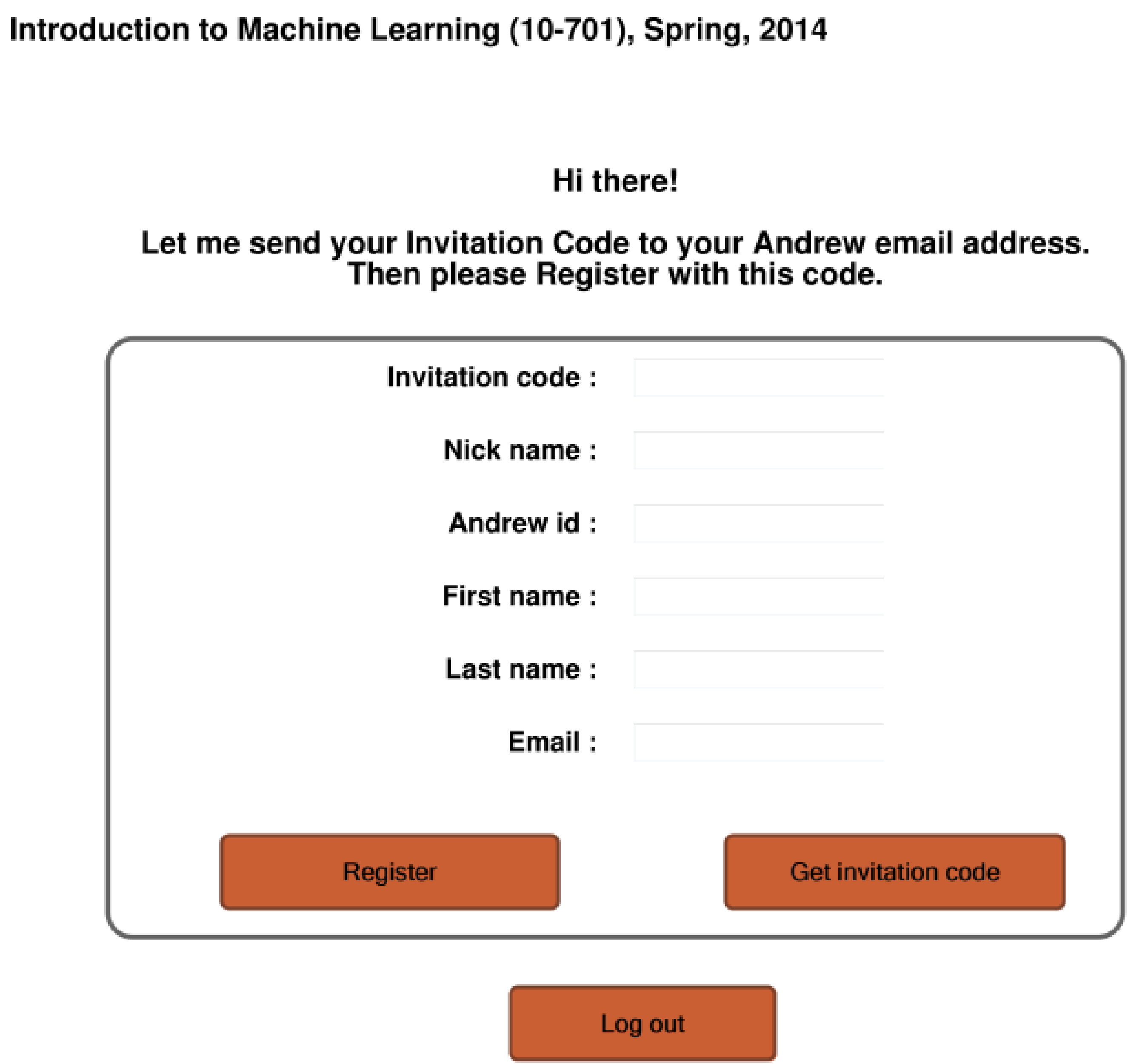
Introduction to Machine Learning CMU-10701

Support Vector Machines

Barnabás Póczos & Aarti Singh

2014 Spring

http://barnabas-cmu-10701.appspot.com/



Introduction to Machine Learning (10-701), Spring, 2014

Welcome test1! Login successful!

Nick name: test1

Andrew Id: bapoczos_v3

First name: test1

Last name: test1

Email: peergrading.test1@gmail.com

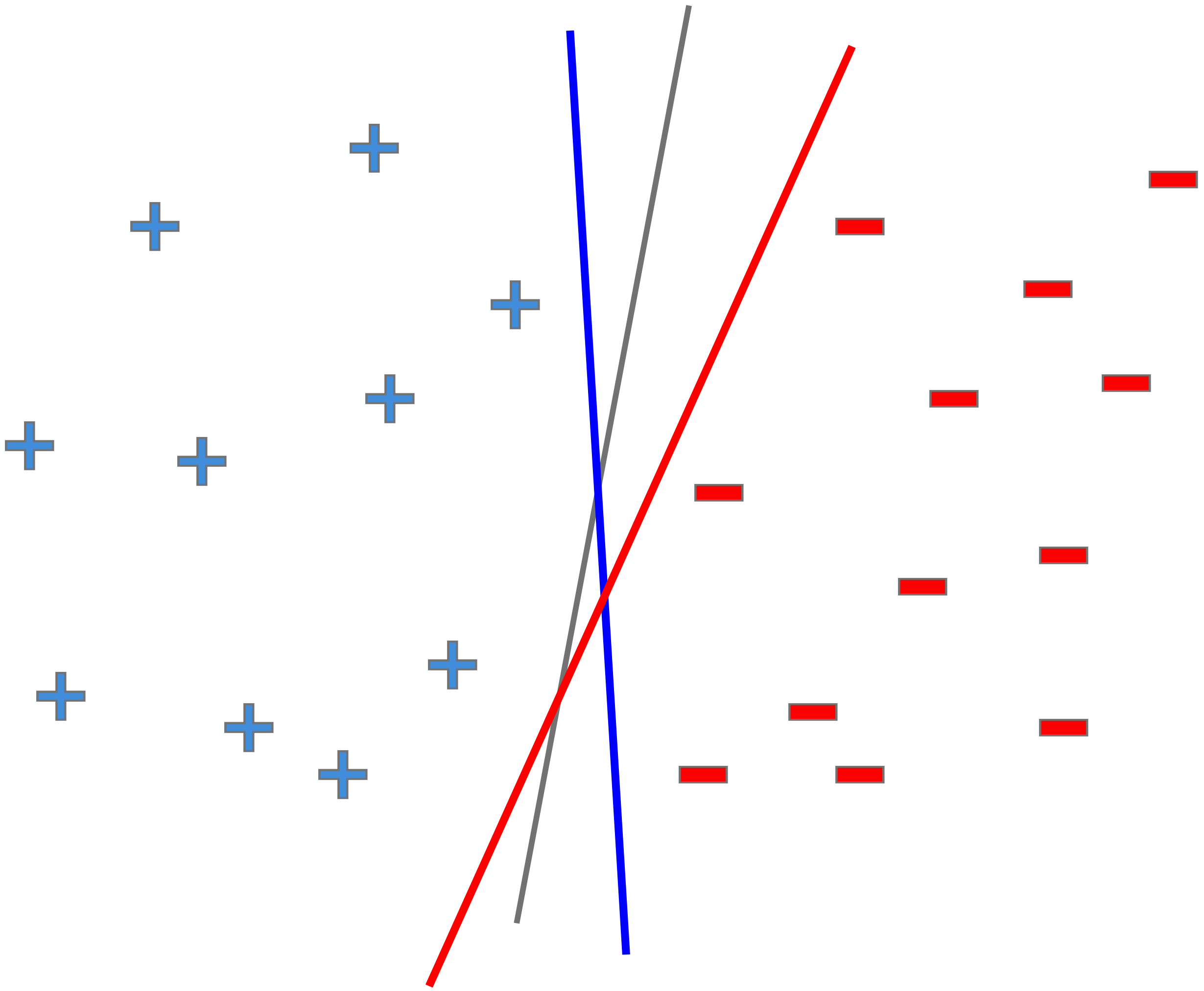
Edit Log out

Your role:

Author of HW1 Author of HW2 Author of HW3 Author of HW4

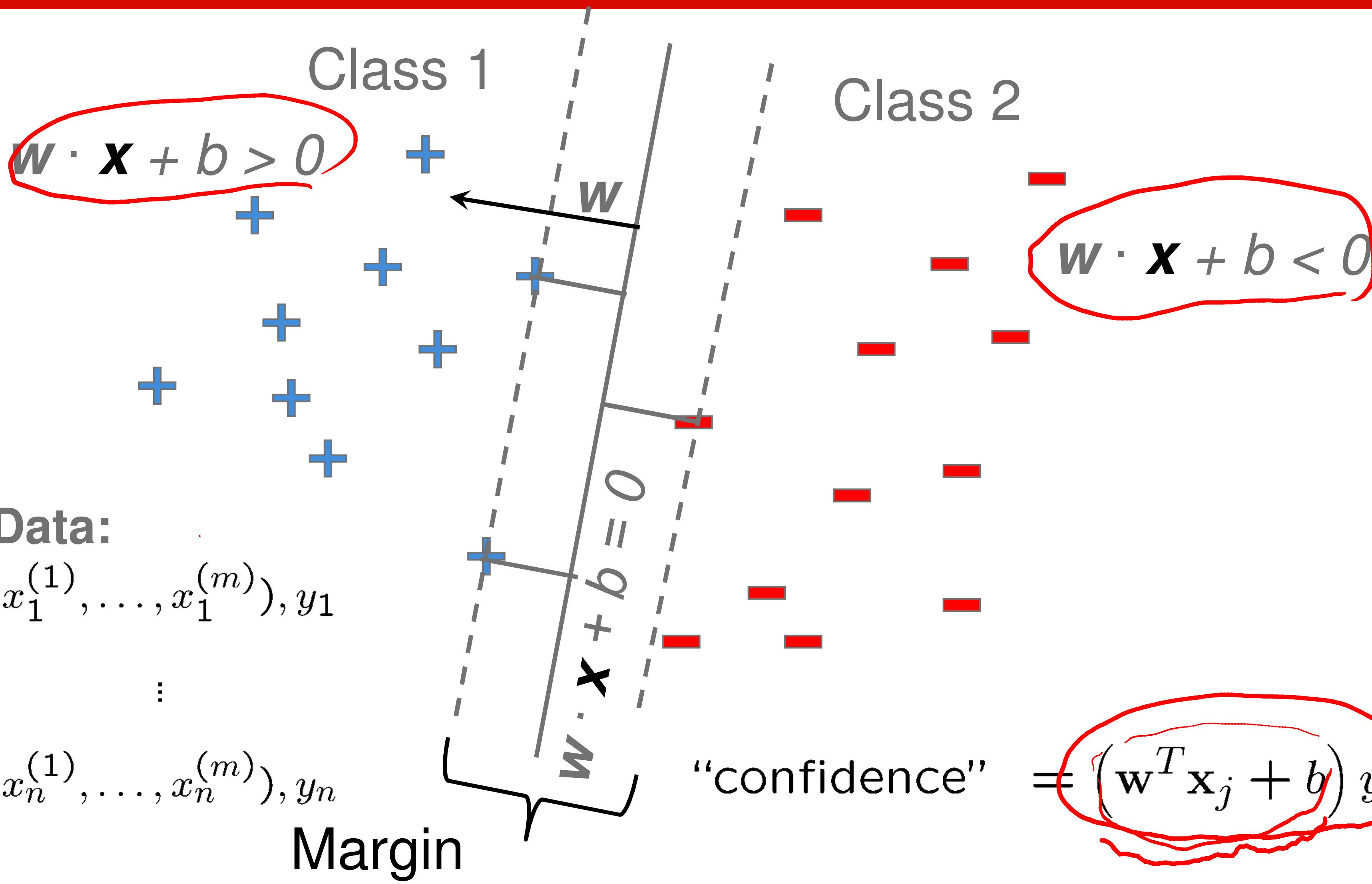
Reviewer of HW1 Reviewer of HW2 Reviewer of HW3 Reviewer of HW4

Linear classifiers
which line is better?



Which decision boundary is better?

Pick the one with the largest margin!



Data:

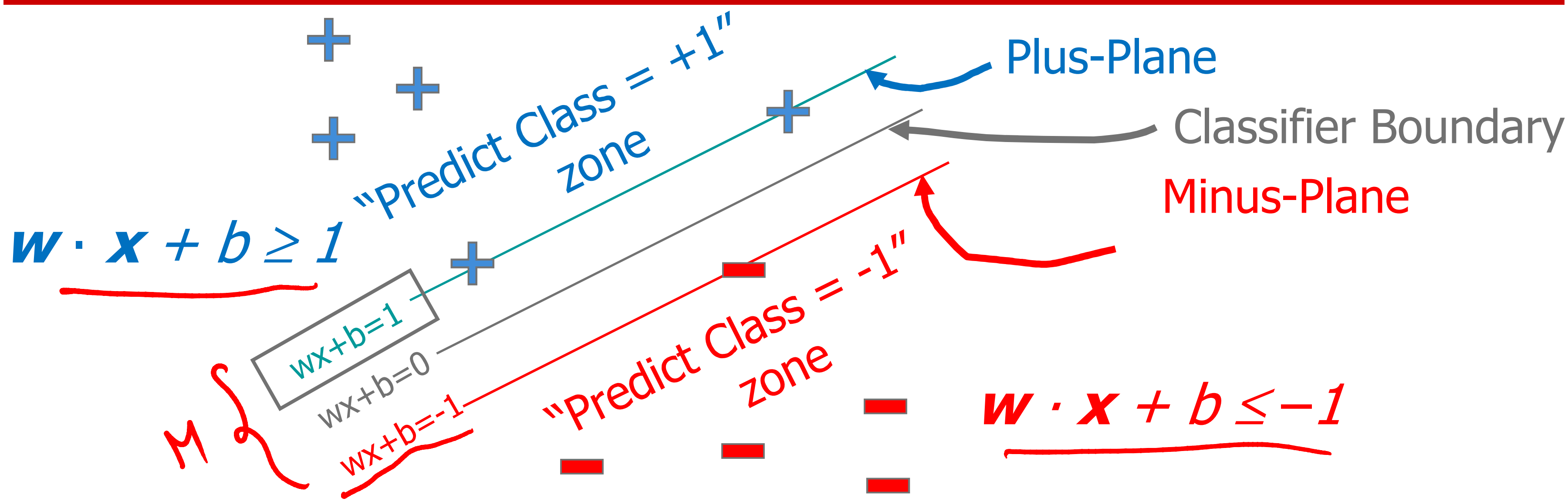
$$(x_1^{(1)}, \ldots, x_1^{(m)}), y_1$$

$$\vdots$$

$$(x_n^{(1)}, \ldots, x_n^{(m)}), y_n$$

"confidence" $= \left(\mathbf{w}^T \mathbf{x}_j + b\right) y_j$

Scaling



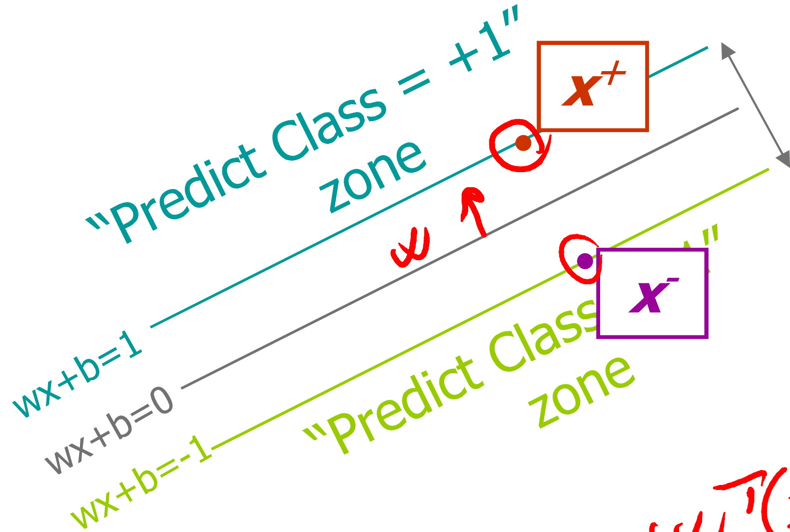
Classification rule:

Classify as..	+1	if	$\boldsymbol{w} \cdot \boldsymbol{x} + b \geq 1$
	−1	if	$\boldsymbol{w} \cdot \boldsymbol{x} + b \leq -1$
	Universe explodes	if	$-1 < \boldsymbol{w} \cdot \boldsymbol{x} + b < 1$

How large is the margin of this classifier?

Goal: Find the maximum margin classifier

Computing the margin width



M = Margin Width $= \dfrac{2}{\sqrt{\mathbf{w}\cdot\mathbf{w}}}$

$$w^T(\underbrace{x^+ - x^-}_{\lambda w}) = 2$$

$$\lambda \cdot w^T w = 2$$

$$\lambda = \frac{2}{w^T w}$$

$$M = |x^+ - x^-| = |\lambda w| = \left|\frac{2w}{w^T w}\right| = \frac{2}{\sqrt{w^T w}}$$

Let $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$ be such that

- $\boldsymbol{w} \cdot \boldsymbol{x}^+ + b = +1$
- $\boldsymbol{w} \cdot \boldsymbol{x}^- + b = -1$
- $\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda\, \boldsymbol{w}$
- $|\boldsymbol{x}^+ - \boldsymbol{x}^-| = M$

Maximize M ≡ minimize **w·w !**

Observations

We can assume b=0

Classify as..	+1	if	$\boldsymbol{w} \cdot \boldsymbol{x} + b \geq 1$
	-1	if	$\boldsymbol{w} \cdot \boldsymbol{x} + b \leq -1$
	Universe explodes	if	$-1 < \boldsymbol{w} \cdot \boldsymbol{x} + b < 1$

This is the same as $y_i \langle \mathbf{x}_i, \mathbf{w} \rangle \geq 1, \ \forall i = 1, \ldots, n$

The Primal Hard SVM

- Given $D = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$ training data set.
- Assume that D is **linearly separable**.

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w} \in \mathbb{R}^m} \frac{1}{2} ||\mathbf{w}||^2$$

$$\text{subject to } y_i \langle \mathbf{x}_i, \mathbf{w} \rangle \geq 1, \ \forall i = 1, \ldots, n$$

Prediction: $f_{\hat{\mathbf{w}}}(\mathbf{x}) = \text{sign}(\langle \hat{\mathbf{w}}, \mathbf{x} \rangle)$

This is a QP problem (m-dimensional)
(Quadratic cost function, linear constraints)

Quadratic Programming

Find $\underset{w \in \mathbb{R}^n}{\mathrm{ARG\,MIN}}\; w^T H w + w^T q + e$

Subject to

$$A w \leq b$$

$$A \in \mathbb{R}^{n \times m} \quad w \in \mathbb{R}^m \quad b \in \mathbb{R}^n$$

and to $C w = d$

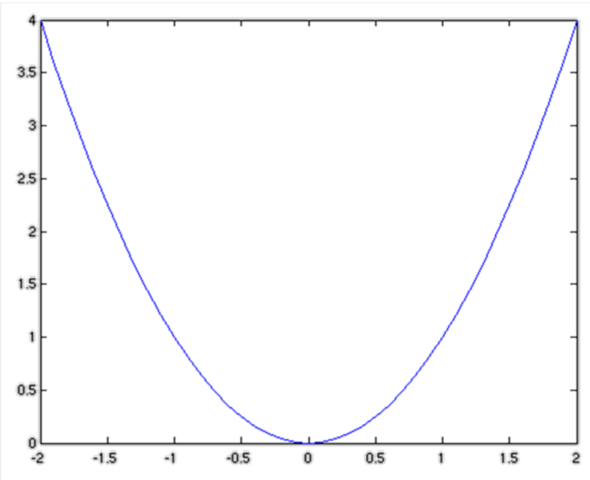
$$C \in \mathbb{R}^{s \times m} \quad d \in \mathbb{R}^s$$

Efficient Algorithms exist for QP.
They often solve the dual problem instead of the primal.

Constrained Optimization

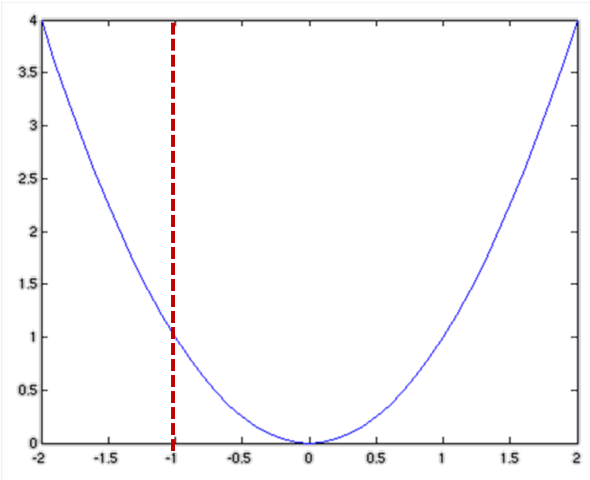
$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$

$$\min_x \; x^2$$



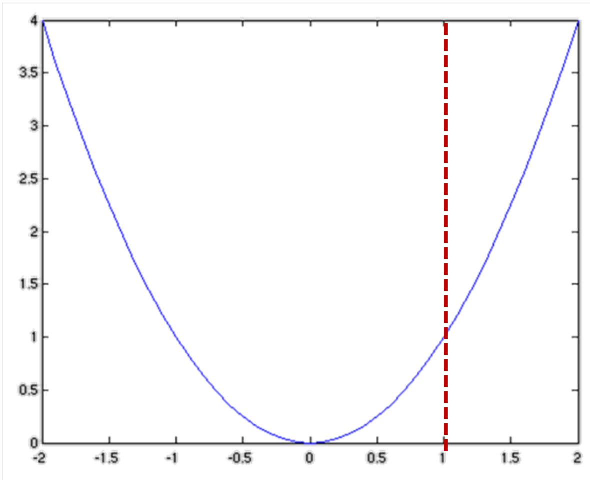
$$x^* = 0$$

$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq -1$$



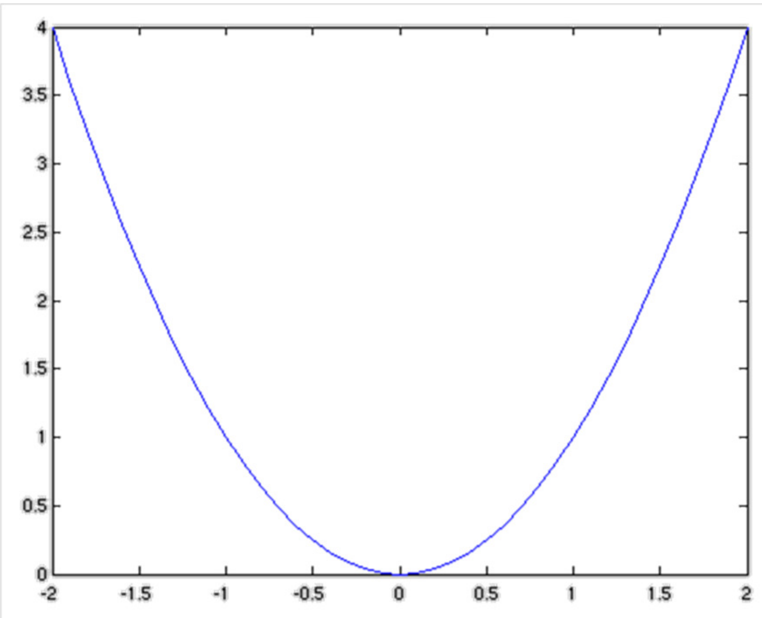
$$x^* = 0$$

$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq 1$$



$$x^* = 1$$

Lagrange Multiplier



$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$

$x - b \geq 0$

Moving the constraint to objective function

Lagrangian:

$$L(x, \alpha) = x^2 - \alpha(x - b)$$
$$\text{s.t.} \quad \alpha \geq 0$$

Solve:

$$\min_x \max_\alpha \; L(x, \alpha)$$
$$\text{s.t.} \quad \alpha \geq 0$$

x^*, α^*

Constraint is active when α > 0

Lagrange Multiplier – Dual Variables

Solving:

$$\begin{cases} \min_x \max_\alpha \ \overbrace{x^2 - \alpha(x-b)}^{L(x,\alpha)} \\ \text{s.t.} \quad \alpha \geq 0 \end{cases}$$

$$\frac{\partial L}{\partial x} = 0 \ \Rightarrow x^* = \frac{\alpha}{2}$$

$$\frac{\partial L}{\partial \alpha} = 0 \ \Rightarrow \alpha^* = \max(2b, 0)$$

$$\frac{\partial L(x,\alpha)}{\partial x} = 2x - \alpha = 0$$

$$x^* = \boxed{\frac{\alpha}{2}}$$

$$L(x^*,\alpha) = \underbrace{\frac{\alpha^2}{4} - \alpha\left(\frac{\alpha}{2} - b\right)}_{-\frac{\alpha^2}{4} + \alpha b}$$

$$\frac{\partial L(x^*,\alpha)}{\partial \alpha} = -\frac{\alpha}{2} + b = 0 \Rightarrow \alpha^* = 2b$$

$$x^* = \mathrm{MAX}(b,0) \qquad \alpha^* = \mathrm{MAX}(2b,0)$$

When α > 0, constraint is tight

From Primal to Dual

Primal problem:

$$\hat{\mathbf{w}} = \text{arg} \min_{\mathbf{w} \in \mathbb{R}^m} \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } y_i \langle \mathbf{x}_i, \mathbf{w} \rangle \geq 1, \ \forall i = 1, \dots, n$$

Lagrange function:

$$\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)^T \geq 0 \text{ Lagrange multipliers}$$

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \big(y_i \langle \mathbf{x}_i, \mathbf{w} \rangle - 1 \big)$$

The Lagrange Problem

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \Big(y_i \langle \mathbf{x}_i, \mathbf{w} \rangle - 1 \Big)$$

The Lagrange problem:

$$(\hat{\mathbf{w}}, \hat{\boldsymbol{\alpha}}) = \arg \min_{\mathbf{w} \in \mathbb{R}^m} \max_{0 \leq \boldsymbol{\alpha} \in \mathbb{R}^n} L(\mathbf{w}, \boldsymbol{\alpha})$$

$$0 = \left. \frac{\partial L(\mathbf{w}, \boldsymbol{\alpha})}{\partial \mathbf{w}} \right|_{\mathbf{w} = \hat{\mathbf{w}}} = \hat{\mathbf{w}} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\boxed{\Rightarrow \hat{\mathbf{w}} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i}$$

The Dual Problem

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \Big(y_i \langle \mathbf{x}_i, \mathbf{w} \rangle - 1 \Big)$$

$$\boxed{\Rightarrow \hat{\mathbf{w}} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i}$$

$$\Rightarrow L(\hat{\mathbf{w}}, \boldsymbol{\alpha}) = \frac{1}{2}\|\hat{\mathbf{w}}\|^2 - \sum_{i=1}^{n} \alpha_i \Big(y_i \langle \mathbf{x}_i, \hat{\mathbf{w}} \rangle - 1 \Big)$$

$$= \underbrace{\frac{1}{2} \| \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \|^2}_{\boldsymbol{\alpha}^T \boldsymbol{Y} \boldsymbol{G} \boldsymbol{Y} \boldsymbol{\alpha}} + \boldsymbol{\alpha}^T \mathbf{1}_n - \underbrace{\sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{x}_i, \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j \rangle}_{\boldsymbol{\alpha}^T \boldsymbol{Y} \boldsymbol{G} \boldsymbol{Y} \boldsymbol{\alpha}}$$

$$\boxed{= \boldsymbol{\alpha}^T \mathbf{1}_n - \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{Y} \boldsymbol{G} \boldsymbol{Y} \boldsymbol{\alpha}}$$

$$\boldsymbol{Y} \doteq diag(y_1, \ldots, y_n), \;\; y_i \in \{-1, 1\}^n$$

$\boldsymbol{G} \in \mathbb{R}^{n \times n} \doteq \{G_{ij}\}_{i,j}^{n,n}$, where $G_{ij} \doteq \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ Gram matrix.

The Dual Hard SVM

$$\boldsymbol{Y} \doteq diag(y_1, \ldots, y_n), \; y_i \in \{-1, 1\}^n$$

$\boldsymbol{G} \in \mathbb{R}^{n \times n} \doteq \{G_{ij}\}_{i,j}^{n,n}$, where $G_{ij} \doteq \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ Gram matrix.

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\arg\max}\, \boldsymbol{\alpha}^T \mathbf{1}_n - \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{Y} \boldsymbol{G} \boldsymbol{Y} \boldsymbol{\alpha}$$

subject to $\alpha_i \geq 0, \; \forall i = 1, \ldots, n$

Quadratic Programming (n-dimensional)

Lemma $\hat{\mathrm{w}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i$

Prediction: $f_{\hat{\mathrm{w}}}(x) = \mathrm{sign}(\langle \hat{\mathrm{w}}, \mathbf{x} \rangle) = \mathrm{sign}(\sum_{i=1}^{n} \hat{\alpha}_i y_i \underbrace{\langle \mathbf{x}_i, \mathbf{x} \rangle}_{k(\mathbf{x}_i, \mathbf{x})})$

The Problem with Hard SVM

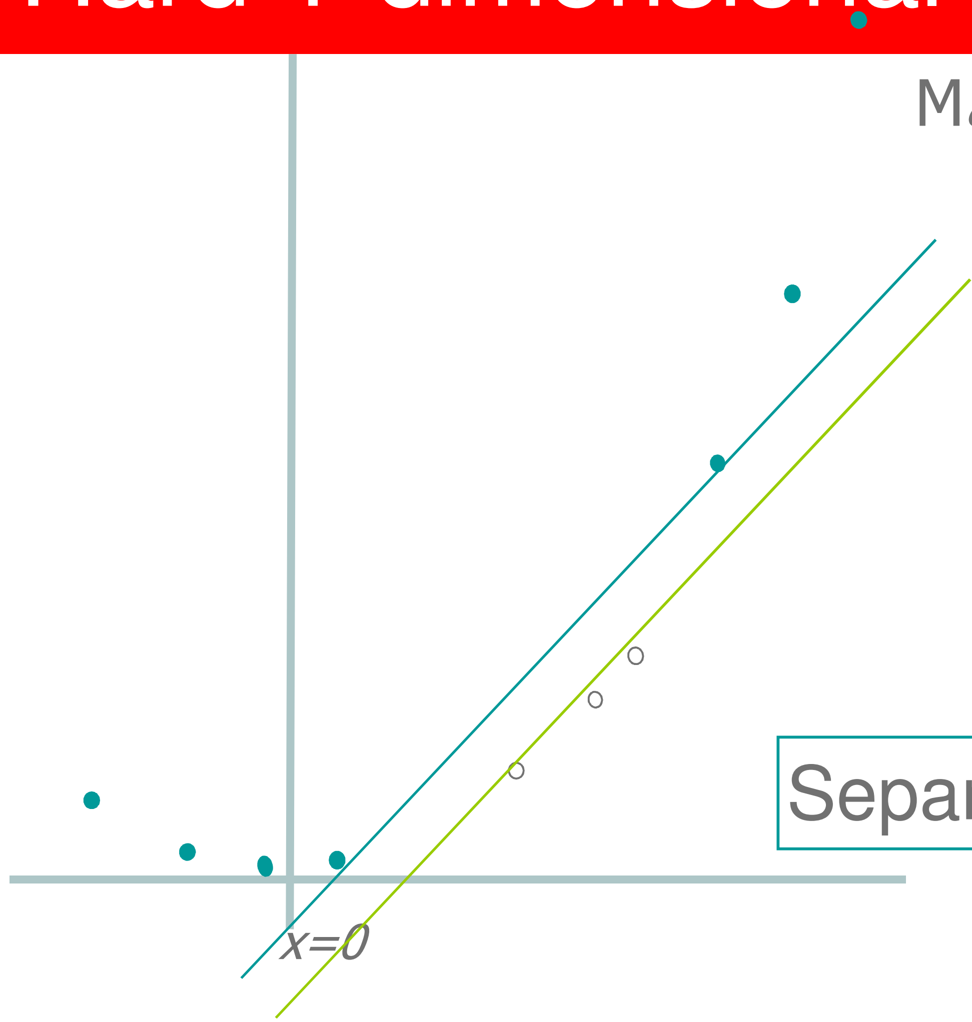
It assumes samples are linearly separable...

What can we do if data is not linearly separable???

Hard 1-dimensional Dataset

If the data set is **not** linearly separable, then adding new features (mapping the data to a larger feature space) the data might become linearly separable

Hard 1-dimensional Dataset



Make up a new feature!

Sort of…
… computed from original feature(s)

$$\mathbf{z}_k = (x_k, x_k^2)$$

Separable! MAGIC!

Now drop this "augmented" data into our linear SVM.

Feature mapping

- ❑ *n* general! points in an *n-1* dimensional space is always linearly separable by a hyperspace!
  $\Rightarrow$ it is good to map the data to high dimensional spaces

- ❑ Having *n* training data, is it always good enough to map the data into a feature space with dimension *n-1*?

  - *Nope... We have to think about the test data as well! Even if we don't know how many test data we have and what they are...*

- ❑ *We might want to map our data to a huge (∞) dimensional feature space*

- ❑ *Overfitting? Generalization error?... We don't care now...*

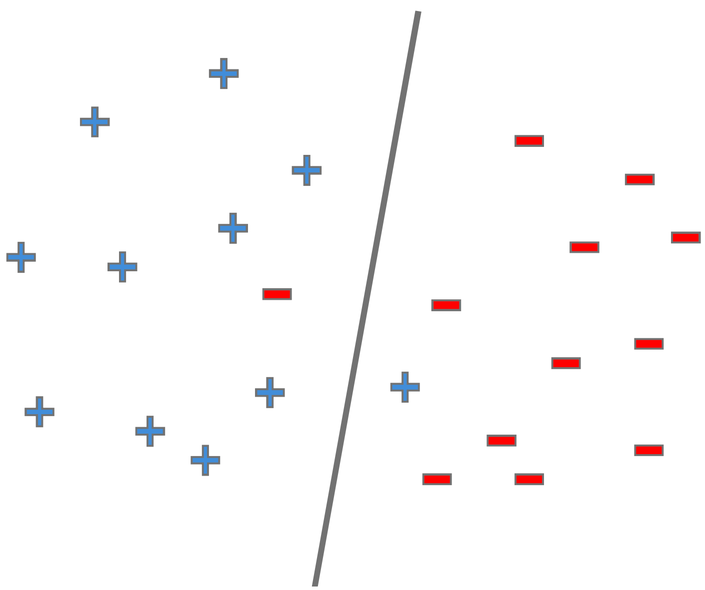
How to do feature mapping?

Let us have n training objects: $\vec{x}_i = [\vec{x}_{i,1}, \vec{x}_{i,2}] \in \mathbb{R}^2$, $i = 1, \ldots, n$

The possible test objects are denoted by $\vec{x} = [\vec{x}_1, \vec{x}_2] \in \mathbb{R}^2$

Use features of features of features of features….

$$\text{Let } \phi(\vec{x}) \doteq \underbrace{[sin(\vec{x}_2), \exp(\vec{x}_2 + \vec{x}_1), \vec{x}_1, \vec{x}_2^{\tan(\vec{x}_1)}, \ldots]}_{\infty}$$



The Problem with Hard SVM

It assumes samples are linearly separable...

Solutions:

1. Use feature transformation to a larger space

   $\Rightarrow$ each training samples are linearly separable in the feature space

   $\Rightarrow$ Hard SVM can be applied ☺

   $\Rightarrow$ overfitting... ☹

2. **Soft margin** SVM instead of Hard SVM

   - We will discuss this now

Hard SVM

The Hard SVM problem can be rewritten:

$$\hat{\mathbf{w}}_{hard} = \arg\min_{\mathbf{w}\in\mathbb{R}^m} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{subject to } y_i\langle\mathbf{x}_i, \mathbf{w}\rangle > 0, \; \forall i = 1, \ldots, n$$

$$\Downarrow$$

$$\boxed{\hat{\mathbf{w}}_{hard} = \arg\min_{\mathbf{w}\in\mathbb{R}^m} \sum_{i=1}^{n} l_{0-\infty}(\langle\mathbf{x}_i, \mathbf{w}\rangle, y_i) + \frac{\lambda}{2}\|\mathbf{w}\|^2}$$

where

$$l_{0-\infty}(a, b) \doteq \begin{cases} \infty : ab < 0 & \text{Misclassification} \\ 0 : ab > 0 & \text{Correct classification} \end{cases}$$

From Hard to Soft constraints

Instead of using hard constraints (points are linearly separable)

$$\hat{\mathbf{w}}_{hard} = \arg\min_{\mathbf{w}\in\mathbb{R}^m} \sum_{i=1}^{n} l_{0-\infty}(\langle \mathbf{x}_i, \mathbf{w}\rangle, y_i) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

We can try to solve the soft version of it:

Your loss is only 1 instead of ∞ if you misclassify an instance

$$\hat{\mathbf{w}}_{soft} = \arg\min_{\mathbf{w}\in\mathbb{R}^m} \sum_{i=1}^{n} l_{0-1}(\langle \mathbf{x}_i, \mathbf{w}\rangle, y_i) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

where

$$l_{0-1}(y, f(\mathbf{x})) = \begin{cases} 1 : y\underbrace{f(\mathbf{x})}_{x^T w} < 0 & \text{Misclassification} \\ 0 : yf(\mathbf{x}) > 0 & \text{Correct classification} \end{cases}$$

Problems with l_{0-1} loss

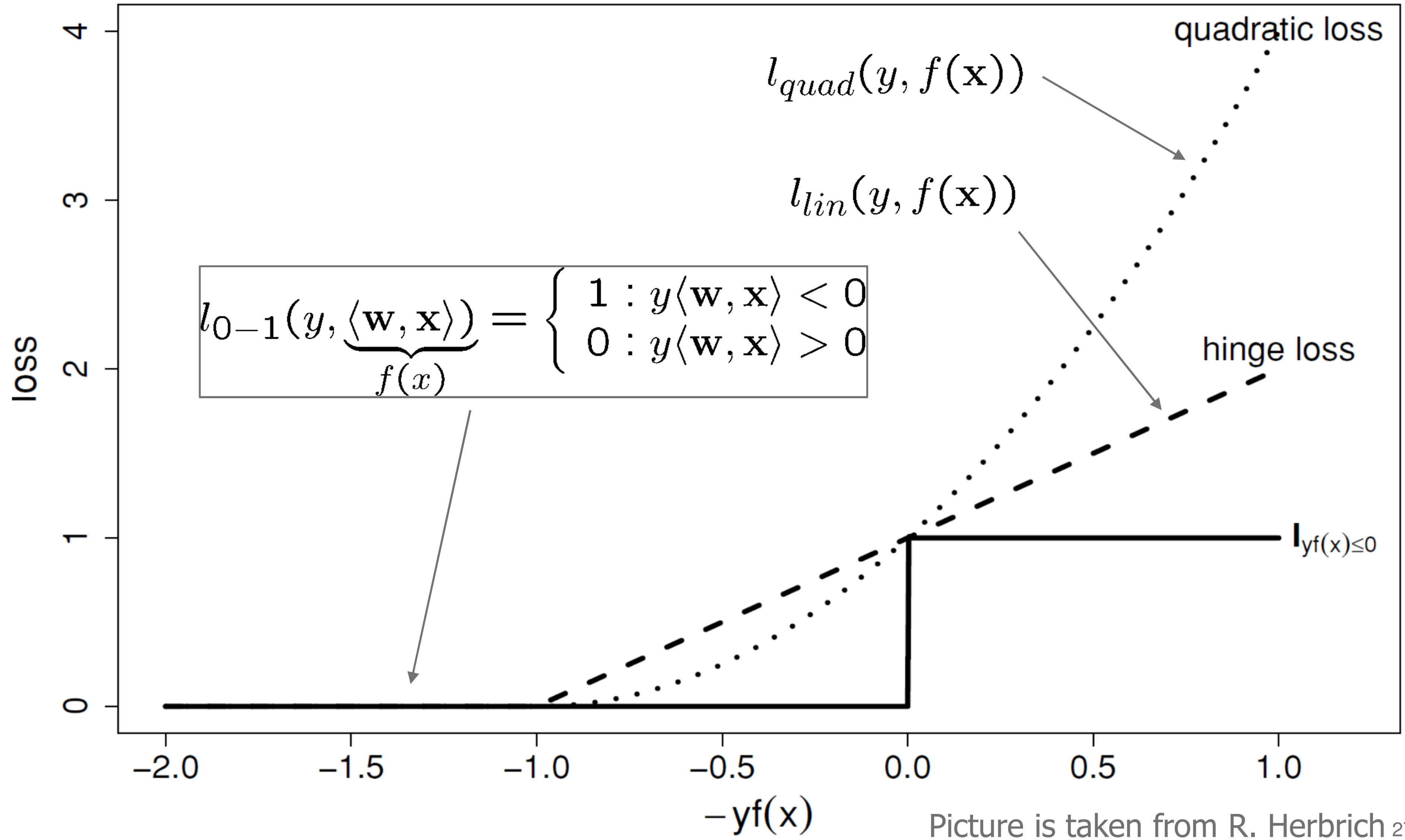
$$\hat{\mathbf{w}}_{soft} = \arg\min_{\mathbf{w}\in\mathbb{R}^m} \sum_{i=1}^{n} l_{0-1}(\langle \mathbf{x}_i, \mathbf{w}\rangle, y_i) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

$$l_{0-1}(y, f(\mathbf{x})) = \begin{cases} 1 : yf(\mathbf{x}) < 0 \\ 0 : yf(\mathbf{x}) > 0 \end{cases}$$

It is not convex in $yf(\mathbf{x})$ $\Rightarrow$ It is not convex in **w**, either...
... and we like only convex functions...

Let us approximate it with convex functions!

Approximation of the Heaviside step function



Picture is taken from R. Herbrich

Approximations of $l_{0\text{-}1}$ loss

- Piecewise linear approximations (hinge loss, l_{lin})

$$l_{lin}(f(\mathbf{x}), y) = \max\{1 - yf(\mathbf{x}), 0)\}$$

[We want $yf(\mathbf{x}) > 1$]

- Quadratic approximation (l_{quad})

$$l_{quad}(f(\mathbf{x}), y) = \max\{1 - yf(\mathbf{x}), 0)\}^2$$

The hinge loss approximation of $l_{0\text{-}1}$

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}\in\mathbb{R}^m} \sum_{i=1}^{n} \underbrace{l_{lin}(\langle \mathbf{x}_i, \mathbf{w}\rangle, y_i)}_{\xi_i \geq 0} + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

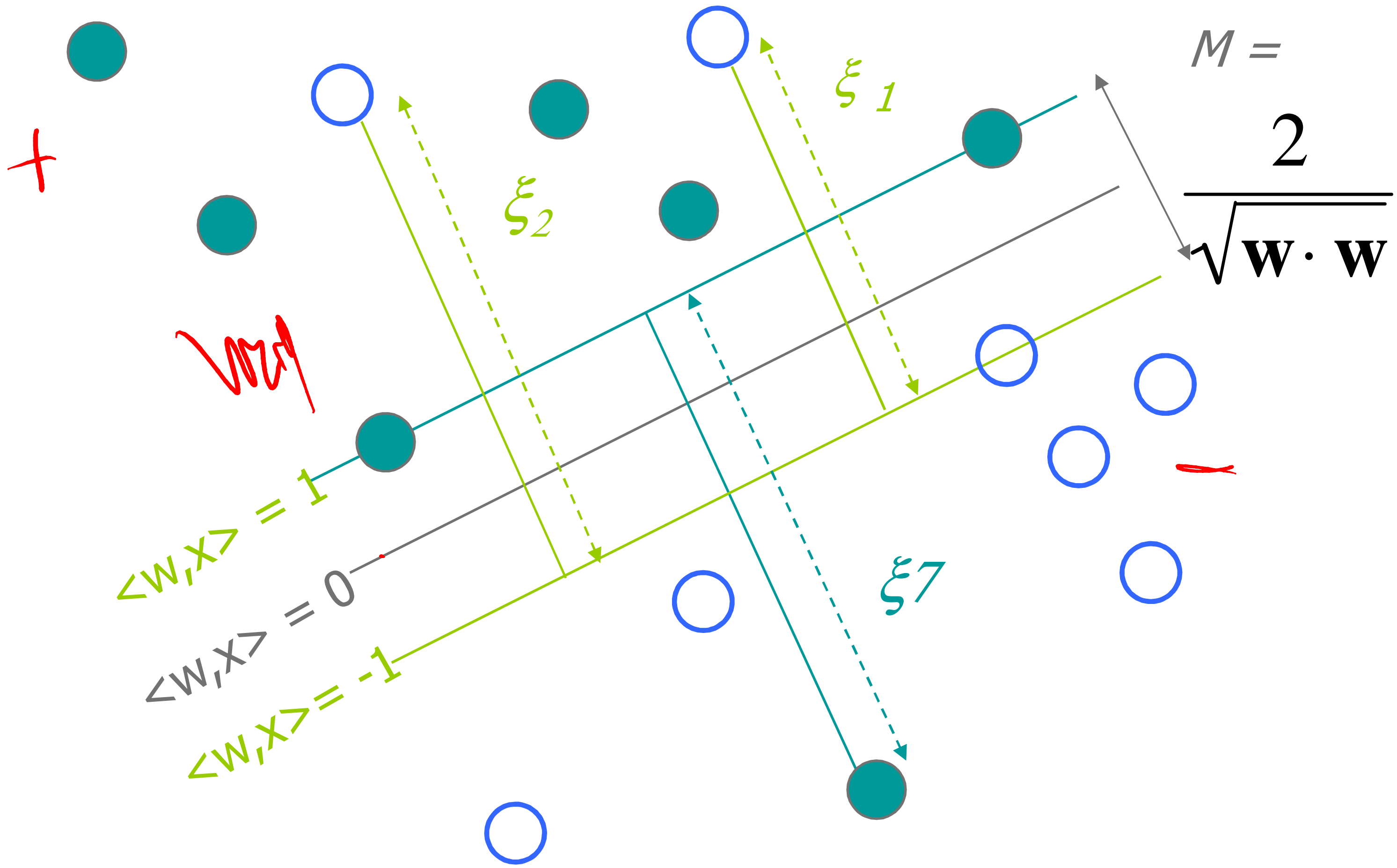
Where,

$$\xi_i \doteq l_{lin}(f(\mathbf{x}_i), y_i) = \max\{1 - y_i f(\mathbf{x}_i), 0)\}$$

$$\geq 1 - y_i \underbrace{\langle \mathbf{w}, \mathbf{x}_i \rangle}_{f(\mathbf{x}_i)} \geq l_{0-1}(y_i, f(\mathbf{x}_i))$$

The hinge loss upper bounds the 0-1 loss

Geometric interpretation: Slack Variables



$$\xi_i \doteq l_{lin}(f(\mathbf{x}_i), y_i) = \max\{1 - y_i f(\mathbf{x}_i), 0)\}$$
$$= \max\{1 - y_i(\mathbf{w}^T\mathbf{x}_i), 0\}$$

The Primal Soft SVM problem

$$\hat{\mathbf{w}}_{soft} = \arg\min_{\mathbf{w}\in\mathbb{R}^m} \sum_{i=1}^{n} \underbrace{l_{lin}(\langle \mathbf{x}_i, \mathbf{w}\rangle, y_i)}_{\xi_i \geq 0} + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

where

$$\xi_i \doteq l_{lin}(f(\mathbf{x}_i), y_i) = \max\{1 - y_i(\mathbf{w}^T\mathbf{x}_i), 0\}$$

Equivalently,

$$\hat{\mathbf{w}}_{soft} = \arg\min_{\mathbf{w}\in\mathbb{R}^m, \boldsymbol{\xi}\in\mathbb{R}^n} \sum_{i=1}^{n} \xi_i + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

$$\text{subject to } y_i\langle \mathbf{x}_i, \mathbf{w}\rangle \geq 1 - \xi_i,\ \forall i = 1, \ldots, n$$

$$\xi_i \geq 0,\ \forall i = 1, \ldots, n$$

ξ_i: Slack variables

The Primal Soft SVM problem

$$\hat{\mathbf{w}}_{soft} = \arg \min_{\mathbf{w}\in\mathbb{R}^m, \boldsymbol{\xi}\in\mathbb{R}^n} \sum_{i=1}^{n} \xi_i + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

$$\text{subject to } y_i\langle \mathbf{x}_i, \mathbf{w}\rangle \geq 1 - \xi_i, \ \forall i = 1, \ldots, n$$

$$\xi_i \geq 0, \ \forall i = 1, \ldots, n$$

Equivalently,

We can use this form, too.:

$$\hat{\mathbf{w}}_{soft} = \arg \min_{\mathbf{w}\in\mathbb{R}^m, \boldsymbol{\xi}\in\mathbb{R}^n} C \underbrace{\sum_{i=1}^{n} \xi_i}_{\boldsymbol{\xi}^T \mathbf{1}_n} + \frac{1}{2}\|\mathbf{w}\|^2$$

where $C = \frac{1}{\lambda}$

What is the dual form of primal soft SVM?

The Dual Soft SVM (using hinge loss)

$$\hat{\mathbf{w}}_{soft} = \arg \min_{\mathbf{w}\in\mathbb{R}^m, \boldsymbol{\xi}\in\mathbb{R}^n} C\sum_{i=1}^{n}\xi_i + \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{subject to } y_i\langle \mathbf{x}_i, \mathbf{w}\rangle \geq 1 - \xi_i, \ \forall i = 1,\dots,n \qquad \alpha_i$$

$$\xi_i \geq 0, \ \forall i = 1,\dots,n \qquad \beta_i$$

- $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)^T \geq 0$ Largrange multipliers
- $\boldsymbol{\beta} = (\beta_1, \dots, \beta_n)^T \geq 0$ Largrange multipliers

$$(\hat{\mathbf{w}}, \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) = \arg \min_{\substack{\mathbf{w}\in\mathbb{R}^m \\ \boldsymbol{\xi}\in\mathbb{R}^n}} \max_{\substack{0\leq\boldsymbol{\alpha} \\ 0\leq\boldsymbol{\beta}}} L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

where

$$L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\Big(y_i\langle \mathbf{x}_i, \mathbf{w}\rangle - 1 + \xi_i\Big) - \sum_{i=1}^{n}\beta_i\xi_i$$

The Dual Soft SVM (using hinge loss)

$$L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\boldsymbol{\xi}^T\mathbf{1}_n - \sum_{i=1}^{n}\alpha_i y_i\langle \mathbf{x}_i,\mathbf{w}\rangle + \boldsymbol{\alpha}^T\mathbf{1}_n - \boldsymbol{\xi}^T(\boldsymbol{\alpha}+\boldsymbol{\beta})$$

MIN MAX
w, ξ α, β ≥ 0

$$0 = \left.\frac{\partial L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta})}{\partial \mathbf{w}}\right|_{\mathbf{w}=\hat{\mathbf{w}}} = \hat{\mathbf{w}} - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i \Rightarrow \boxed{\hat{\mathbf{w}} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i}$$

$$0 = \left.\frac{\partial L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\hat{\boldsymbol{\xi}}} = C\mathbf{1}_n - \boldsymbol{\alpha} - \boldsymbol{\beta} \Rightarrow \boldsymbol{\beta} = C\mathbf{1}_n - \boldsymbol{\alpha} \geq 0$$

$$\boxed{\Rightarrow 0 \leq \boldsymbol{\alpha} \leq C}$$

$$\Rightarrow (\hat{\boldsymbol{\alpha}},\hat{\boldsymbol{\beta}}) = \arg \max_{\substack{0 \leq \boldsymbol{\alpha} \leq C \\ 0 \leq \boldsymbol{\beta}}} L(\hat{\mathbf{w}},\hat{\boldsymbol{\xi}},\boldsymbol{\alpha},\boldsymbol{\beta})$$

$$\boxed{\Rightarrow \hat{\boldsymbol{\alpha}} = \arg \max_{0 \leq \boldsymbol{\alpha} \leq C} \boldsymbol{\alpha}^T\mathbf{1}_m - \tfrac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{YGY}\boldsymbol{\alpha}}$$

$$Y = \begin{pmatrix} y_1 & & 0 \\ & \ddots & \\ 0 & & y_n \end{pmatrix}$$

$$G_{ij} = \langle x_i, x_j \rangle$$

The Dual Soft SVM (using hinge loss)

$\boldsymbol{Y} \doteq diag(y_1, \ldots, y_n) \in \{-1, 1\}^n$ $\quad x_i \Rightarrow \phi(x_i) \quad \langle a, b \rangle = a^T b$

$\boldsymbol{G} \in \mathbb{R}^{n \times n} \doteq \{G_{ij}\}_{i,j}^{n,n}$, where $G_{ij} \doteq \overbrace{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}^{k(\mathbf{x}_i, \mathbf{x}_j)}$, Gram matrix.

$k_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right)$ $\quad \langle \phi(x_i), \phi(x_j) \rangle = k(x_i, x_j)$

$$\hat{\boldsymbol{\alpha}} = \arg\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \boldsymbol{\alpha}^T \mathbf{1}_n - \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{Y} \boldsymbol{G} \boldsymbol{Y} \boldsymbol{\alpha}$$

subject to $0 \leq \alpha_i \leq C$

where $C = \frac{1}{\lambda}$

$\boxed{\text{If } \lambda \to \mathbf{0} \Rightarrow \text{soft-SVM} \to \text{hard-SVM}}$

This is the same as the dual hard-SVM problem, but now we have the additional $0 \leq \alpha_i \leq C$ constraints.

SVM classification in the dual space

Solve the dual problem

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\text{arg max}}\ \boldsymbol{\alpha}^T \mathbf{1}_n - \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{YGY}\boldsymbol{\alpha}$$

$$\text{subject to } 0 \leq \alpha_i \leq C \quad \forall i$$

where $C = \frac{1}{\lambda}$. Let $\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i$.

$k(x_i, x)$

$\langle \varphi(x_i), \varphi(x) \rangle$

On test data $\mathbf{x}$**:** $f_{\hat{\mathbf{w}}}(\mathbf{x}) = \langle \hat{\mathbf{w}}, \mathbf{x} \rangle = \sum_{i=1}^{n} \hat{\alpha}_i y_i \underbrace{\langle \mathbf{x}_i, \mathbf{x} \rangle}_{k(\mathbf{x}_i, \mathbf{x})}$

Why is it called Support Vector Machine?

$\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^T \geq 0$ Lagrange multipliers

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \underbrace{\left(y_i \langle \mathbf{x}_i, \mathbf{w} \rangle - 1\right)}_{0}$$

KKT conditions

COMPLEMENTARY SLACKNESS CONDITION

$\alpha_i > 0 \Rightarrow y_i \langle x_i, w \rangle = 1$

$\langle x_i, w \rangle = \begin{matrix} +1 \\ -1 \end{matrix}$

EITHER $\alpha_i = 0$ OR $\underbrace{(y_i \langle x_i, w \rangle - 1) = 0}$ AND $\alpha_i > 0$

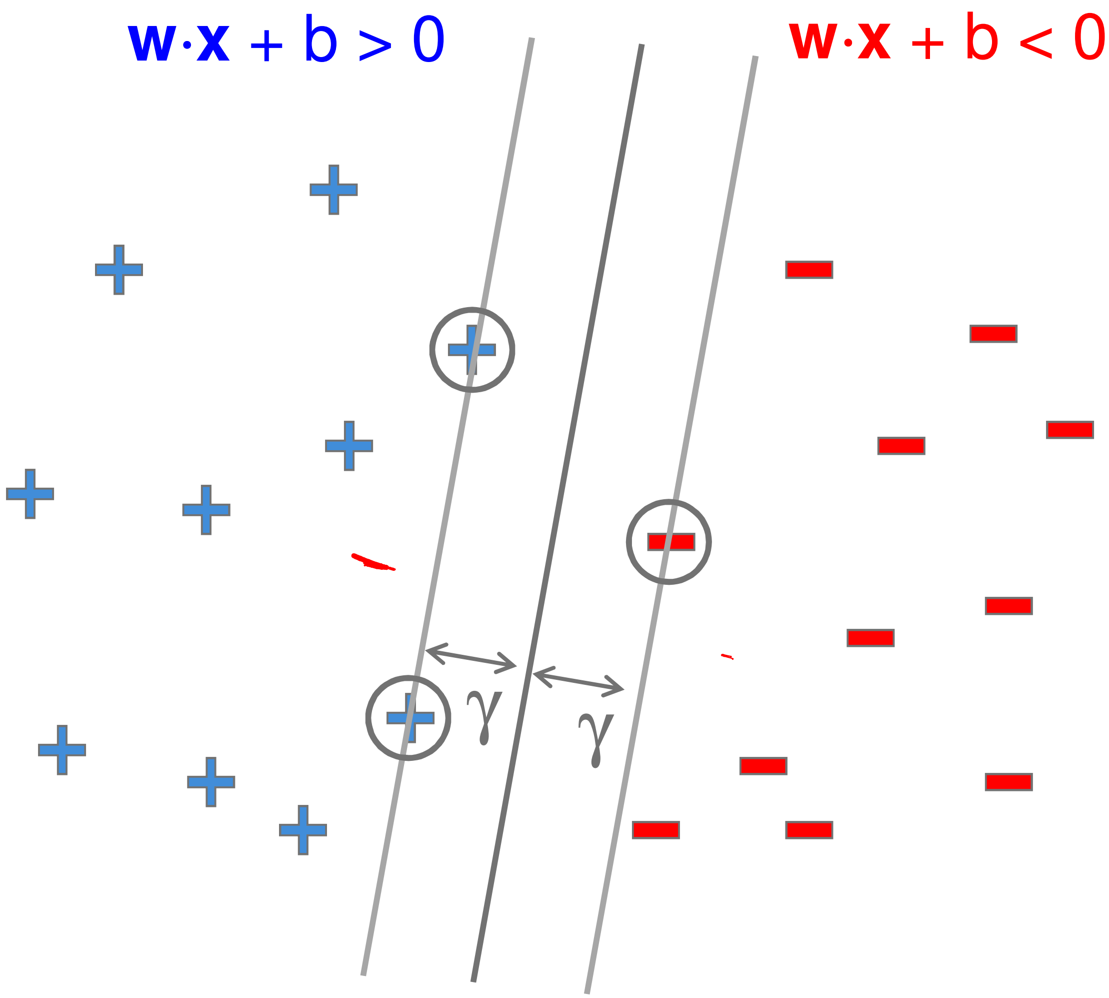
$(y_i \langle x_i, w \rangle - 1) \neq 0$

$\Downarrow$

x_i IS ON THE MARGIN LINES

SUPPORT VECTORS

Why is it called Support Vector Machine?



Hard SVM:

Linear hyperplane defined by "support vectors"

Moving other points a little doesn't effect the decision boundary

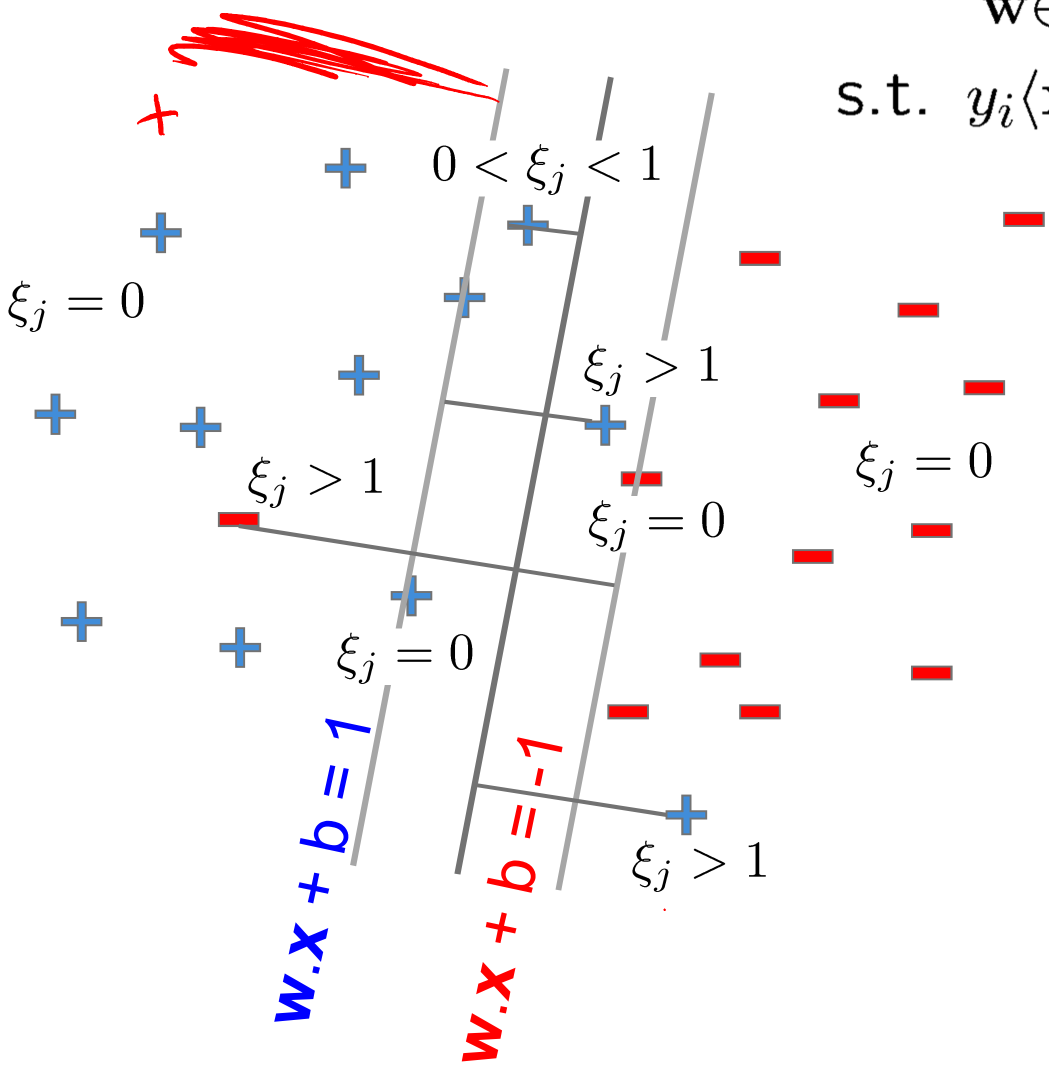
only need to store the support vectors to predict labels of new points

Support vectors in Soft SVM

$$\hat{\mathbf{w}}_{soft} = \arg \min_{\mathbf{w}\in\mathbb{R}^m, \boldsymbol{\xi}\in\mathbb{R}^n} C\sum_{i=1}^{n} \xi_i + \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t. } y_i\langle \mathbf{x}_i, \mathbf{w}\rangle \geq 1 - \xi_i, \ \forall i = 1, \ldots, n$$

$$\xi_i \geq 0, \ \forall i = 1, \ldots, n$$



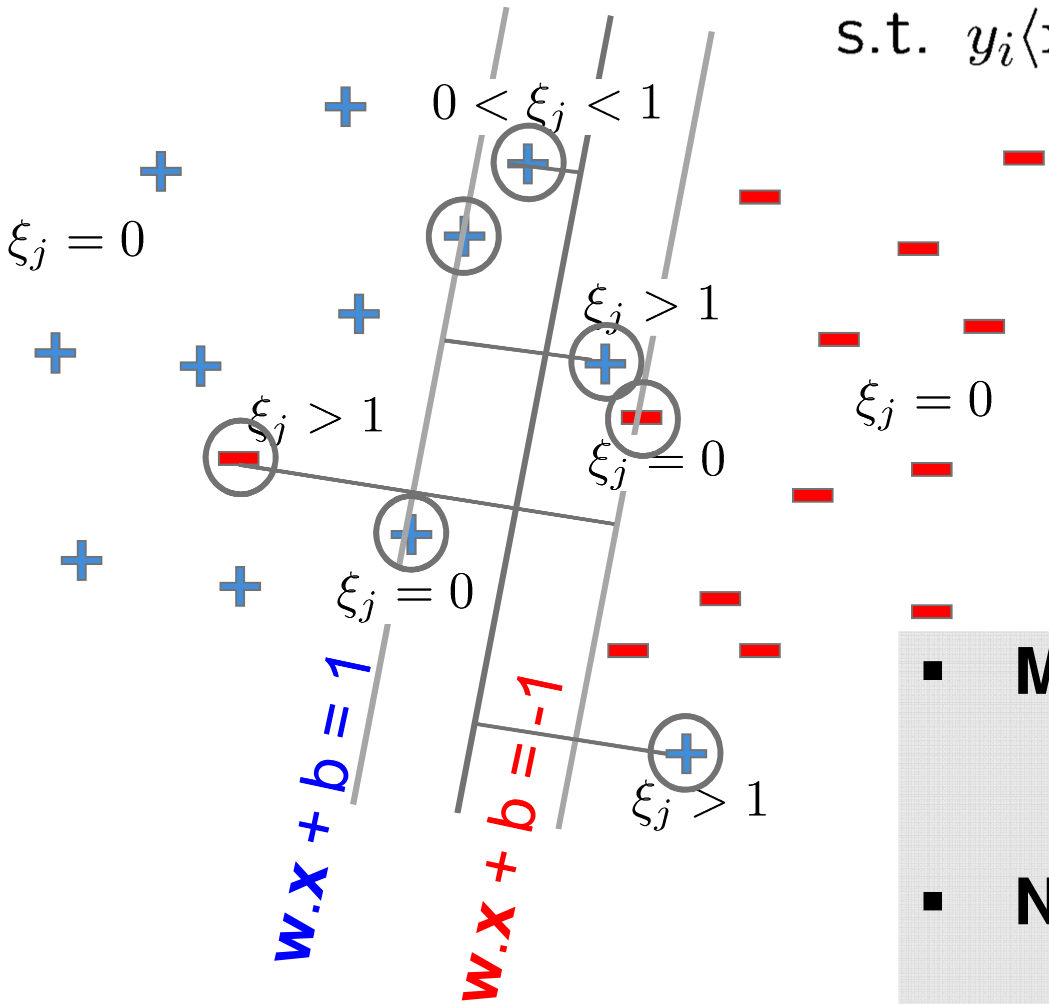
$\xi_j > 1$

Support vectors in Soft SVM

$$\widehat{\mathbf{w}}_{soft} = \arg\min_{\mathbf{w}\in\mathbb{R}^m, \boldsymbol{\xi}\in\mathbb{R}^n} C\sum_{i=1}^{n} \xi_i + \frac{1}{2}\|\mathbf{w}\|^2$$

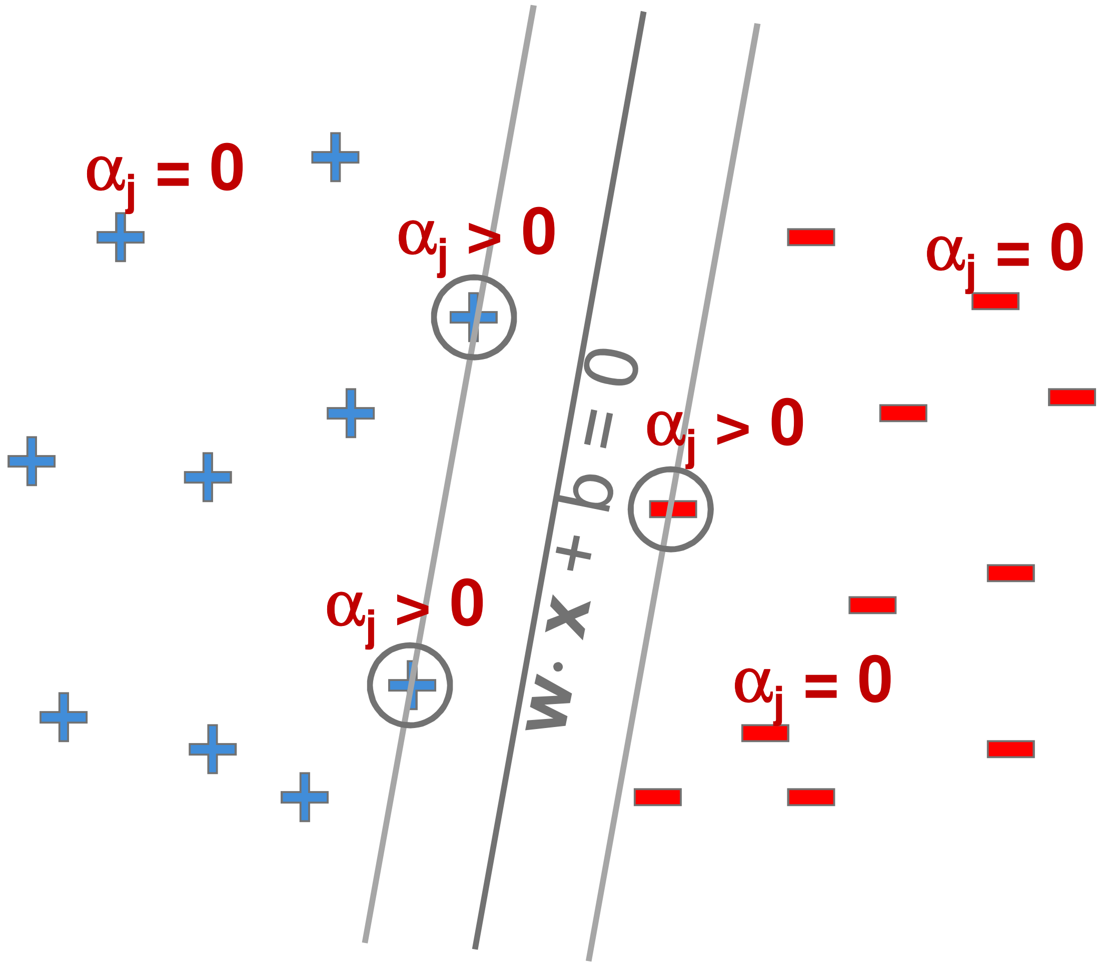
$$\text{s.t. } y_i\langle \mathbf{x}_i, \mathbf{w}\rangle \geq 1 - \xi_i, \ \forall i = 1,\ldots,n$$

$$\xi_i \geq 0, \ \forall i = 1,\ldots,n$$



- **Margin support vectors**
  $y_i\langle \mathbf{x}_i, \mathbf{w}\rangle = 1$
- **Nonmargin support vectors**
  $\xi_i > 0$

Dual SVM Interpretation: Sparsity

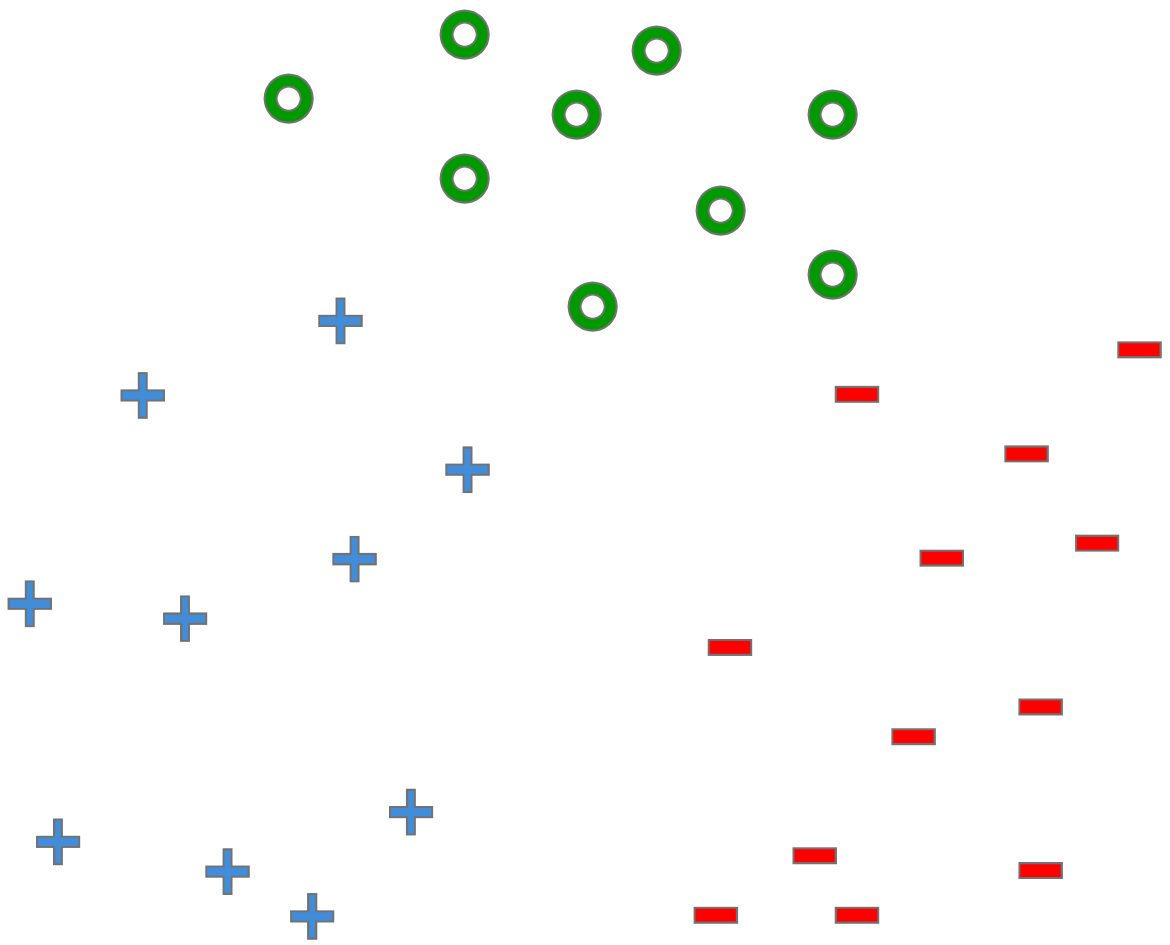


$$\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

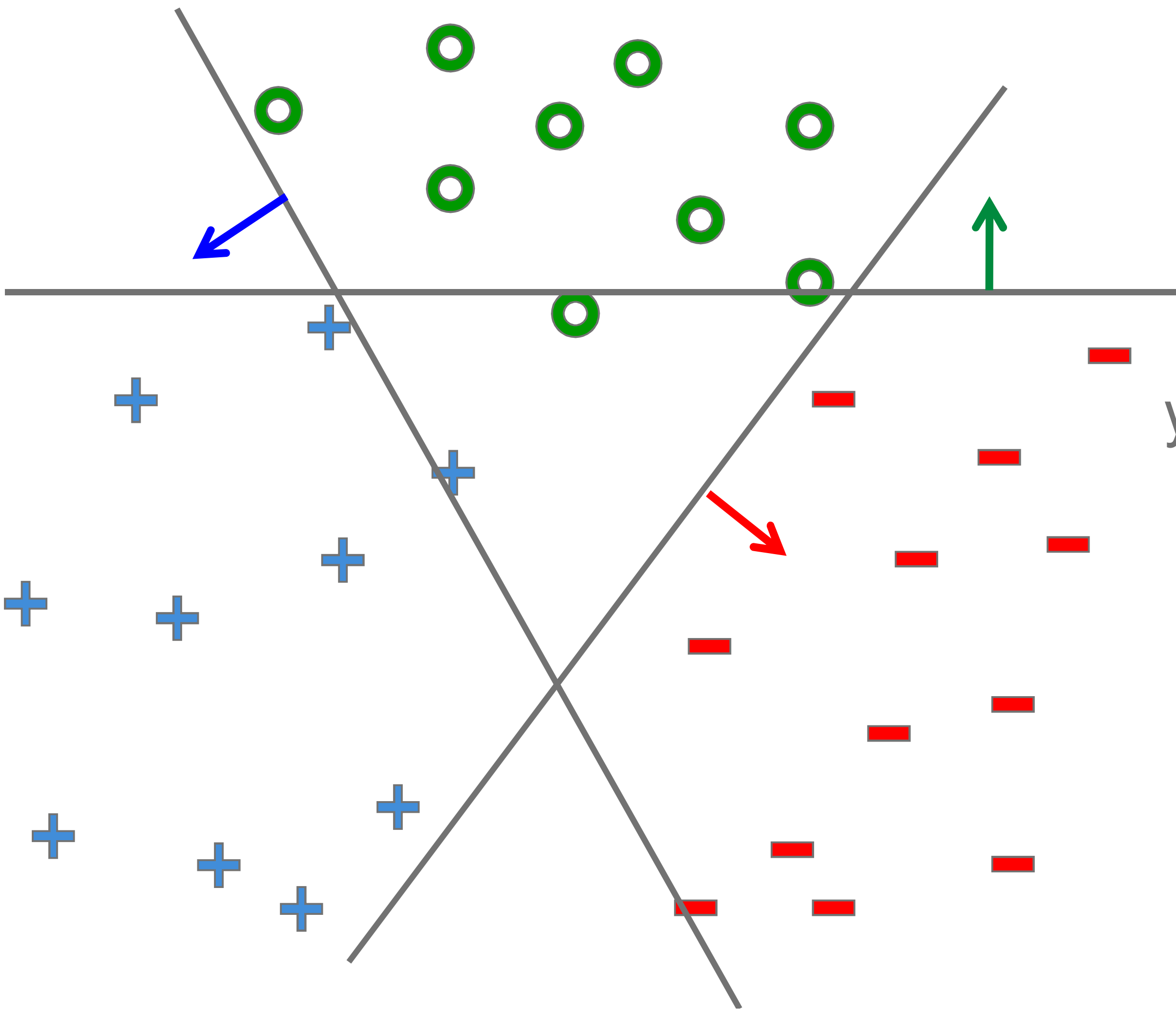
Only few α_js can be non-zero : where constraint is tight

$(<\mathbf{w},\mathbf{x}_j> + b)y_j = 1$

What about multiple classes?



One against all



Learn 3 classifiers separately:
Class k vs. rest
$(\mathbf{w}_k, b_k)_{k=1,2,3}$

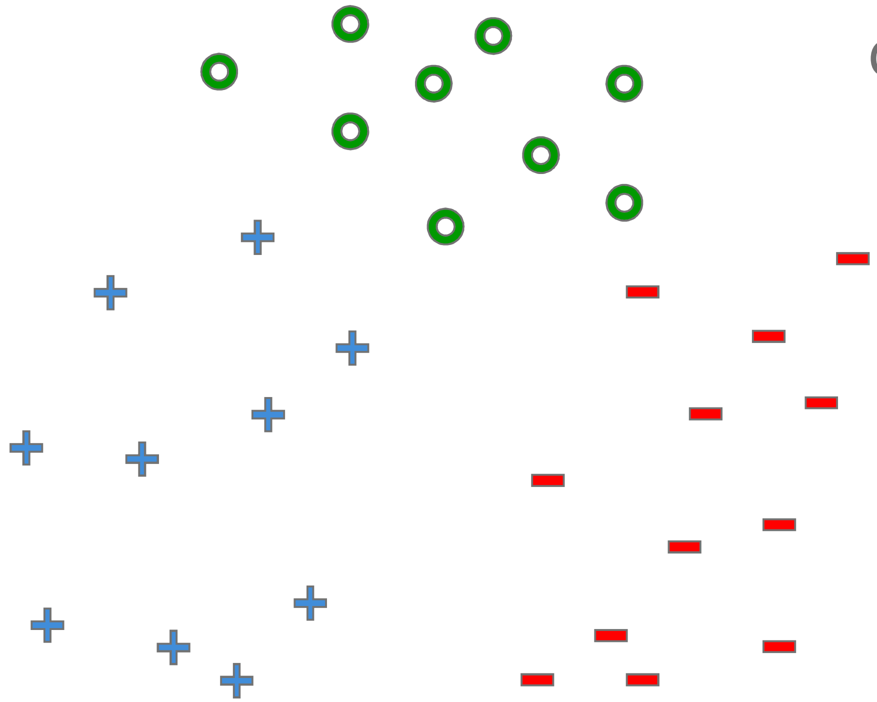
$$y = \arg\max_k \mathbf{w}_k \cdot x + b_k$$

But $\mathbf{w}_k$s may not be based on the same scale.
Note: $(a\mathbf{w}) \cdot x + (ab)$ is also a solution

Learn 1 classifier: Multi-class SVM

Simultaneously learn 3 sets of weights.
Constraints:

$$\mathbf{w}^{(y_j)} \cdot \mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y')} \cdot \mathbf{x}_j + b^{(y')} + 1, \ \forall y' \neq y_j, \ \forall j$$



Margin - gap between correct class and nearest other class

$y = \arg\max_k \mathbf{w}^{(k)} \cdot x + b^{(k)}$

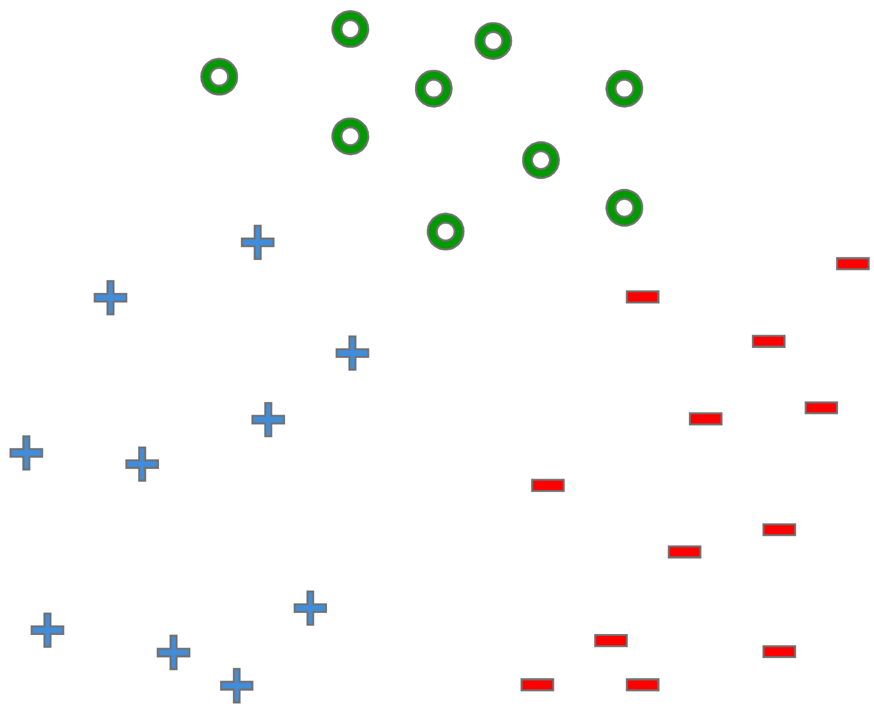
Learn 1 classifier: Multi-class SVM

Simultaneously learn 3 sets of weights

$$\text{minimize}_{\mathbf{w},b} \quad \sum_y \mathbf{w}^{(y)} \cdot \mathbf{w}^{(y)} + C \sum_j \sum_{y \neq y_j} \xi_j^{(y)}$$

$$\mathbf{w}^{(y_j)} \cdot \mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y)} \cdot \mathbf{x}_j + b^{(y)} + 1 - \xi_j^{(y)}, \ \forall y \neq y_j, \ \forall j$$

$$\xi_j^{(y)} \geq 0 \quad , \ \forall y \neq y_j, \ \forall j$$



$y = \arg\max_k \mathbf{w}^{(k)} \cdot x + b^{(k)}$

Joint optimization: $\mathbf{w}_k$s have the same scale.

What you need to know

- Maximizing margin
- Derivation of SVM formulation
- Slack variables and hinge loss
- Relationship between SVMs and logistic regression
  - 0/1 loss
  - Hinge loss
  - Log loss
- Tackling multiple class
  - One against All
  - Multiclass SVMs

SVM vs. Logistic Regression

SVM : **Hinge loss:** $\mathrm{loss}(f(\mathbf{x}_j), y_j) = (1 - (\mathbf{w} \cdot \mathbf{x}_j + b)y_j))_+$

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}\in\mathbb{R}^m} \sum_{i=1}^{n} \underbrace{\mathrm{loss}(\mathbf{x}_i \cdot \mathbf{w} + b, y_i)}_{\xi_i \geq 0} + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Logistic Regression : **Log loss** (log conditional likelihood)

$$\mathrm{loss}(f(x_j), y_j) = -\log P(y_j \mid x_j, \mathbf{w}, b) = \log(1 + e^{-(\mathbf{w}\cdot x_j + b)y_j})$$

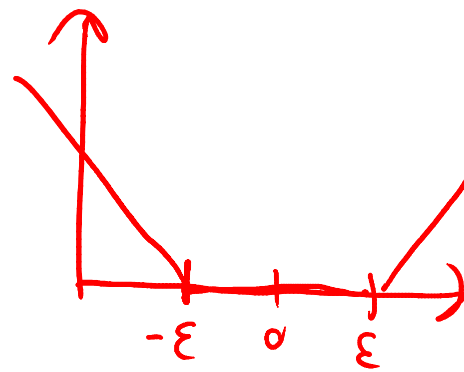
$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}\in\mathbb{R}^m} \sum_{i=1}^{n} \mathrm{loss}(\mathbf{x}_i \cdot \mathbf{w} + b, y_i)$$

SVM for Regression

$$\text{LOSS}(y, w^T x) = (y - w^T x)^2$$

$$0 \text{ IF } |y - w^T x| \leq \varepsilon$$

$$|y - w^T x| \text{ OTHERWISE}$$



SVM classification in the dual space

"Without b"

$$\hat{\boldsymbol{\alpha}} = \text{arg} \max_{\boldsymbol{\alpha} \in \mathbb{R}^m} \boldsymbol{\alpha}^T \mathbf{1}_m - \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{Y} \boldsymbol{G} \boldsymbol{Y} \boldsymbol{\alpha}$$

subject to $0 \leq \alpha_i \leq C$

"With b" $$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \Big(y_i (\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \Big)$$

$$\hat{\boldsymbol{\alpha}} = \text{arg} \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \boldsymbol{\alpha}^T \mathbf{1}_n - \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{Y} \boldsymbol{G} \boldsymbol{Y} \boldsymbol{\alpha}$$

subject to $0 \leq \alpha_i \leq C$

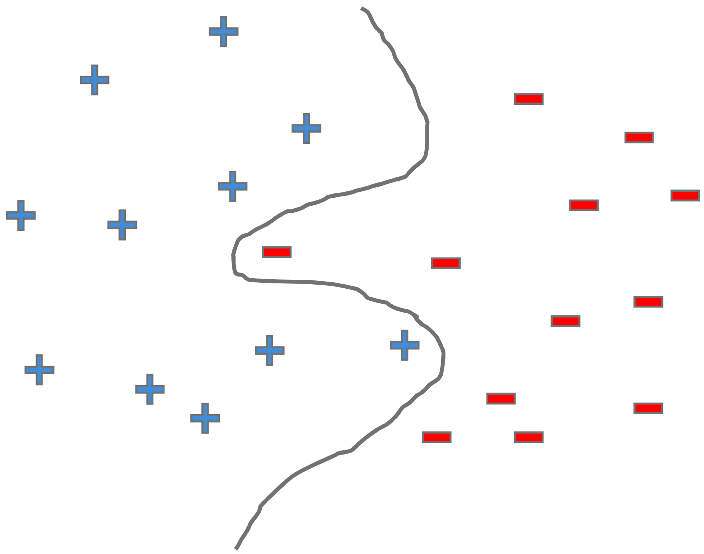
$$\sum_i \alpha_i y_i = 0$$

So why solve the dual SVM?

- There are some quadratic programming algorithms that can solve the dual faster than the primal, specially in high dimensions m>>n

- But, more importantly, the "**kernel trick**"!!!

What if data is not linearly separable?



Use features of features of features of features….

For example polynomials

$$\Phi(\mathbf{x}) = (x_1^3, x_2^3, x_3^3, x_1^2 x_2 x_3, \ldots.,)$$

Dot Product of Polynomials

$\Phi(\mathbf{x}) =$ polynomials of degree exactly d

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

d=1 $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = x_1 z_1 + x_2 z_2 = \mathbf{x} \cdot \mathbf{z}$

d=2

$$\begin{aligned} \Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) = \begin{bmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{bmatrix} \cdot \begin{bmatrix} z_1^2 \\ \sqrt{2} z_1 z_2 \\ z_2^2 \end{bmatrix} &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 \\ &= (x_1 z_1 + x_2 z_2)^2 \\ &= (\mathbf{x} \cdot \mathbf{z})^2 \end{aligned}$$

Dot Product of Polynomials

$d = 3$

$$\begin{pmatrix} x_1^3 \\ \sqrt{3}x_1^2 x_2 \\ \sqrt{3}x_1 x_2^2 \\ x_2^3 \end{pmatrix} \cdot \begin{pmatrix} z_1^3 \\ \sqrt{3}z_1^2 z_2 \\ \sqrt{3}z_1 z_2^2 \\ z_2^3 \end{pmatrix}$$

$$x_1^3 z_1^3 + 3x_1^2 x_2 z_1^2 z_2 + 3x_1 z_1 x_2^2 z_2^2 + x_2^3 z_2^3$$

$$= (x_1 z_1 + z_2 z_2)^3$$

d $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) = K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^d$

Higher Order Polynomials

Feature space becomes really large very quickly!

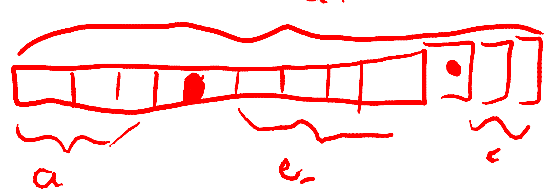
m – input features d – degree of polynomial

$$\text{num. terms} = \binom{d+m-1}{d} = \frac{(d+m-1)!}{d!(m-1)!} \sim m^d$$

grows fast: d = 6, m = 100, about 1.6 billion terms

$x_1^a \; x_2^b \; x_3^c$ $a+b+c=d$

$a+b+c+m-1 = d+m-1$



$\binom{d+m-1}{m-1}$

Dual formulation only depends on dot-products, not on w!

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \underbrace{\mathbf{x}_i \cdot \mathbf{x}_j}$$

$$C \geq \alpha_i \geq 0$$

$$\Downarrow$$

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \underbrace{K(\mathbf{x}_i, \mathbf{x}_j)}$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$
$$C \geq \alpha_i \geq 0$$

$\Phi(\mathbf{x})$ – High-dimensional feature space, but never need it explicitly as long as we can compute the dot product fast using some Kernel K

Finally: The Kernel Trick!

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

$$\boxed{\begin{array}{c} \mathbf{w} = \sum_i \alpha_i y_i \Phi(\mathbf{x}_i) \\[1em] b = y_k - \mathbf{w} \cdot \Phi(\mathbf{x}_k) \\ \text{for any } k \text{ where } C > \alpha_k > 0 \end{array}}$$

- Never represent features explicitly
  - Compute dot products in closed form
- Constant-time high-dimensional dot-products for many classes of features

Common Kernels

- Polynomials of degree d $K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v})^d$
- Polynomials of degree up to d $K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v}+1)^d$
- Gaussian/Radial kernels (polynomials of all orders – recall series expansion)

$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||^2}{2\sigma^2}\right)$$

- Sigmoid

$$K(\mathbf{u},\mathbf{v}) = \tanh(\eta\mathbf{u}\cdot\mathbf{v}+\nu)$$

Which functions can be used as kernels???
…and why are they called kernels???

Overfitting

- Huge feature space with kernels, what about overfitting???
  - Maximizing margin leads to sparse set of support vectors
  - Some interesting theory says that SVMs search for simple hypothesis with large margin
  - Often robust to overfitting

What about classification time?

$$\mathbf{w} = \sum_i \alpha_i y_i \Phi(\mathbf{x}_i)$$

$$b = y_k - \mathbf{w} \cdot \Phi(\mathbf{x}_k)$$

for any k where $C > \alpha_k > 0$

- For a new input **x**, if we need to represent Φ(**x**), we are in trouble!
- Recall classifier: sign(**w**·Φ(**x**)+b)
- Using kernels we are cool!

$$K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v})$$

Kernels in Logistic Regression

$$P(Y=1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1+e^{-(\mathbf{w}\cdot\Phi(\mathbf{x})+b)}}$$

- Define weights in terms of features:

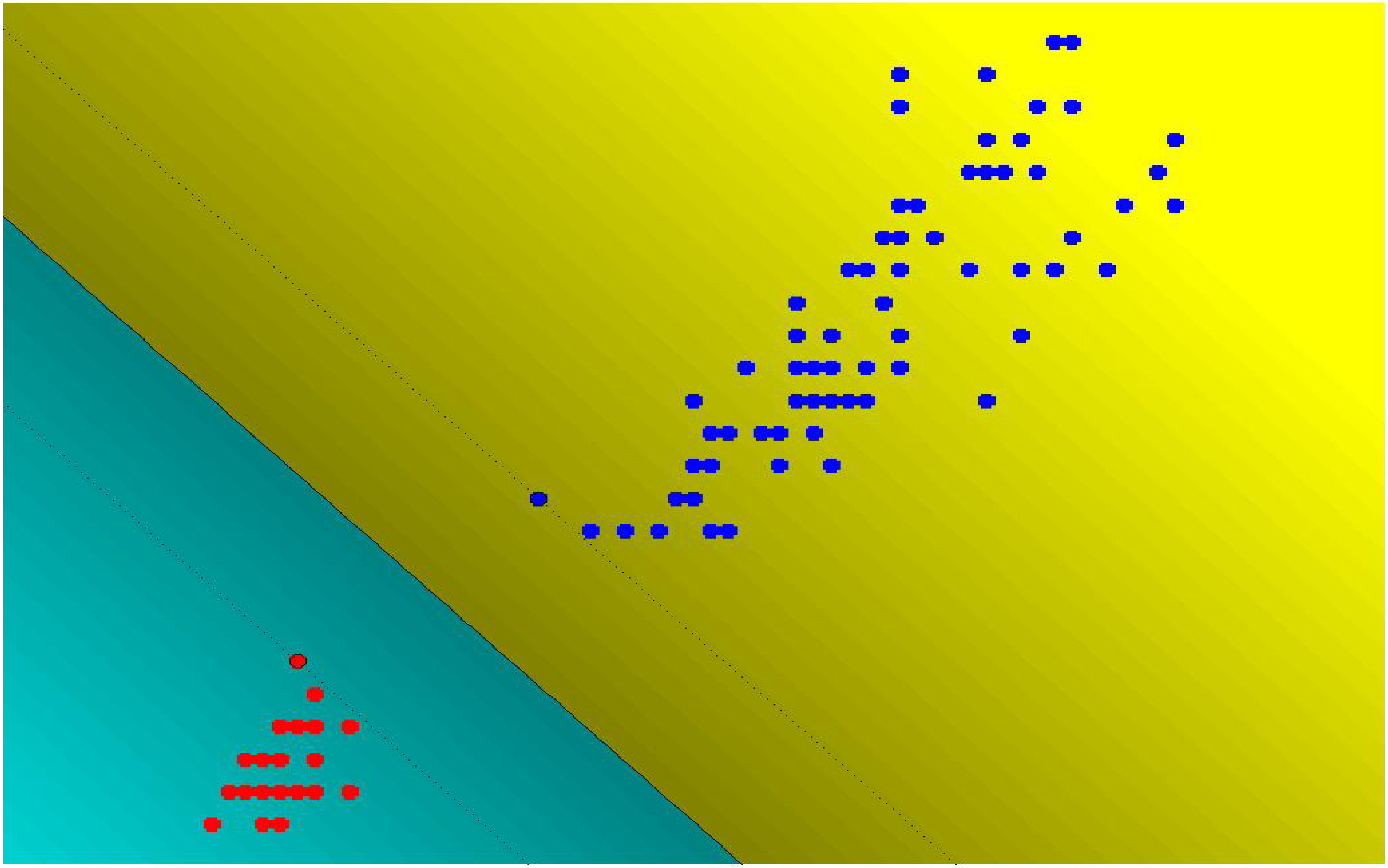
$$\mathbf{w} = \sum_i \alpha_i \Phi(\mathbf{x}_i)$$

$$\begin{aligned} P(Y=1 \mid \mathbf{x}, \mathbf{w}) &= \frac{1}{1+e^{-(\sum_i \alpha_i \Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x})+b)}} \\ &= \frac{1}{1+e^{-(\sum_i \alpha_i K(\mathbf{x},\mathbf{x}_i)+b)}} \end{aligned}$$

- Derive simple gradient descent rule on α_i

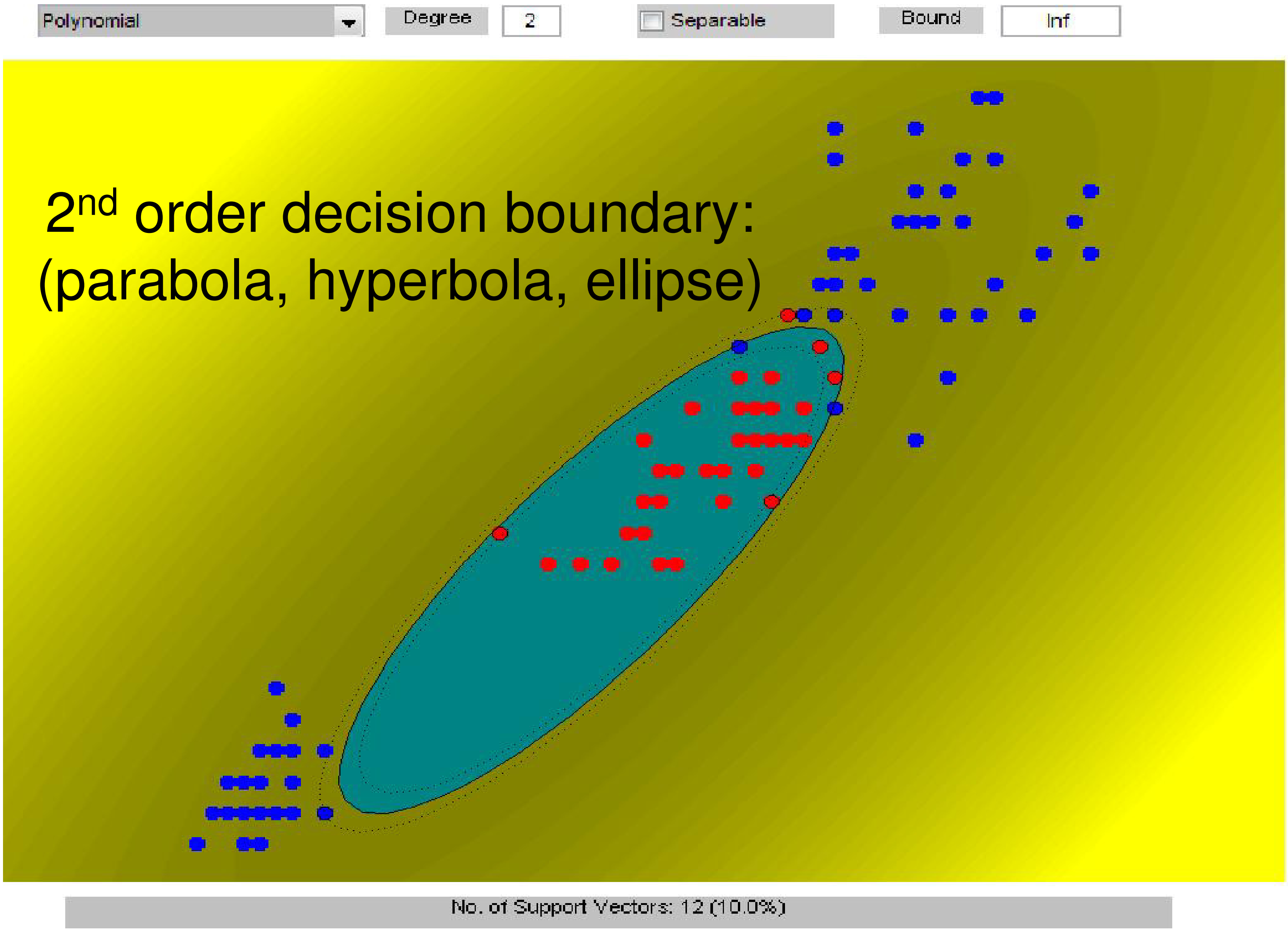
A few results

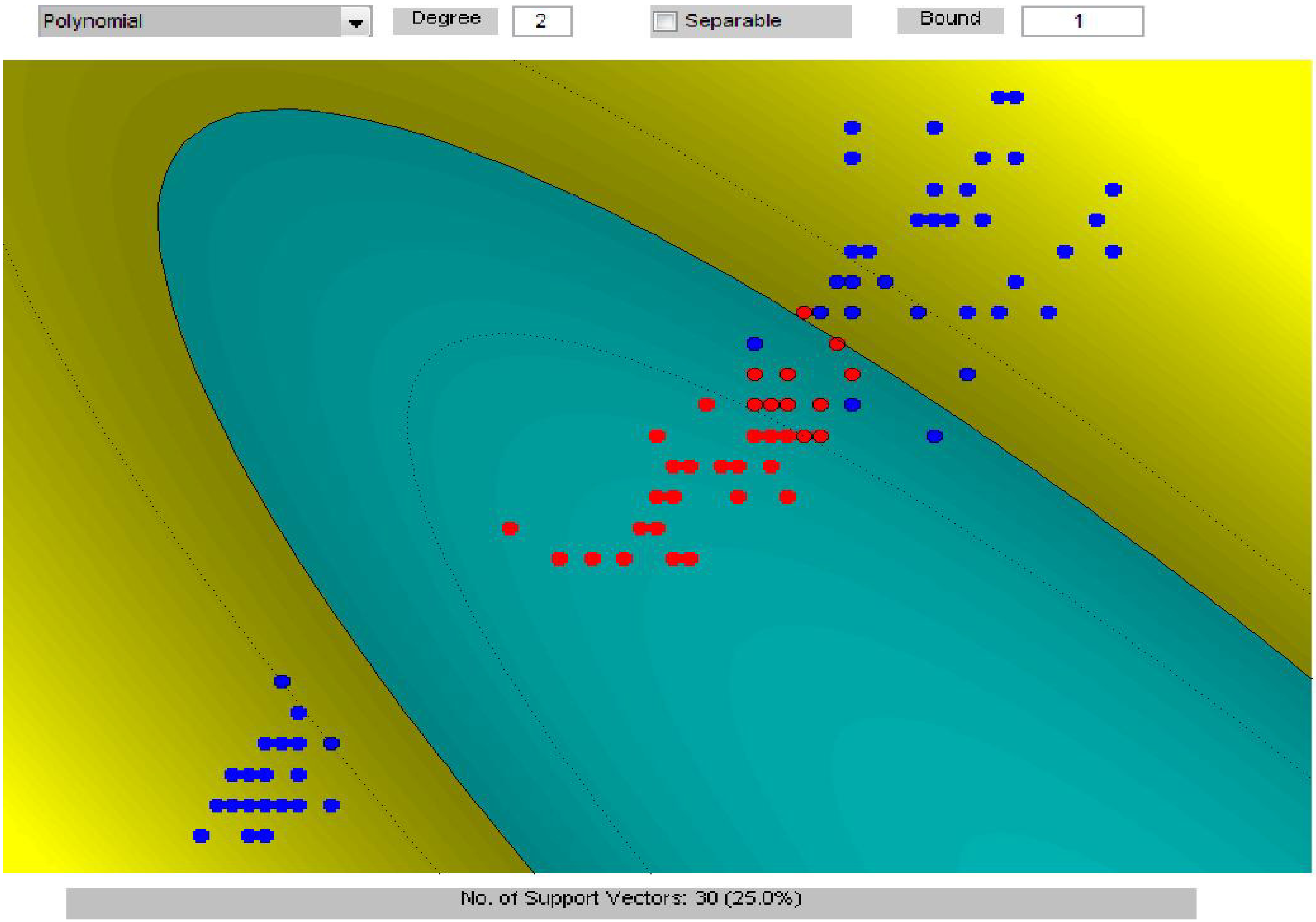
Steve Gunn's svm toolbox Results, Iris 2vs13, Linear kernel



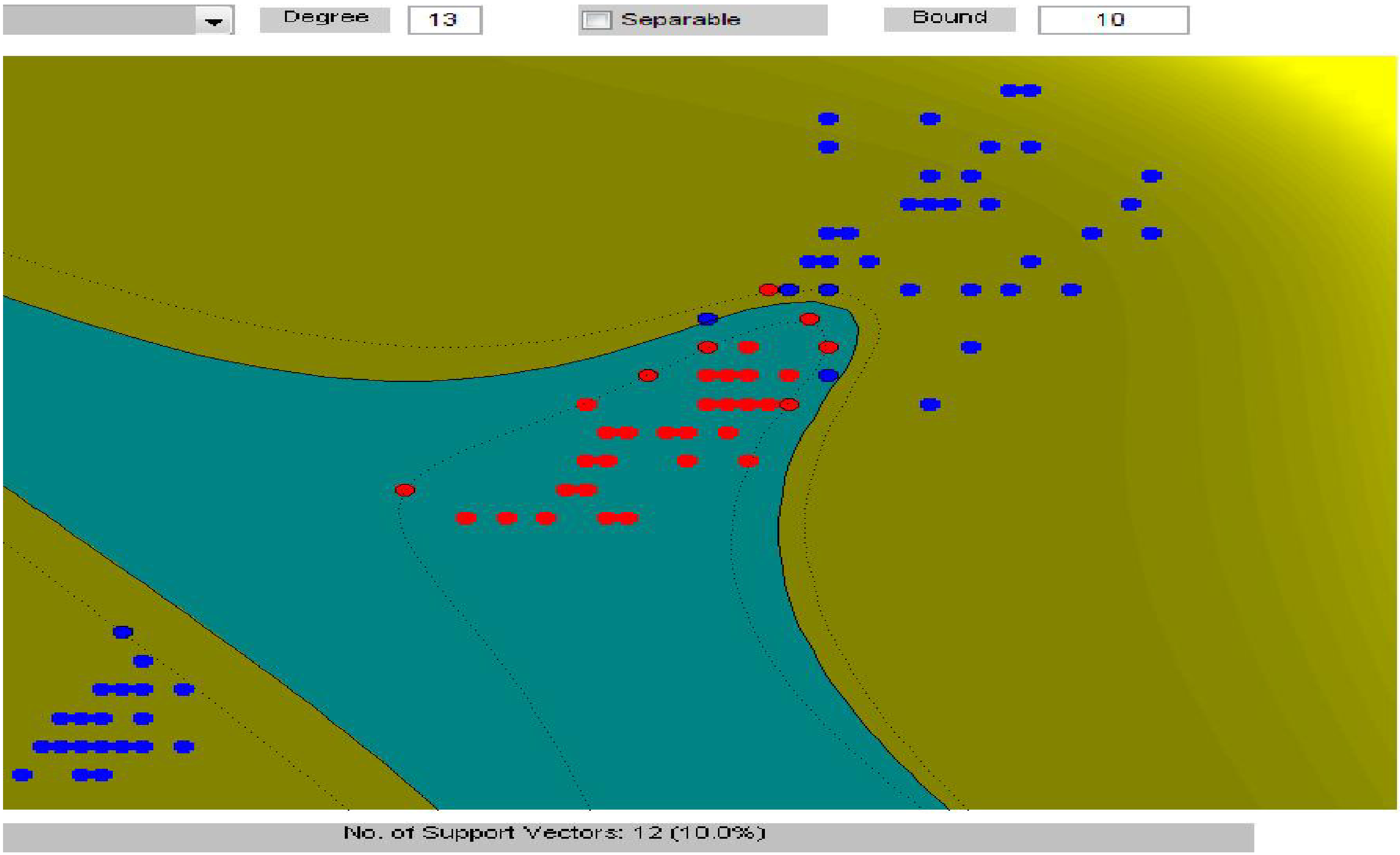
No. of Support Vectors: 2 (1.7%)

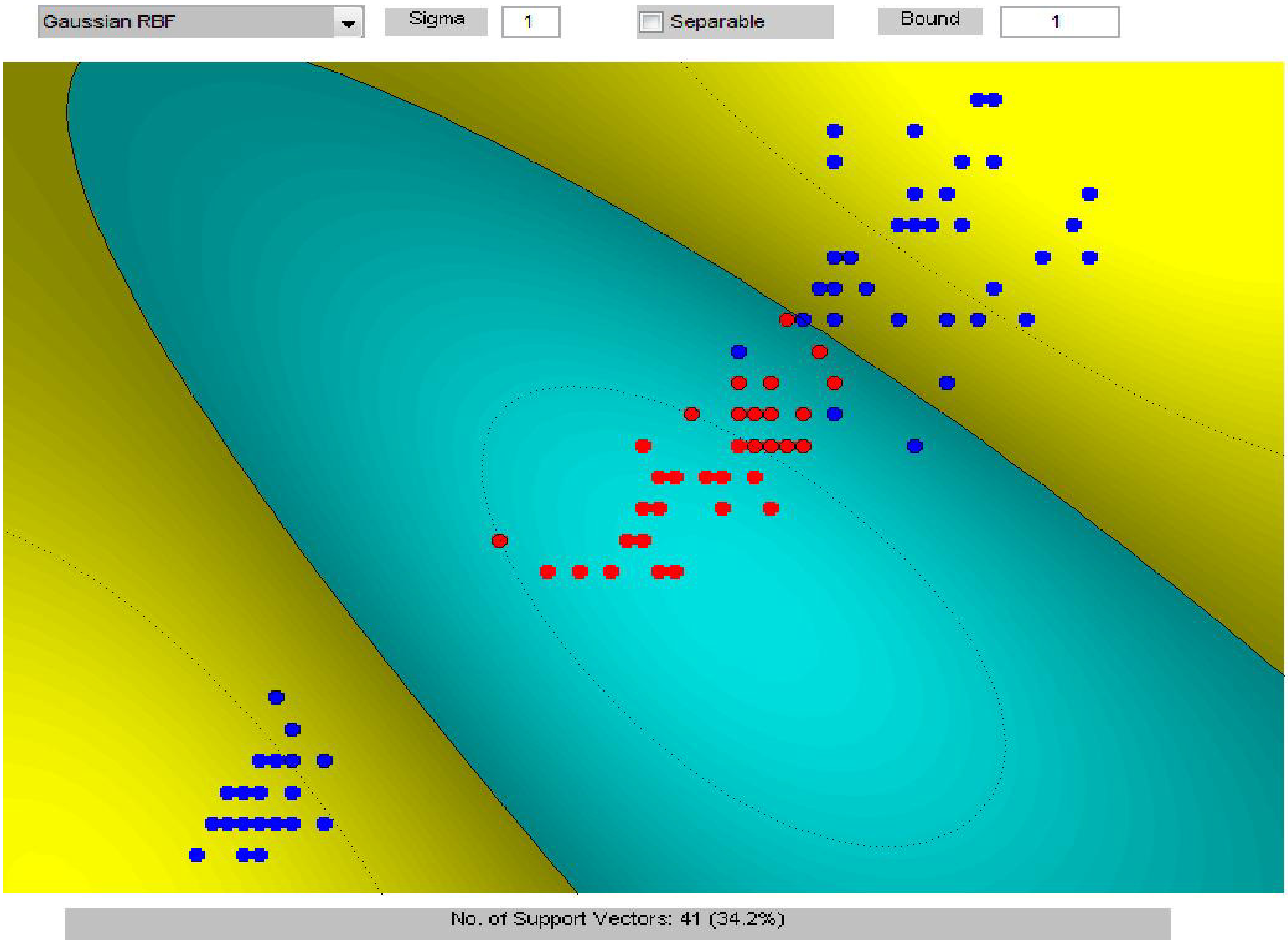
Results, Iris 1vs23, 2^{nd} order kernel




Polynomial
Degree
2
Separable
Bound
1
No. of Support Vectors: 30 (25.0%)

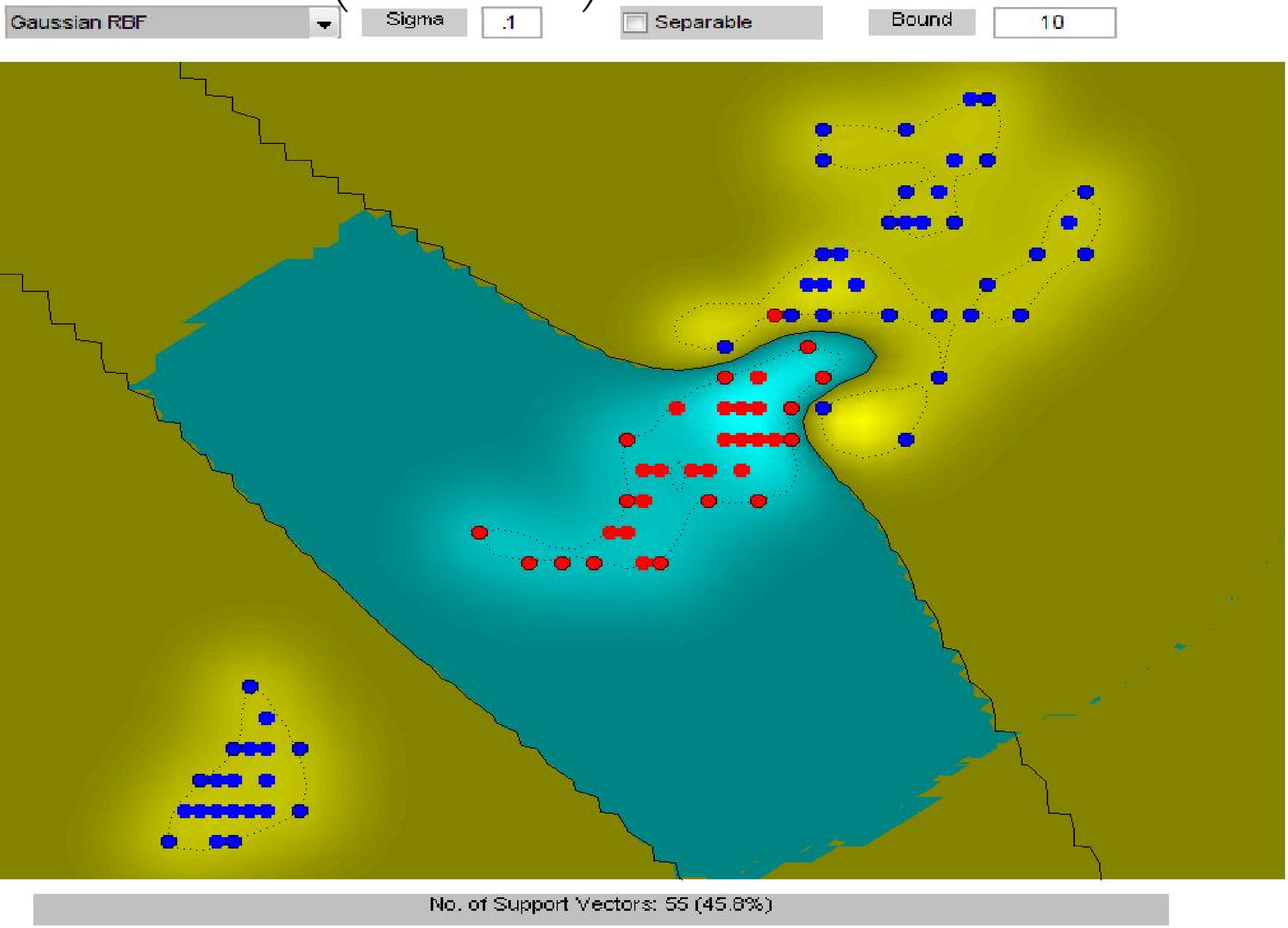
Results, Iris 1vs23, 13th order kernel



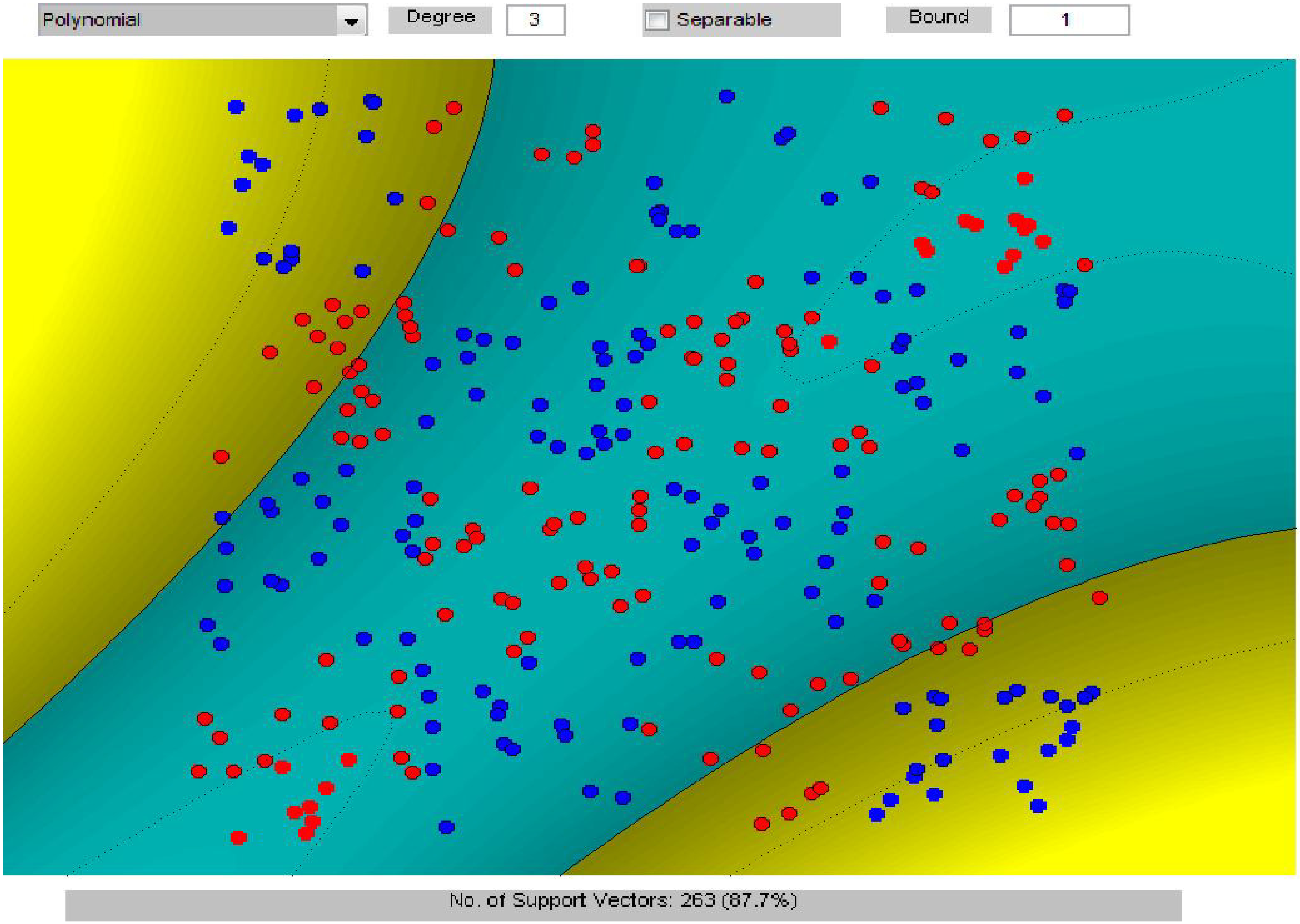

Gaussian RBF
Sigma
1
Separable
Bound
1
No. of Support Vectors: 41 (34.2%)

