Classifying Large Data Sets Using SVMs with Hierarchical Clusters

Presented by :Limou Wang

Overview

- SVM Overview
- Motivation
- Hierarchical micro-clustering algorithm
- Clustering-Based SVM (CB-SVM)
- Experimental Evaluation
- Conclusion & Future work

SVM Overview

- What is Support Vector Machine ?

  analyze data & recognize patterns

- How does SVM work ?

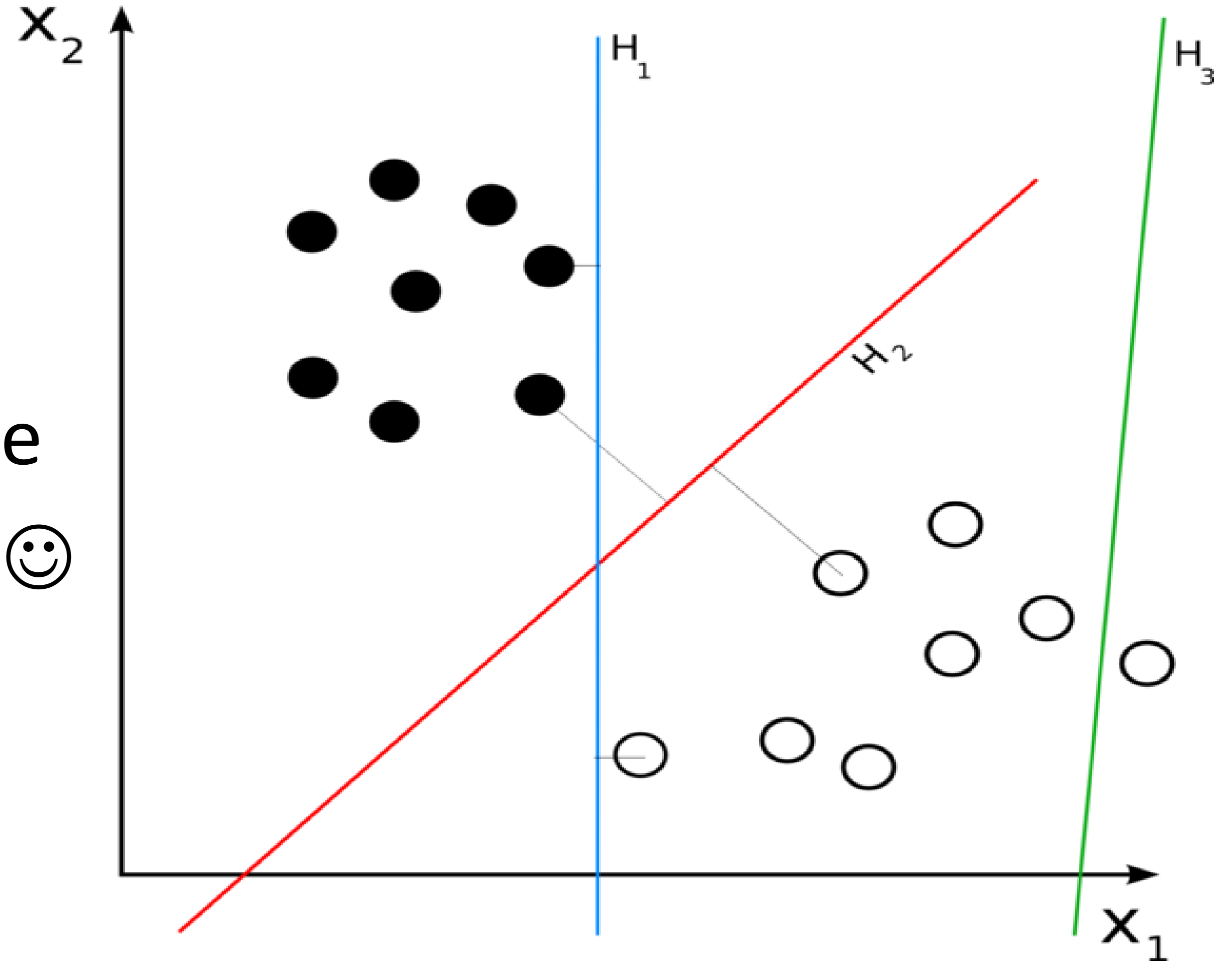
  Training phase : Given a sequence of training examples, find a hyperplane that separate these data points.

  Predicting phase : When a new data points arrive, correctly classifies it.

SVM Overview

Example :

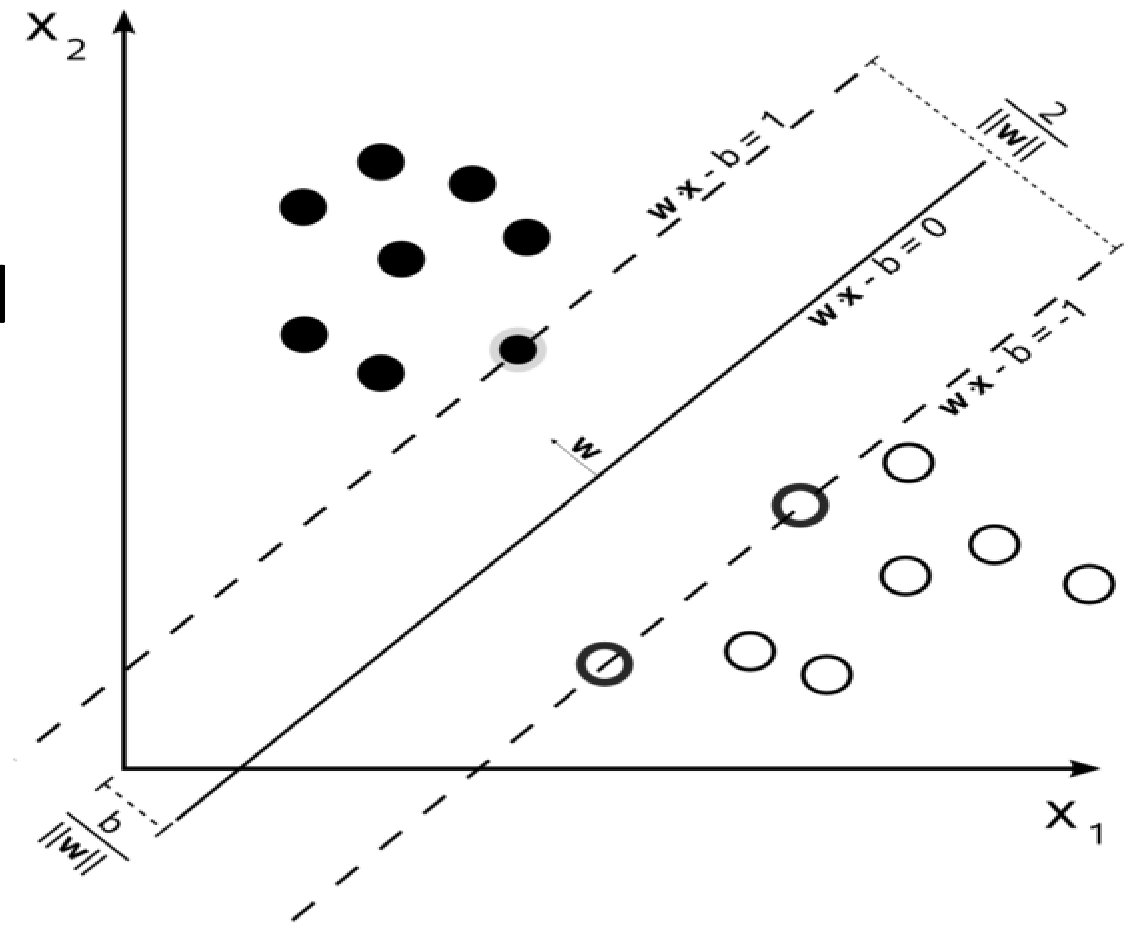
- H1(blue) & H2(red) separate the data points ☺
- H3(green) does not ☹



SVM Overview

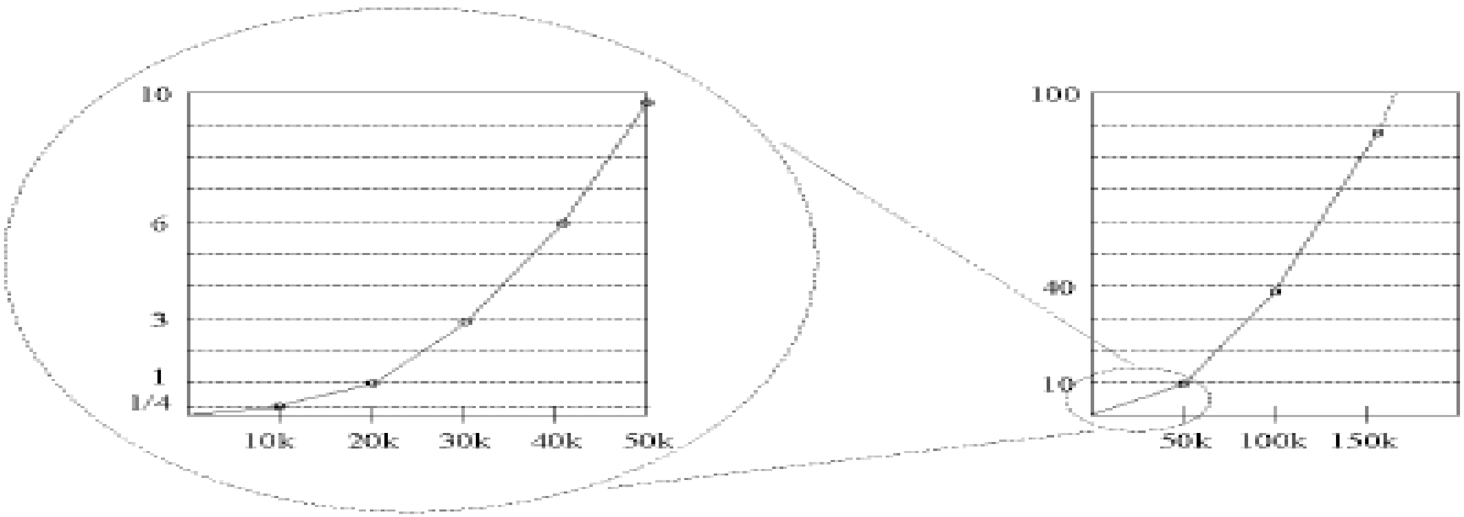
Maximum Margin Principle

Samples on the margin are called support vectors



Motivation

- SVM has been promising methods
- However, training time of the standard SVM is $O(N^3)$
- Therefore, not scalable for very large data sets



Hierarchical micro-clustering algorithm

Goals

- Minimize running time and data scans, thus formulating the problem for large data sets
- Clustering decisions made without scanning the whole data
- Exploit the non uniformity of data – treat dense areas as one, and remove outliers (noise)

Hierarchical micro-clustering algorithm

Clustering Feature (CF)

- CF is a compact storage for data on points in a cluster
- Has enough information to calculate the intra-cluster distances

$$C = \frac{\sum_{i=1}^{N} x_i}{N} \qquad R = \left(\frac{\sum_{i=1}^{N} ||x_i - C||^2}{N}\right)^{\frac{1}{2}}$$

- Additivity theorem allows us to merge sub-clusters

Hierarchical micro-clustering algorithm

CF (contd.)

- Given N d-dimensional data points in a cluster: {Xi} where i = 1, 2, …, N,
- CF = (N, LS, SS)
- N is the number of data points in the
- cluster,
- LS is the linear sum of the N data points,
- SS is the square sum of the N data points.

Hierarchical micro-clustering algorithm

CF Tree

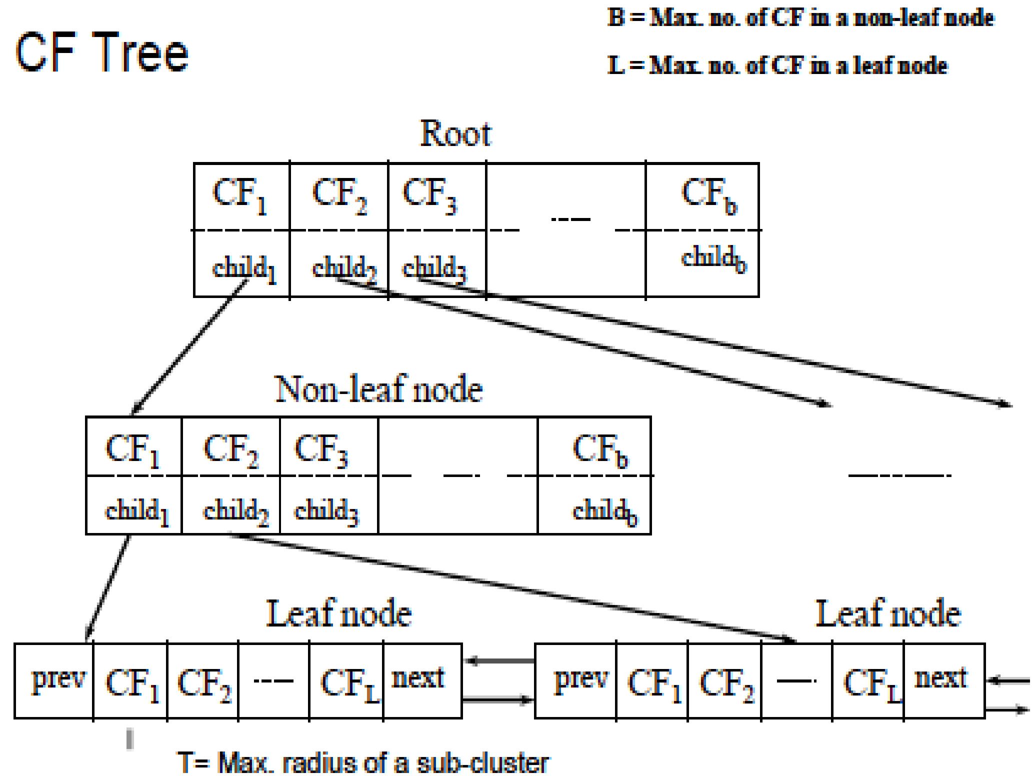
- Comprising of a root node, non-leaf nodes and leaf nodes
- Two parameters : branching factor b & threshold t
- Non-leaf node at most b entries
- Radius of an leaf node entry must less than t

Hierarchical micro-clustering algorithm

CF Tree (contd.)

Example

Thus, CF Tree is a compact representation of a large data set



Hierarchical micro-clustering algorithm

CF Tree (contd.)

Algorithm :

1. Identifying the appropriate leaf

   Root -> closet centroid

2. Modifying the leaf

   Absorb, add an entry or split

3. Modifying the path to the leaf

   Recursively back to the root

Hierarchical micro-clustering algorithm

CF Tree (contd.)

Outlier handling : After building the CF tree, remove the entries that contains far fewer data points than the average

Running time : If we only consider the dependence of the size of the data set, the computation complexity of the algorithm is O(N)

Clustering-Based SVM (CB-SVM)

Train data using the hierarchical micro-cluster

1. Construct two CF trees
2. Train an SVM boundary function from the centroids of the root entries
3. Decluster the entries near the boundary into the next level
4. Update the SVM boundary function from the centroids of the entries in the training set, repeat 3 until nothing is accumulated

Clustering-Based SVM (CB-SVM)

Decluster the low-margin cluster :

Let D_s be the distance from the boundary to the centroid of a support cluster

D_i be the distance from the boundary to the centroid of a cluster E_i

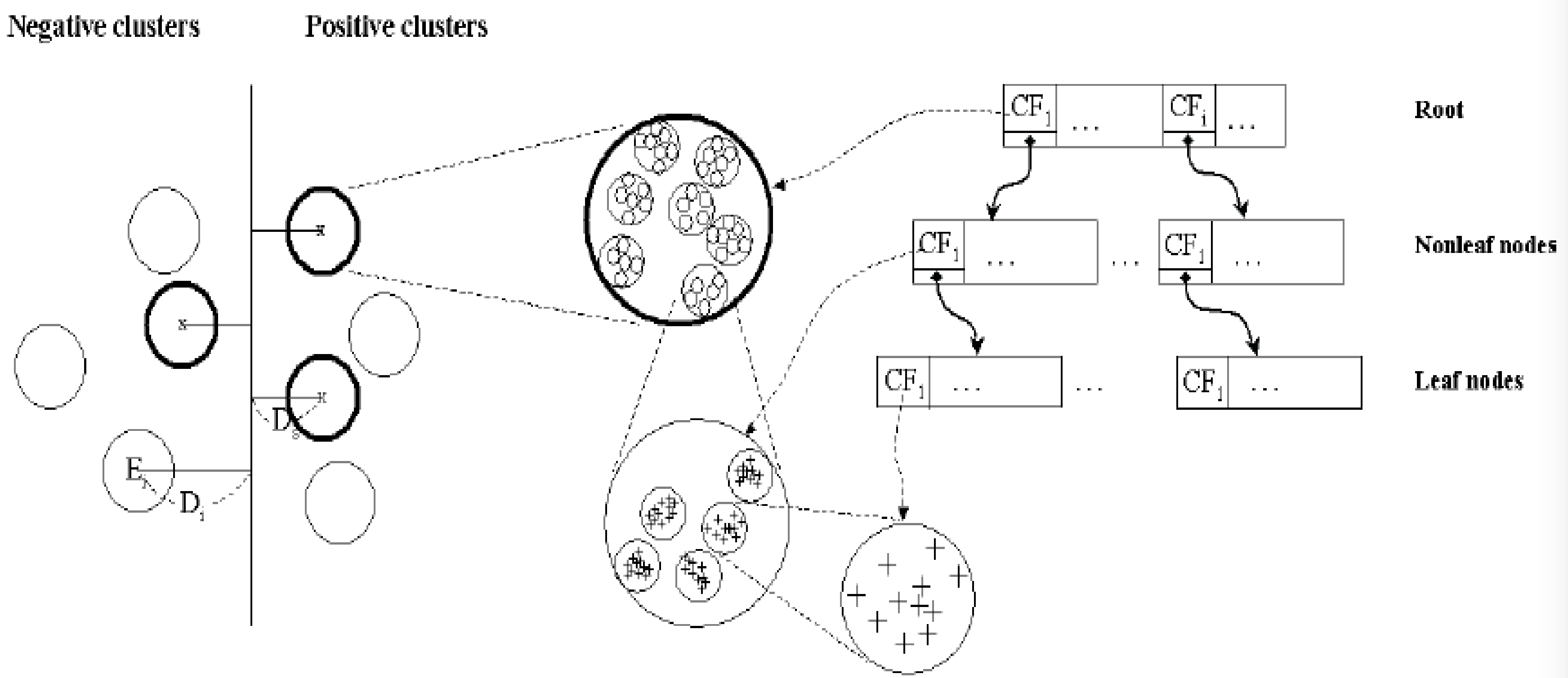
Then

$$\text{if} \qquad D_i - R_i < D_s$$

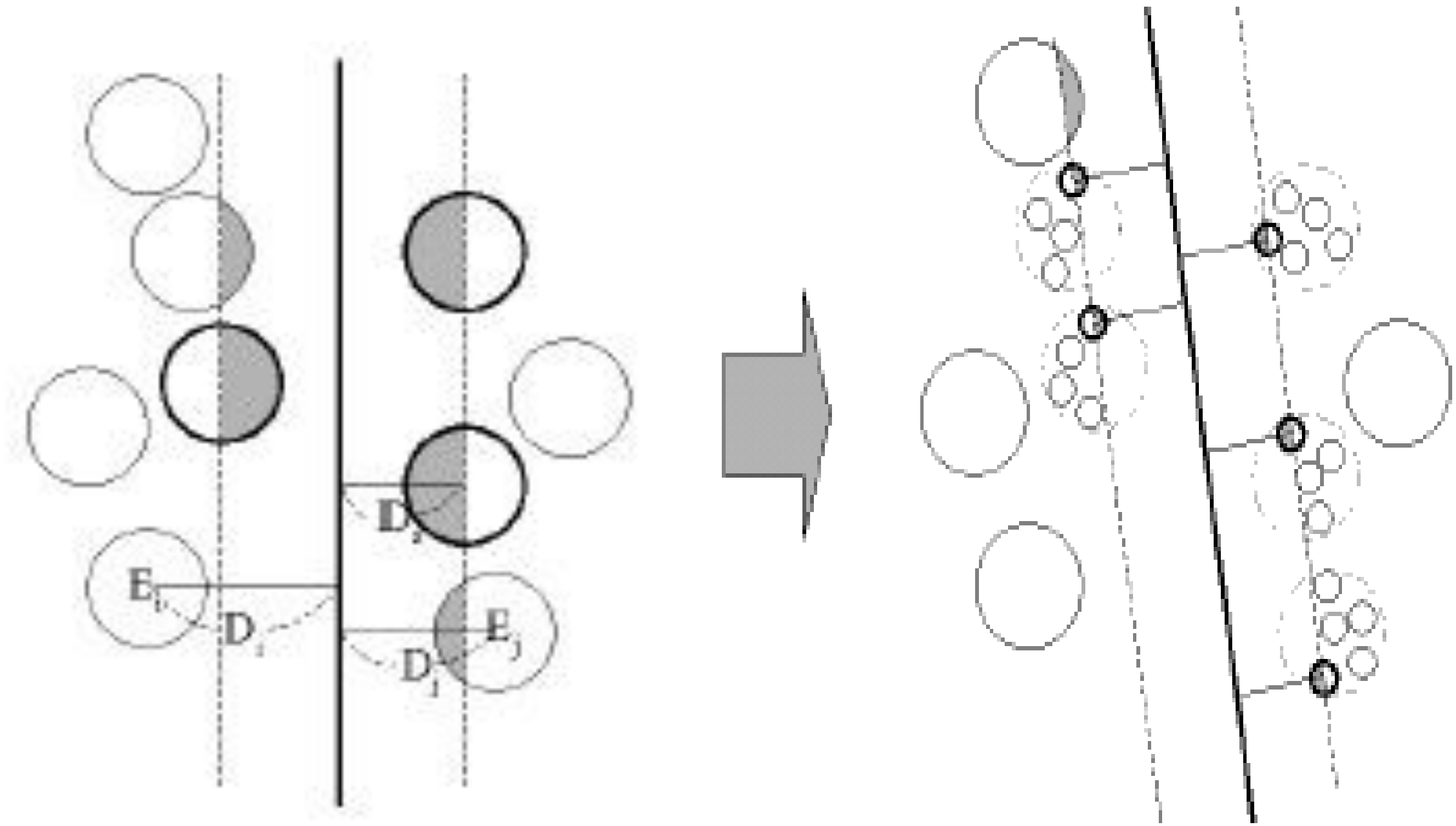
E_i is considered to be a low margin cluster

Clustering-Based SVM (CB-SVM)

Next, we look at how CB-SVM works



Clustering-Based SVM (CB-SVM)



Clustering-Based SVM (CB-SVM)

Running time :

Let $r = s / b$

s : average number of support entries

$0 < r << 1$

Therefore, CB-SVM trains from leaf entries is $O(b^{2h})$

If the # of leaf entries = number of data points,

It trains $1/r^{2h-2}$ faster than the standard SVM

which is a huge improvement in performance compared with the standard SVM as the data set becomes larger

Experimental Evaluation

Environment :

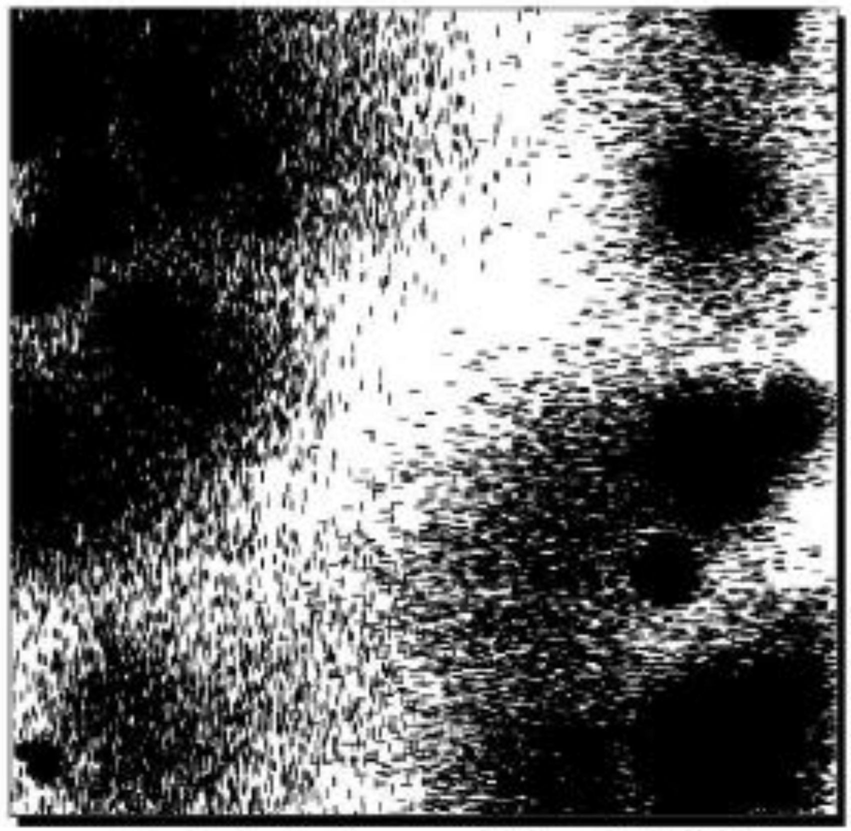
All experiments are done in a Pentium III 800Mhz machine with 906MB memory

Premise :

Perform binary classification on 2 dimensional data sets

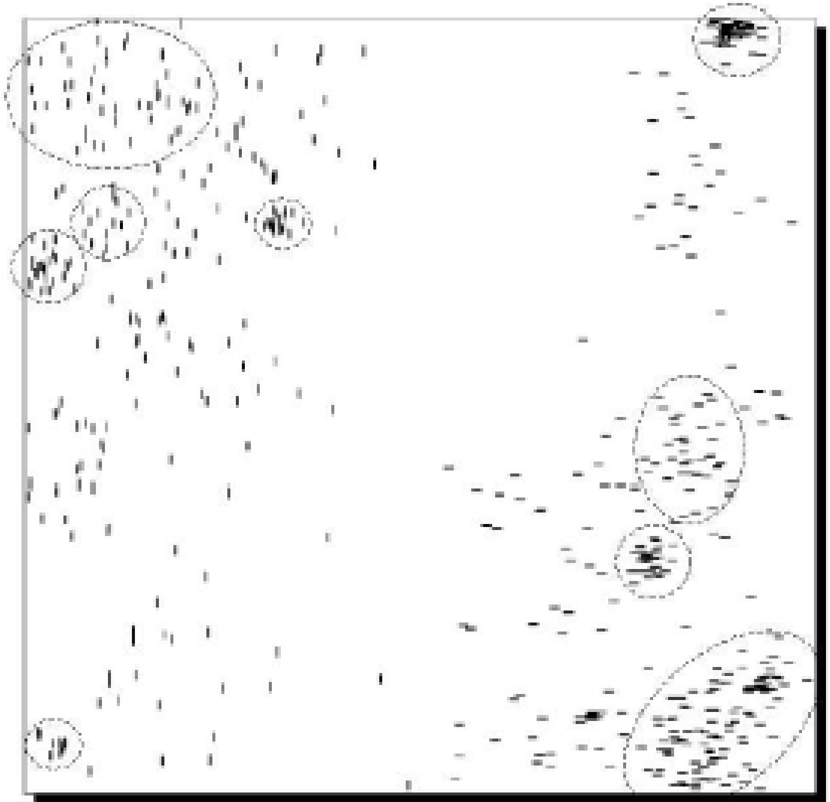
Note : training and testing data are drawn from the same distribution

