Pattern recognition systems – Lab 12

Support Vector Machine – Classification

1. Objectives

In this lab session we will study and implement a simple linear classifier for linearly separable dataset and we will study the mechanisms of support vector classification based on soft margin classifiers.

2. Theoretical Background

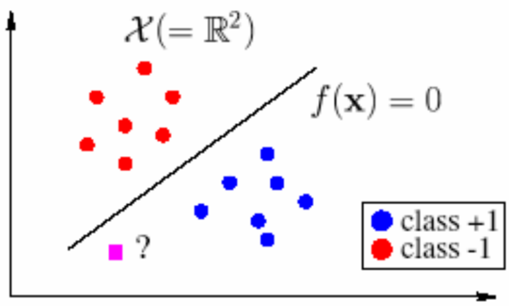
2.1. Linear classification

The goal of classification is to group items that have similar feature values into classes or groups. A linear classifier achieves this by making a classification decision based on the value of the linear combination of the features.

There are some issues in the learning classifier problem:

- Limited number of training data;
- Learning algorithm (how to build the classifier?);
- Generalization: the classifier should correctly classify test data.

Example of a linear classifier,

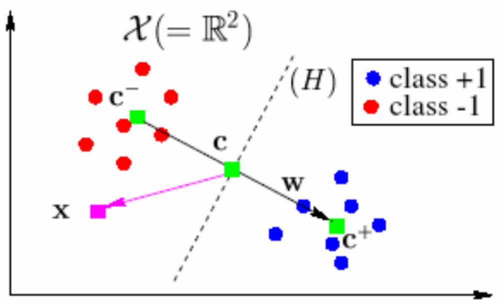


where X is the space of data (feature space), called input space, Y is the target space (class space) with values from {-1, 1} and $f:X\rightarrow Y$ is the classifier.

Given a test data $x\in X$ we choose y such that (x, y) is in some sense similar to training examples. Thus we need a notion of similarity in X and in {-1, 1}.

2.2. A simple linear classifier

A simple type of similarity is the dot product (inner product or scalar product). The idea of this classifier is to assign a new point to the class whose mean is the closest.



$$\mathbf{c}^+ = \frac{1}{m^+} \sum_{\{i:y_i=+1\}} \mathbf{x}_i$$

$$\mathbf{c}^- = \frac{1}{m^-} \sum_{\{i:y_i=-1\}} \mathbf{x}_i$$

$$\mathbf{c} = \tfrac{1}{2}(c^+ + c^-)$$

$$\mathbf{w} = c^+ - c^-$$

For $x \in X$ it is sufficient to take the sign of the inner product between w and x-c.
If $h(x)=\langle w, x-c\rangle$, we have the classifier $f(x)=sign(h(x))$. The dotted hyperplane (H), of the normal vector w, is the decision surface ($h(x)=0$).
We obtain:

$$h(x) = \sum_{i=1,\dots,m} \alpha_i \langle x_i, x\rangle + b$$

Where: m is the total number of samples, m^+ is the total number of samples from blue class, m^- is the total number of samples from red class, $\alpha_i = \frac{1}{m^+} \forall i, if\ y_i = 1$ and $\alpha_i = -\frac{1}{m^-} \forall i, if\ y_i = -1$, and the value $b = \langle c, c^- \rangle - \langle c, c^+ \rangle$.

Remark: A generic linear classifier is $g(x) = w^T x + w_0$. We can identify from the equality $g(x) = h(x)$ that: $w^T = \sum_{i=1,\dots,m} \alpha_i x_i$ and $w_0 = b$

2.3. Hard-margin classifiers

In explaining the problem of hard and soft margin classifiers we will start from a simple problem of linearly separating a set of points in the Cartesian plane, as depicted in Fig. 1.1

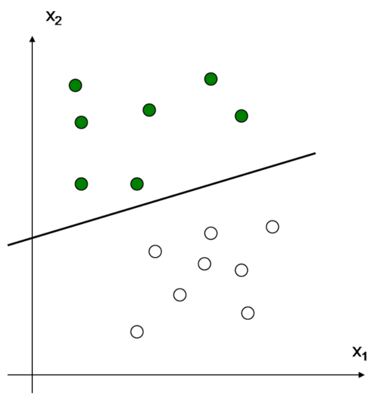


Fig. 1.1 A set of points linearly separable

The question here is how can we classify these points using a linear discriminant function in order to minimize the error rate? We have an infinite number of answers, as shown in Fig. 1.2:

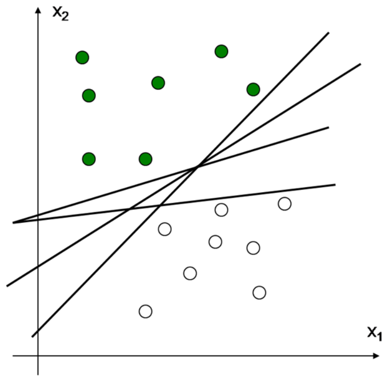


Fig. 1.2 Linear classifiers that can discriminate between the set of points

From the multitude of solutions we need to find out which is the best one. The answer is given by the linear discriminant function (classifier) with the maximum margin is the best. Margin is defined as the width that the boundary could be increased by before hitting a data point as in Fig. 1.3.

