Training Support Vector Machines using Gilbert's Algorithm

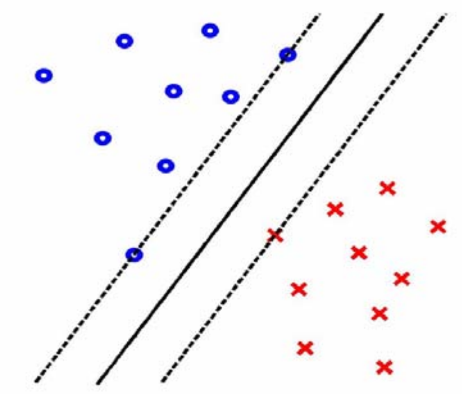
Shawn Martin

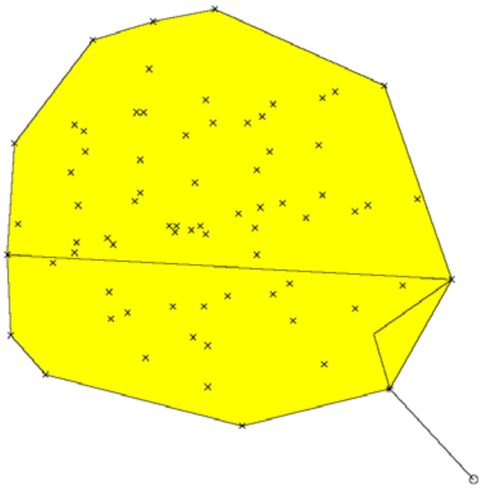
Sandia National Laboratories
Albuquerque, NM, USA

Nov. 30th, 2005

Outline of Talk

- Support Vector Machines
  - Background
  - Nonlinear Extension
  - Geometric Version
- Gilbert's Algorithm
  - Background
  - Problems
  - Modifications
- Examples/Comparisons
- Conclusions





Support Vector Machines (SVMs)

1) Starting with a dataset

$$\{(\mathbf{x}_i, y_i)\} \subseteq \mathbb{R}^n \times \{\pm 1\}$$

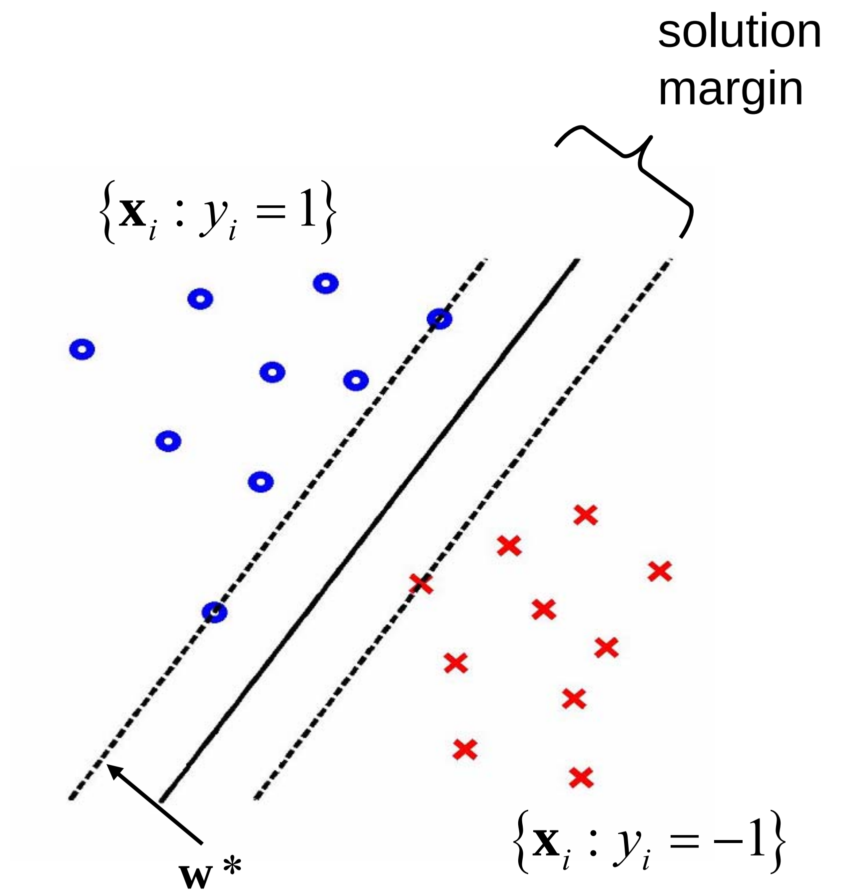
2) we solve the quadratic program

$$\max \ \sum_i \alpha_i - \tfrac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{s.t.} \quad \alpha_i \geq 0, \ \sum_i y_i \alpha_i = 0$$

