Recognition of Traffic Signs
Exam assignment
Statistical Methods for Machine Learning

Kasper Nybo Hansen
`nybo@diku.dk`

April 7, 2011

Question 1

The downloaded dataset contains 26640 vector samples, where each vector is of dimension 1568×1. The 26640 samples is distributed throughout 43 classes, where each class is a different traffic sign. The samples are HOG features of the observed traffic sign.

Each folder in the downloaded dataset, `HOG_01`, corresponds to one class. Each file corresponds to one sample. The number of samples, are not equal distributed throughout the classes. I.e. the number of samples taken from each of the 43 traffic signs are not equal. Figure 1 shows the *normalized* histogram of the class frequencies found in the dataset.

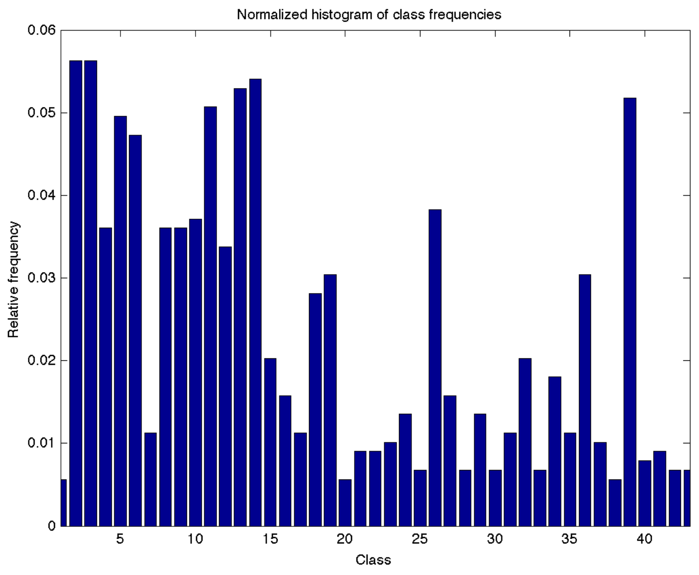


Figure 1 – Histogram of normalized frequencies of the classes found in the dataset.

The source code to this question can be found in the files `loadDataSet`, `initializaDataSet.m` and `q1.m`. All the following implementations, assume that `initializaDataSet.m` has been run.

Question 2

Principal Component analysis (PCA) can be used for dimensionality reduction, giving the possibility to visualize parts of a high dimensional dataset. The dimensionality reduction is done by finding principal components of a dataset, an projecting the data on to these. It is important to note, that only a part of the variance in the dataset will be visualized this way.

I have done PCA on the dataset, X, by first centering the dataset i.e

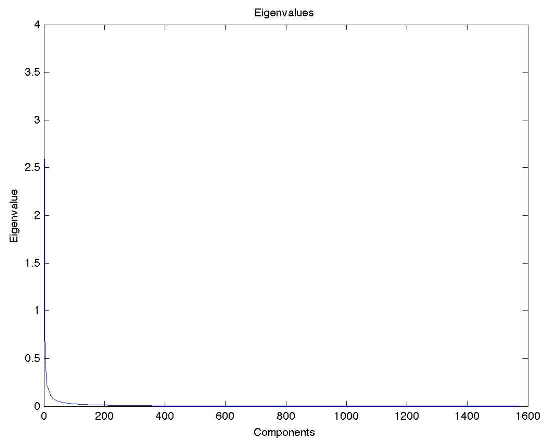
$$B = X - \bar{X} \tag{1}$$

where $\bar{X}$ is the mean vector of X. I then calculate the covariance matrix C of B. The eigenvectors of the covariance matrix is the PCA components, and the eigenvalues describes the variation contained in the PCA components.

