Classification

Abstract

Classification consists of predicting a certain outcome based on a given input. Support Vector Machine (SVM) is a new promising technique for pattern classification. In this project, we develop a classification model for specific datasets using Matlab. Simulations are given for further analysis and conclusions.

1. Introduction

Classification consists of predicting a certain outcome based on a given input. Classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.

In the terminology of machine learning, classification is considered an instance of supervised learning, i.e. learning where a training set of correctly identified observations is available.

Common algorithms for classification include linear classifiers (logistic regression, naive Bayes classifier), support vector machine, k-nearest neighbour, neural networks and so on. In this project, we mainly focus on support vector machine(SVM).

2. Background

There are several best-known supervised classification techniques in machine learning. In the paper *Supervised Machine Learning: A Review of Classification Techniques (S. B. Kotsiantis, 2007)*, the author completes a review for the major supervised machine learning classification algorithms. The performances of each classification algorithms are demonstrated as follow:

	Decision Trees	Neural Networks	Naïve Bayes	kNN	SVM	Rule-learners
Accuracy in general	**	***	*	**	****	**
Speed of learning with respect to number of attributes and the number of instances	***	*	****	****	*	**
Speed of classification	****	****	****	*	****	****
Tolerance to missing values	***	*	****	*	**	**
Tolerance to irrelevant attributes	***	*	**	**	****	**
Tolerance to redundant attributes	**	**	*	**	***	**
Tolerance to highly interdependent attributes (e.g. parity problems)	**	***	*	*	***	**
Dealing with discrete/binary/continuous attributes	****	***(not discrete)	***(not continuous)	***(not directly discrete)	**(not discrete)	***(not directly continuous)
Tolerance to noise	**	**	***	*	**	*
Dealing with danger of overfitting	**	*	***	***	**	**
Attempts for incremental learning	**	***	****	****	**	*
Explanation ability/transparency of knowledge/classifications	****	*	****	**	*	****
Model parameter handling	***	*	****	***	*	***

Table 4. Comparing learning algorithms (**** stars represent the best and * star the worst performance)

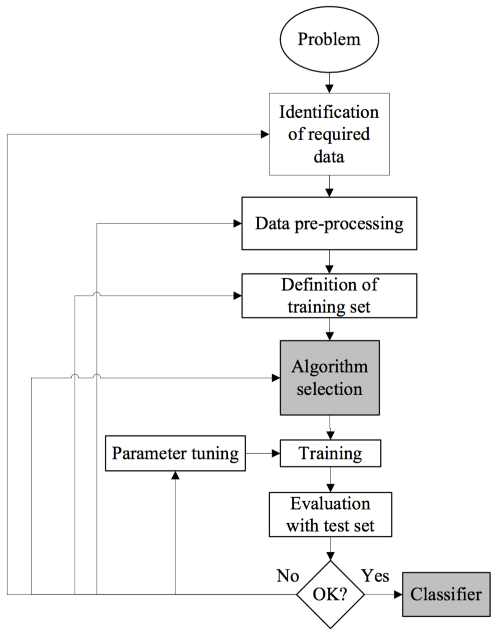
Based on the result table shown above, in order to achieve high accuracy and overall good performances for general datasets, it is better to choose SVM as our model for the project.

Support Vector Machines (SVMs) are the one of the newest supervised machine learning techniques according to the paper. Thus, in this study apart from a brief description of SVMs we will refer to some more recent works and the landmark that were published before these works. SVMs revolve around the notion of a "margin"—either side of a hyperplane that separates two data classes. Maximizing the margin and thereby creating the largest possible distance between the separating hyperplane and the instances on either side of it has been proven to reduce an upper bound on the expected generalisation error.

Sequential Minimal Optimization (SMO) is a simple algorithm that can quickly solve the SVM quadratic programming problem without any extra matrix storage and without using numerical quadratic programming optimization steps at all. SMO decomposes the overall quadratic programming problem into quadratic programming sub-problems.

3. Process

The process of building a classification model is to construct a set of rules from instances (examples in a training set), or more generally speaking, creating a classifier that can be used to generalize from new instances. The process can be described as follow:



The first step is collecting the dataset. It contains in most cases noise and missing feature values, and therefore requires significant pre-processing.

The second step is the data preparation and data pre-processiong. Depending on the circumstances, there is a variety of procedures for sampling instances.

The third step is definition of training set. This process is to identify and remove as many irrelevant and redundant features.

In this project, the datasets have been given and thus these three steps have been solved. Each file in the folder contains a dataset (independent of the datasets in other files). Each file contains two matrices, class 1 and class 2. The column vectors in class are the feature vectors observed in Class.

The fourth step is the selection of algorithm. It is critical for the classifier because it determines the overall performance of the model.

As is mentioned above, the classification algorithm to be implemented will be SMO for SVM.

The fifth step is training and parameter tuning. In this project, we will split each dataset and use four-fifths of the data for training. The major parameters for SVM are C(regularization parameter), tolerance, maximum iteration and sigma.

The six step is evaluation using the remaining one-fifth of each dataset. We will measure the accuracy rate and total number of errors.

4. Model & Algorithm

In the linear case, the margin is defined by the distance of the hyperplane to the nearest of the positive and negative examples. The formula for the output of a linear SVM is

$$u = \vec{w} \cdot \vec{x} - b,$$

The separating hyperplane is the plane u=0. The nearest points lie on the planes u = ±1. The margin m is thus

$$m = \frac{1}{\|w\|_2}.$$

Maximizing margin can be expressed as

$$\min_{\vec{w},b,\vec{\xi}} \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{N} \xi_i \text{ subject to } y_i(\vec{w}\cdot\vec{x}_i - b) \geq 1 - \xi_i, \forall i,$$

where ξi are slack variables that permit margin failure and C is a parameter which trades off wide margin with a small number of margin failures.

In SVM, there is a one-to-one relationship between each Lagrange multiplier and each training example. Once the Lagrange multipliers are determined, the normal vector w and the threshold b can be derived from the Lagrange multipliers:

$$\vec{w} = \sum_{i=1}^{N} y_i \alpha_i \vec{x}_i, \quad b = \vec{w} \cdot \vec{x}_k - y_k \text{ for some } \alpha_k > 0.$$

The output of a non-linear SVM is explicitly computed from the Lagrange multipliers:

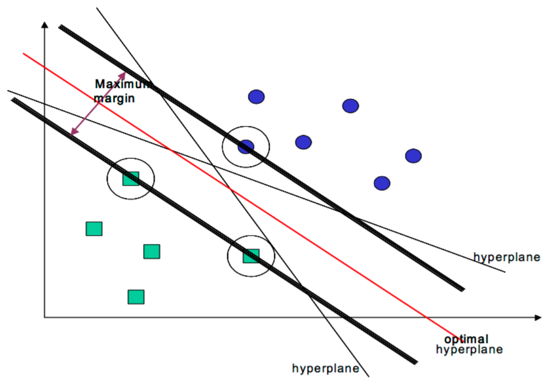
$$u = \sum_{j=1}^{N} y_j \alpha_j K(\vec{x}_j, \vec{x}) - b,$$

where K is a kernel function that measures the similarity or distance between the input vector x and the stored training vector xj . Examples of K include Gaussians.

The Karush-Kuhn-Tucker (KKT) conditions are necessary and can be evaluated on one example at a time, which will be useful in the construction of the SMO algorithm. The problem is solved when, for all i:

$$\begin{aligned} \alpha_i = 0 &\Leftrightarrow y_i u_i \geq 1, \\ 0 < \alpha_i < C &\Leftrightarrow y_i u_i = 1, \\ \alpha_i = C &\Leftrightarrow y_i u_i \leq 1. \end{aligned}$$

where ui is the output of the SVM for the ith training example. For case 1, it indicates that ai is the correct classification. For case 2, it indicates that ai is a support vector point which lies on its margin. For case 3, it indicates that ai is a point between the two margins.



Without loss of generality, the algorithm first computes the second Lagrange multiplier α2 and computes the ends of the diagonal line segment in terms of α2.

$$\begin{cases} L = \max(0, \alpha_j - \alpha_i), H = \max(C, C + \alpha_j - \alpha_i) & \text{if } y_i \neq y_j \\ L = \max(0, \alpha_j + \alpha_i - C), H = \max(C, \alpha_j - \alpha_i) & \text{if } y_i = y_j \end{cases}$$

Therefore, SMO computes the minimum along the direction of the constraint :

$$\alpha_2^{\text{new}} = \alpha_2 + \frac{y_2(E_1 - E_2)}{\eta},$$

where Ei = ui - yi is the error on the ith training example. As a next step, the constrained minimum is found by clipping the unconstrained minimum to the ends of the line segment:(s=y1*y2)

$$\alpha_2^{\text{new,clipped}} = \begin{cases} H & \text{if} \quad \alpha_2^{\text{new}} \geq H; \\ \alpha_2^{\text{new}} & \text{if} \quad L < \alpha_2^{\text{new}} < H; \\ \text{L} & \text{if} \quad \alpha_2^{\text{new}} \leq L. \end{cases}$$

$$\alpha_1^{\text{new}} = \alpha_1 + s(\alpha_2 - \alpha_2^{\text{new,clipped}}).$$

The threshold b is re-computed after each step, so that the KKT conditions are fulfilled for both optimized examples.

$$b_1 = E_1 + y_1(\alpha_1^{\text{new}} - \alpha_1)K(\vec{x}_1, \vec{x}_1) + y_2(\alpha_2^{\text{new,clipped}} - \alpha_2)K(\vec{x}_1, \vec{x}_2) + b.$$
$$b_2 = E_2 + y_1(\alpha_1^{\text{new}} - \alpha_1)K(\vec{x}_1, \vec{x}_2) + y_2(\alpha_2^{\text{new,clipped}} - \alpha_2)K(\vec{x}_2, \vec{x}_2) + b.$$

$$b := \begin{cases} b_1 & \text{if } 0 < \alpha_i < C \\ b_2 & \text{if } 0 < \alpha_j < C \\ (b_1 + b_2)/2 & \text{otherwise} \end{cases}$$

Summing up all the new values for each iteration from calculation (new a,b,y,k(xj,x)), the outcome for u can be computed. If u>=0 then the sample is predicted to be in Class A. If u<0 then the sample is predicted to be in Class B.

5. Implementation

According to SVM model and SMO algorithm mentioned above, we can implement the classifier for this project using the following pseudo-code:

```
1) Introduce positive Lagrange multipliers, one for each of the inequality constraints (1). This gives Lagrangian:
   $L_P \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i y_i (x_i \cdot w - b) + \sum_{i=1}^{N} \alpha_i$
2) Minimize $L_P$ with respect to w, b. This is a convex quadratic programming problem.
3) In the solution, those points for which $\alpha_i > 0$ are called "support vectors"
```

5.1 modelTraining

The function modelTraining() mainly uses SMO algorithm to train the SVM model. The receiving matrix X is the raw dataset. Another matrix Y is the classification result dataset with number 1 and 0 (representing Class A and Class B). C is a parameter for controlling errors. Tol represents the parameter for tolerance. We set the iteration for Lagrangian as 5. As for the kernel function, we use Gaussian Kernel for more precise learning. Using the steps mentioned above, we can implement the training section for the classifier. At the end of the function, the model structures will be save as m files in the "model" folder. Parameters includes alpha(a), b, the training matrix X, result matrix y and normal vector w.

5.2 modelTesting

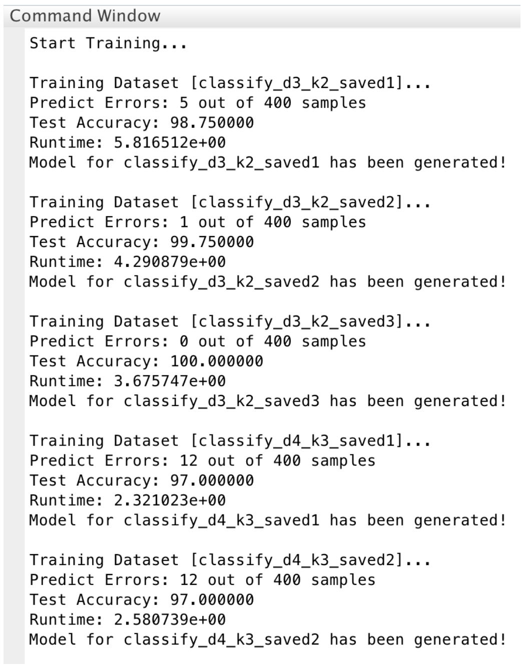
The function modelTesting() mainly uses the trained model to complete tests and evaluations based on the prediction results. It returns a vector of predictions with 0 or 1. Function bsxfun() Apply element-wise operation to two arrays with implicit expansion. We will vectorized the model with model parameters from the trained model structures so as to calculate the kernel. If the prediction result is no less than 0 then the sample is predicted to be in Class A. Otherwise the sample is predicted to be in Class B.

5.3 mainClassification

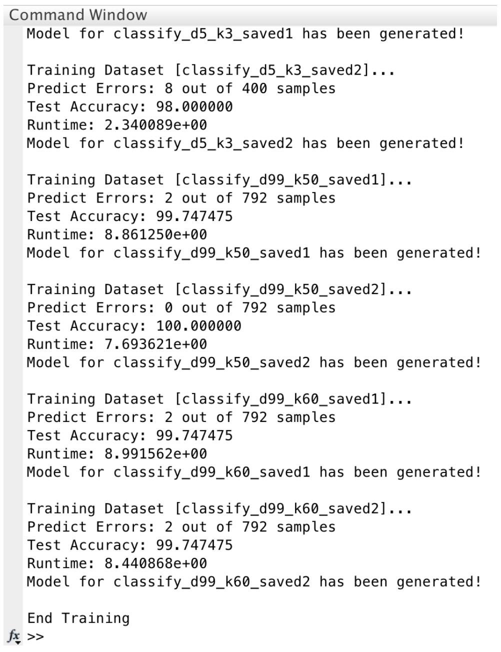
The main file acts as the main classifier. It will classify all the datasets given as the original resources. First of all, the classifier load the given dataset. There are some pre-processing operations for the raw data. For convenience only, the data matrix will do transpose operation. During the training phase, 4/5 of the dataset from both A and B class will be assigned to training matrix M. The rest 1/5 of the data will be mapped to 1 and 0 (for Class A and B) and stored in matrix N. During the testing phase, 4/5 of the dataset from both A and B class will be assigned to training matrix T. The rest 1/5 of the data will be mapped to 1 and 0 (for Class A and B) and stored in matrix R. By using the modelTraining and modelTesting functions, models will be generated and stored and prediction p will be created. The evaluation phase is to count the total number of errors in the prediction matrix and further calculate the accuracy for the implemented SVM model.

6. Results & Performance

For performance evaluation, we test the trained models using 11 given datasets. We mainly focus on the performance of accuracy rate and runtime(seconds). Test results are demonstrated as follow:



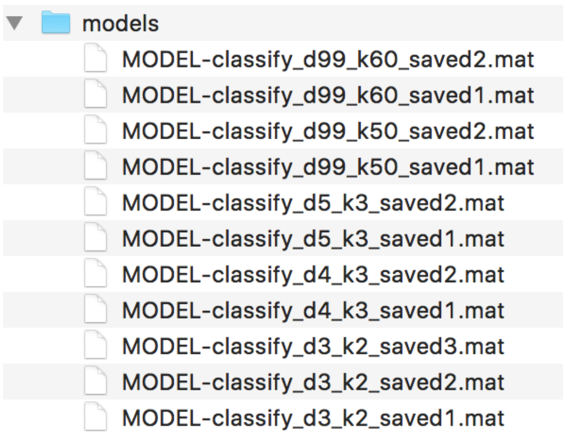
```
Command Window
Start Training...

Training Dataset [classify_d3_k2_saved1]...
Predict Errors: 5 out of 400 samples
Test Accuracy: 98.750000
Runtime: 5.816512e+00
Model for classify_d3_k2_saved1 has been generated!

Training Dataset [classify_d3_k2_saved2]...
Predict Errors: 1 out of 400 samples
Test Accuracy: 99.750000
Runtime: 4.290879e+00
Model for classify_d3_k2_saved2 has been generated!

Training Dataset [classify_d3_k2_saved3]...
Predict Errors: 0 out of 400 samples
Test Accuracy: 100.000000
Runtime: 3.675747e+00
Model for classify_d3_k2_saved3 has been generated!

Training Dataset [classify_d4_k3_saved1]...
Predict Errors: 12 out of 400 samples
Test Accuracy: 97.000000
Runtime: 2.321023e+00
Model for classify_d4_k3_saved1 has been generated!

Training Dataset [classify_d4_k3_saved2]...
Predict Errors: 12 out of 400 samples
Test Accuracy: 97.000000
Runtime: 2.580739e+00
Model for classify_d4_k3_saved2 has been generated!
```



```
Command Window
Model for classify_d5_k3_saved1 has been generated!

Training Dataset [classify_d5_k3_saved2]...
Predict Errors: 8 out of 400 samples
Test Accuracy: 98.000000
Runtime: 2.340089e+00
Model for classify_d5_k3_saved2 has been generated!

Training Dataset [classify_d99_k50_saved1]...
Predict Errors: 2 out of 792 samples
Test Accuracy: 99.747475
Runtime: 8.861250e+00
Model for classify_d99_k50_saved1 has been generated!

Training Dataset [classify_d99_k50_saved2]...
Predict Errors: 0 out of 792 samples
Test Accuracy: 100.000000
Runtime: 7.693621e+00
Model for classify_d99_k50_saved2 has been generated!

Training Dataset [classify_d99_k60_saved1]...
Predict Errors: 2 out of 792 samples
Test Accuracy: 99.747475
Runtime: 8.991562e+00
Model for classify_d99_k60_saved1 has been generated!

Training Dataset [classify_d99_k60_saved2]...
Predict Errors: 2 out of 792 samples
Test Accuracy: 99.747475
Runtime: 8.440868e+00
Model for classify_d99_k60_saved2 has been generated!

End Training
>>
```

In fact, the parameter sigma for Gaussian kernel function has been tuned. When sigma is bigger, the accuracy rate is much lower. For example, when sigma = 1, the accuracy rate for smaller datasets will be around 65% to 72% but relatively higher for bigger datasets which will be up to 90%. When sigma is tuned to 0.1, the performance in terms of accuracy will increase dramatically. It shows the importance of parameter tuning in modelling in machine learning.

At the same time, the model files have been automatically generated for further practice as follow:



```
models
    MODEL-classify_d99_k60_saved2.mat
    MODEL-classify_d99_k60_saved1.mat
    MODEL-classify_d99_k50_saved2.mat
    MODEL-classify_d99_k50_saved1.mat
    MODEL-classify_d5_k3_saved2.mat
    MODEL-classify_d5_k3_saved1.mat
    MODEL-classify_d4_k3_saved2.mat
    MODEL-classify_d4_k3_saved1.mat
    MODEL-classify_d3_k2_saved3.mat
    MODEL-classify_d3_k2_saved2.mat
    MODEL-classify_d3_k2_saved1.mat
```

PERFORMANCE OF SVM MODELS USING SMO ALGORITHM		
DATASET NAME	ACCURACY RATE (%)	RUMTIME (s)
classify_d3_k2_saved1	98.75	5.81
classify_d3_k2_saved2	99.75	4.29
classify_d3_k2_saved3	100.00	3.68
classify_d4_k3_saved1	97.00	2.32
classify_d4_k3_saved2	97.00	2.58
classify_d5_k3_saved1	98.00	3.02
classify_d5_k3_saved2	99.25	2.34
classify_d99_k50_saved1	99.75	8.86
classify_d99_k50_saved2	100.00	7.69
classify_d99_k60_saved1	99.75	8.99
classify_d99_k60_saved2	99.75	8.44

7. Conclusion

Classification models and algorithms have different performances under different circumstances. Support vector machines provide powerful techniques for classifications. By constructing SVM classifier using SMO algorithm, the results for classifying the given datasets have been nearly perfect with accuracy rates over 97% and runtime within 10 seconds for all datasets.

8. Reference

[1] S. B. Kotsiantis, “Supervised Machine Learning: A Review of Classification Techniques” Informatica 31 (2007).

[2] John C. Platt,“Sequential Minimal Optimization:A Fast Algorithm for Training Support Vector Machines” Technical Report MSR-TR-98-14,April 21, 1998.

[3]Osuna, E., Freund, R., Girosi, F., "Improved Training Algorithm for Support Vector Machines," Proc. IEEE NNSP ’97, (1997).