7. Kernel SVM

EEC E454 Introduction to Machine Learning Systems

2023 Fall, Jaeho Lee

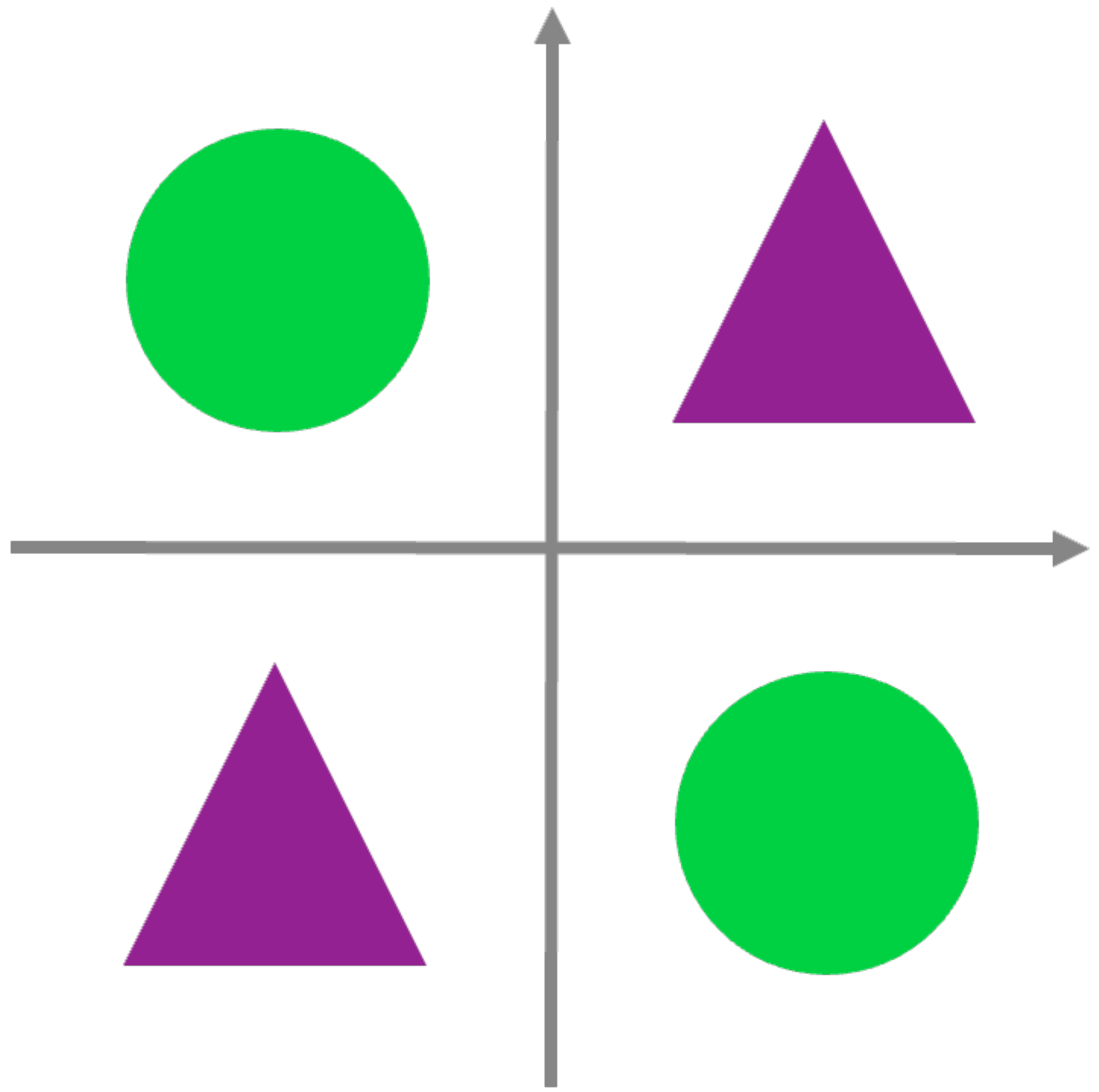
Recap

- SVM, a linear classifier that maximizes ***margin***.
  - **Hard.** Data is linearly separable.
  - **Soft.** NOT linearly separable.
- Both hard & soft SVM are formulated as ***constrained optimization***.
  - Constraints can be made cleaner by the ***method of Lagrange multipliers***, becoming a quadratic optimization.
    - Can be solved by off-the-shelf solvers.
    - Solution takes the form of $\mathbf{w}^* = \sum_i a_i \cdot \mathbf{x}_i$