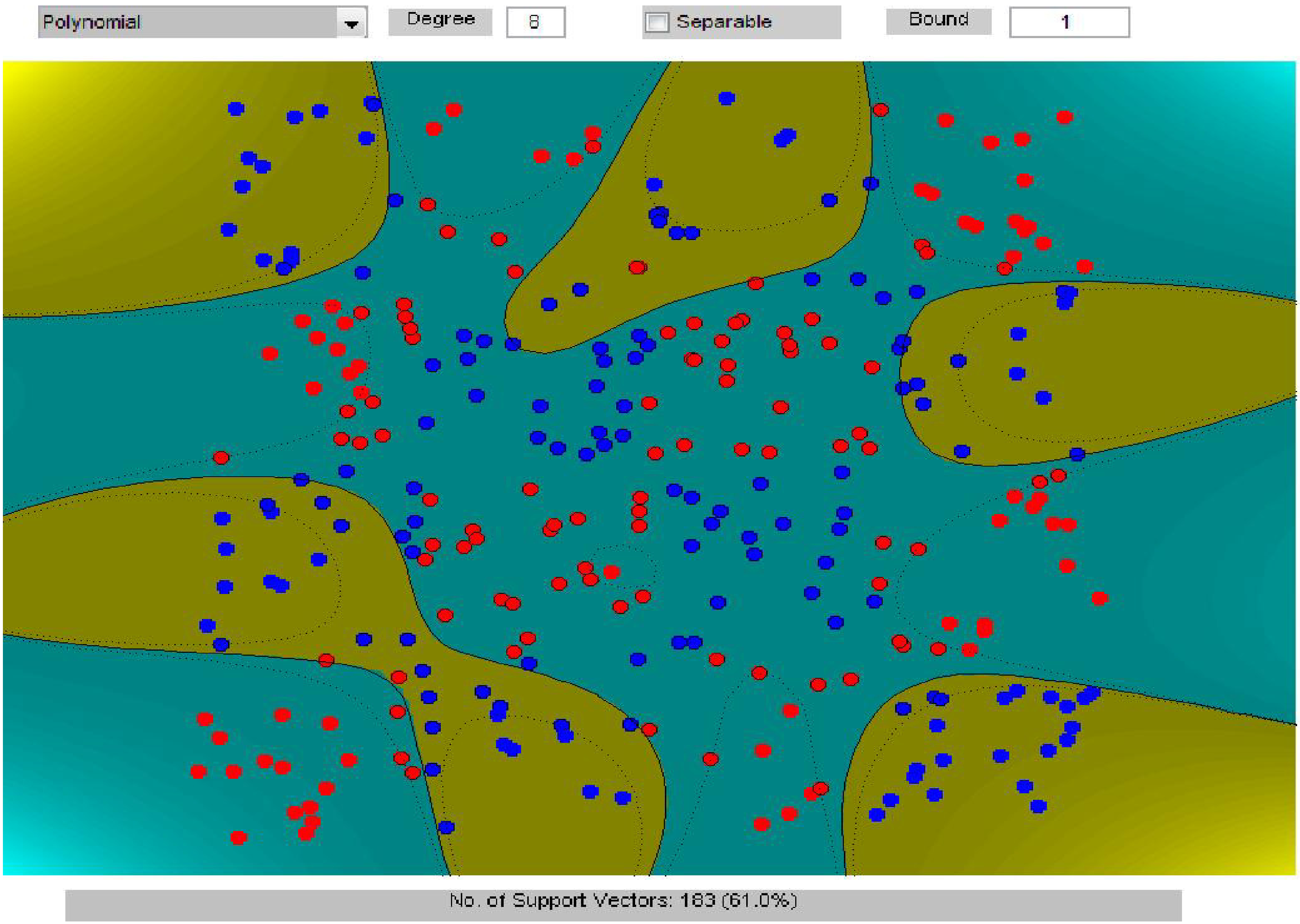
$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||^2}{2\sigma^2}\right)$$

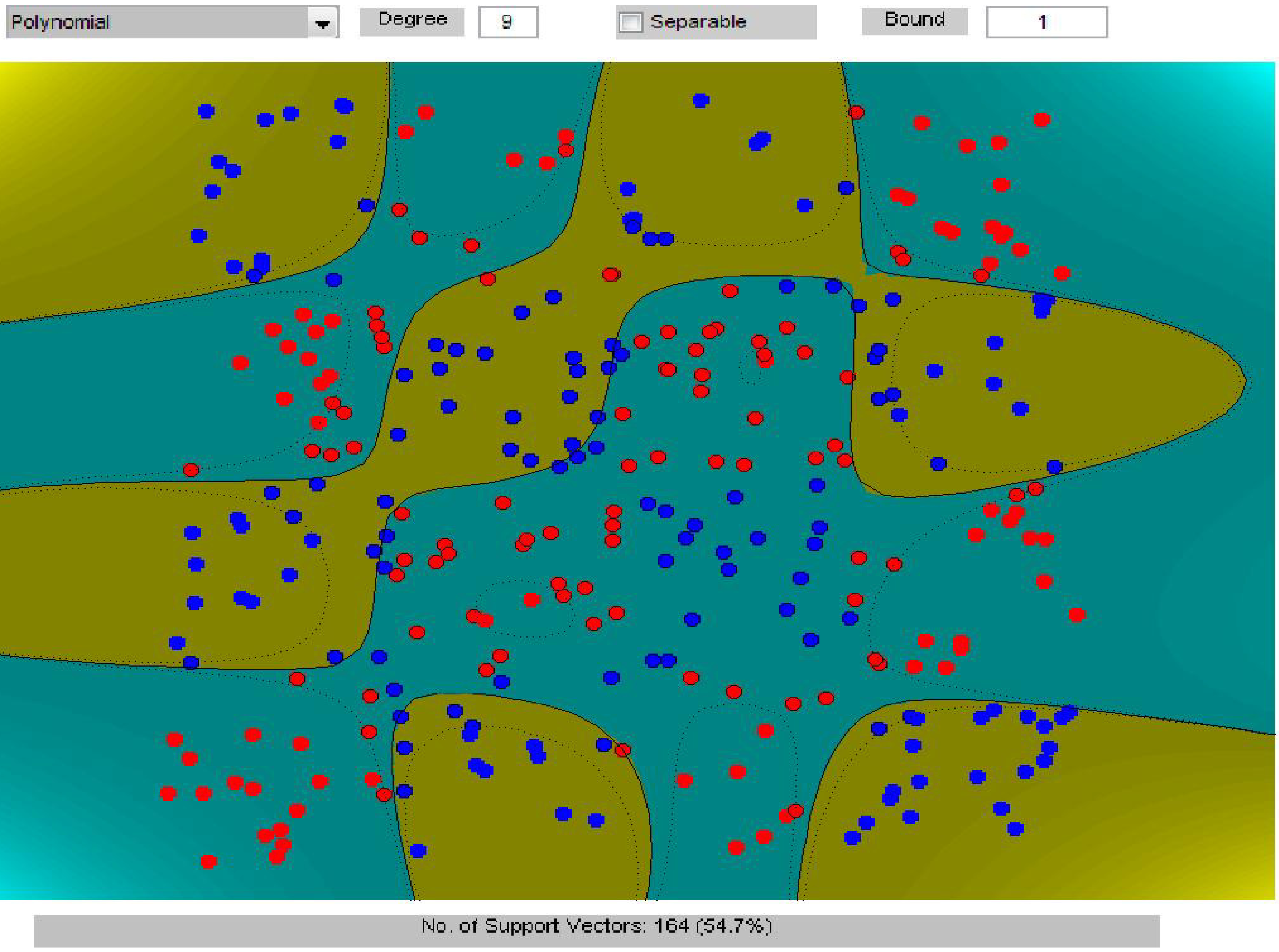
$\sigma \to 0 \Rightarrow$ MORE SUPPORT VECTORS

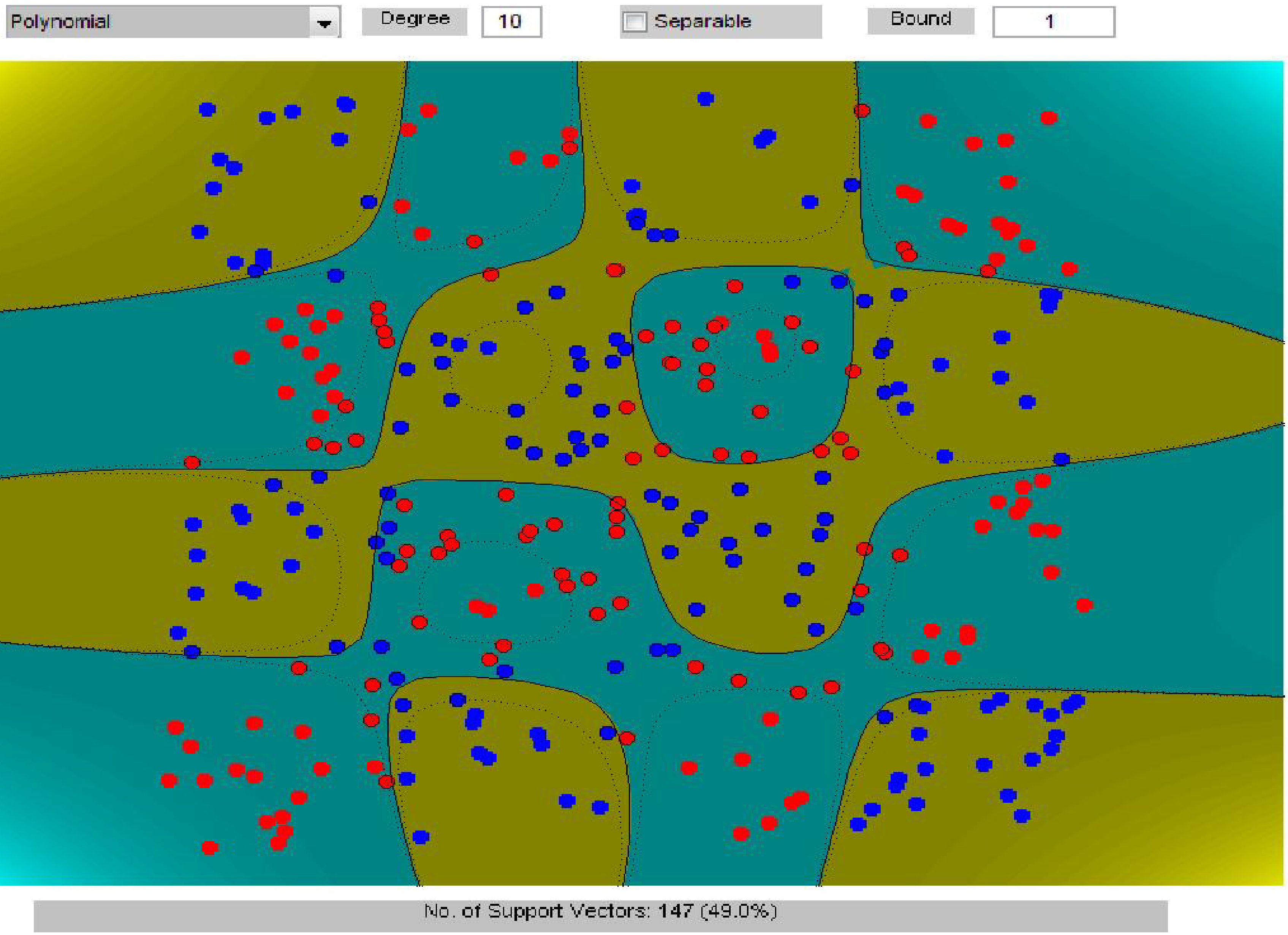


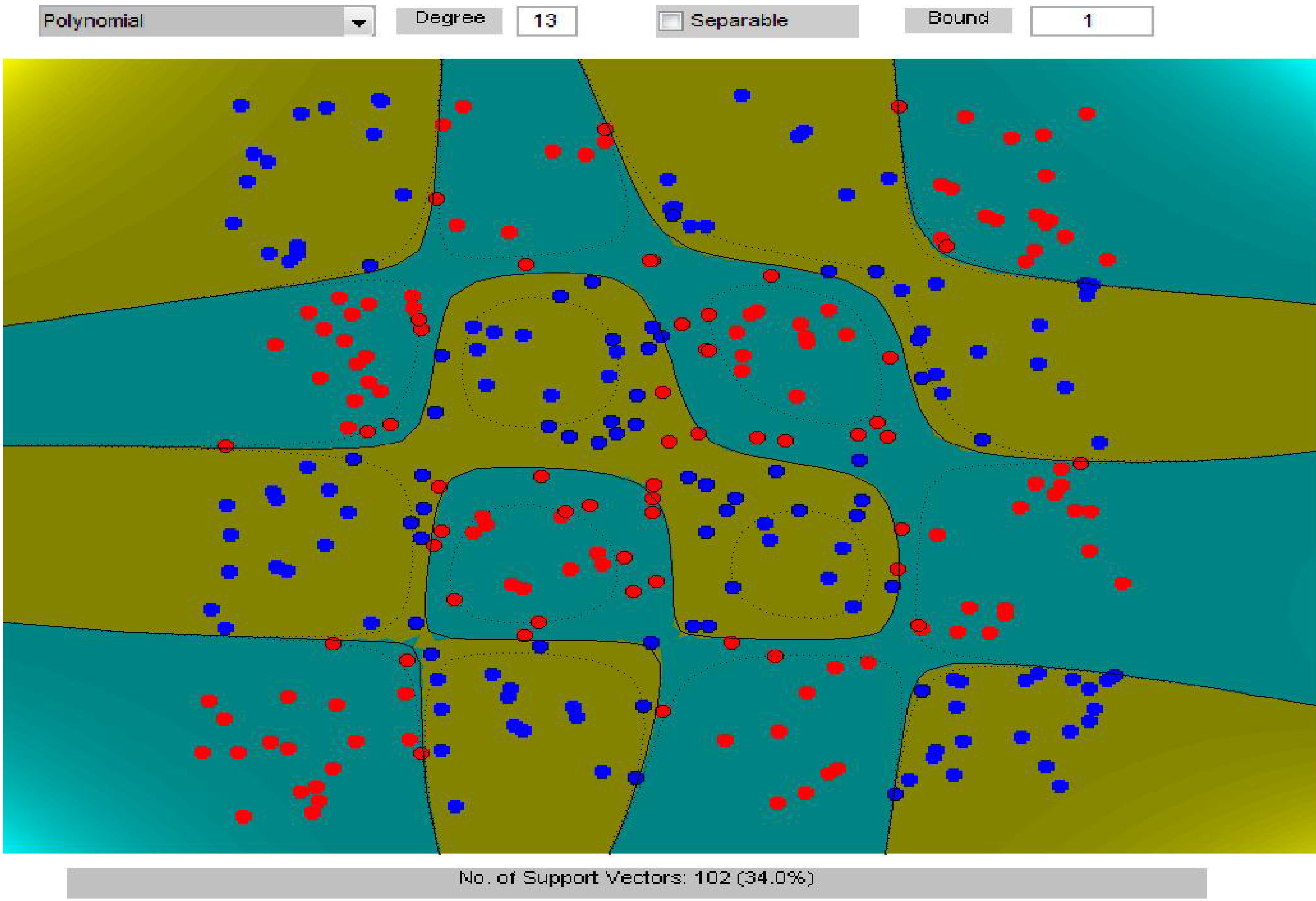
Chessboard dataset




Polynomial
Degree
8
Separable
Bound
1
No. of Support Vectors: 183 (61.0%)


Polynomial
Degree
9
Separable
Bound
1
No. of Support Vectors: 164 (54.7%)


Polynomial
Degree
10
Separable
Bound
1
No. of Support Vectors: 147 (49.0%)


Polynomial
Degree
13
Separable
Bound
1
No. of Support Vectors: 102 (34.0%)

Results, Chessboard, RBF kernel