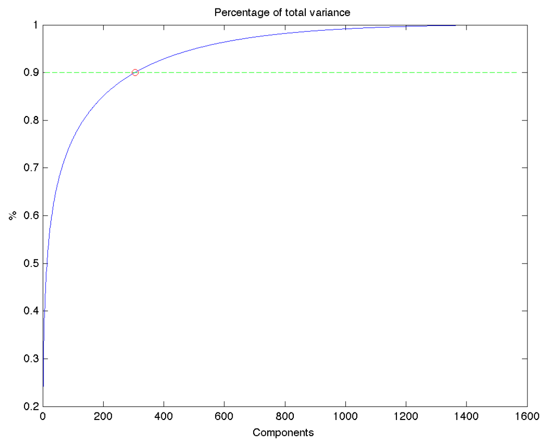
I calculate the eigenvalues and eigenvectors of C. The eigenvalues are stored as the diagonal elements of a matrix D, and the eigenvectors are stored column wise in the matrix V. I then sort the eigenvalues in descending order, making sure that the eigenvalues matrix V are sorted the same way. The $i'th$ column in V is now the $i'th$ PCA component.

The eigenvalues in D contains information about the variation in the corresponding eigenvector in V. The total variance is calculated as the sum of all the diagonal elements in D. A plot can be made showing the percentage of total variance in the first k components, by calculating the cumulative sum of D, and plotting this as a function of the number of components. An example of such a plot, can be seen in figure 2(b).

Figure 2(a) shows a plot of the eigenvalues of the sorted components. Figure 2(b) shows that 90% of the variance is contain within the first 305 PCA components (This is also confirmed programmatically!).



(a) Eigenvalues of the sorted components



(b) 90% of the variance is contained within the first 305 principal components.

I have projected the dataset on to the first and second PCA components, by multiplying the transposed two principal components on to the centered dataset. I.e.

$$Y = E^T * B \tag{2}$$

where E is a matrix consisting of two column vectors representing the two PCA components, and B is the centered dataset. The result can be seen on figure 2

The source code to this question can be found in the file `q2_3.m`.

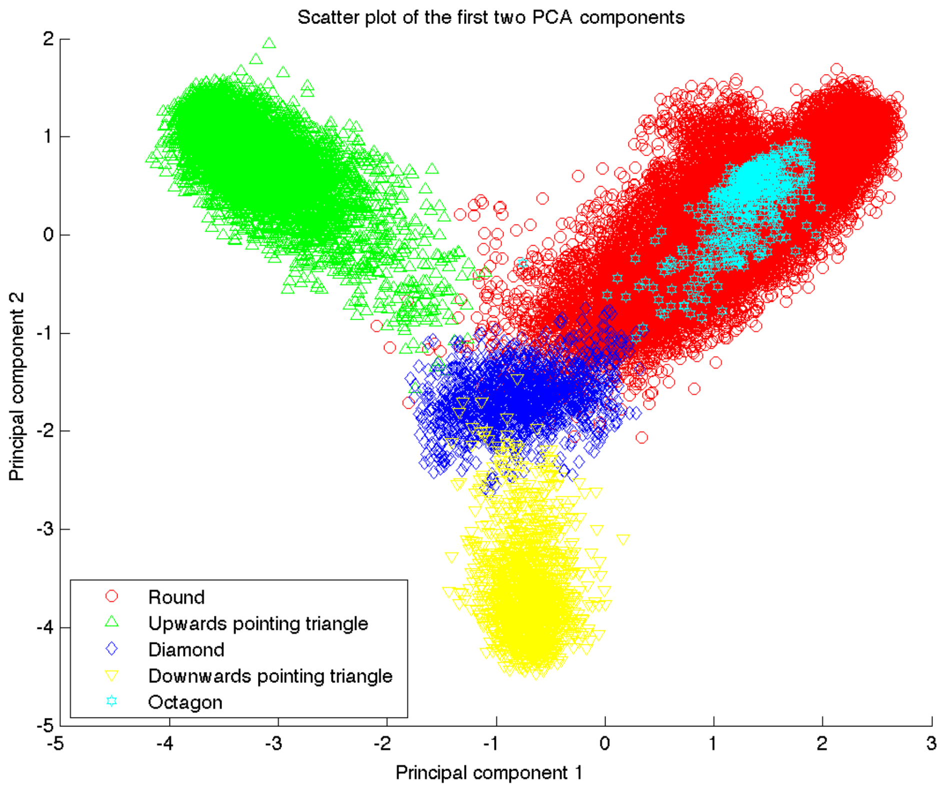


Figure 2 – The dataset projected on to the first two PCA components. Each color/shape represent a different class. There are in total five classes.

