CS145: INTRODUCTION TO DATA MINING

4: Vector Data: Support Vector Machine

Instructor: Yizhou Sun

yzsun@cs.ucla.edu

October 15, 2017

Announcements

- Homework 1

  - Due end of the day of this Thursday (11:59pm)

- Reminder of late submission policy

  - original score $^*\mathbf{1}(t <= 24)e^{-(ln(2)/12)*t}$

  - E.g., if you are t = 12 hours late, maximum of half score will be obtained; if you are 24 hours late, 0 score will be given.

Methods to Learn: Last Lecture

	Vector Data	Set Data	Sequence Data	Text Data
Classification	**Logistic Regression;** **Decision Tree**; KNN SVM; NN			Naïve Bayes for Text
Clustering	K-means; hierarchical clustering; DBSCAN; Mixture Models			PLSA
Prediction	**Linear Regression** GLM*			
Frequent Pattern Mining		Apriori; FP growth	GSP; PrefixSpan	
Similarity Search			DTW	

Methods to Learn

|	Vector Data	Set Data	Sequence Data	Text Data
Classification	**Logistic Regression; Decision Tree**; KNN **SVM**; NN			Naïve Bayes for Text
Clustering	K-means; hierarchical clustering; DBSCAN; Mixture Models			PLSA
Prediction	**Linear Regression** GLM*			
Frequent Pattern Mining		Apriori; FP growth	GSP; PrefixSpan	
Similarity Search			DTW	

Support Vector Machine

- Introduction

- Linear SVM

- Non-linear SVM

- Scalability Issues*

- Summary

Math Review

- Vector
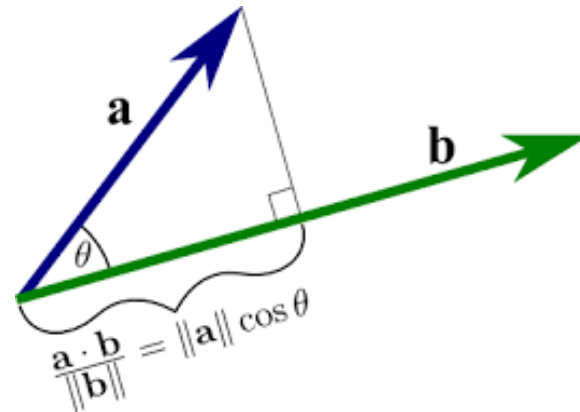 - $x = (x_1, x_2, \ldots, x_n)$
  - Subtracting two vectors: $x = b - a$
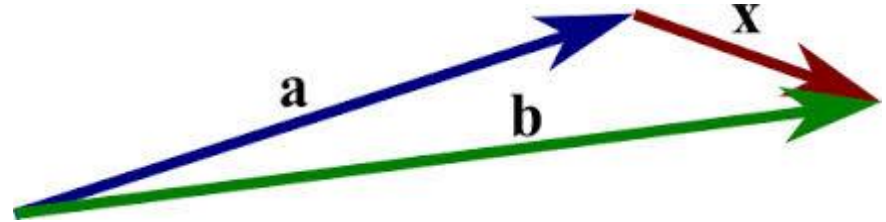
- Dot product
  - $a \cdot b = \sum a_i b_i$
  - Geometric interpretation: projection
  - If a *and* b are orthogonal, $a \cdot b = 0$

Math Review (Cont.)
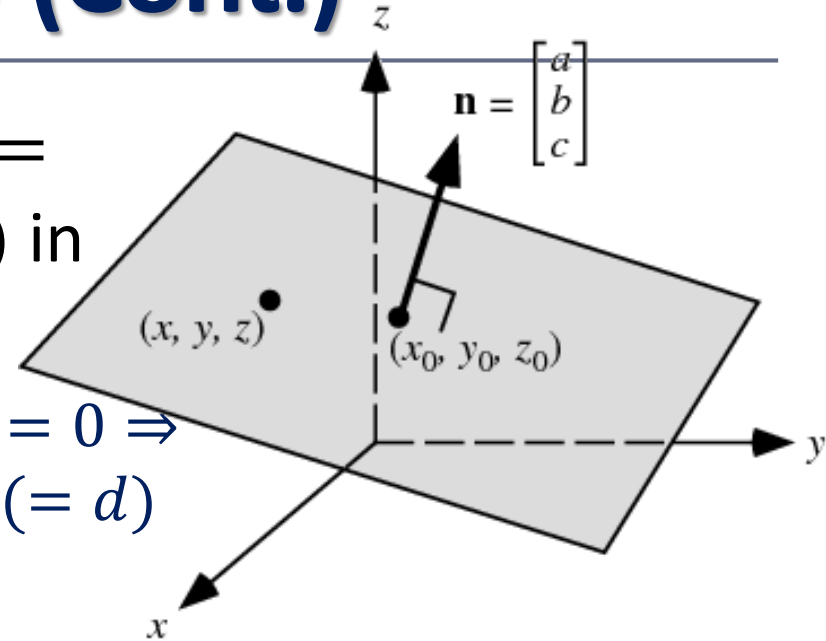
- Plane/Hyperplane

  - $a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = c$
  - Line (n=2), plane (n=3), hyperplane (higher dimensions)

- Normal of a plane

  - $\boldsymbol{n} = (a_1, a_2, \ldots, a_n)$
  - a vector which is perpendicular to the surface

Math Review (Cont.)

- Define a plane using normal $\boldsymbol{n} = (a, b, c)$ and a point (x_0, y_0, z_0) in the plane:

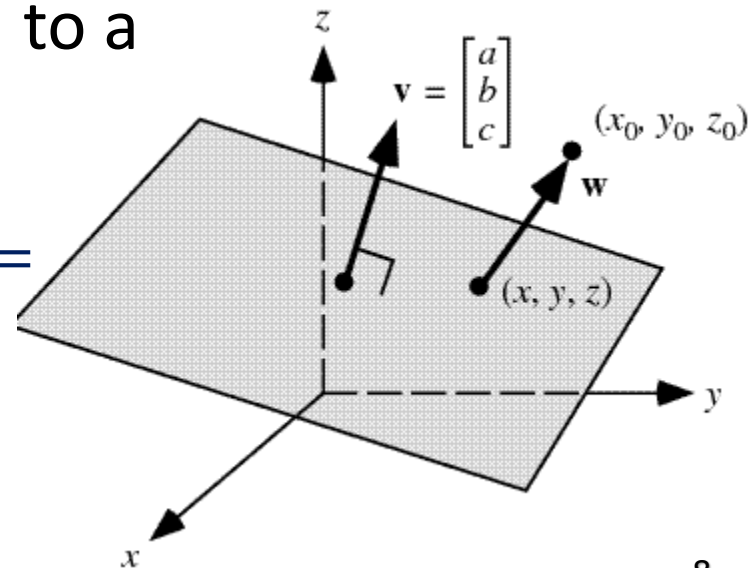 - $(a, b, c) \cdot (x_0 - x, y_0 - y, z_0 - z) = 0 \Rightarrow$
    $ax + by + cz = ax_0 + by_0 + cz_0 (= d)$