3) to obtain the normal to the separating hyperplane

$$\mathbf{w}^* = \sum_i \alpha_i \mathbf{x}_i$$



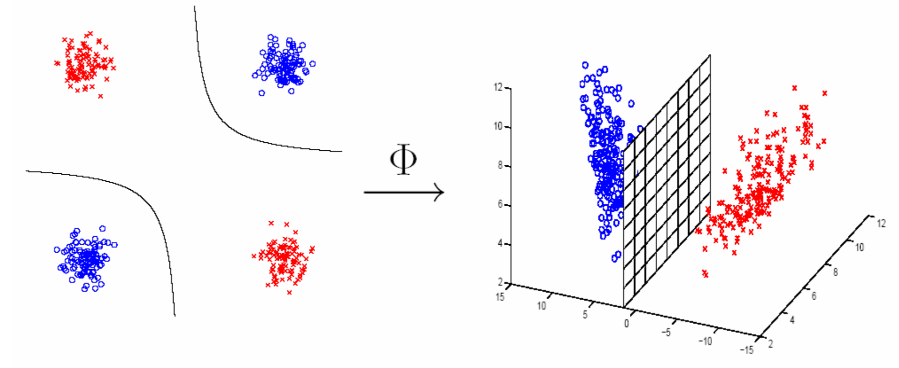
4) Support Vectors are $\mathbf{x}_i$ such that $\alpha_i \neq 0$, shown as lying on dashed lines. Distance between dashed lines is known as solution margin.

Nonlinear/Non-separable Extension of SVMs

1) Map the dataset into a higher dimensional space using a nonlinear map

$$\Phi : \mathbb{R}^n \to F.$$

2) Use the linear SVM classifier in the higher dimensional space.



3) Do this by replacing the inner products $(\mathbf{x}_i,\mathbf{x}_j)$ in the SVM problem with a kernel function, where a kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ corresponds to Φ such that

$$k\left(\mathbf{x}_i,\mathbf{x}_j\right) = \left(\Phi\left(\mathbf{x}_i\right),\Phi\left(\mathbf{x}_j\right)\right).$$

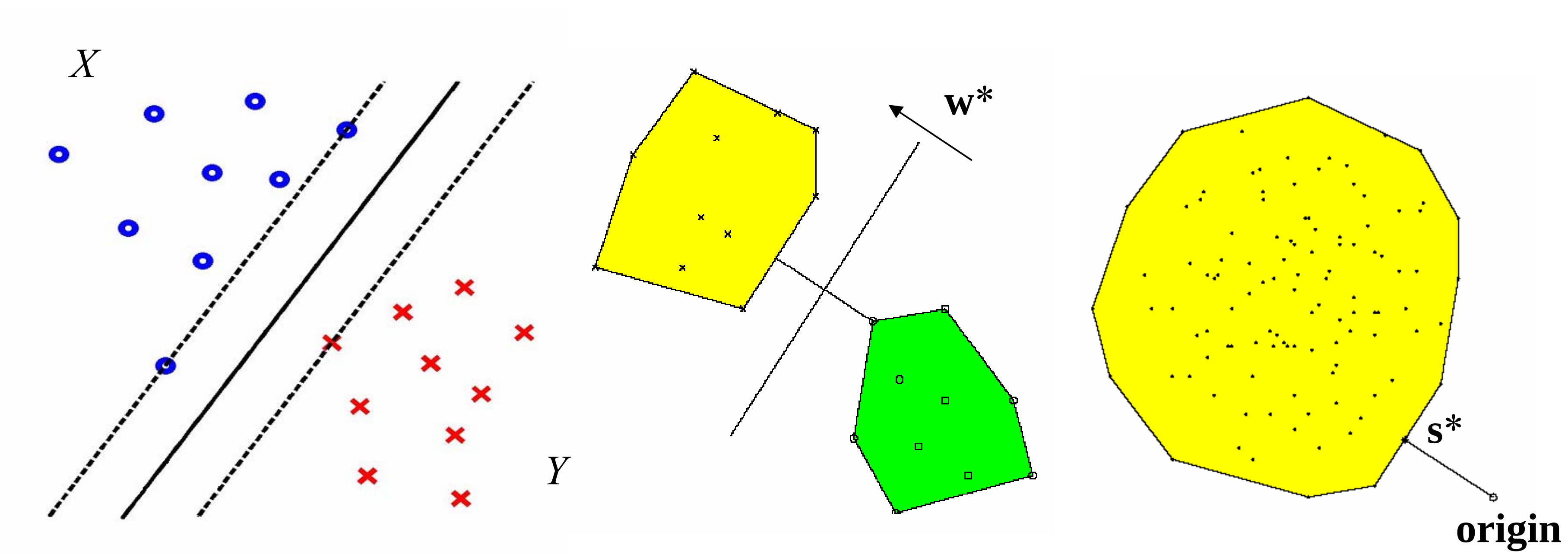
4) If our dataset is non-separable, we can use a kernel function of the form

$$\tilde{k}\left(\mathbf{x}_i,\mathbf{x}_j\right) = k\left(\mathbf{x}_i,\mathbf{x}_j\right) + \delta_{ij}/\tilde{C}.$$

Geometric Version of the SVM Problem

Let $X = \{\mathbf{x}_i : y_i = 1\}$, $Y = \{\mathbf{x}_i : y_i = -1\}$, and $S = X - Y$.