Features for nonlinear data

Nonlinear data

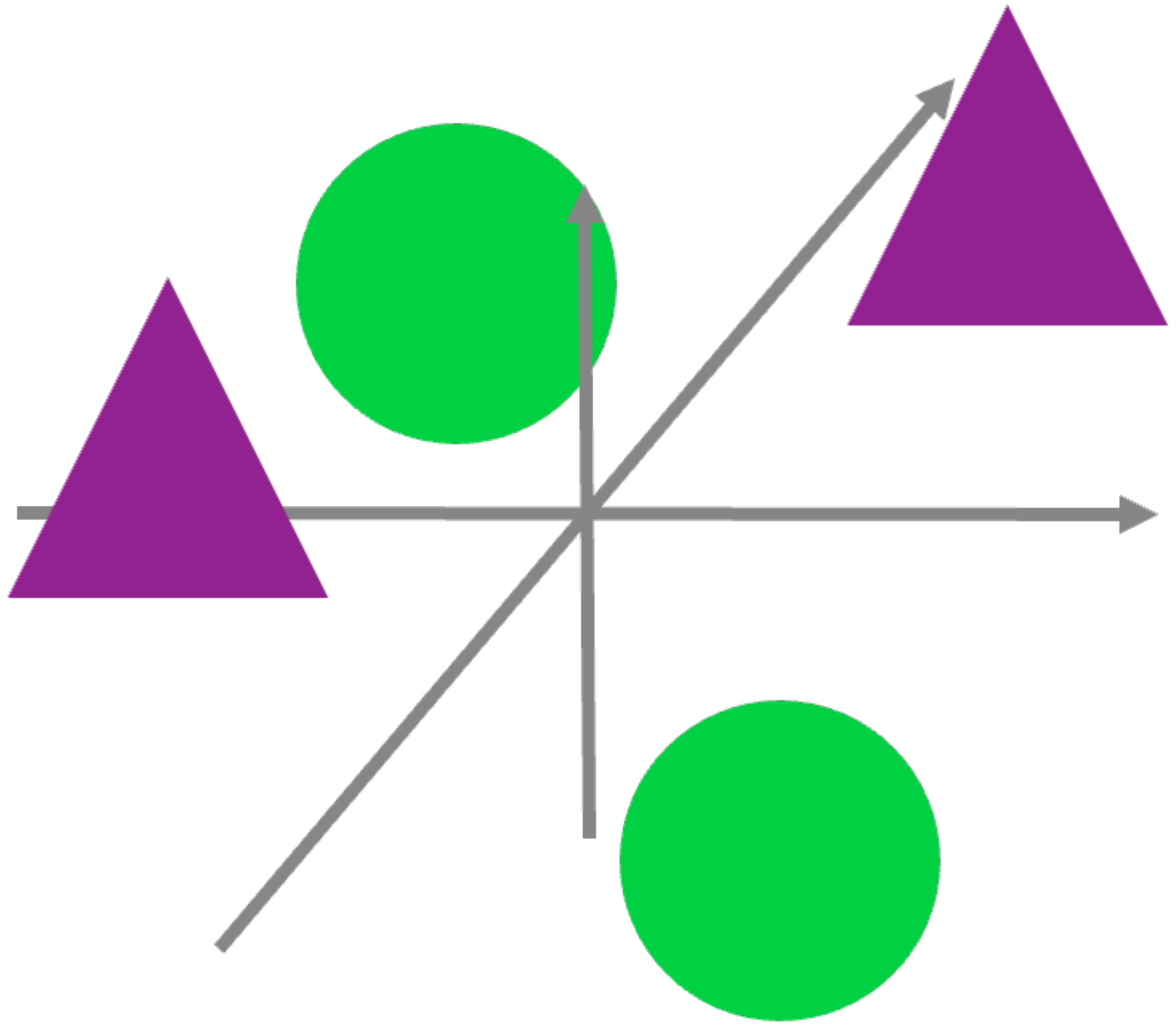
- Suppose that we have a data that looks like ***XOR***.
  - Not linearly separable, and no satisfactory linear classifier exists.



(x_1, x_2)

Nonlinear data

- Suppose that we map it to a ***higher-dimensional space***.
  - Then, there exists a clean linear classifier!



$(x_1, x_2, \boxed{x_1 x_2})$

$$f(\mathbf{x}) = \operatorname{sign}\left(\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right)$$

SVM with a polynomial Kernel visualization

Created by:
Udi Aharoni

More formally...

- We map the data to a high-dim ***feature*** $\Phi(\,\cdot\,) : \mathbb{R}^d \to \mathbb{R}^k$.
    - Typically $d < k$ (but not necessarily)
- Our predictor takes the form

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} a_i \cdot \boxed{\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle} + b \right)$$

    - Similar to original SVMs

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum a_i \cdot \langle \mathbf{x}_i, \mathbf{x} \rangle + b \right)$$

Choosing the feature

- **Q.** How to choose $\Phi(\,\cdot\,)$?
- **Naïve.** Just throw in many features—SVM will choose useful features.

$$\Phi(\mathbf{x}) = (x_1, \cdots, x_d, x_1x_2, \cdots, x_{d-1}x_d, \cdots, x_k^{100})$$

  - This is a bad idea.
    - Overfitting
    - Computation
      - for (1) computing features, and (2) inner products.

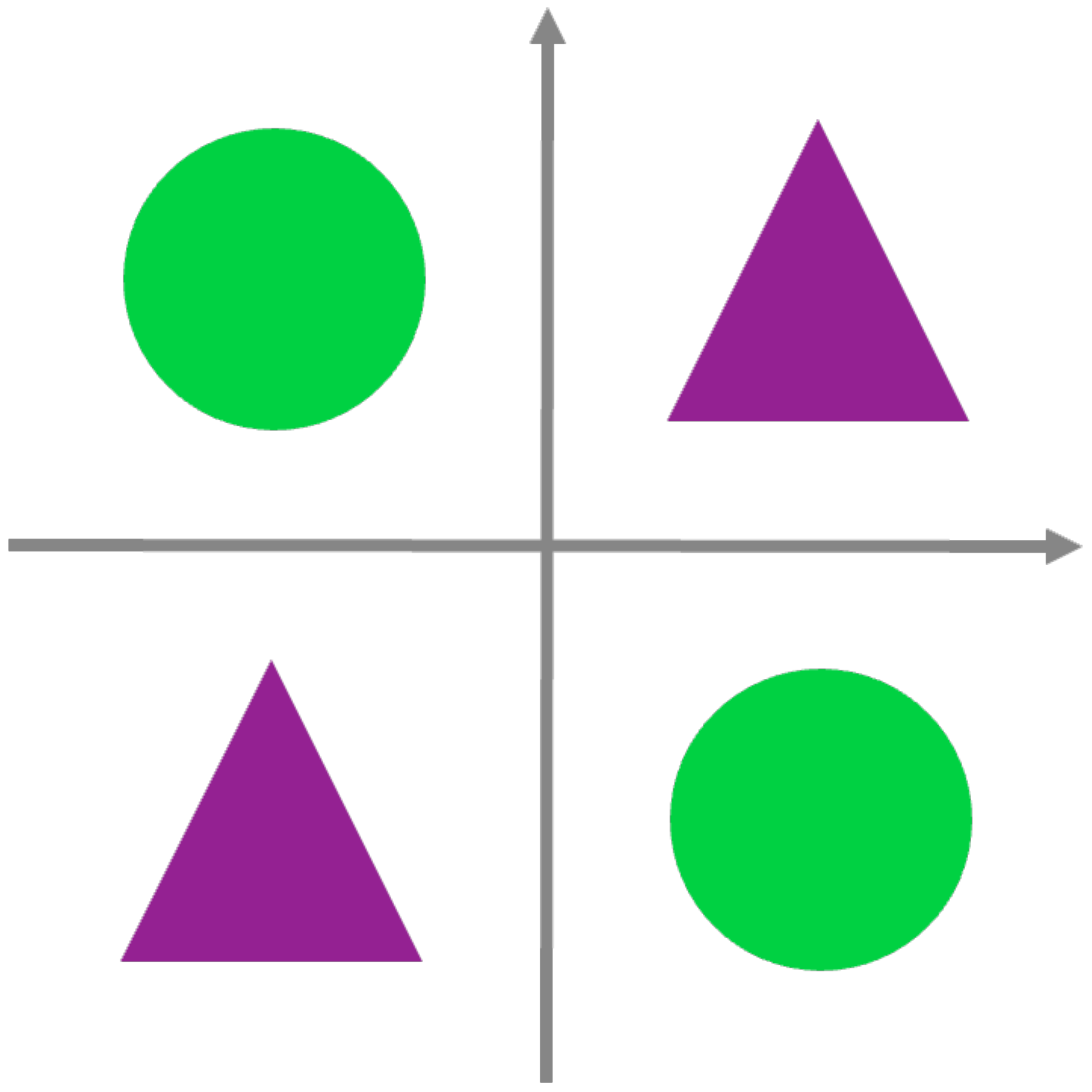
Choosing the feature

- Interestingly, some features admit easy computational shortcuts.
  - *Example*. Recall the XOR case, and think of two features:

$$\Phi_a((x_1, x_2)) = (x_1, x_2, x_1 x_2)$$

$$\Phi_b((x_1, x_2)) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2)$$

  - Looks similar, but one is better than the other.