- Distance from a point (x_0, y_0, z_0) to a plane $ax + by + cz = d$

  - $\left| (x_0 - x, y_0 - y, z_0 - z) \cdot \dfrac{(a,b,c)}{||(a,b,c)||} \right| =$
    $\dfrac{|ax_0 + by_0 + cz_0 - d|}{\sqrt{a^2 + b^2 + c^2}}$

Linear Classifier

- Given a training dataset $\{x_i, y_i\}_{i=1}^{N}$

  ■ A separating hyperplane can be written as a linear combination of attributes

  $$\mathbf{W} \bullet \mathbf{X} + b = 0$$

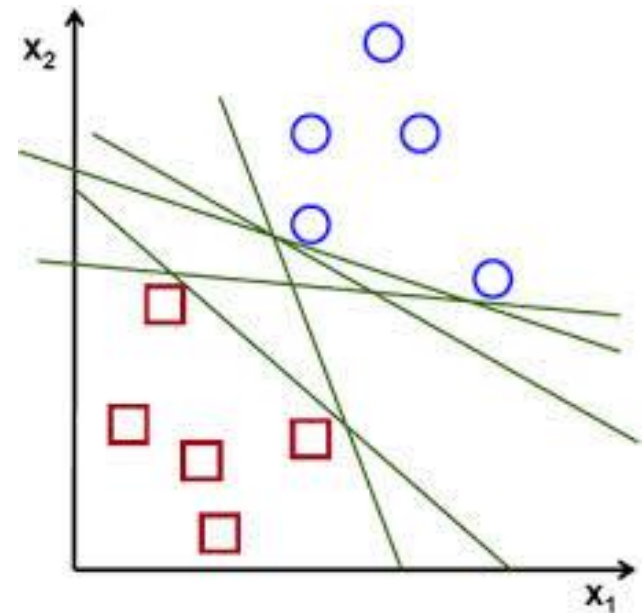 where $\mathbf{W} = \{w_1, w_2, ..., w_n\}$ is a weight vector and b a scalar (bias)

  ■ For 2-D it can be written as

  $$w_0 + w_1 x_1 + w_2 x_2 = 0$$

  ■ Classification:

  $$w_0 + w_1 x_1 + w_2 x_2 > 0 \Rightarrow y_i = +1$$
  $$w_0 + w_1 x_1 + w_2 x_2 \leq 0 \Rightarrow y_i = -1$$

Recall

- Is the decision boundary for logistic regression linear?

- Is the decision boundary for decision tree linear?

Simple Linear Classifier: Perceptron

$$\mathbf{x} = (1, x_1, x_2, \ldots, x_d)^T \qquad \mathbf{w} = (\omega_0, \omega_1, \omega_2, \ldots, \omega_d)^T$$

$$y = \{1, -1\} \qquad\qquad \alpha \in (0, 1] \text{ (learning rate)}$$

Initialize $\mathbf{w} = \mathbf{0}$ (can be any vector)

Repeat:

- For each training example $(\mathbf{x}_i, y_i)$:
  - Compute $\qquad \hat{y}_i = \text{sign}(\mathbf{w^T x_i})$
  - if $(y_i \neq \hat{y}_i) \qquad \mathbf{w} = \mathbf{w} + \alpha(y_i \mathbf{x_i})$
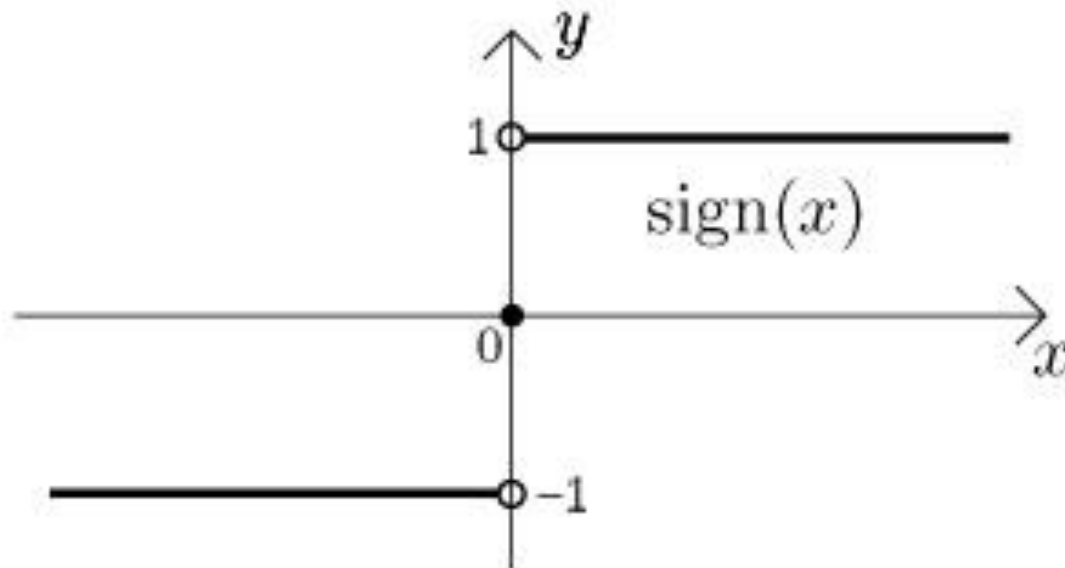
Until $(y_i = \hat{y}_i \quad \forall i = 1 \ldots N)$

Return $\mathbf{w}$

Loss function: $\max\{0, -y_i * w^T x_i\}$

More on Sign Function

-

$$\text{sign}(x) = \begin{cases} 1, & x > 0; \\ 0, & x = 0; \\ -1, & x < 0. \end{cases}$$