Then the normal to the separating hyperplane **w*** can be obtained from the point **s*** closest to the origin in the convex hull of the secant set S.



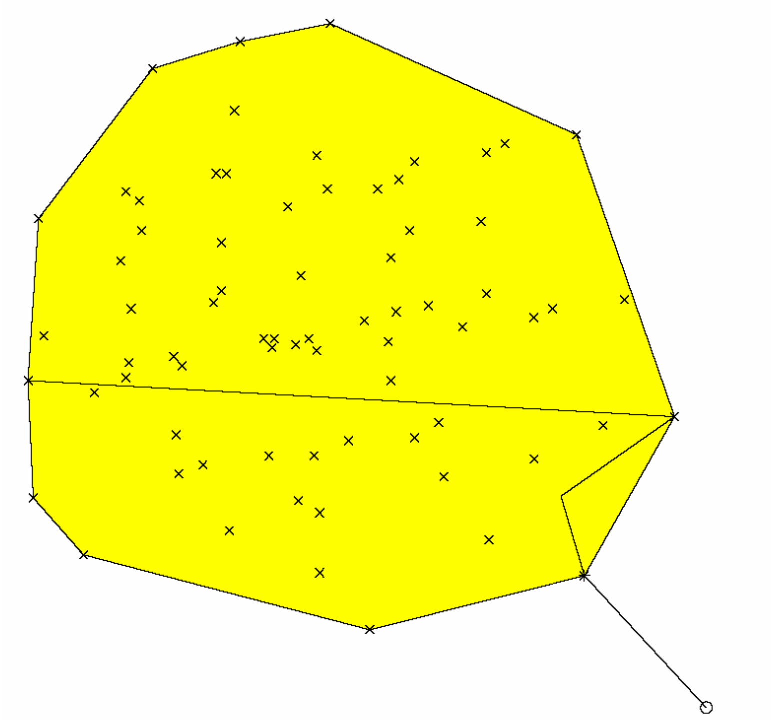
Finding Closest Point on Convex Hull

Q. How can we find the point **s*** on the convex hull of *S* closest to the origin?

A. One solution is to use Gilbert's Algorithm (1966). This was originally attempted in (Keerthi *et al*., 2000).

<u>Overview of Gilbert's Algorithm</u>

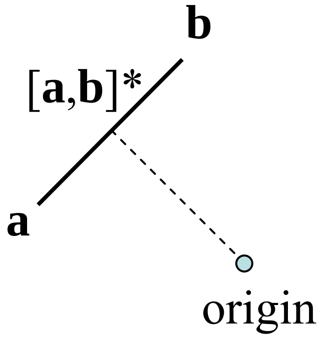
1. Choose a point $\mathbf{w}_1$ in *S*.
2. Identify the point $g^*(-\mathbf{w}_1)$ in *S* closest to the origin in the direction of $-\mathbf{w}_1$.
3. Identify the point $\mathbf{w}_2$ on the line from $\mathbf{w}_1$ to $g^*(-\mathbf{w}_1)$ closest to the origin.
4. Repeat 2-3.



Formalizing Gilbert's Algorithm (Definitions)

For $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ we set

$$[\mathbf{a}, \mathbf{b}]^* = \begin{cases} \mathbf{a} & \text{if} \quad -(\mathbf{a}, \mathbf{b}-\mathbf{a}) \leq 0 \\ \mathbf{a} + \frac{-(\mathbf{a},\mathbf{b}-\mathbf{a})}{\|\mathbf{b}-\mathbf{a}\|^2}(\mathbf{b}-\mathbf{a}) & \text{if} \quad 0 < -(\mathbf{a}, \mathbf{b}-\mathbf{a}) < \|\mathbf{b}-\mathbf{a}\|^2 \\ \mathbf{b} & \text{if} \quad \|\mathbf{b}-\mathbf{a}\|^2 \leq -(\mathbf{a}, \mathbf{b}-\mathbf{a}) \end{cases}$$



The point $[\mathbf{a}, \mathbf{b}]^*$ is the point on the line segment from $\mathbf{a}$ to $\mathbf{b}$ closest to the origin.

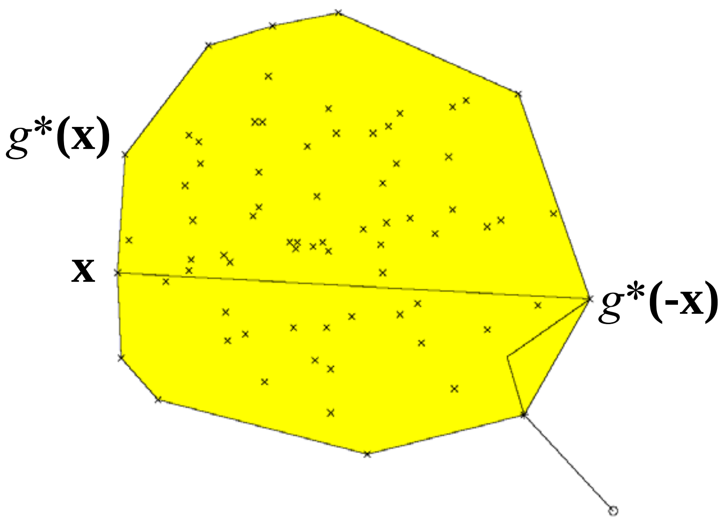
We define the support function $g : \mathbb{R}^n \to \mathbb{R}$ by

$$g(\mathbf{x}) = \max_m \{(\mathbf{x}, \mathbf{s}_m)\},$$

and the contact function $g^* : \mathbb{R}^n \to \mathbb{R}^n$ by

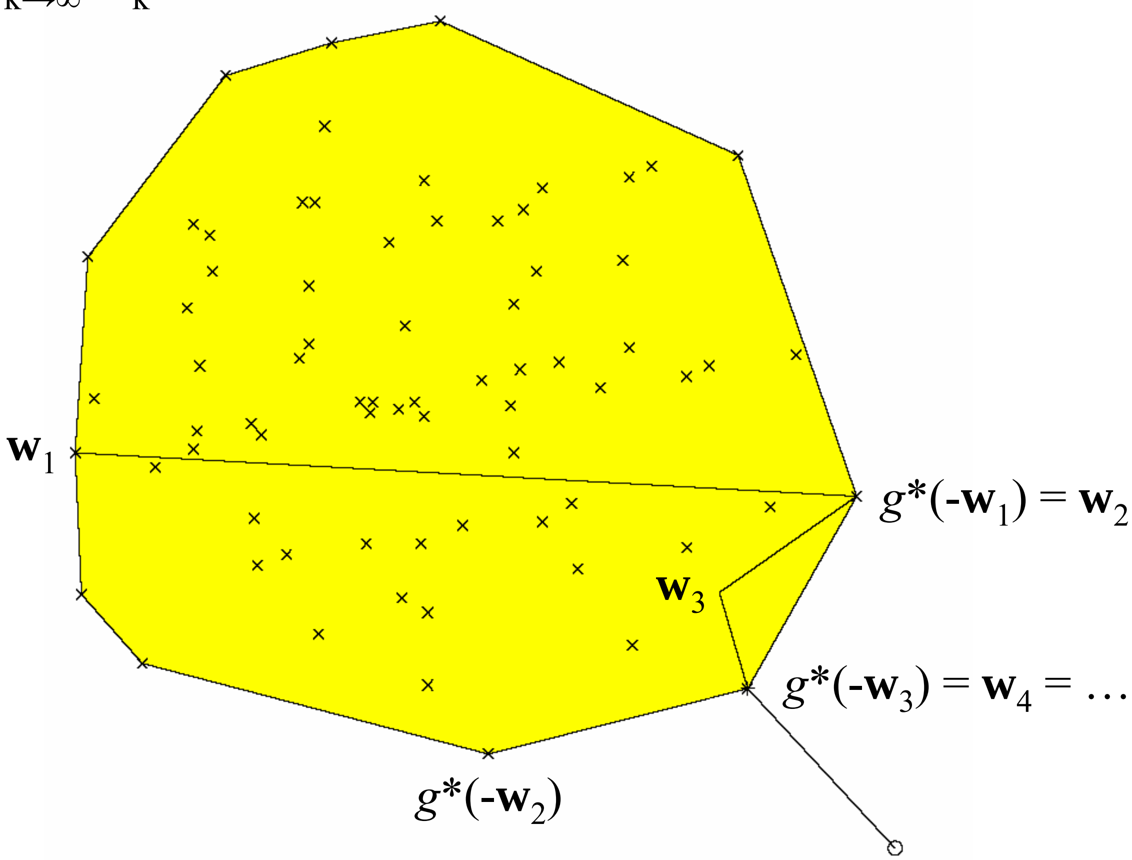
$$g^*(\mathbf{x}) = \mathbf{s}_{m_0},$$

for some uniquely defined m_0.