Choosing the feature

- **Answer.** Φ_b is better, computationally.

$$\Phi_a((x_1, x_2)) = (x_1, x_2, x_1 x_2)$$

$$\langle \Phi_a(\mathbf{x}), \Phi_a(\mathbf{y}) \rangle = x_1 y_1 + x_2 y_2 + 2 x_1 x_2 y_1 y_2$$

(1) Compute 3-d features $\phi_{\mathbf{x}} = \Phi_a(\mathbf{x})$

$$\phi_{\mathbf{y}} = \Phi_a(\mathbf{y})$$

(2) Compute 3-d inner product $\langle \phi_{\mathbf{x}}, \phi_{\mathbf{y}} \rangle$.

Choosing the feature

- **Answer.** Φ_b is better, computationally.

$$\Phi_b((x_1, x_2)) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\begin{aligned}\langle \Phi_b(\mathbf{x}), \Phi_b(\mathbf{y}) \rangle &= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 \\ &= (\langle \mathbf{x}, \mathbf{y} \rangle)^2\end{aligned}$$

(1) Perform 2d inner product $\langle \mathbf{x}, \mathbf{y} \rangle$.

(2) Square.

(less memory & less computation, especially for higher-dim)

Kernel SVM

- **Idea.** Maybe we can do...
    - Choose an easy-to-compute "similarity metric" $K(\,\cdot\,,\,\cdot\,)$.
    - Construct predictors of form

$$f(\mathbf{x}) = \mathrm{sign}\left(\sum a_i \cdot K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

    and fit a_i, b.

- **Question.** Is it equivalent to doing SVM with features?

Kernel SVM

- **Answer.** Yes, if $K(\cdot\,,\cdot)$ is a Mercer kernel.
- **Definition.** A real-valued function $K(\cdot\,,\cdot)$ is called a *Mercer kernel* if
  - $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x})$ # symmetric
  - $\lim_{n\to\infty} K(\mathbf{x}^{(n)}, \mathbf{x}) \to K\left(\lim_{n\to\infty} \mathbf{x}^{(n)}, \mathbf{x}\right)$ # continuous
  - $\sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0, \quad \forall \alpha_i, \alpha_j, \mathbf{x}_i, \mathbf{x}_j$ # positive-semidefinite

Kernel SVM

- **Mercer's Theorem.** For a Mercer kernel $K(\,\cdot\,,\cdot\,)$, there exists a corresponding $\Phi(\,\cdot\,)$ such that

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$$

- That is, if we choose a nice enough $K(\,\cdot\,,\cdot\,)$, we are effectively doing SVM-like thing with some features.

Optimizing Kernel SVM

- In typical SVM, we solved

$$\max_{\alpha}\left(-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j + \sum_{i=1}^{n}\alpha_i\right)$$