Question 3

K-means is an unsupervised clustering algorithm. Given a K and a dataset, the algorithm partitions the data into K clusters. Unsupervised means that it requires no information about the features present in the dataset.

Let the dataset consists of sample vectors of size $\mathbb{R}^d$. Then the algorithm is initialized by creating K arbitrary centroids all belonging to $\mathbb{R}^d$. Let each centroid has it's own label.

For each sample vector in the dataset, the distance to all the centroids is calculated. The label of the centroid closest to the datapoint determines the label of the sample vector. When all sample vectors have been assigned a label, the centroids are recalibrated.

Recalibration is done by moving each centroid to the mean of the sample vectors having it's label.

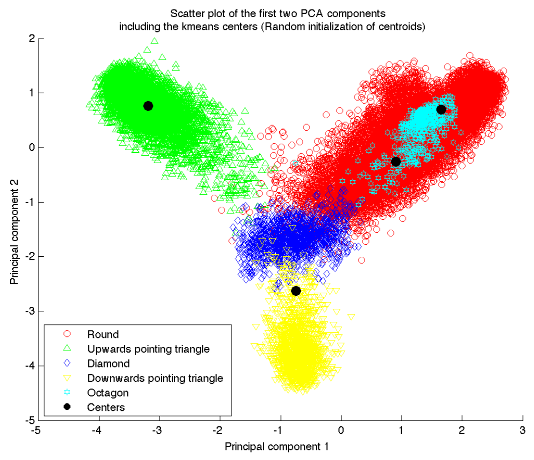
When the recalibration is done, the algorithm restarts and for each sample vector, the distance to the new centroids are calculated. The sample vector is again given the label of the closest centroid.

The algorithm stops when the centroids doesn't move.

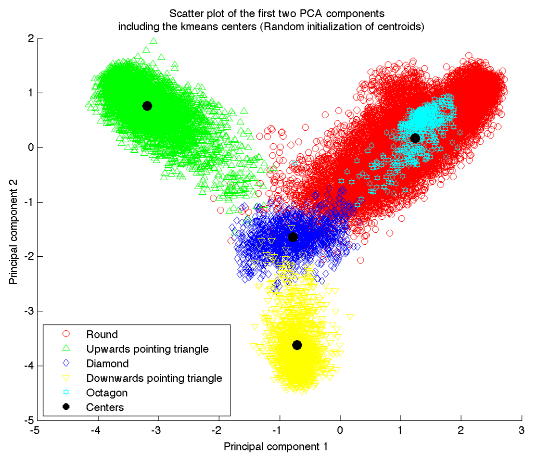
For this question I have made my own implementation of the K-means algorithm. The source code to this algorithm can be found in the file `kmeans.m`.

I have applied my implementation to the dataset from the previous assignment. I applied it with $K = 4$. The result can be seen on figure 3(a). The figure shows the found centroids projected on to the first two PCA components. The algorithm does finds the green class nicely, but it is clear that the three other centroids are placed a bit of, compared to the original classes.

The reason for the misplacement of some of the centroids, is the K-means initialization step. The K-means algorithm is very sensitive when it comes to the initial guess of the centroids of the clusters. The difference between figure 3(a) and 3(b) shows this. Figure 3(a) and 3(b) are both made by initializing the algorithm with 4 randomly selected sample vectors as centroids. It is clear that the cluster centroids are placed very differently between the two runs.



(a) Random initialization of cluster centroids. Two centroids inside 'red/cyan' class, none in 'blue' class.



(b) Random initialization of cluster centroids. One centroid inside 'red/cyan' and one in 'blue' class.

Figure 3 – Illustration of the sensitivity in the initialization process. Both figures where made by choosing the cluster centroids randomly from the dataset, but finds completely different centroids.

The source code to this question can be found in the file `q2_3.m`.

Question 4

In this question I will relate the different types of overfitting described in hunch.net[4] to this exam set.

Traditional overfitting can occur if we train e.g. the multi-class classification on only a very small subset of the the total training set. Luckily for us, we have plenty of data to play around with so this shouldn't be a problem.

Parameter tweak overfitting can occur if we adjust the parameters of e.g. SVM to perform good on the test set. Parameter tweak overfitting would also occur if we didn't make sure that all 30 instances of the same physical sign is placed in the same subset of the cross validation. By not insuring this is the case, we could train our model on something that is very 'similar' to the test set.

Brittle measure can occur if we instead of 5-fold cross validation, did n-fold cross validation, where n is the number of samples. This type of cross fold validation is also known as *Leave-one-out cross-validation*.

Choice of measure can occur if the accuracy calculation is calculated differently. HOW?

Incomplete Prediction can occur if we used the models found in the binary classification in the multi-class classification.

Data set selection, say that the performance of the kNN was especially good in one of the folds in the 5 fold cross validation. We could then choose this subset as the whole test set and get better results.

I don't see how the following methods of overfitting can occur: **Bad statistics**, **Human-loop overfitting** and **reprobleming** in this ecxam assignment. It is hard to do reprobleming, since the assignment text is very explicit. Theres no humans involved in the training nor testing phase, so Human-loop overfitting seems impossible.

Fold no.	Accuracy
1	0.87962
2	0.88947
3	0.89814
4	0.89298
5	0.86481
Mean accuracy	0.88500

Table 1 – This table shows the average accuracy over a 5-fold cross validation when running LDA on the entire test dataset.

Question 5

In this question I have filtered the training set, such that only the classes 1 and 5 are present.

Cross-validation

Each physical traffic sign has 30 sample vectors present in the dataset. This causes problems, because this can lead to overoptimistic results. Although we are not given the details of the HOG features, it is clear that the 30 sample vectors taken from the same physical traffic sign is positioned close together in the 1568 dimensional space.

This poses a problem, since, unintentionally, we can train on a subset of a physical sign, and test on the other part of the same subset. This would definitely lead to overoptimistic results and should be avoided.

I have partitioned the training data, by randomizing the physical signs into 5 bins. After the randomization, each sign is replaced with it's 30 samples. This ensures that the same physical sign is placed in the same cross validation subset.

The cross validation is done as a 5-fold cross validation. More precisely I have done cross validation in the following way. Let B denote the bin numbers, e.g. $B = \{1, 2, 3, 4, 5\}$. Foreach bin $b_i \quad i \in B$ i have used the samples in b_i as test data, and samples in the rest of the bins as training data. Thus each subset of the 5-fold cross validation acts as a test set once, while acting as part of a training set 4 times.

Computation of test errors

When doing 5-fold cross validation, the result is 5 intermediate results. I have used the mean of these 5 intermediate result, as a measurement of how good the classifier is. In the binary case, I have measured the error as a sum of the number of misclassifications the classifier makes. I.e. foreach classification the classifier makes, I compare the result with the label of the training set, and if the two are different I consider it as an misclassification.

The accuracy is measured as

$$\text{accuracy} = \text{misclassification}/\text{total number of labels} \tag{3}$$

Linear classification

I have chosen Linear Discriminant analysis (LDA) as my linear classification method. I have chosen this method because it gives good result in practice and is highly recommended as a baseline method[6].