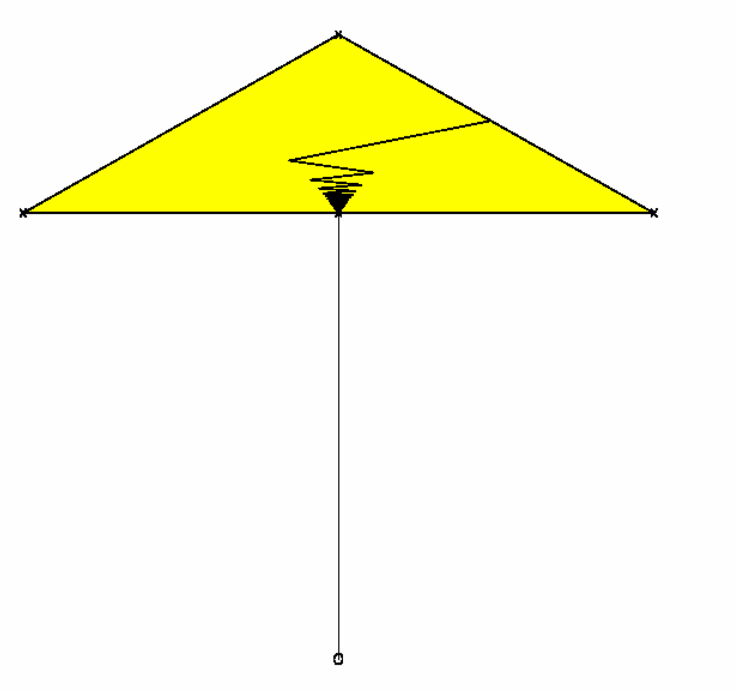
Gilbert's Algorithm

1. Choose a point $\mathbf{w}_1$ in S.
2. Identify the point $g^*(-\mathbf{w}_1)$ in S closest to the origin in the direction of $-\mathbf{w}_1$.
3. Identify the point $\mathbf{w}_2 = [\mathbf{w}_1, g^*(-\mathbf{w}_1)]^*$.
4. Repeat 2-3 indefinitely.
5. $\mathbf{s}^* = \lim_{k\to\infty} \mathbf{w}_k$.



Problem with Gilbert's Algorithm

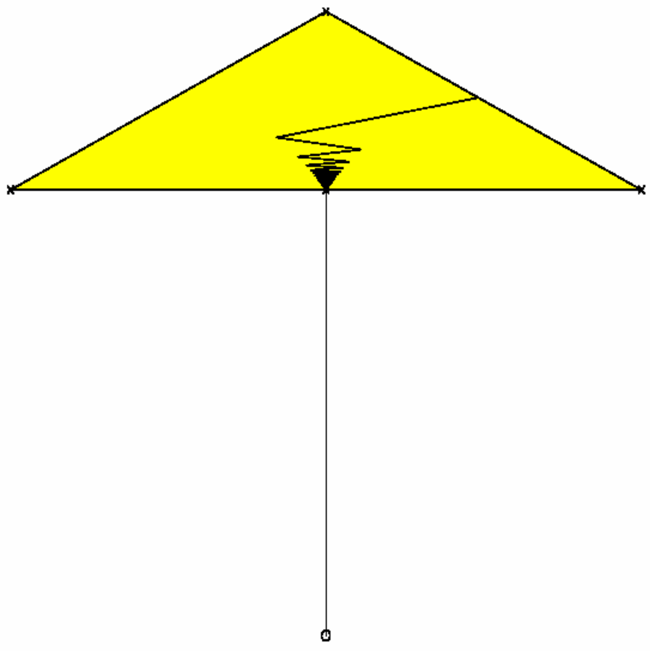
Gilbert's Algorithm often gets "stuck" in very slow (~1/n) asymptotic convergence.



Can we fix this?

Observations about Gilbert's Algorithm

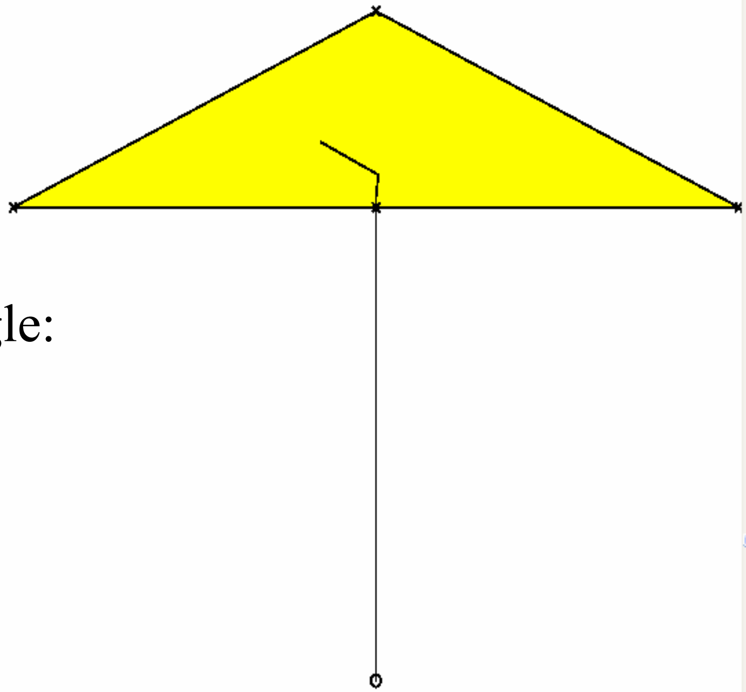
1) Gilbert's Algorithm identifies a subset *S'* of *S* and iterates between the vectors in the subset indefinitely.
2) Gilbert's Algorithm appears to converge faster in angle than in norm: $(\mathbf{w}_k,\mathbf{s}^*)/(\|\mathbf{w}_k\|\,\|\mathbf{s}^*\|) \sim 1/n^2$.