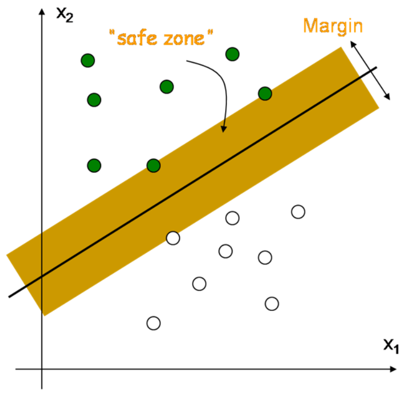


Fig. 1.3 The margin of a linear classifier

This classifier is the best because it is robust to outliners and thus has strong generalization ability.

Given a set of data points: $\{x_i, y_i\}, i = 1,2,\dots,n$ where

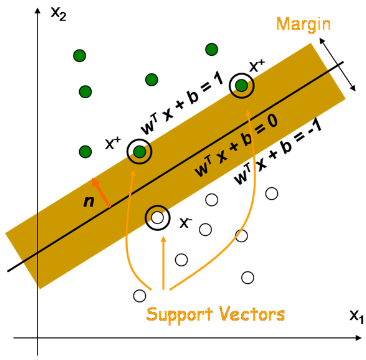
$$\text{For } y_i = +1, w^T x_i + b > 0$$
$$\text{For } y_i = -1, w^T x_i + b < 0$$

With a scale transformation on both *w* and *b*, the above is equivalent to:

$$\text{For } y_i = +1, w^T x_i + b \geq 1$$
$$\text{For } y_i = -1, w^T x_i + b \leq -1$$

We know that:

$$w^T x^+ + b = 1$$
$$w^T x^- + b = -1$$



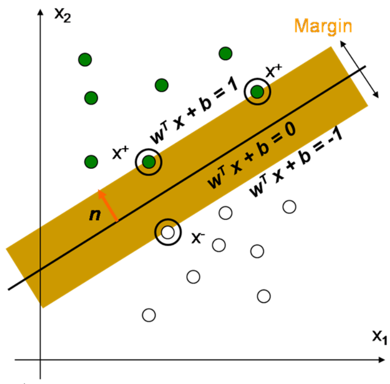
The margin width is:

$$M = (x^+ - x^-) \cdot n = (x^+ - x^-) \cdot \frac{w}{||w||} = \frac{2}{||w||}$$

This margin should be maximized. The maximization problem is difficult to solve because it depends on ||**w**||, the norm of **w**, which involves a square root. Fortunately it is possible to alter the equation by substituting ||**w**|| with $\frac{1}{2}||w||^2$ without changing the solution (the minimum of the original and the modified equation have the same *w* and *b*).

This is a quadratic programming (QP) optimization problem. More clearly we need to:

- minimize $\frac{1}{2}||w||^2$ such that:
- For $y_i = +1, w^T x_i + b \geq 1$
- For $y_i = -1, w^T x_i + b \leq -1$



Which is equivalent to minimize $\frac{1}{2}||w||^2$ such that $y_i(w^T x_i + b) \geq 1$.
The solution to this optimization problem is found by Lagrangian multipliers, but it is not the purpose of this lab.

2.4. Soft-margin classifiers

In 1995, Corinna Cortes and Vladimir Vapnik suggested a modified maximum margin idea that allows for mislabeled examples. If there exists no hyperplane that can split the "yes" and "no" examples, the Soft Margin method will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. The method introduces slack variables, ξ_i, which measure the degree of misclassification of the datum x_i.

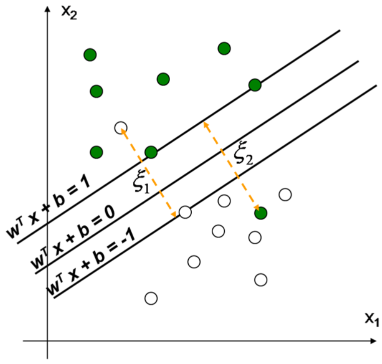


Fig. 1.4 Classification using soft margin

By minimizing $\sum\xi_i$, we can obtain ξ_i by:

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

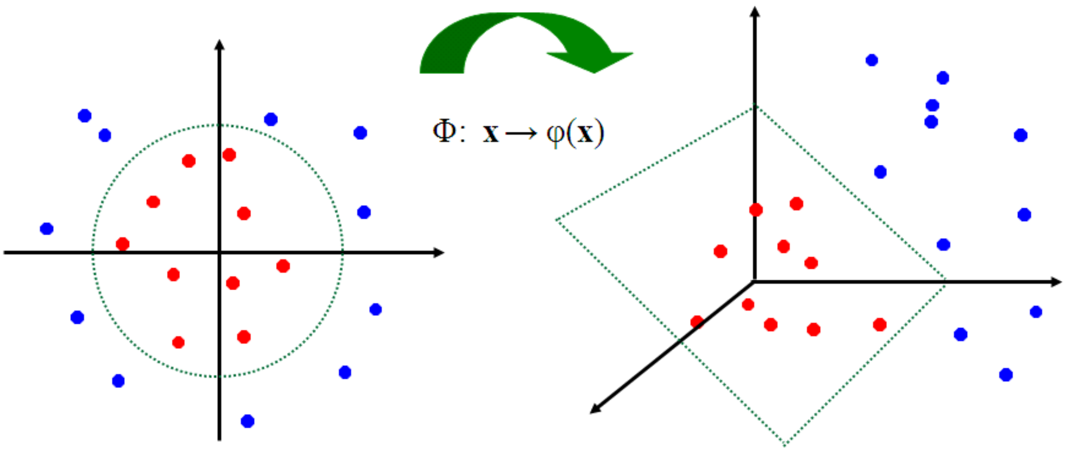
- ξ_i are "slack variables" in optimization
- ξ_i =0 if there is no error for $\mathbf{x}_i$
- $\sum\xi_i$ is an upper bound of the number of errors

So we have to minimize $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$ such that $y_i(w^T x_i + b) \geq 1 - \xi_i$ and ξ_i≥0. Parameter C can be viewed as a tradeoff parameter between error and margin.

2.5. Support vector machine with soft margin classification

If the data is non-linearly separable a transformation is applied to each sample x_i such that $x_i \rightarrow \phi(x_i)$ such that the original input space is be mapped to some higher-dimensional feature space where the training set is separable.

Suppose we have a nonlinearly separable dataset $\{(x_1,y_1),\ldots,(x_m,y_m)\}$. The idea is to choose a nonlinear mapping $\Phi : X \rightarrow H$ and each x in input space is assigned $\Phi(x)$ in H. H is an inner product space called feature space. Then we can find a classifier (a separating hyperplane) in H to classify $\{(\Phi(x_1),y_1),\ldots,(\Phi(x_m),y_m)\}$.



Given a training set of instance-label pairs (x_i, y_i); $i = 1\ldots l$ where $x_i \in R^n$ and $y \in \{+1,-1\}$[1], the support vector machines (SVM) require the solution of the following optimization problem:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i$$

Subject to:

$$y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0.$$

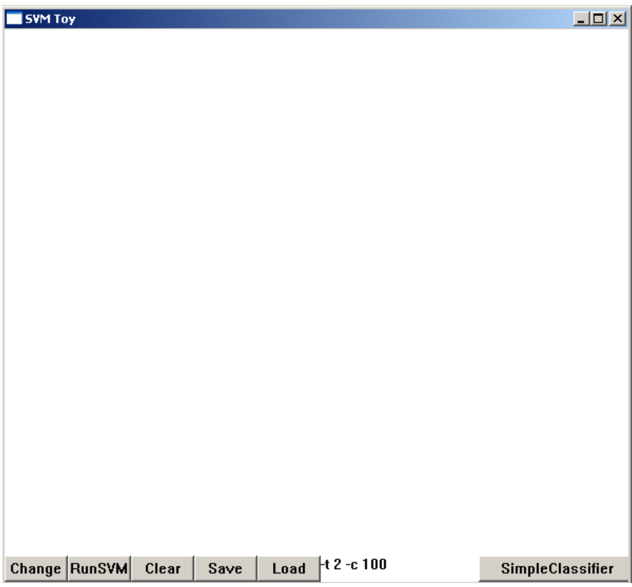
Here training vectors x_i are mapped into a higher (maybe infinite) dimensional space by the function Φ. Then SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. Furthermore, $K(x_i, x_j) = \Phi(x_i)^T\,\Phi(x_j)$ is called the kernel function. Though new kernels are being proposed by researchers, beginners may find in SVM books the following four basic kernels:

- Linear kernel: $k(u,v) = u^T v$
- Gaussian kernel: $k(u,v) = \exp\left(-\frac{\|u-v\|^2}{2\sigma^2}\right)$
- Sigmoid: $k(u,v) = \tanh(\beta_0 u^T v + \beta_1)$
- Polynomial kernel: $k(u,v) = \left(\langle u,v\rangle + c\right)^d, c \in \mathbb{R}, d \in \mathbb{N}$

3. Exercises

For the practical work you will be given a framework called *SVM-toy*, that provides a C++ implementation of soft-margin classifiers using different types of kernels.