Experimental Evaluation



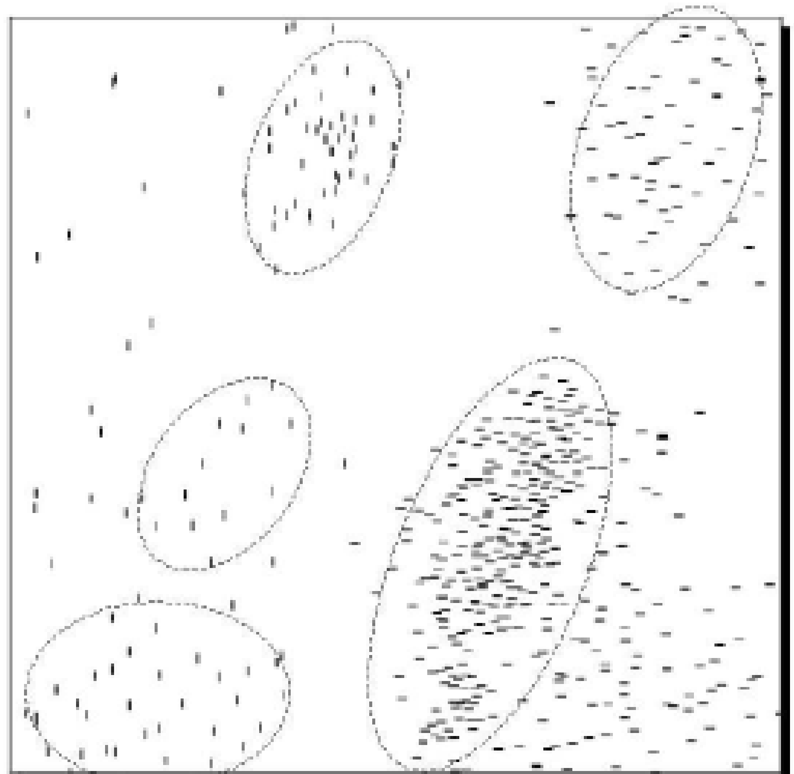
(a) original data set ($N = 113601$)

Experimental Evaluation



(b) 0.5% randomly sampled data
($N = 603$)

Experimental Evaluation



(c) data distribution at the last iteration in CB-SVM ($N = 597$)

Experimental Evaluation

	Original	CB-SVM	0.5% samples
Number of data points	113601	597	603
SVM Training time (sec.)	160.792	0.003	0.003
Sampling time (sec.)	0.0	10.586	4.111
# of false predictions (# of FP, # of FN)	69 (49, 20)	86 (73, 13)	243 (220, 23)

Table 2: Performance results on synthetic data set (# of training data = 113601, # of testing data = 107072). FP:false positive; FN:false negative; Sampling time for CB-SVM: time for contructing the CF tree

Experimental Evaluation

S-Rate	# of data	# of errors	T-Time	S-Time
0.0001%	23	6425	0.000114	822.97
0.001%	226	2413	0.000972	825.40
0.01%	2333	1132	0.03	828.61
0.1%	23273	1012	6.287	835.87
1%	230380	1015	1192.793	838.92
5%	1151714	1020	20705.4	842.92
CB-SVM	2893	876	1.639	2528.213

Table 4: Performance results on the very large data set (# of training data = 23066169, # of testing data = 233890). S-Rate: sampling rate; T-Time: training time; S-Time: sampling time;

Conclusion & Future work

- CB-SVM very scalable for large data sets
- Generating high classification accuracy

However,

- Limited to the usage of linear kernels
- Radius and distances will not be preserved in a high-dimensional feature space

Future work

Constructing an efficient indexing structure for nonlinear kernels

Questions ?

Thank you !