Modifications to Gilbert's Algorithm

1) Construct $\bar{\mathbf{m}}_1$ from $\mathbf{w}_1$, $\mathbf{w}_2$, … by using the subset of $S' = \{\mathbf{s}_j,\ldots,\mathbf{s}_k\}$ identified by Gilbert's Algorithm:

$$\bar{\mathbf{m}}_1 = \frac{1}{k-j}\sum_{i=j+1}^{k}\mathbf{w}_i$$

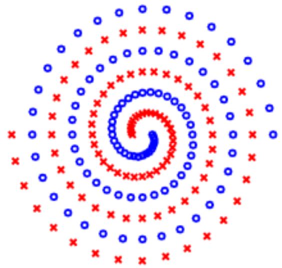


2) Repeat to obtain $\bar{\mathbf{m}}_2$, $\bar{\mathbf{m}}_3$, …

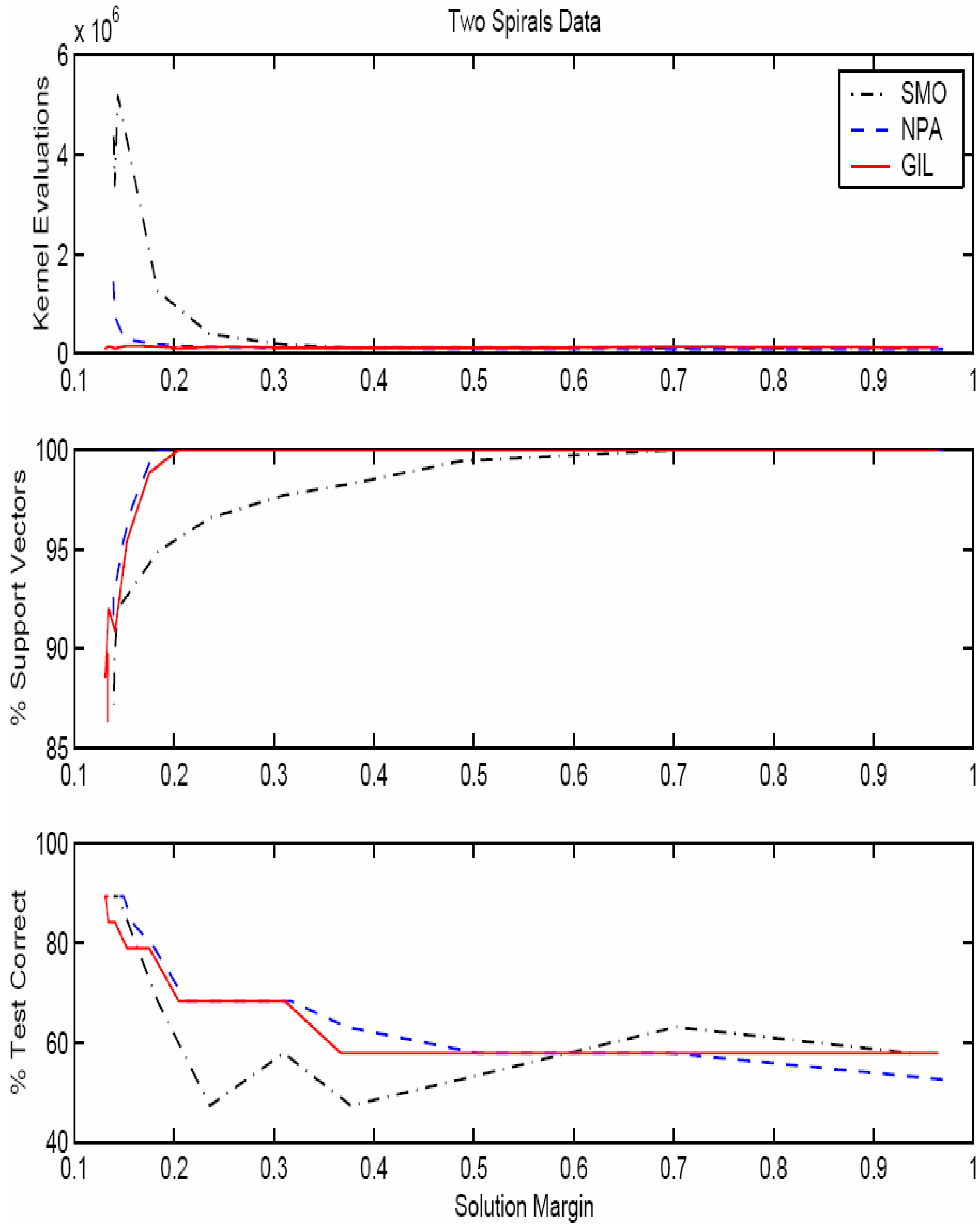
3) Stop when $\bar{\mathbf{m}}_1$, $\bar{\mathbf{m}}_2$, … converges in angle:

$$\frac{\left(\bar{\mathbf{m}}_l,\bar{\mathbf{m}}_{l-1}\right)}{\left\|\bar{\mathbf{m}}_l\right\|\left\|\bar{\mathbf{m}}_{l-1}\right\|} < \varepsilon.$$