Example

| x0	x1	x2	true label	w before update	predicted label	w after update
1	0	1	Y	(0.0, 0.0, 0.0)	N	(0.9, 0.0, 0.9)
1	1	1	N	(0.9, 0.0, 0.9)	Y	(0.0, -0.9, 0.0)
1	0	0	Y	(0.0, -0.9, 0.0)	N	(0.9, -0.9, 0.0)
1	1	0	Y	(0.9, -0.9, 0.0)	N	(1.8, 0.0, 0.0)
1	0	1	Y	(1.8, 0.0, 0.0)	Y	(1.8, 0.0, 0.0)
1	1	1	N	(1.8, 0.0, 0.0)	Y	(0.9, -0.9, -0.9)
1	0	0	Y	(0.9, -0.9, -0.9)	Y	(0.9, -0.9, -0.9)
1	1	0	Y	(0.9, -0.9, -0.9)	N	(1.8, 0.0, -0.9)
1	0	1	Y	(1.8, 0.0, -0.9)	Y	(1.8, 0.0, -0.9)
1	1	1	N	(1.8, 0.0, -0.9)	Y	(0.9, -0.9, -1.8)
1	0	0	Y	(0.9, -0.9, -1.8)	Y	(0.9, -0.9, -1.8)
1	1	0	Y	(0.9, -0.9, -1.8)	N	(1.8, 0.0, -1.8)

Support Vector Machine

- Introduction

- Linear SVM

- Non-linear SVM

- Scalability Issues*

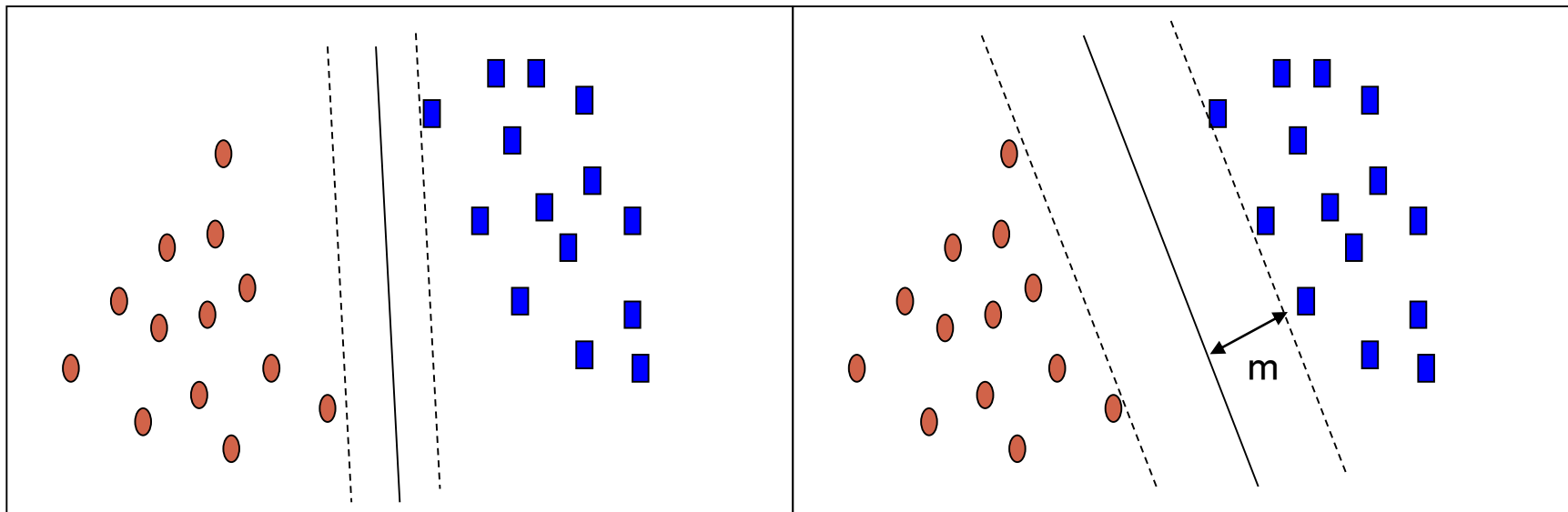- Summary

Can we do better?

- Which hyperplane to choose?
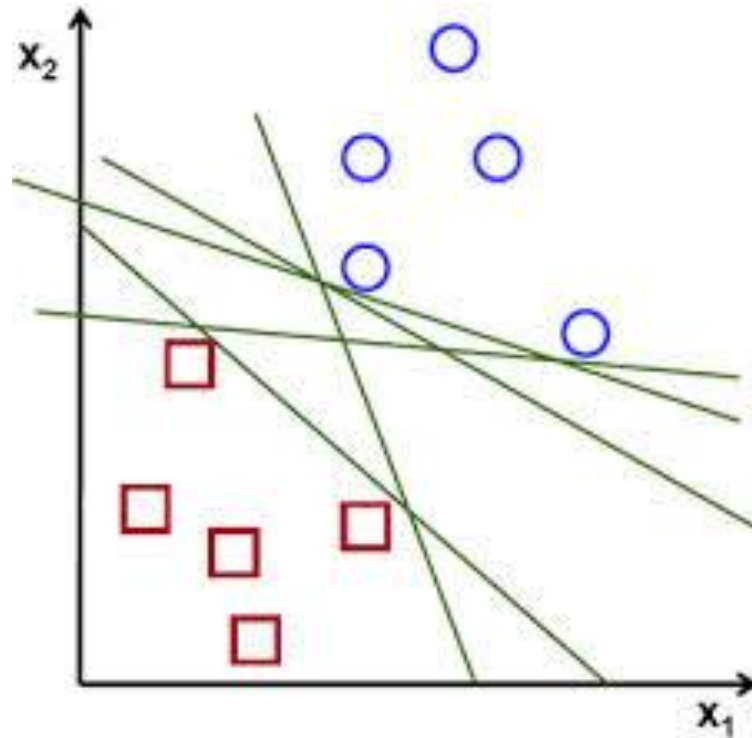
SVM—Margins and Support Vectors

Small Margin

Large Margin

Support Vectors
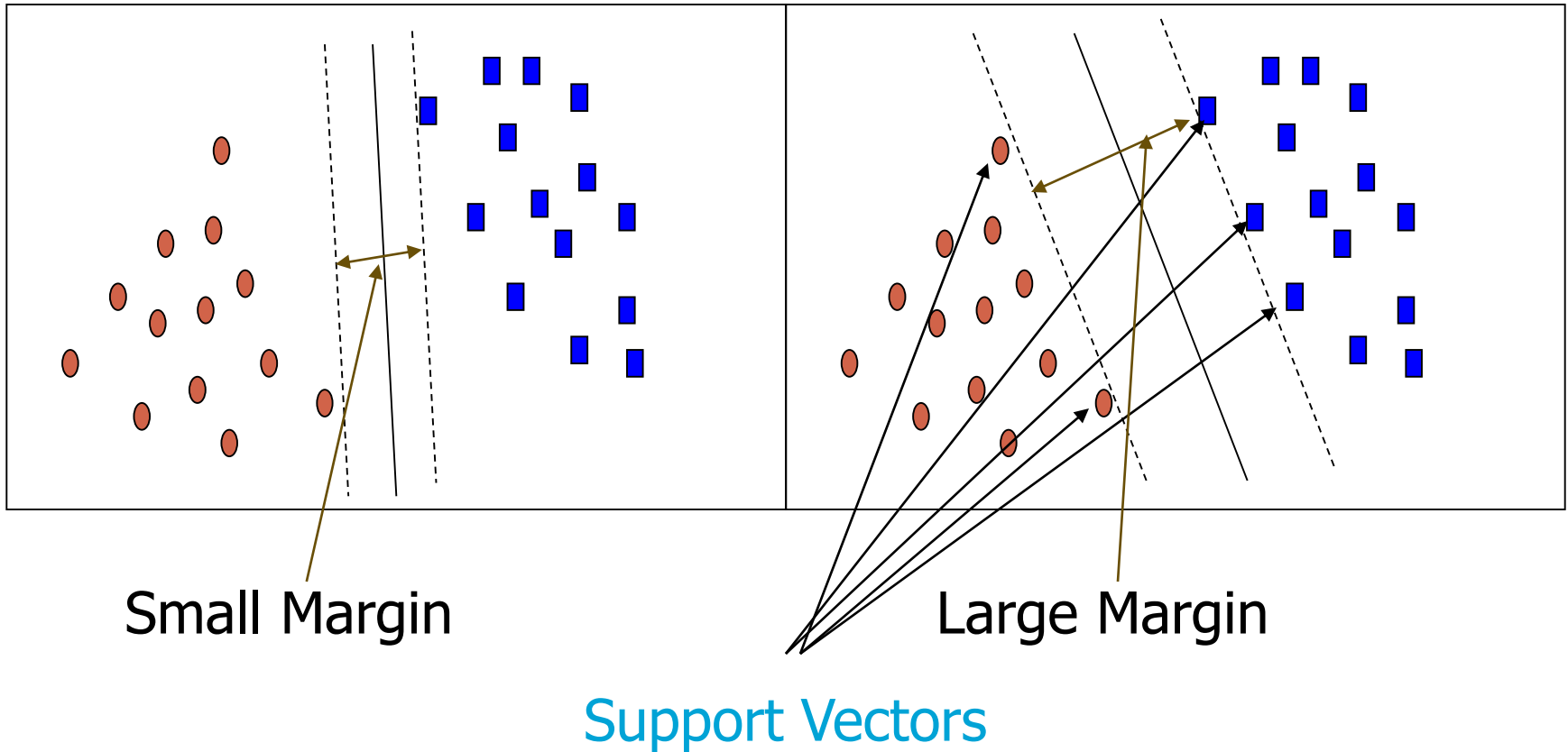
SVM—When Data Is Linearly Separable

Let data D be $(\mathbf{X}_1, y_1), \ldots, (\mathbf{X}_{|D|}, y_{|D|})$, where $\mathbf{X}_i$ is the set of training tuples associated with the class labels y_i

There are infinite lines (<u>hyperplanes</u>) separating the two classes but we want to <u>find the best one</u> (the one that minimizes classification error on unseen data)

SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)

SVM—Linearly Separable

- A separating hyperplane can be written as

    $\mathbf{W} \bullet \mathbf{X} + b = 0$

- The hyperplane defining the sides of the margin, e.g.,:
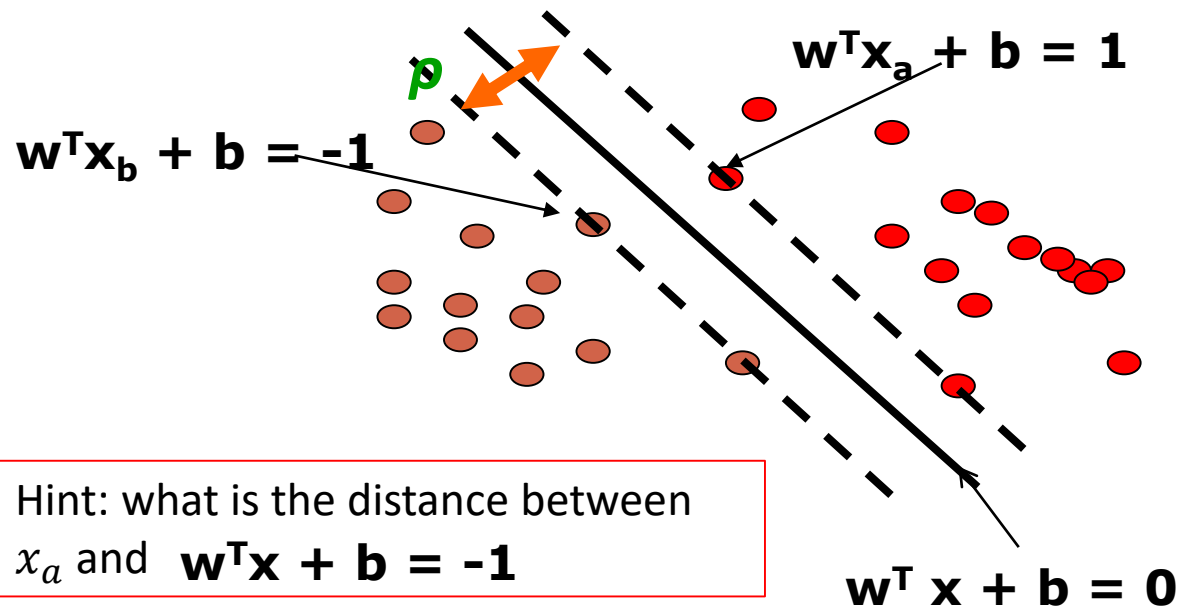
    $H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

    $H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$

- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**

- This becomes a **constrained (convex) quadratic optimization** problem: Quadratic objective function and linear constraints → *Quadratic Programming (QP)* → Lagrangian multipliers

Maximum Margin Calculation

- **w**: decision hyperplane normal vector

- $\mathbf{x}_i$: data point i

- y_i: class of data point i (+1 or -1)

$\boldsymbol{\rho}$

$\mathbf{w^T x_b} + b = -1$

$\mathbf{w^T x_a} + b = 1$

$margin: \rho = \dfrac{2}{||\boldsymbol{w}||}$

Hint: what is the distance between x_a and $\mathbf{w^T x} + b = -1$

$\mathbf{w^T x} + b = 0$

SVM as a Quadratic Programming

- QP

Objective: Find **w** and b such that $\rho = \frac{2}{||w||}$ is maximized;

Constraints: For all $\{(\mathbf{x_i}, y_i)\}$

$\mathbf{w^T x_i} + b \geq 1$ if $y_i = 1$;

$\mathbf{w^T x_i} + b \leq -1$ if $y_i = -1$

- A better form

Objective: Find **w** and b such that $\mathbf{\Phi(w)} = \frac{1}{2} \mathbf{w^T w}$ is minimized;

Constraints: for all $\{(\mathbf{x_i}, y_i)\}$: $y_i (\mathbf{w^T x_i} + b) \geq 1$

Solve QP

- This is now optimizing a *quadratic* function subject to *linear* constraints

- Quadratic optimization problems are a well-known class of mathematical programming problem, and many (intricate) algorithms exist for solving them (with many special ones built for SVMs)

- The solution involves constructing a *dual problem* where a *Lagrange multiplier α_i* is associated with every constraint in the primary problem:

Lagrange Formulation

Minimize

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} - \sum_{i=1}^{N} \alpha_i(y_i(\mathbf{w}^{\mathsf{T}}\mathbf{x}_i + b) - 1)$$

Take the partial derivatives w.r.t $\mathbf{w}, b$:

$$\nabla_{\mathbf{w}} L = \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = 0 \implies \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0$$

Primal Form and Dual Form

Primal

> Objective: Find $\mathbf{w}$ and b such that $\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ is minimized;
>
> Constraints: for all $\{(\mathbf{x_i}, y_i)\}$: $\quad y_i(\mathbf{w^T x_i} + b) \geq 1$

Equivalent under some conditions: KKT conditions

Dual

> Objective: Find $\alpha_1...\alpha_n$ such that
> $\mathbf{Q}(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
>
> Constraints
> (1) $\Sigma\alpha_i y_i = 0$
> (2) $\alpha_i \geq 0$ for all α_i

- More derivations:
  http://cs229.stanford.edu/notes/cs229-notes3.pdf

The Optimization Problem Solution

- The solution has the form:

$$\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w^T x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0$$

- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

$$f(\mathbf{x}) = \Sigma \alpha_i y_i \mathbf{x_i^T x} + b$$

- Notice that it relies on an *inner product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x_i}$
  - We will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i^T x_j}$ between all pairs of training points.

Soft Margin Classification

- If the training data is not linearly separable, *slack variables ξ_i* can be added to allow misclassification of difficult or noisy examples.

- Allow some errors

  - Let some points be moved to where they belong, at a cost

- Still, try to minimize training set errors, and to place hyperplane "far" from each class (large margin)
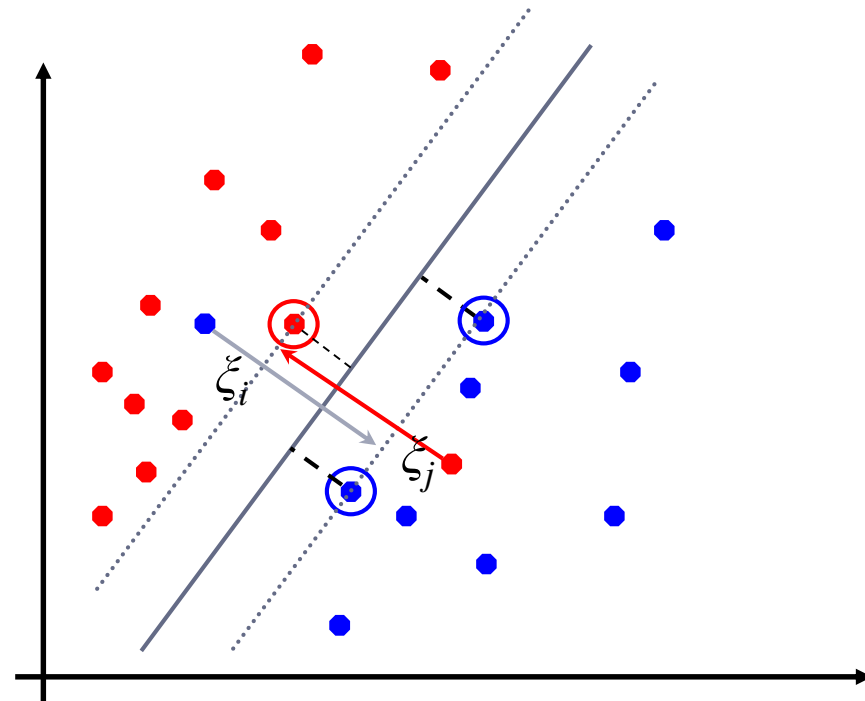
Soft Margin Classification Mathematically

- The old formulation:

> Find $\mathbf{w}$ and b such that
>
> $\Phi(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
>
> $y_i\,(\mathbf{w^T x_i} + b) \geq 1$

- The new formulation incorporating slack variables:

> Find $\mathbf{w}$ and b such that
>
> $\Phi(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
>
> $y_i\,(\mathbf{w^T x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- Parameter *C* can be viewed as a way to control overfitting
  - A regularization term (L1 regularization)

Soft Margin Classification – Solution

- The dual problem for soft margin classification:

> Find $\alpha_1 \ldots \alpha_N$ such that
> $\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma \alpha_i - \frac{1}{2} \Sigma \Sigma \alpha_i \alpha_j y_i y_j \mathbf{x_i^T x_j}$ is maximized and
> (1) $\Sigma \alpha_i y_i = 0$
> (2) $0 \leq \alpha_i \leq C$ for all α_i

- Neither slack variables ξ_i nor their Lagrange multipliers appear in the dual problem!

- Again, $\mathbf{x_i}$ with non-zero α_i will be **support vectors**.

- Solution to the dual problem is:

> $\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x_i}$
> $b = y_k(1 - \xi_k) - \mathbf{w^T} \mathbf{x}_k$ where $k = \underset{k'}{\mathrm{argmax}} \; \alpha_{k'}$

w is not needed explicitly for classification!
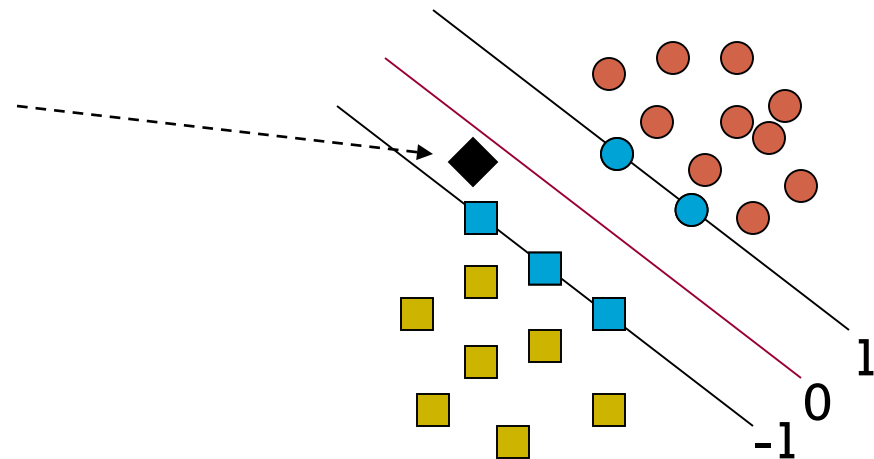
> $f(\mathbf{x}) = \Sigma \alpha_i y_i \mathbf{x_i^T x} + b$

Classification with SVMs

- Given a new point **x**, we can score its projection onto the hyperplane normal:

  - I.e., compute score: $\mathbf{w^T x} + b = \Sigma \alpha_i y_i \mathbf{x_i^T x} + b$
    - Decide class based on whether < or > 0

  - Can set confidence threshold t.