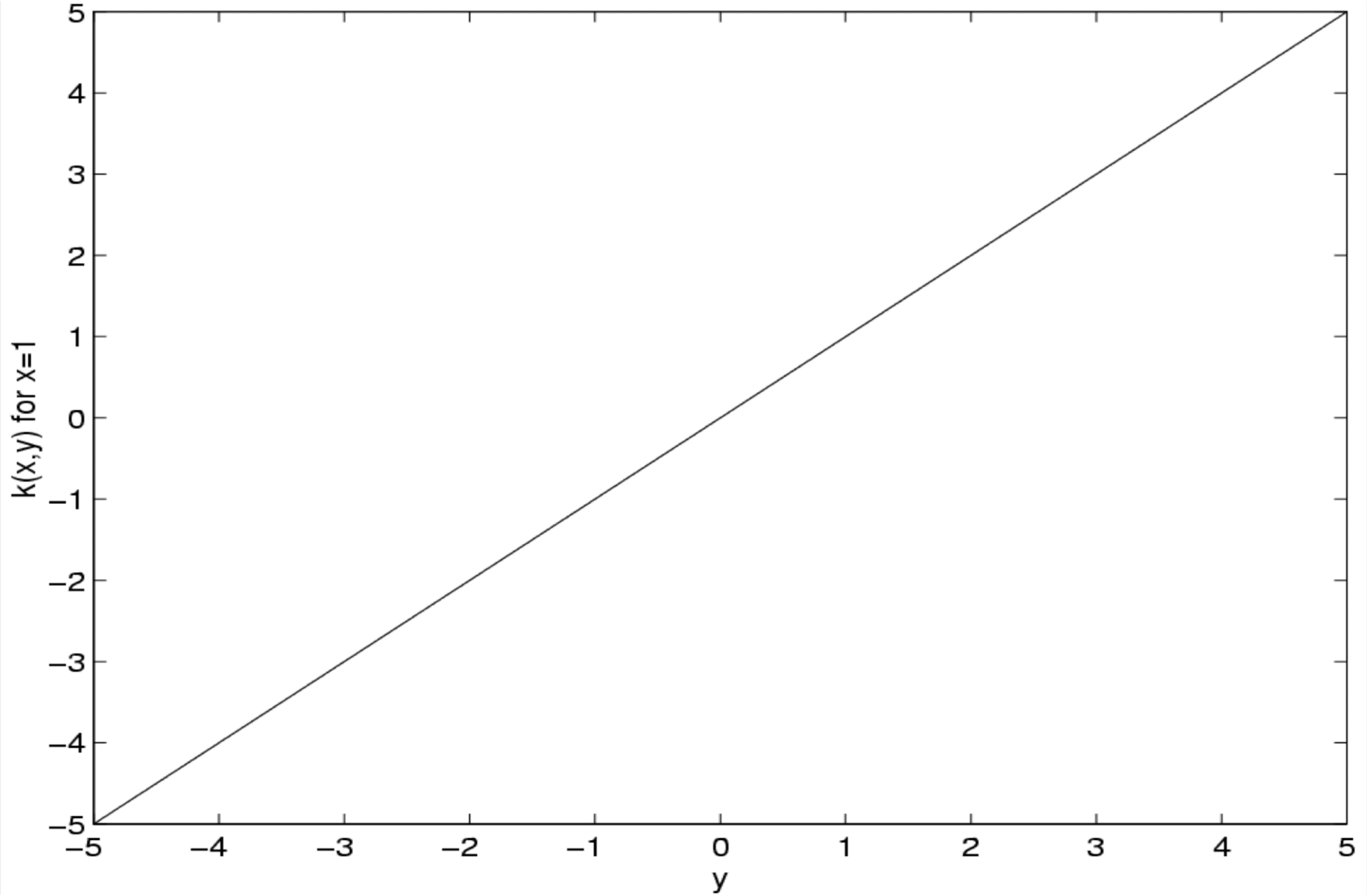
- In kernel SVM, we solve

$$\max_{\alpha}\left(-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i\right)$$

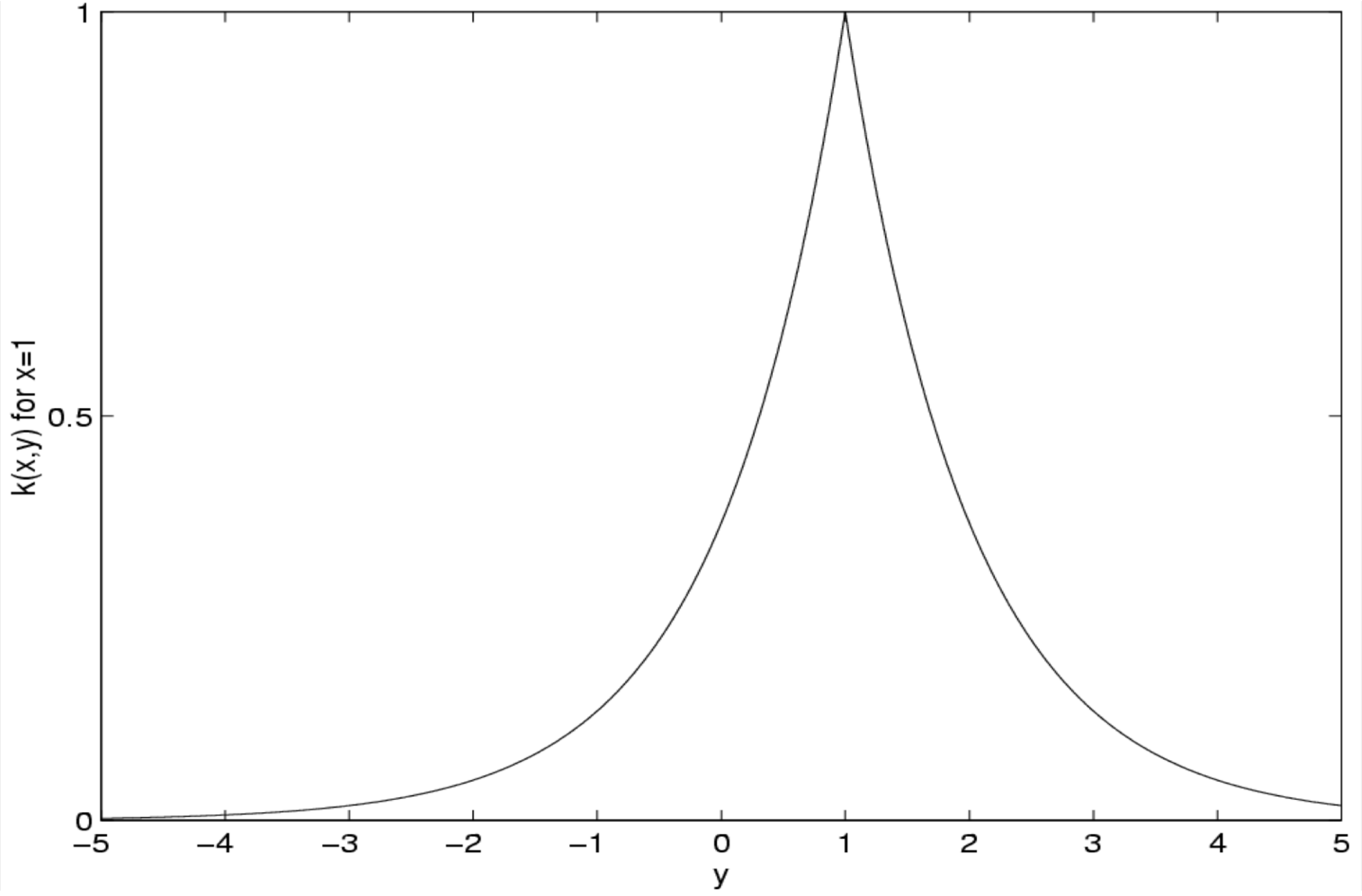
Popular kernels

- **Linear.** $\langle \mathbf{x}, \mathbf{x}' \rangle$.
- **Laplacian RBF**. $\exp(-\lambda \|\mathbf{x} - \mathbf{x}'\|_2)$
- **Gaussian RBF**. $\exp(-\lambda \|\mathbf{x} - \mathbf{x}'\|_2^2)$
- **Polynomial.** $(\langle \mathbf{x}, \mathbf{x}' \rangle + c)^d$
- **B-Spline**

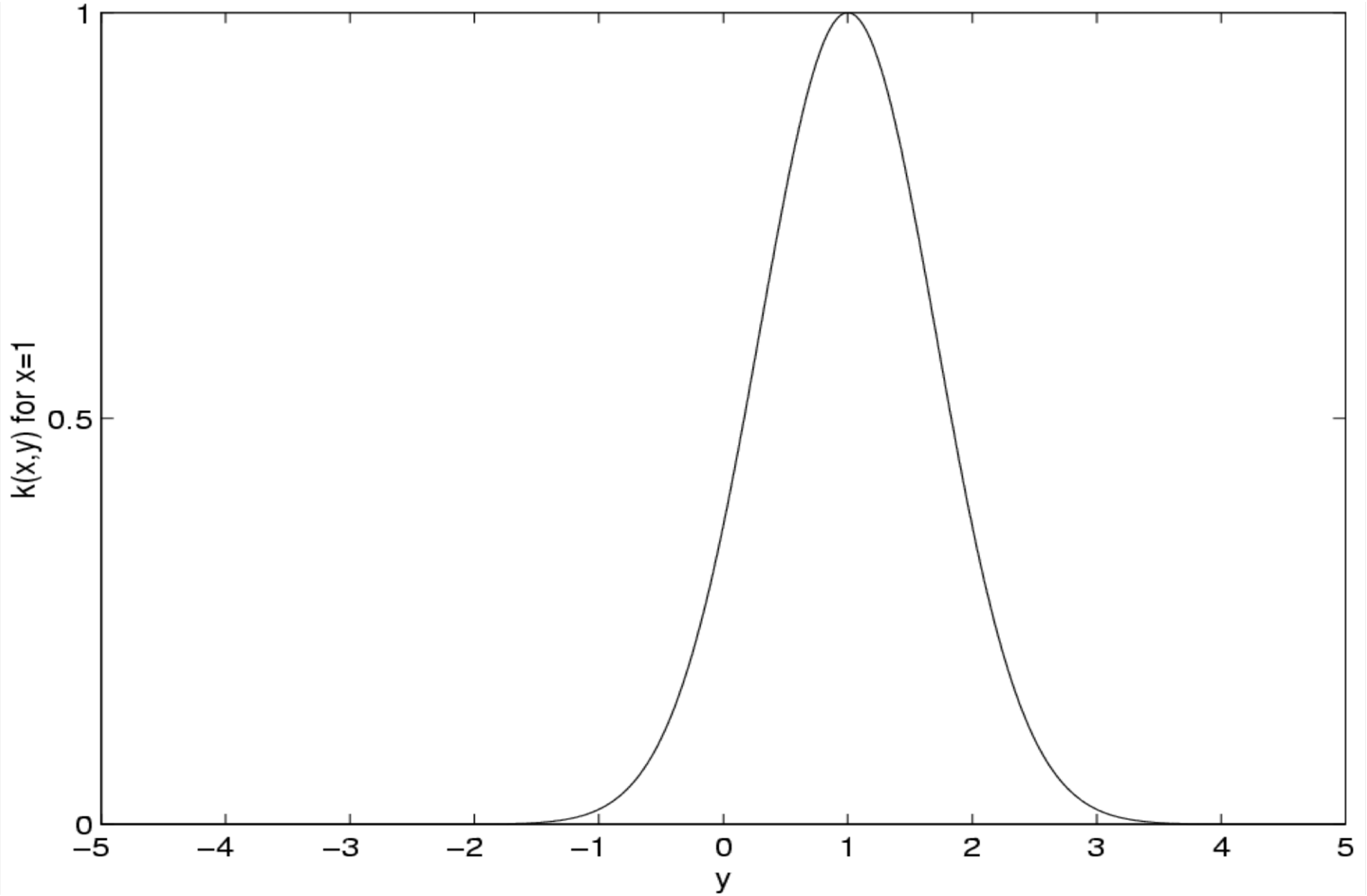
Linear Kernel



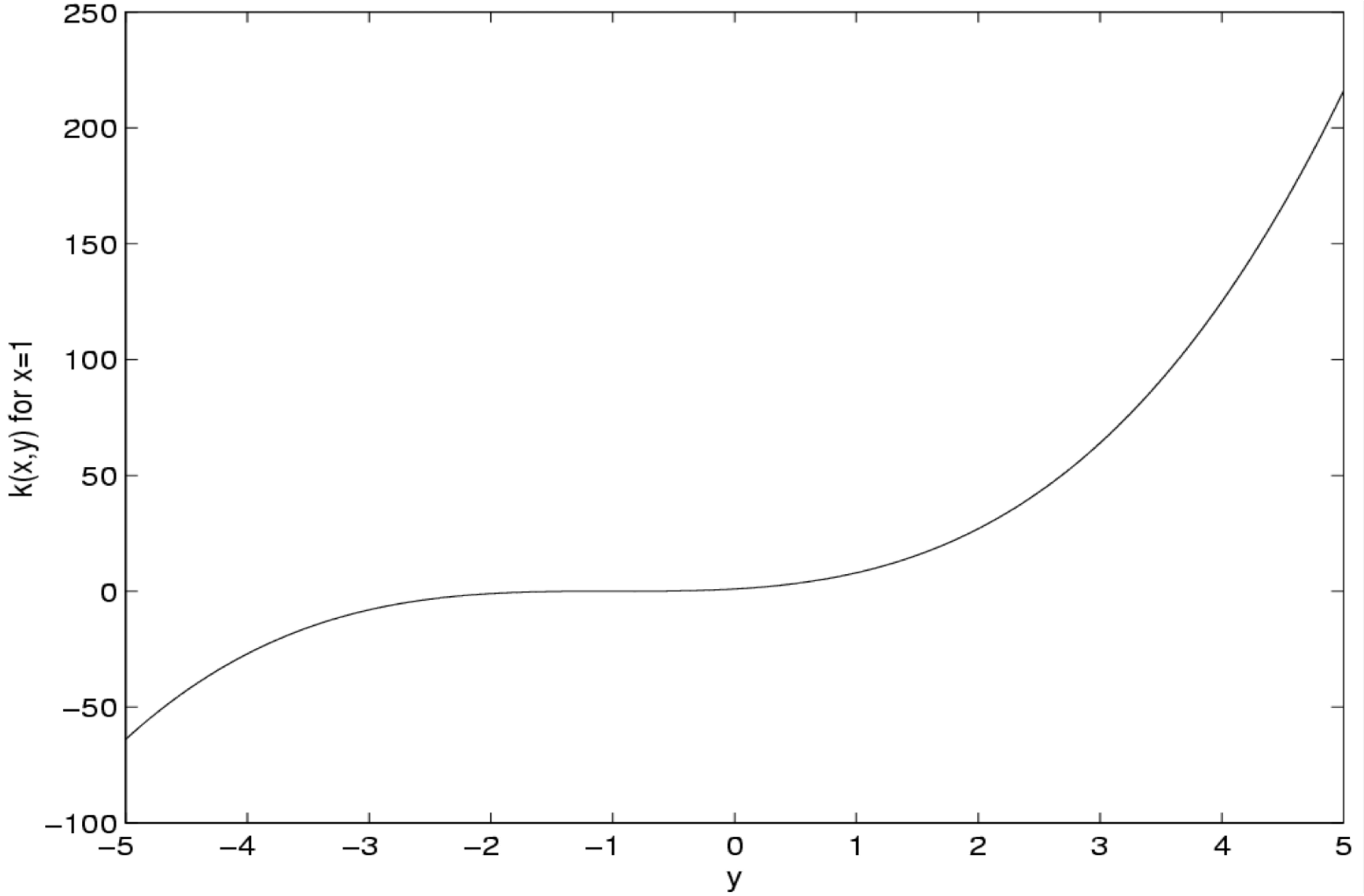