Example: Two Spirals Dataset

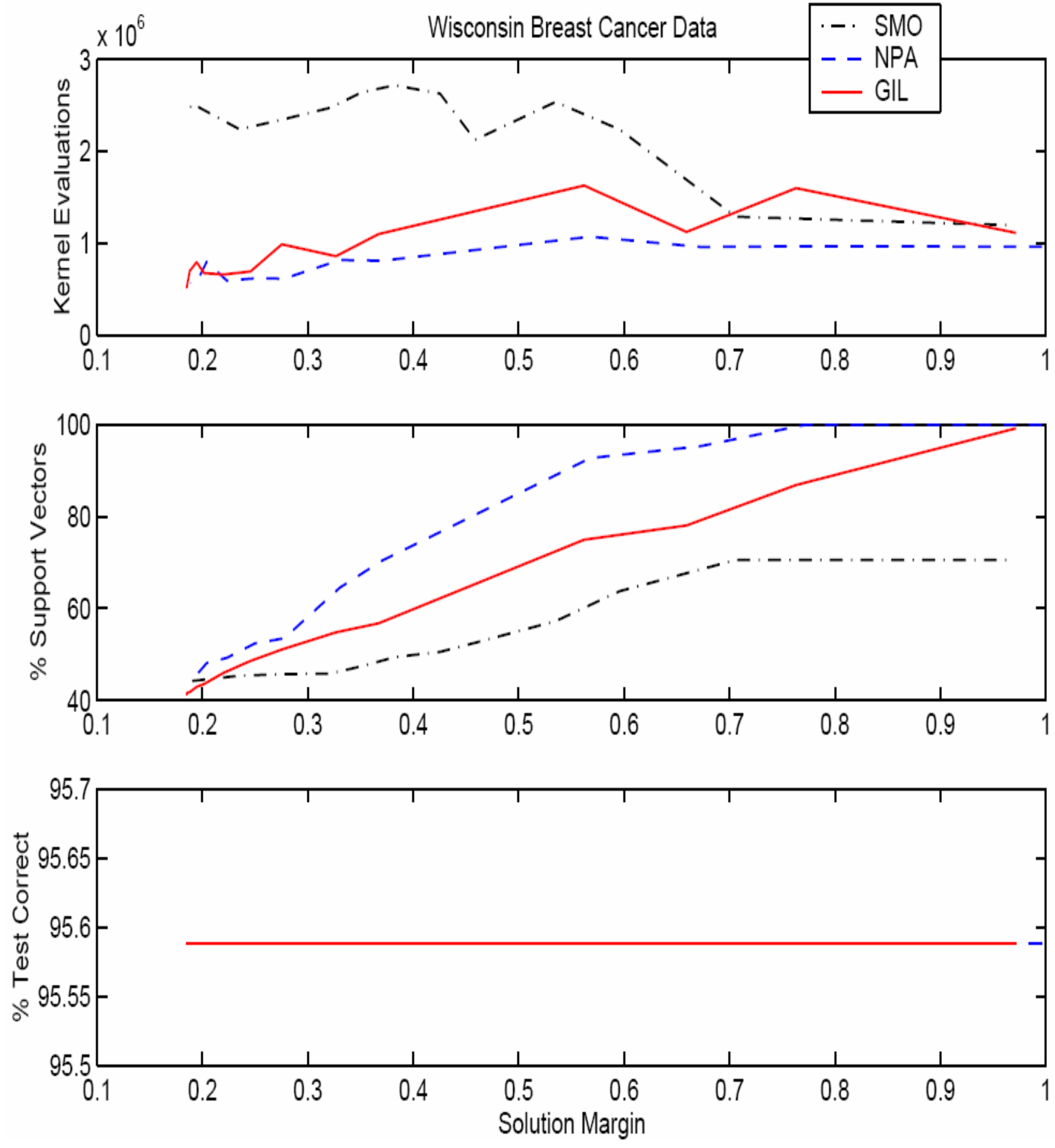


- We compared our method to Sequential Minimal Optimization (SMO) and the Nearest Point Algorithm (NPA) in (Keerthi *et al*., 2000).
- We measured speed using number of kernel evaluations.
- We compared the final solution using the percent of support vectors.
- We compared performance accuracy by using a test set.
- In all cases we used solution margin (distance between two classes) to measure classifier similarity.