In group assignment 2, we made a LDA implementation[7]. I tried solving the classification problem with this implementation, but the performance was very bad. Instead I substituted my own implementation with the MATLAB function `classify` and the results where very much improved. The result of doing binary classification with MATLAB's LDA implementation can be seen in table 1. The table shows the accuracy for each of the 5-folds cross validation runs, and the mean accuracy.

The source code to this part of the assignment can be found in `q5_1.m`.

i/C	0.1	1	10	100	1000	10000
-3	67.33138	87.08966	91.66861	91.85380	91.89083	91.89083
-1	81.30409	91.61208	92.60623	92.71734	92.71734	92.71734
-0	84.55555	92.33723	93.07017	93.07017	93.07017	93.07017
1	86.25341	92.74658	93.38986	93.38986	93.38986	93.38986
3	64.77582	88.99415	89.35672	89.35672	89.35672	89.35672

Table 2 – Accuracy given different SVM model parameters. The model is most accurate when $i = 1$ and $C = 10$, yielding a accuracy of $\approx 93\%$

Non-linear classification

For the non linear classifier I have chosen a Support Vector Machine (SVM) with a Gaussian Radial Basis Function - the same approach as we used in a previous group assignment[8]. I have chosen to use SVM, because it performed well in the group assignment, and is highly recommended by the course responsible.

The gaussian kernel is on the form

$$k(x, z) = e^{-\gamma||x-z||^2} \tag{4}$$

where $\gamma > 0$ is a bandwidth hyperparameter that can be tweaked accordingly. The SVM also has a cost parameter C that can be tweaked. As in the previous group assignment, we can use

$$\sigma = \sqrt{1/(2\gamma)} \tag{5}$$

instead of γ. We can then compute γ from 5 as

$$\gamma = 1/(2\sigma^2) \tag{6}$$

Jaakkola's heuristic, provides a reasonable guess of the bandwidth parameter σ[8]. Let

$$G = \{||x_i - x_j|| \quad | \quad (x_i, y_i), (x_j, y_j) \in S \wedge y_i \neq y_j\} \tag{7}$$

for a training set S. I.e G is the difference in input space between all pairs in S. Jaakkola's heuristic can then be computed as the median of G.

For finding the appropriate SVM hyperparameters I have made a grid search. The grid can be seen in table 2.

As can be seen from table 2 the model is most accurate when $i = 1$ and $C = 10$. The accuracy when running the model with these hyperparameters is $\approx 93\%$.

The source code to this part of the assignment can be found in `q5_2.m` and `calcSVMModel.m`. Note the implemented code needs the LIBSVM framework[5].

Question 6

For this question, I have chosen to use linear description method (LDA) as my linear classifier and K nearest neighbor (kNN) as my non-linear classification method. In both cases I have used 5-fold cross validation as described in question 5. Furthermore I have calculated the accuracy of the method as described in question 5.

Linear classification

For my linear classifier I have chosen to use LDA, again using the MATLAB function `classify`. The result if running the LDA can be seen in table 3.

The source code to this question can be found in the file `q6_1.m`.

Fold no.	Accuracy
1	0.91412
2	0.93913
3	0.93333
4	0.92453
5	0.92322
Average	0.92686

Table 3 – This table shows the average accuracy over a 5-fold cross validation when running LDA on the entire test dataset.

k	Average accuracy over 5-fold cross validation
1	0.69996
2	0.69483
5	0.70937
10	0.71256
15	0.71167
20	0.71329
30	0.71262
40	0.71063
50	0.70920
100	0.69793
250	0.66310

Table 4 – This table shows the average accuracy over a 5-fold cross validation when running kNN on the entire test dataset. As can be seen $k = 20$ yields the best accuracy.

Non-linear classification

As my non-linear classification method, I have chosen the basic kernel known as the k nearest neighbor (knn). K nearest neighbor is a very versatile, yet simple, algorithm that usually performs pretty good regardless of the data it is working on. It is actually in the top 10 data mining algorithms[10].