Score $>$ t: yes

Score $<$ $-t$: no

Else: don't know

Linear SVMs: Summary

- The classifier is a *separating hyperplane.*

- The most "important" training points are the support vectors; they define the hyperplane.

- Quadratic optimization algorithms can identify which training points $\mathbf{x_i}$ are support vectors with non-zero Lagrangian multipliers α_i.

- Both in the dual formulation of the problem and in the solution, training points appear only inside inner products:

Find $\alpha_1...\alpha_N$ such that
$\mathbf{Q(\alpha)} = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \boxed{\mathbf{x_i^T x_j}}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

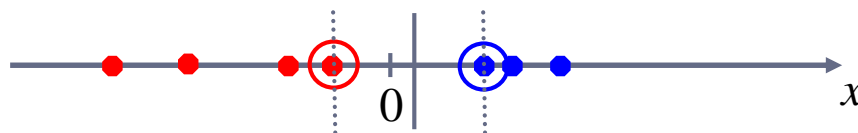$$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{\mathbf{x_i^T x}} + b$$

Support Vector Machine

- Introduction

- Linear SVM

- Non-linear SVM

- Scalability Issues*

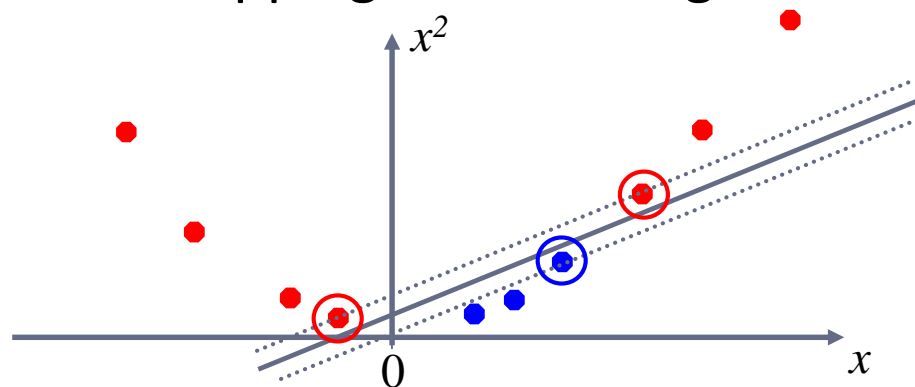- Summary

Non-linear SVMs

- Datasets that are linearly separable (with some noise) work out great:

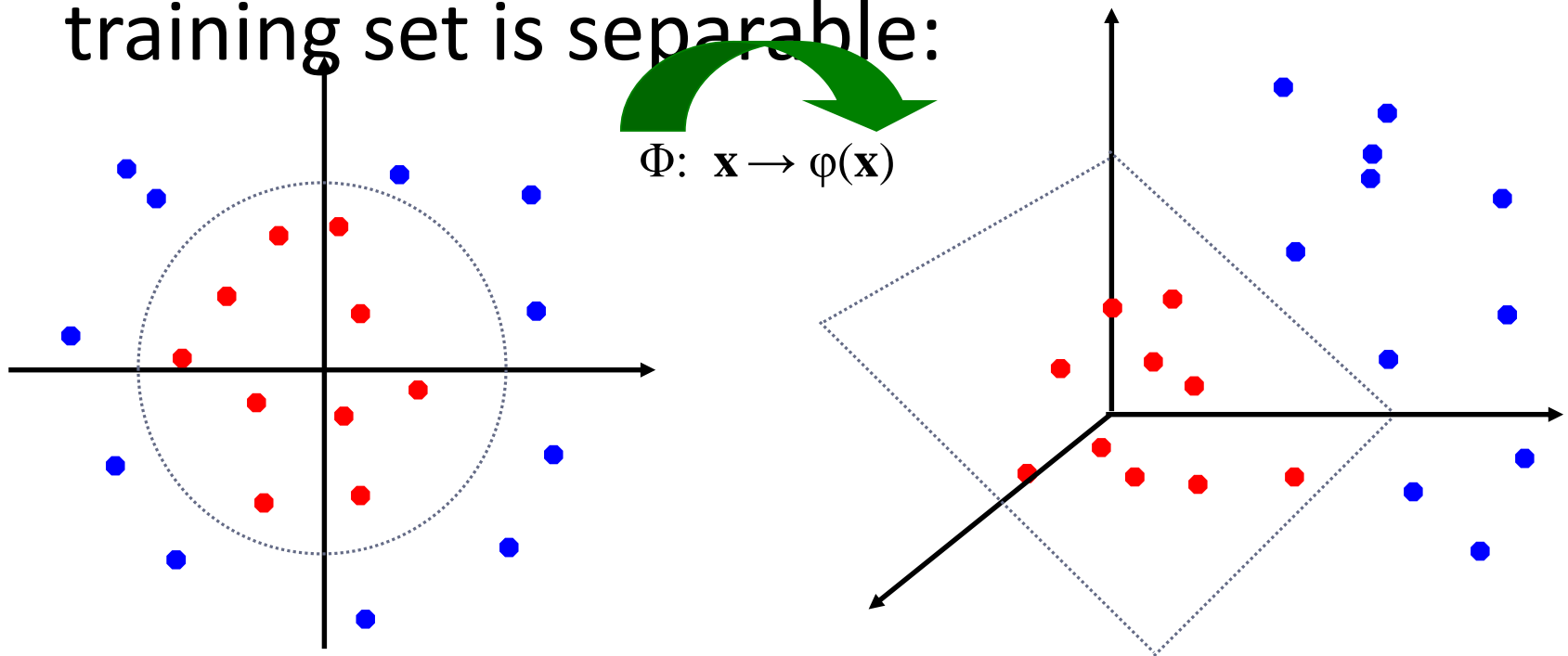- But what are we going to do if the dataset is just too hard?

- How about ... mapping data to a higher-dimensional space:

Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:

$$\Phi: \ \mathbf{x} \rightarrow \varphi(\mathbf{x})$$

The "Kernel Trick"

- The linear classifier relies on an inner product between vectors $K(\mathbf{x_i},\mathbf{x_j})=\mathbf{x_i}^\mathsf{T}\mathbf{x_j}$

- If every data point is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \phi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x_i},\mathbf{x_j})= \phi(\mathbf{x_i})^\mathsf{T}\phi(\mathbf{x_j})$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.

- Example:

2-dimensional vectors $\mathbf{x}=[x_1\ \ x_2]$; let $K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^\mathsf{T}\mathbf{x_j})^2$,

Need to show that $K(\mathbf{x_i},\mathbf{x_j})= \phi(\mathbf{x_i})^\mathsf{T}\phi(\mathbf{x_j})$:

$K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^\mathsf{T}\mathbf{x_j})^2= 1+ x_{i1}^2x_{j1}^2 + 2\ x_{i1}x_{j1}\ x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$

$\quad = [1\ \ x_{i1}^2\ \ \sqrt2\ x_{i1}x_{i2}\ \ x_{i2}^2\ \ \sqrt2 x_{i1}\ \ \sqrt2 x_{i2}]^\mathsf{T}\ [1\ \ x_{j1}^2\ \ \sqrt2\ x_{j1}x_{j2}\ \ x_{j2}^2\ \ \sqrt2 x_{j1}\ \ \sqrt2 x_{j2}]$

$\quad = \phi(\mathbf{x_i})^\mathsf{T}\phi(\mathbf{x_j})\quad$ where $\phi(\mathbf{x}) = [1\ \ x_1^2\ \ \sqrt2\ x_1x_2\ \ x_2^2\ \ \sqrt2 x_1\ \ \sqrt2 x_2]$

SVM: Different Kernel functions

- Instead of computing the dot product on the transformed data, it is math. equivalent to applying a kernel function K($\mathbf{X}_i$, $\mathbf{X}_j$) to the original data, i.e., K($\mathbf{X}_i$, $\mathbf{X}_j$) = $\Phi(\mathbf{X}_i)^\top \Phi(\mathbf{X}_j)$

- Typical Kernel Functions

$$\text{Polynomial kernel of degree } h: \quad K(X_i, X_j) = (X_i \cdot X_j + 1)^h$$

$$\text{Gaussian radial basis function kernel}: \quad K(X_i, X_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$$

$$\text{Sigmoid kernel}: \quad K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)$$

- *SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional parameters)

Non-linear SVM

- Replace inner-product with kernel functions
  - Optimization problem

  > Find $\alpha_1 \ldots \alpha_N$ such that
  > $\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma \alpha_i - \frac{1}{2} \Sigma \Sigma \alpha_i \alpha_j y_i y_j \mathbf{K}(\mathbf{x_i}, \mathbf{x_j})$ is maximized and
  > (1) $\Sigma \alpha_i y_i = 0$
  > (2) $0 \leq \alpha_i \leq C$ for all α_i

  - Decision boundary

  > $f(\mathbf{x}) = \Sigma \alpha_i y_i \mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) + b$

Support Vector Machine

- Introduction

- Linear SVM

- Non-linear SVM

- Scalability Issues*
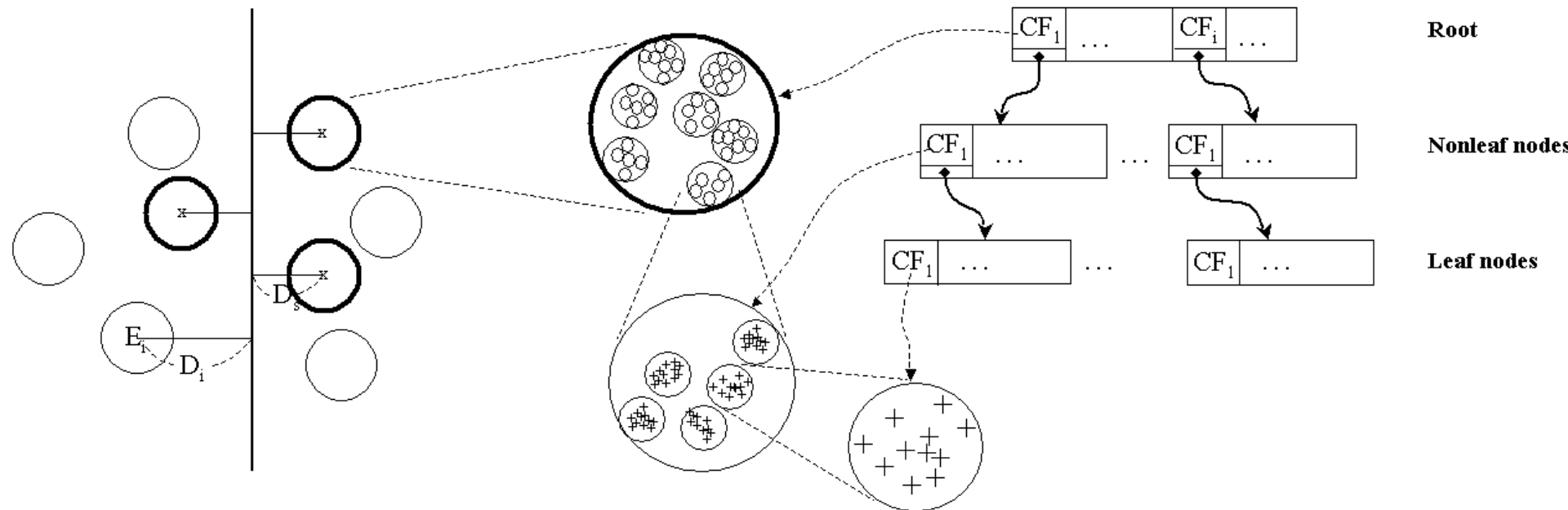
- Summary

*Scaling SVM by Hierarchical Micro-Clustering

- SVM is not scalable to the number of data objects in terms of training time and memory usage

- H. Yu, J. Yang, and J. Han, "Classifying Large Data Sets Using SVM with Hierarchical Clusters", KDD'03)

- CB-SVM (Clustering-Based SVM)

  - Given limited amount of system resources (e.g., memory), maximize the SVM performance in terms of accuracy and the training speed

  - Use micro-clustering to effectively reduce the number of points to be considered

  - At deriving support vectors, de-cluster micro-clusters near "candidate vector" to ensure high classification accuracy

*CF-Tree: Hierarchical Micro-cluster

- Read the data set once, construct a statistical summary of the data (i.e., hierarchical clusters) given a limited amount of memory

- Micro-clustering: Hierarchical indexing structure
  - provide finer samples closer to the boundary and coarser samples farther from the boundary

*Selective Declustering: Ensure High Accuracy

- CF tree is a suitable base structure for selective declustering

- De-cluster only the cluster E_i such that

  - $D_i - R_i < D_s$, where D_i is the distance from the boundary to the center point of E_i and R_i is the radius of E_i