1. Download *TestSVM.zip*. Compile *SVM-toy* and run it. Its interface should look like:



The buttons of the interface have the following meaning:

- ‘Change’ button: the application allows the user to add points in the classification space (the white window) by mouse left click; this button allows to change the color of the points (each color corresponds to a class). A maximum number of three colors is allowed (hence three classes)
- ‘RunSVM’ button – runs the SVM classifier with the parameters specified in the edit box
- ‘Clear’ button – clears the classification space
- ‘Save’ button – saves the points (normalized coordinates) from the classification space to a file
- ‘Load’ button – loads a bitmap image (loads and draws the points into the classification space)
- The Edit box where parameters are specified, the default values are ‘–t 2 –c 100’
  The application allows several parameters, but we will use two of them, naming:
  - ‘-t kernel_type’ specifies the kernel type: set type of kernel function (default 2); ‘kernel_type’ can be one of the following:
    0 – linear kernel: u*v
    1 –polynomial kernel: (gamma*u'*v + coef0)^degree
    2 – radial basis function: exp(-gamma*|u-v|^2)
    3 – sigmoid: tanh(gamma*u'*v + coef0)
  - ‘-c cost’ specifies the parameter C from the soft margin classification problem
- ‘SimpleClassifier’ button – runs the simple classifier (to do!).

2. For each image in *svm_images.zip* run the default SVM classifier (with different kernels and costs)
3. Implement the ‘SimpleClassifier’ (according to section 2.2) and compare it to the SVM classifier that uses a linear kernel.

   Write the code in the file *svm-toy.cpp* for the case branch:

```
case ID_BUTTON_SIMPLE_CLASSIFIER:
{

/* ****************************************
      TO DO:
      WRITE YOUR CODE HERE FOR THE SIMPLE CLASSIFIER
**************************************** */

}
```

   For implementing the simple classifier you should know that in the *svm_toy.cpp* file the coordinates of the points are stored in the structure

```
list<point> point_list;
```

   and a point is defined by the structure:

```
struct point {
	double x, y;
	signed char value;
};
```

The variable ‘value’ represents the class label.

The coordinates of the points are normalized between 0 and 1 and the (0,0) point is located in the top left corner.
Notice that the dimension of the classification space is XLEN x YLEN. Hence to a normalized point (x,y) we have other coordinates in the classification space (drawing space) which are (x*XLEN, y*YLEN).

The drawing of a segment between two points is done by the method:

```
DrawLine(window_dc,x1, y1, x2, y2, RGB(255,0,0));
```

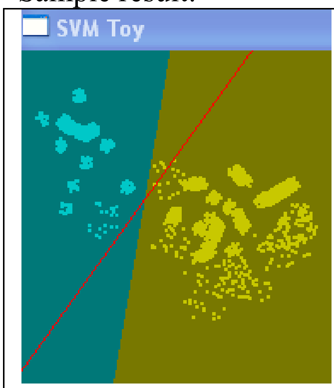
In order to iterate over all the points and count how many points are in class ‘1’ and in class ‘2’ you should do the following:

```
//declare an iterator
list<point>::iterator p;
int nrSamples1=0;
int nrSamples2=0;
double xC1=0,xC2=0,yC1=0,yC2=0;

for(p = point_list.begin(); p != point_list.end(); p++)
{
	if ((*p).value==1) //point from class '1'
	{
		nrSamples1++;
		xC1 =(*p).x; //normalized x coordinate of the current point
		yC1 =(*p).y; //normalized y coordinate of the current point

	}
	if ((*p).value==2)  //point from class '2'
	{
		nrSamples2++;
		xC2 =(*p).x; //normalized x coordinate of the current point
		yC2 =(*p).y; //normalized y coordinate of the current point
	}
}
```

Sample result:

	Details: - 2D points to be classified - 2 classes, 2 features (x_1 and x_2) - Red line separation obtained by implementing the ‘Simple Classifier’ algorithm - Cyan/Brown line separation obtained by SVM linear kernel (-t 0) and cost C=100 (-c 100) Observe - The maximized margin obtained with SVM - The points incorrectly classified by simple classifier

4. References

[1] Jinwei Gu - *An Introduction to SVM*: http://www1.cs.columbia.edu/~belhumeur/courses/biometrics/2009/svm.ppt

[2] J. Shawe-Taylor, N. Cristianini: *Kernel Methods for Pattern Analysis.* Pattern Analysis (Chapter 1)

[3] B. Scholkopf, A. Smola: *Learning with Kernels*. A Tutorial Introduction (Chapter 1), MIT University Press.

[4] LIBSVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm/