presence of a unique polypeptide. However, we wanted to save time and started with this simple approach. This leaves 88 spots (of which 66 are in fact absent in the test cases), *i. e.* a 95% dimensionality reduction. These spot intensities were then projected onto the 20 first principal components of the training data, yielding a data matrix of 40 observations and 20 variables for each observation.

The exact preprocessing steps were:

1. Select only those spots which are present in all the training samples. This reduces the number of spots from 1553 down to 88.
2. Compute principal components by taking the eigenvalues and eigenvectors of the covariance matrix for the observations. Keep 20 of these, since about 99.9% of the signal power is retained in the first 20 eigenvalues.

We used a model based variable selection method. The principal component variables were included in the model one by one, in order of decreasing eigenvalues, and the model generalization performance was estimated using cross validation. In each step was the variable selected which resulted in the largest decrease in classification error, as estimated using cross validation. An example of this process is shown for the Gaussian classifiers in Figure 3.1, and the same general approach was used also in the MLP case.

#### 3.3.2 Constructing the classifiers

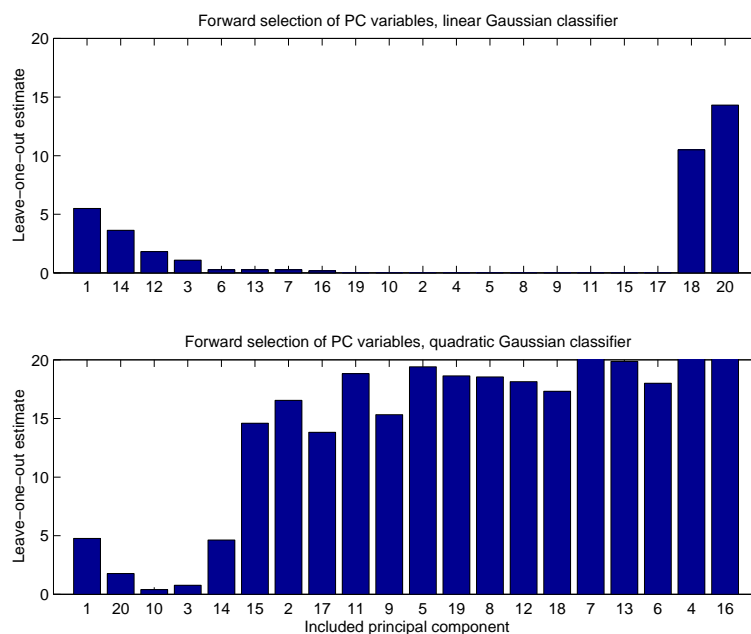
The Gaussian classifiers were very simple to construct (the whole construction process takes a few hours including the experimenting). In the case of the linear Gaussian classifier, the means and the covariances of the three classes were computed, and then a pooled covariance matrix was computed for the entire data set according to

$$\Sigma = \frac{N_{benign}\Sigma_{benign} + N_{borderline}\Sigma_{borderline} + N_{malignant}\Sigma_{malignant}}{N_{benign} + N_{borderline} + N_{malignant}}. \quad (3.1)$$

Here,  $\Sigma_{nnn}$  denotes the covariance matrices for the individual categories.

In the case of the quadratic Gaussian classifier the individual covariance matrices were retained.

The MLP, however, required considerable more effort. Mostly because it takes longer to train for each model setup. The input variables were selected in a forward manner and the generalization error estimated using  $n$ -fold cross validation (not leave-one-out because this took too much time). The final MLP was



**Figure 3.1:** Examples of the model based forward variable selection process. The top panel shows the result of including more and more principal components in a linear Gaussian classifier, and the bottom panel shows the same for a quadratic Gaussian classifier. The abscissa is the leave-one-out estimate of the generalization classification error. The very large errors (*i. e.* making more than 15 errors out of 22 possible) occur because the matrices in the Gaussian classifiers become ill-conditioned. The top panel shows *e. g.* that the estimated generalization error for a linear Gaussian classifier is zero if we include principal components  $\{1, 14, 12, 3, 6, 13, 7, 16, 19\}$  in the model. The bottom panel shows that the lowest generalization error for a quadratic Gaussian classifier is achieved using principal components  $\{1, 20, 10\}$ .

a committee combination of 420 different MLPs (trained using Levenberg-Marquardt minimization and different initial conditions).