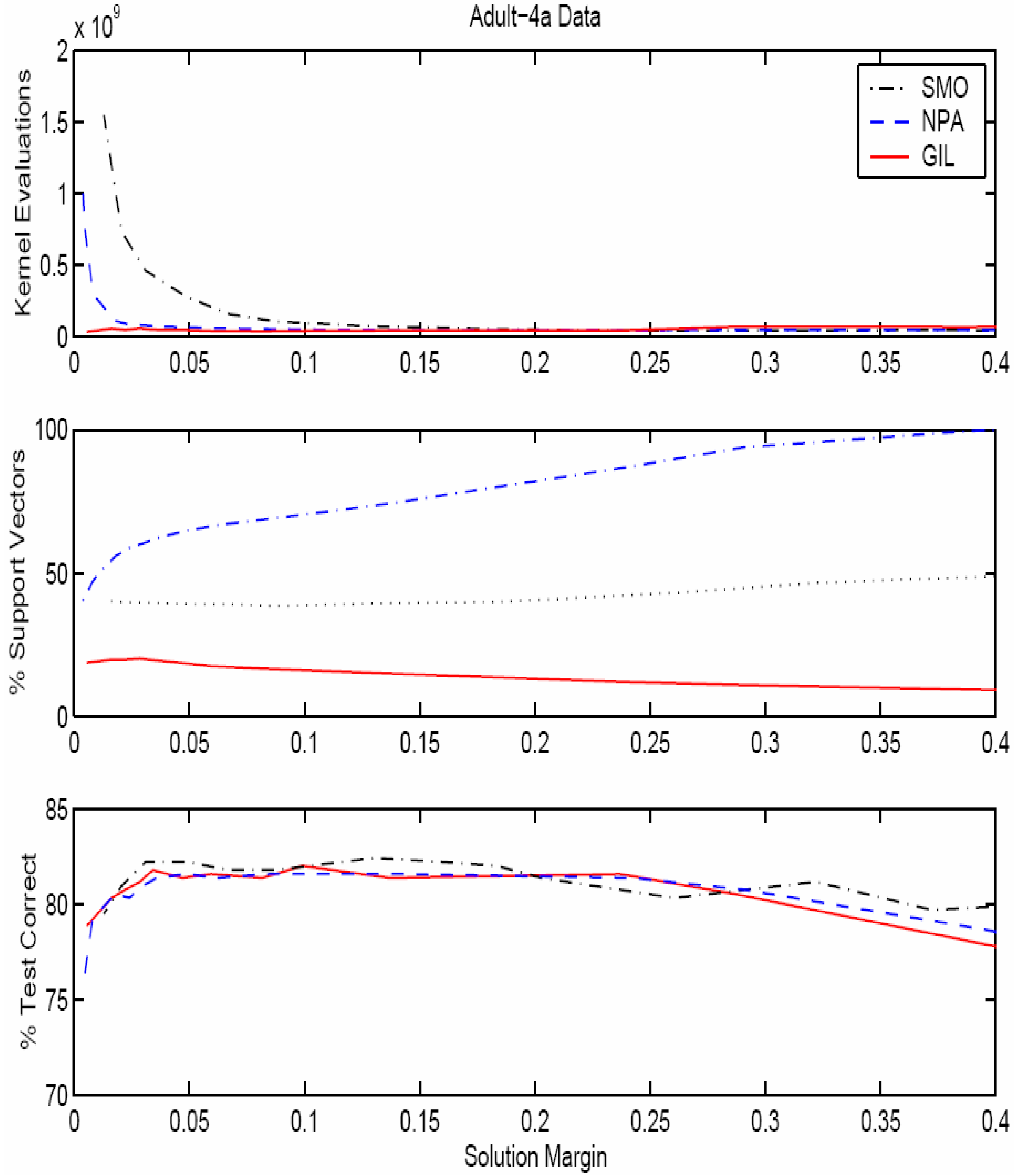
Example: Wisconsin Breast Cancer Dataset

- Our comparisons indicate that our method is as fast and as accurate as standard methods.



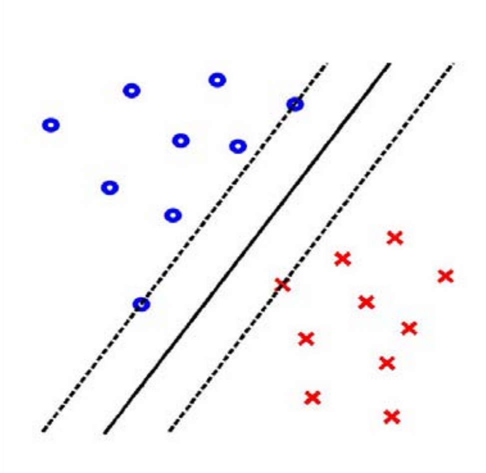
Example: Adult-4a Dataset

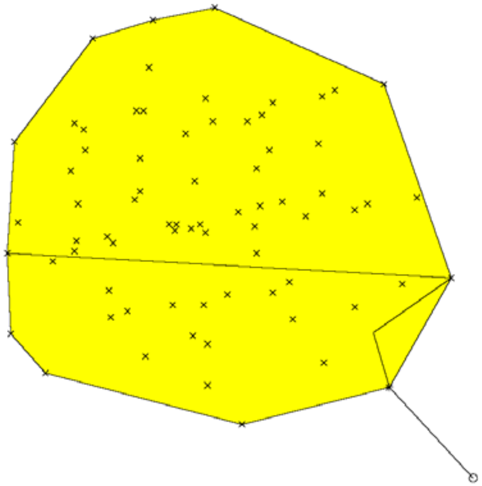
- In some cases we also get fewer support vectors.



Conclusions

- Modified Gilbert's Algorithm to successfully train SVMs.
- New algorithm appears to be fast.
- Results are as accurate as other methods.
- New algorithm may identify fewer SVs than other methods.
- Theoretical results should be derived to support/refute this approach.





Future Work

- Another possible direction:
  1) Identify subset *S'* of *S* using Gilbert's Algorithm.
  2) Solve for **s*** directly using *S'*.