The algorithm works as follows. The user specifies a k to the algorithm and a data and test set. For each point in the test set, the k nearest neighbors of training set are found. The most frequently label of the k nearest neighbors becomes the label of the test point. This is also the reason why kNN can be slow. It is computational very expensive.

K nearest neighbor is a lazy algorithm. This mean that the algorithm doesn't create a model for the training set.

k is a parameter that can be tweaked to suit the dataset. In order to find an appropiate k i have done 5-fold cross validation, on different k values. The result can be seen in table 4. Each row in the table corresponds to a 5-fold cross validation with the k specified in column 1. As can be seen the most accurate result is archieved when $k = 20$

I have implemented the kNN algorithm in the file `kNN.m`. Other source code files related to this question includes `knnHelper.m` and `q6_2.m`.

Question 7

I have tested the previous implemented classifiers on the supplied test set. For each classifier implemented in the previous assignments, I have chosen the best performing model. I have then used this model to classify the traffic signs in the the test set.

The following sub sections elaborate on the results.

Testing binary classification using LDA

In the testing phase, the 5-fold cross validation yielded a mean accuracy of 0.88500. I have testet the implementation on the test set by using the whole training set as training, and the test set as test. The result is a average accuracy of 0.92196. The classifier actually performs 3.596% better on the test set than the training set.

The source code to this question can be found in `q7_1.m`.

Testing binary classification using SVM

I found a SVM model with hyperparameters, $i = 1$ and $C = 10$, to perform best in question 5. I have therefore used these parameters in the testing phase. Running the SVM with these chosen hyperparameters and the whole training set as training data, and the test set as test data I get an accuracy of 0.93389. Compared to the mean accuracy off the 5-fold cross validation this is 4.35% better than the training phase.

The source code to this question can be found in `q7_2.m`.

Testing multi-class classification with LDA

The accuracy of the multi-class LDA classifier was 0.92686. When i run the LDA classiffier with the whole traning setas training data, and the whole test set as test data, I get an accuracy of 0.94510. The classifier is thus 1.824% more accurate on the test set.

The source code to this question can be found in `q7_3.m`.

Testing multi-class classification using kNN

As the non linear method, i made use of kNN when finding doing multi-class classification. From Question 6, I have that the kNN performed best when $k = 20$. I have therefore chosen to set $k = 20$ when classifying the test set.

Again the whole training set is used as training, and the whole test set is used as test data. The result of classifying the test set is a accuracy of 0.76776, which is 5.447% better than the average of the 5-folds cross validation yielded.

The source code to this question can be found in `q7_4.m`.

Overall performance

All the implementations performed better on the testing data than the training data. One possible explanation for this behavior could be that we use the whole test set as the testing data, Thereby dramatically increasing the number of training points.

It comes to no surprise that the LDA implementations archives a high degree of accuracy. Looking at the results page of the *The German Traffic Sign Recognition Benchmark*[3] the combination of LDA and the `HOG01` test set is scoring high.

What comes as a surprise, is how good the kNN implementation performs compared to how simple it is. Even in the multi-class classification the kNN archives a accuracy of 76%.

Question 8

I have used linear regression to solve the binary classification and multi-class classification problem. In both cases i have used 5-fold cross validation, partitioning the data is explained in question 5. Furthermore I have calculated the accuracy as explained in question 5.

I have constructed the design matrix as defined in Bishop 3.16[1]. Each row of the design matrix is a sample vector. The *Moore-Penrose pseudo inverse* of the design matrix is calculated by and the maximum

Fold no.	Accuracy binary case	Accuracy multi-class case
1	0.87962	0.13032
2	0.83157	0.10244
3	0.91666	0.10168
4	0.88421	0.10468
5	0.92592	0.12209
Average	0.88760	0.11224

Table 5 – Performance of using linear regression when doing binary and multi-class classification. In average the classifier is correct in $\approx 88\%$ of the cases when doing binary classification, and $\approx 11\%$ when doing multi-class classification.

likelihood is calculated as Bishop 3.15. Finally the model is applied to each test set as defined in 3.3 of Bishop.