### 3.3.3 Results

Several different variable setups yield zero generalization error for the linear Gaussian classifier. We can not choose reliably whether to use 9, 10, 11 or 12 (or more) principal component variables for the linear Gaussian model. We therefore used four different setups (9, 10, 11, and 12 input variables) that all gave zero generalization error and combined them into a committee. We did similarly with the quadratic Gaussian classifier, but used only two setups (3 and 4 input variables). The Gaussian classifier committees were voting committees.

A large set of MLPs (420 of them to be exact) were combined into an averaging committee before the final cancer categories were registered.

The final results are listed in Table 3.4.

	True category	Benchmark	Lin. Gauss	Quad. Gauss	MLP
	A	B	B	B	B
	B	B	C	C	C
	B	B	B	B	B
	C	C	B/C	B	B
	C	C	C	C	C
	A	A	A	C	A
	B	A	B	B	B
	C	C	C	C	B
	C	B	A	C	B
	C	C	C	C	C
	C	C	C	B	C
	C	B	A	B/C	B
	A	A	A	B	A
	A	B	A/B	B	A
	A	B	B	B	B
	C	C	A	B	B
	B	B	B	B	B
	A	B	A	B	A
Mistakes	–	7	6 (8)	10 (11)	8

**Table 3.4:** Summary of results on the ovarian cancer hold-out test data. The classes are: A = Benign cancer, B = Borderline, and C = Malignant cancer. The column “Benchmark” shows the results reported in [Alaiya *et al.*, 2000]. Just random guessing with equal a priori probabilities of the cancer categories would give about 12 mistakes on average. The 95% significance limit for a non-random result is about 8 mistakes (*i. e.* if the model makes more than 8 mistakes then we cannot with 95% confidence discard the hypothesis that it is no better than a random guess). Apart from the Benchmark, it is only the linear Gaussian classifier, the simplest model of all, that is significantly better than a random guess. Both the Gaussian classifier results were done using voting member committees.

### 3.3.4 Conclusion

The simplest model of them all, a linear Gaussian classifier, did the best on this problem. This is not an unusual finding in cases with many input variables and few observations (we had only 22 observations). None of the other models tested here were significantly (95%) better than a random guess of the cancer types.

### 3.4 The model used at Skövde University

Henrik Jacobsson

As mentioned above, the data consisted of 22 training examples, 18 test examples and 1553 variables. The variables was either instantiated with a value typically in the range  $10^3$  to  $10^5$  or with a -1. -1 was interpreted as a missing value. All, except 88 variables had missing values. The data was sorted according to how many missing values it had. 88 variables had no missing values, these were considered as more informative than the other variables.

To reduce the data we chose to use a principal component analysis (PCA). We extracted 21 principal components from the data. This is quite straightforward for the 88 variables with no missing data, but missing data cannot be handled by the PCA. The ANN was then trained on this preprocessed data.

The model we used for training on this preprocessed data was a feedforward neural network. We tried with several different topologies, output representations and training parameters. The target values of the test set was at this time unknown to us and the models were all evaluated on the training set. When the full test set became available, it turned out that none of our models generalized to the test set at all. After several attempts to train the networks on this model we tried to incorporate the variables with missing data in the principal component analysis. There are no given method to do this, so we tried a few *ad hoc* techniques. No significant effect on the results could be seen however. As we did not achieve any good results, the parameters of the training algorithm are not presented in detail.

It is important to bear in mind that the known classification of the test examples were not given to us. This made it quite difficult to determine whether a specific network performed well in terms of generalization etc. It might also be worth noting that by having only 22 training examples for 1553 variables, any significant results or reliable networks are highly unlikely to be found using this type of model.

## 3.5 Discussion

Anders Holst

This task is, as mentioned above, a quite hard problem due to the extreme under-determination of the model. The methods that gave reasonable results here were all very simple models with few parameters or very strong regularization: PLS with four latent variables in Alaiya *et al.* [2000] (11/18 correct), a linear Gaussian classifier at Halmstad University (12/18 correct), and a naive Bayesian classifier at SICS (13/18 correct).

These approaches all gave approximately the same results, and the differences between them can not be considered very significant. It seems feasible to believe that it is hard to get any further on exactly this data set, with any method. Specifically since the classes are not really homogeneous: there seems to be several types of cancer in each of the three classes, some of which occur only once in the 40 samples.

## 3.6 References

Alaiya A., Franzén B., Hagman A., Silfverswärd C., Moberger B., Linder S., and Auer G. (2000). Classification of human ovarian tumors using multivariate data analysis of polypeptide expression patterns. *International Journal of Cancer* **86**: 731–736.