  - Decluster only the cluster whose subclusters have possibilities to be the support cluster of the boundary

    - "Support cluster": The cluster whose centroid is a support vector
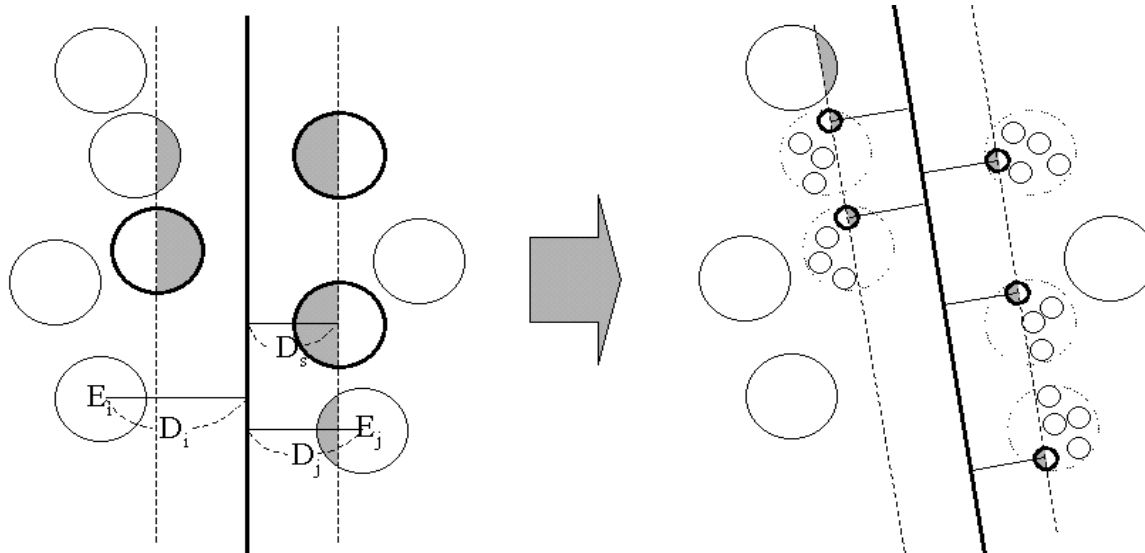
*CB-SVM Algorithm: Outline

- Construct two CF-trees from positive and negative data sets independently

    - Need one scan of the data set

- Train an SVM from the centroids of the root entries

- De-cluster the entries near the boundary into the next level

    - The children entries de-clustered from the parent entries are accumulated into the training set with the non-declustered parent entries

- Train an SVM again from the centroids of the entries in the training set

- Repeat until nothing is accumulated
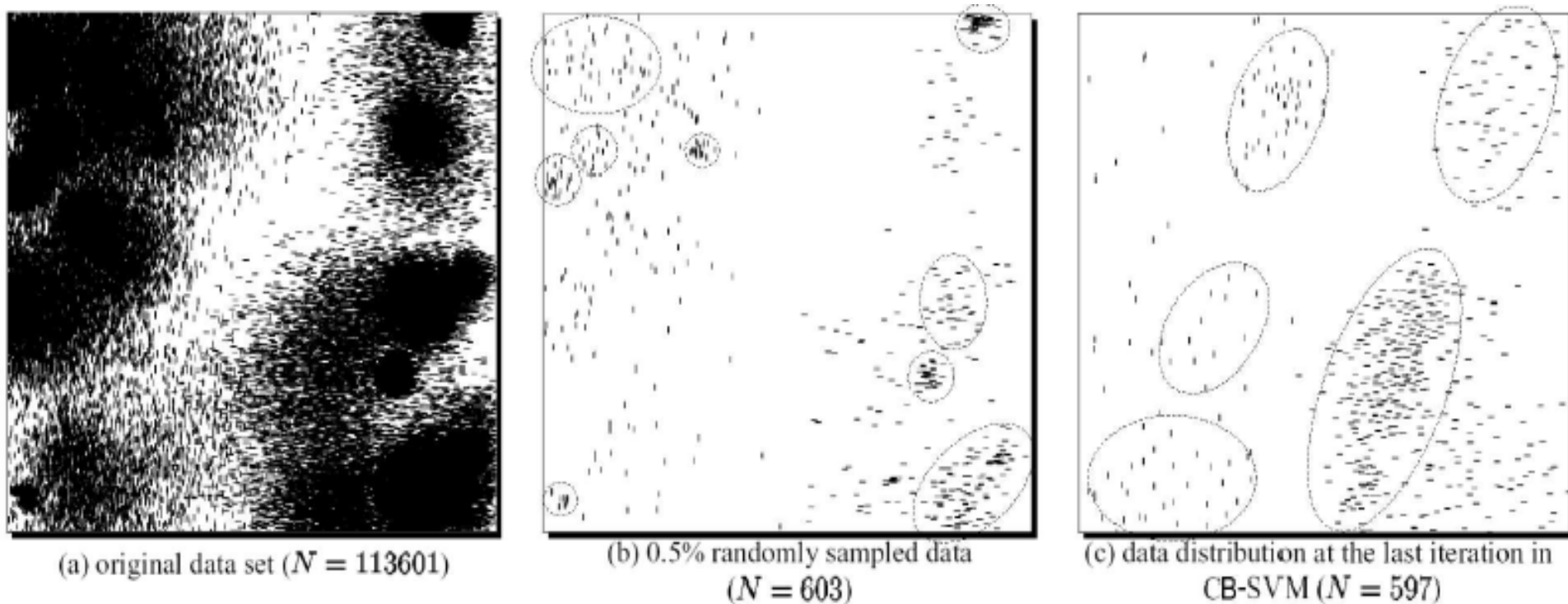
*Accuracy and Scalability on Synthetic Dataset

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

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

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

Figure 6: Synthetic data set in a two-dimensional space. '|': positive data; '−': negative data

- Experiments on large synthetic data sets shows better accuracy than random sampling approaches and far more scalable than the original SVM algorithm

Support Vector Machine

- Introduction

- Linear SVM

- Non-linear SVM

- Scalability Issues*

- Summary

Summary

- Support Vector Machine
  - Linear classifier; support vectors; kernel SVM

SVM Related Links

- SVM Website: http://www.kernel-machines.org/

- Representative implementations

  - **LIBSVM**: an efficient implementation of SVM, multi-class classifications, nu-**SVM**, one-class **SVM**, including also various interfaces with java, python, etc.

  - **SVM-light:** simpler but performance is not better than **LIBSVM**, support only binary classification and only in C

  - **SVM-torch**: another recent implementation also written in C

- From classification to regression and ranking:

  - http://www.dainf.ct.utfpr.edu.br/~kaestner/Mineracao/hwanjoyu-svmtutorial.pdf