Laplacian Kernel



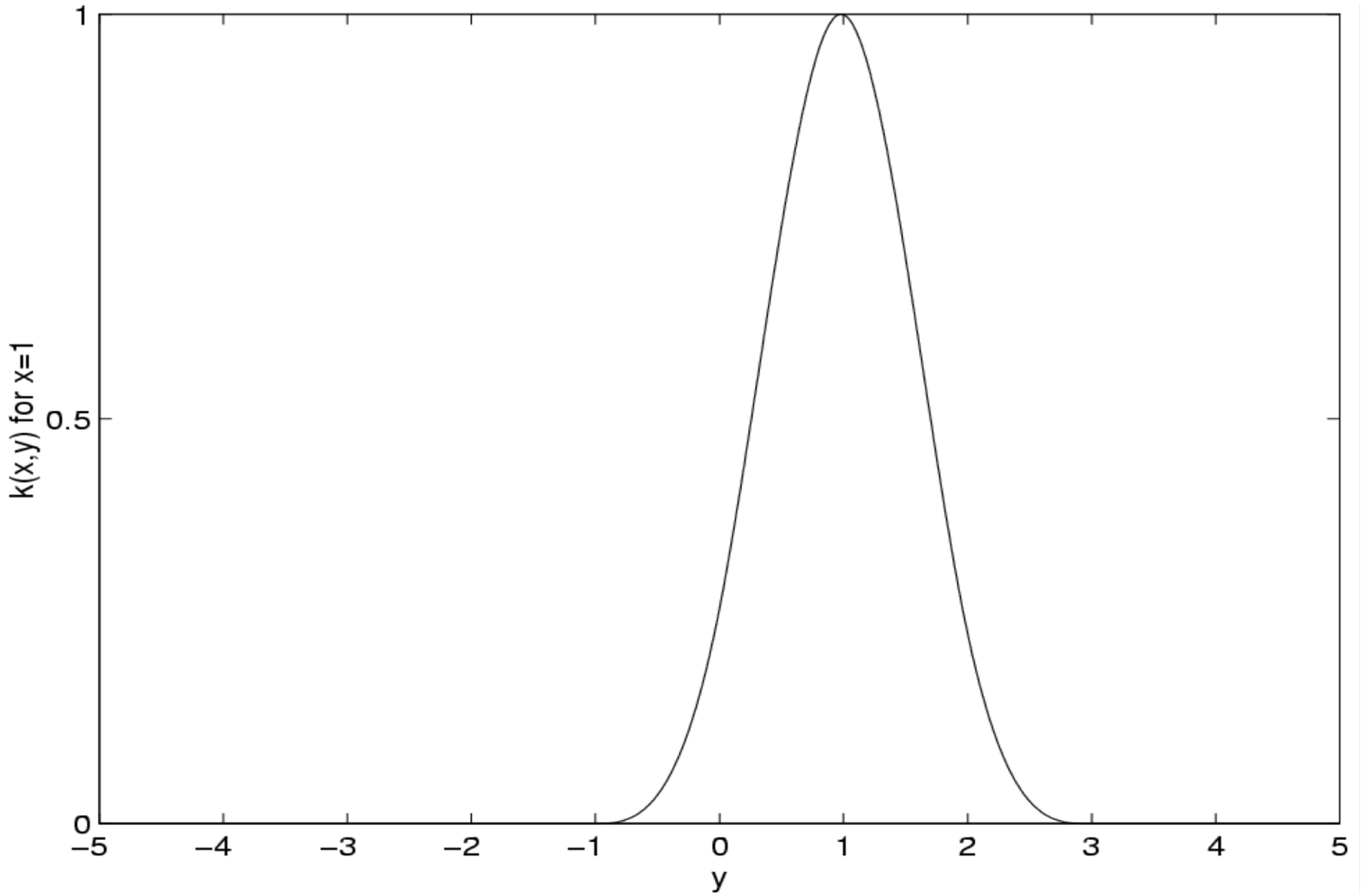
Gaussian Kernel



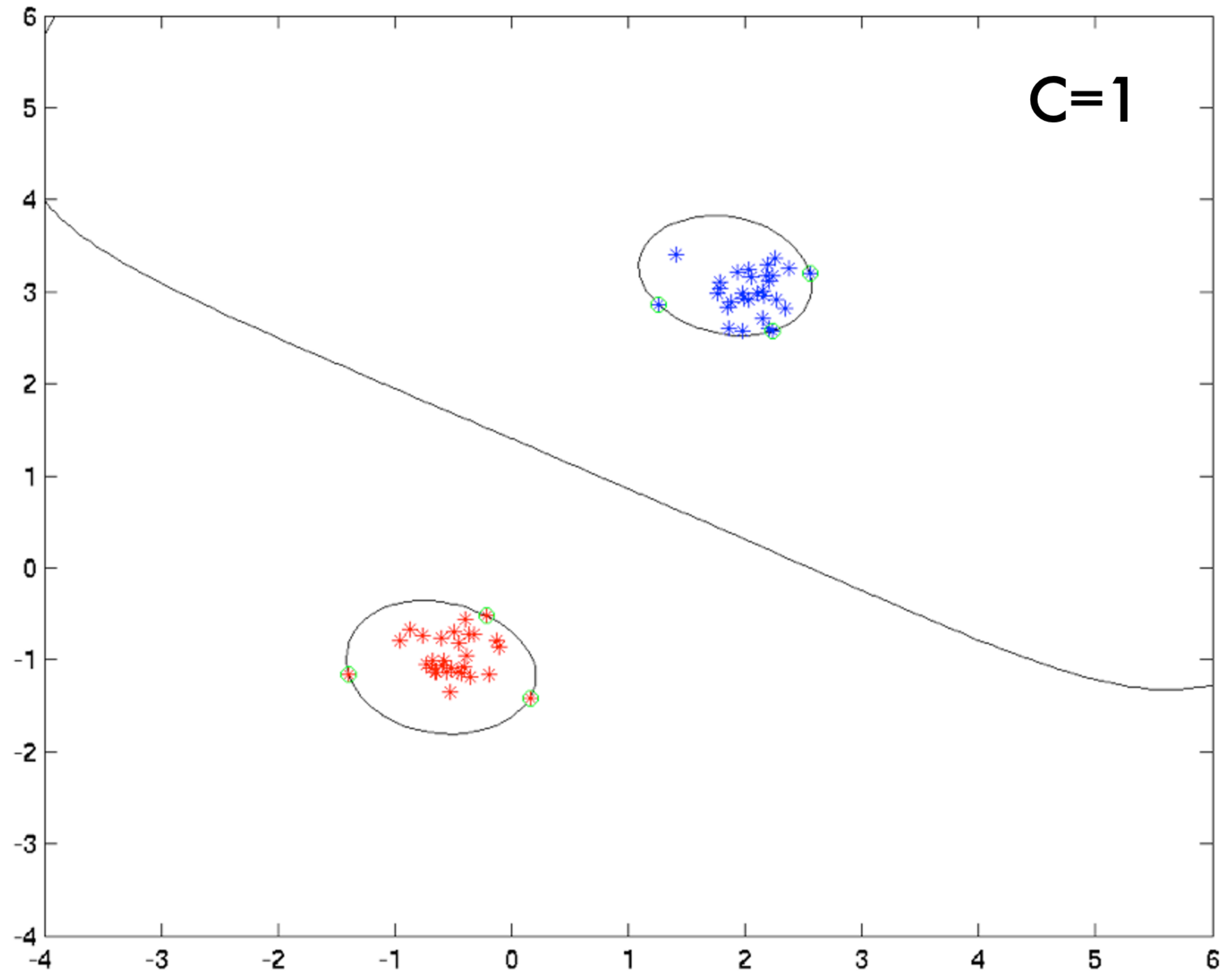