The *Moore-Penrose pseudo inverse* is computational expensive to calculate. In order to speed things up, I have used a alternative implementation taken from[2]. The optimized Moore-Penrose pseudo inverse function can be found in the file `geninv.m`.

The result can be seen in table 5

Compared to the result we got in question 5 (Binary case), the abuse of regression for doing classification, actually performs pretty good. In Average the regression method is 0.88760 accurate, whereas the best binary classifier SVM was 0.93389 accurate. This means that the SVM is only 4.269% better, compared to the regression method.

If we compare the regression method with the LDA implementation from the binary case, then the LDA is only 3.925% better.

As can be seen from table 5, the regression method fails miserably when used on the multi-class case. The best classifier in the multi-class case was LDA. LDA archived an accuracy of 0.92686. This is 81.462% better than regression method.

The source code to this question can be found in the files `regression.m`, `q8_1.m` and Q8_2.

Question 9

We know that 1 out of 10^5 ROIs is a traffic sign. Let $P(A)$ define the probability of observing a traffic sign, then $P(A) = 10^{-5}$. Let $P(!A)$ define the probability of not observing a traffic sign, then $P(!A) = 1 - 10^{-5}$. Let $P(B)$ define the probability that the classifier classifies the ROI as belonging to the positive class. Let $P(!B)$ be the probability that the classifier classifies the ROI as belonging to the negative class.

It is stated that the probability, that the classifier classifies a positive class as positive is 99%[9]. Thus $P(B|A) = 0.99$ and $P(!B|A) = 0.01$.

Furthermore it is stated that the probability, that the classifier classifies a negative class as negative, is 99%[9]. Thus $P(!B|!A) = 0.99$ and $P(B|!A) = 0.01$.

The probability of a observing a sign, $P(B)$, can then be calculated as the sum

$$P(B) = P(B|A)P(A) + P(B|!A)P(!A) = 0.99 \cdot 10^{-5} + 0.01 \cdot (1 - 10^{-5}) = 0.0100098 \tag{8}$$

With 10^6 ROI's this yields that 1.00098% would be classified as a sign corresponding to ≈ 10010 ROI's classified as positive.

Bayes' theorem states

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{9}$$

The probability of a false positive, $P(!A|B)$, can be calculated by (9)

$$P(!A|B) = \frac{P(B|!A)P(!A)}{P(B)} = \frac{0.01 \cdot (1 - 10^{-5})}{0.0100098} = 0.999010969 \tag{10}$$

So given 10010 ROIs classified as positive, the expectation is that 99.9010969% is false positives. This corresponds to ≈ 10000. Note that I have rounded the numbers, since fractions of a ROI doesn't make sense in this scenario.

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] P. Courrieu. Fast computation of moore-penrose inverse matrices. *CoRR*, abs/0804.4809, 2008.

[3] http://benchmark.ini.rub.de/index.php?section=results, april 2011.

[4] http://hunch.net/?p=22, april 2011.

[5] http://www.csie.ntu.edu.tw/ cjlin/libsvm/matlab, april 2011.

[6] C. Igel. Teaching material. linear classification statistical methods for machine learning. February 2011.

[7] C. Igel and K. S. Pedersen. Assignment 2: Basic learning algorithms. February 2011.

[8] C. Igel and K. S. Pedersen. Assignment 3: Neural networks and support vectpr machines. March 2011.

[9] C. Igel and K. S. Pedersen. Statistical methods for machine learning. final exam assignment. recognition of traffic signs. March 2011.

[10] X. Wu, V. Kumar, J. Ross, Q. Joydeep, G. Q. Yang, H. Motoda, G. J. Mclachlan, A. Ng, B. Liu, P. S. Yu, D. Steinberg, X. W. (b, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, and H. Motoda. Doi 10.1007/s10115-007-0114-2 survey paper top 10 algorithms in data mining, 2007.