Polynomial of order 3



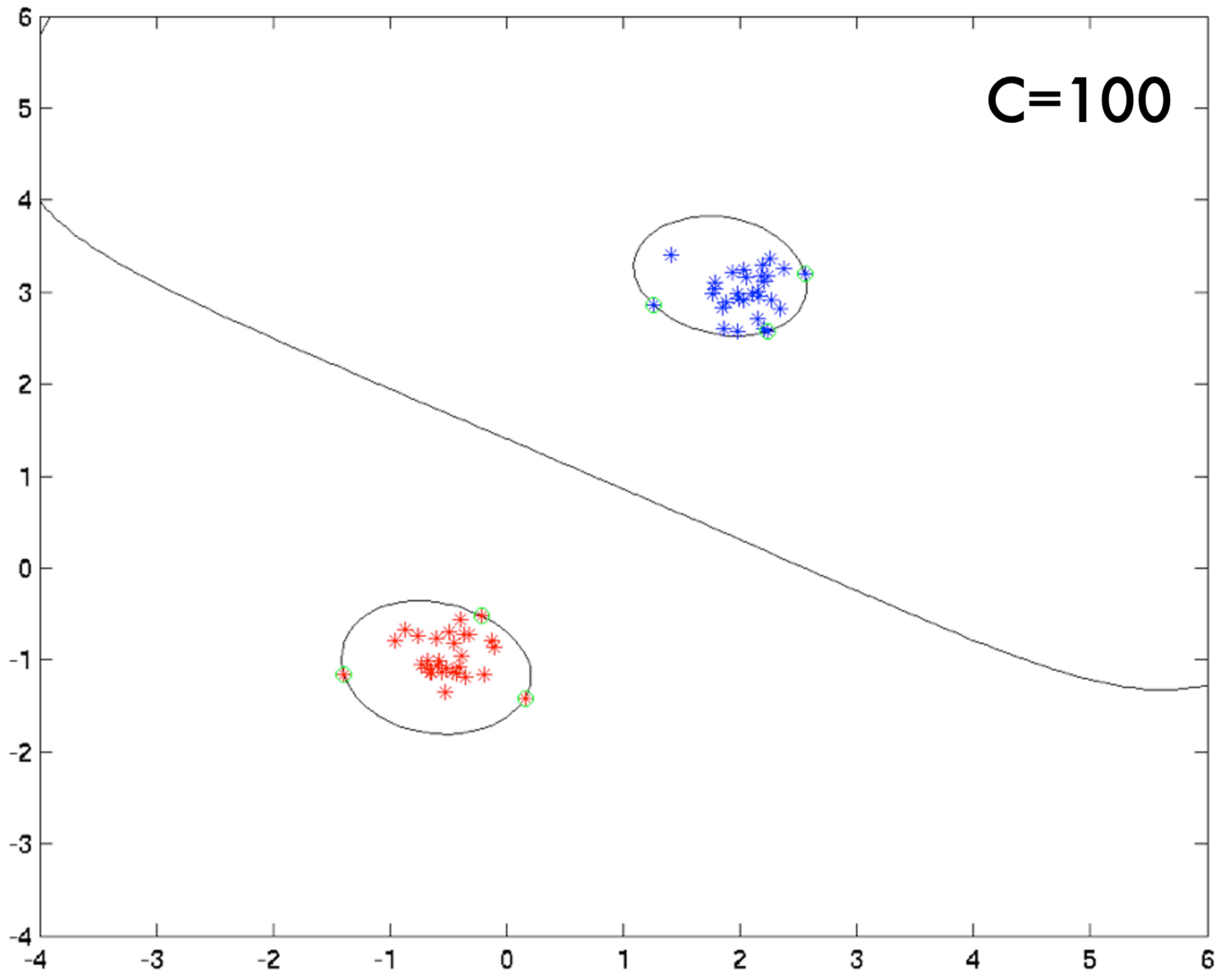
B_3 Spline Kernel



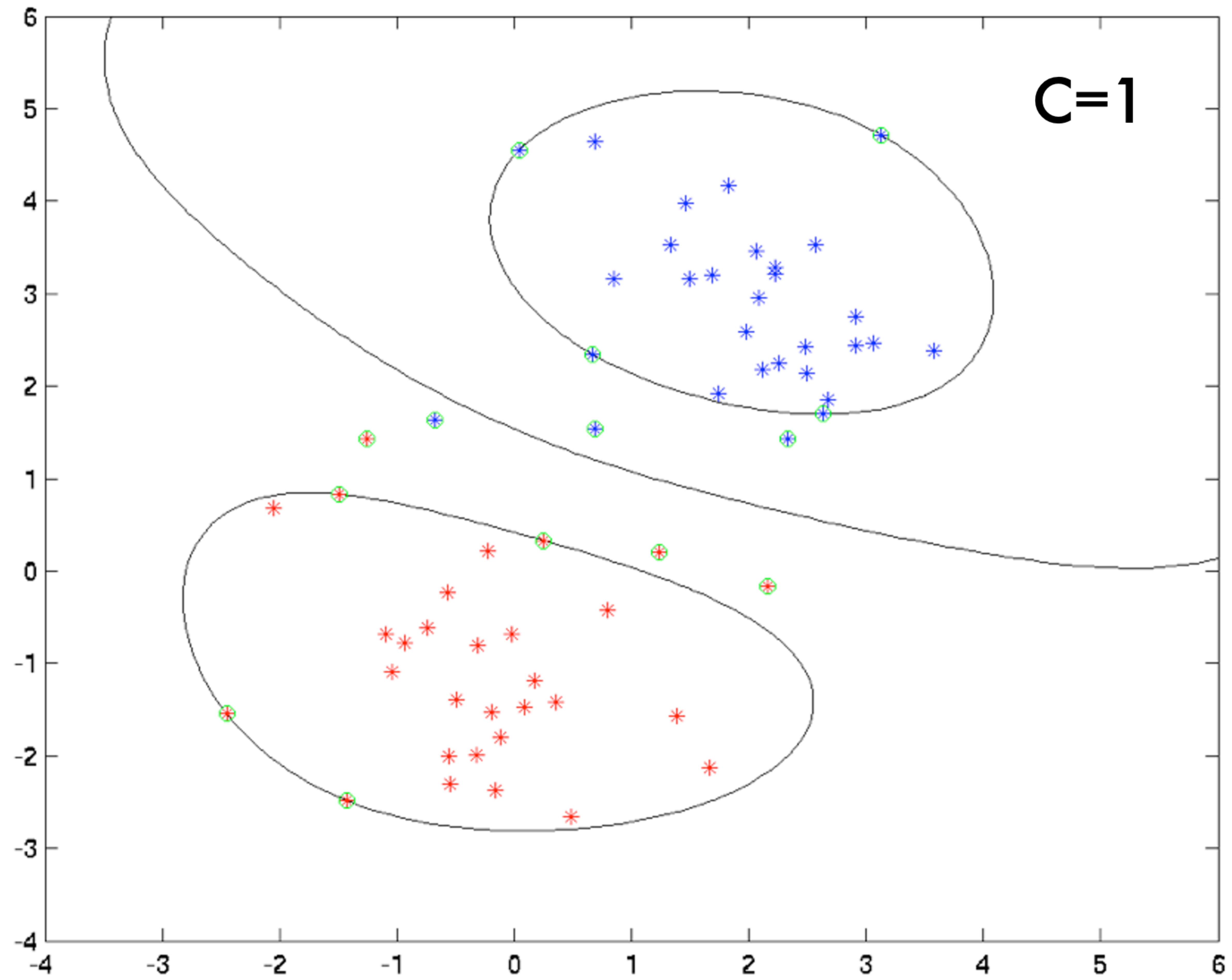
Well Separable Case



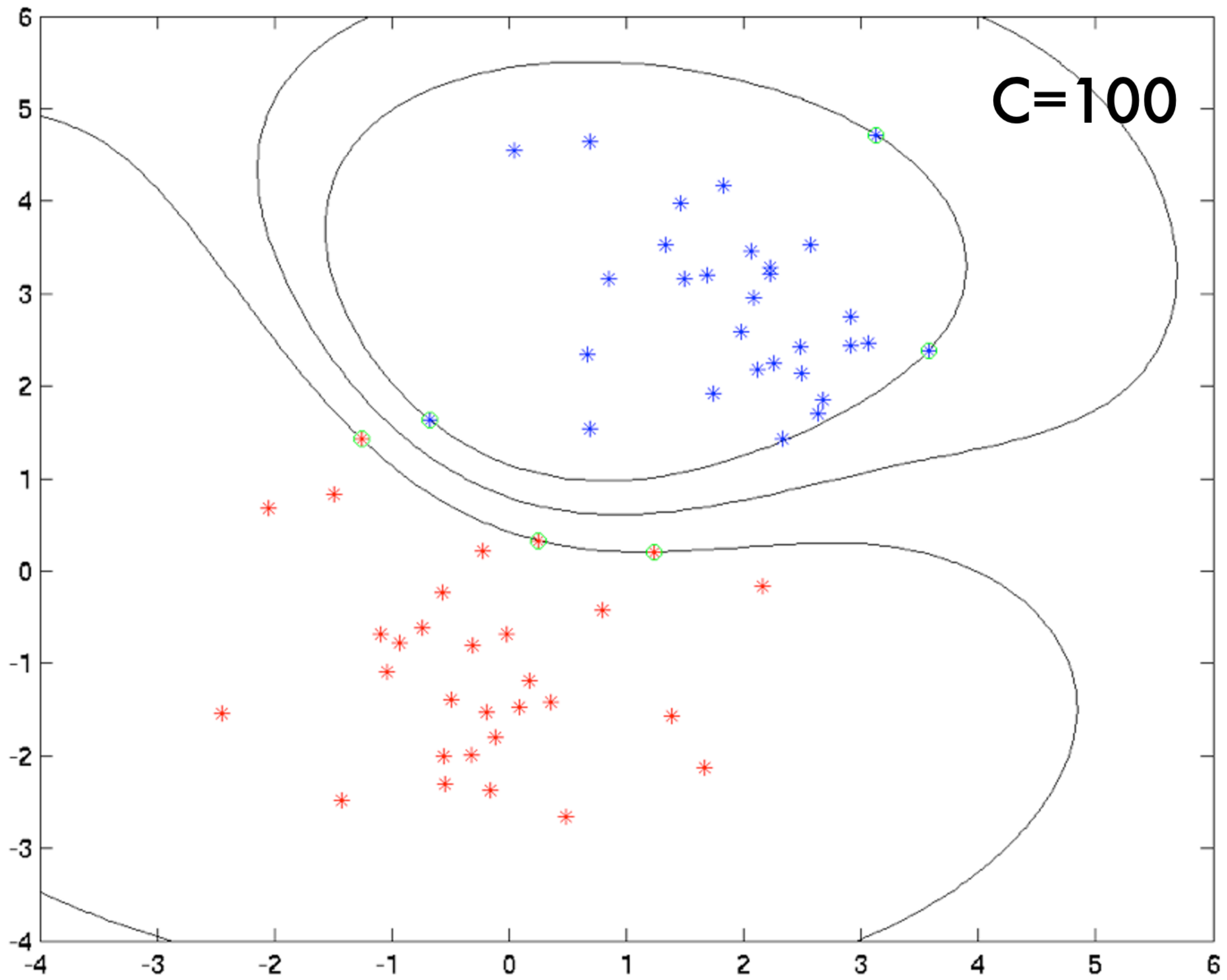
Well Separable Case



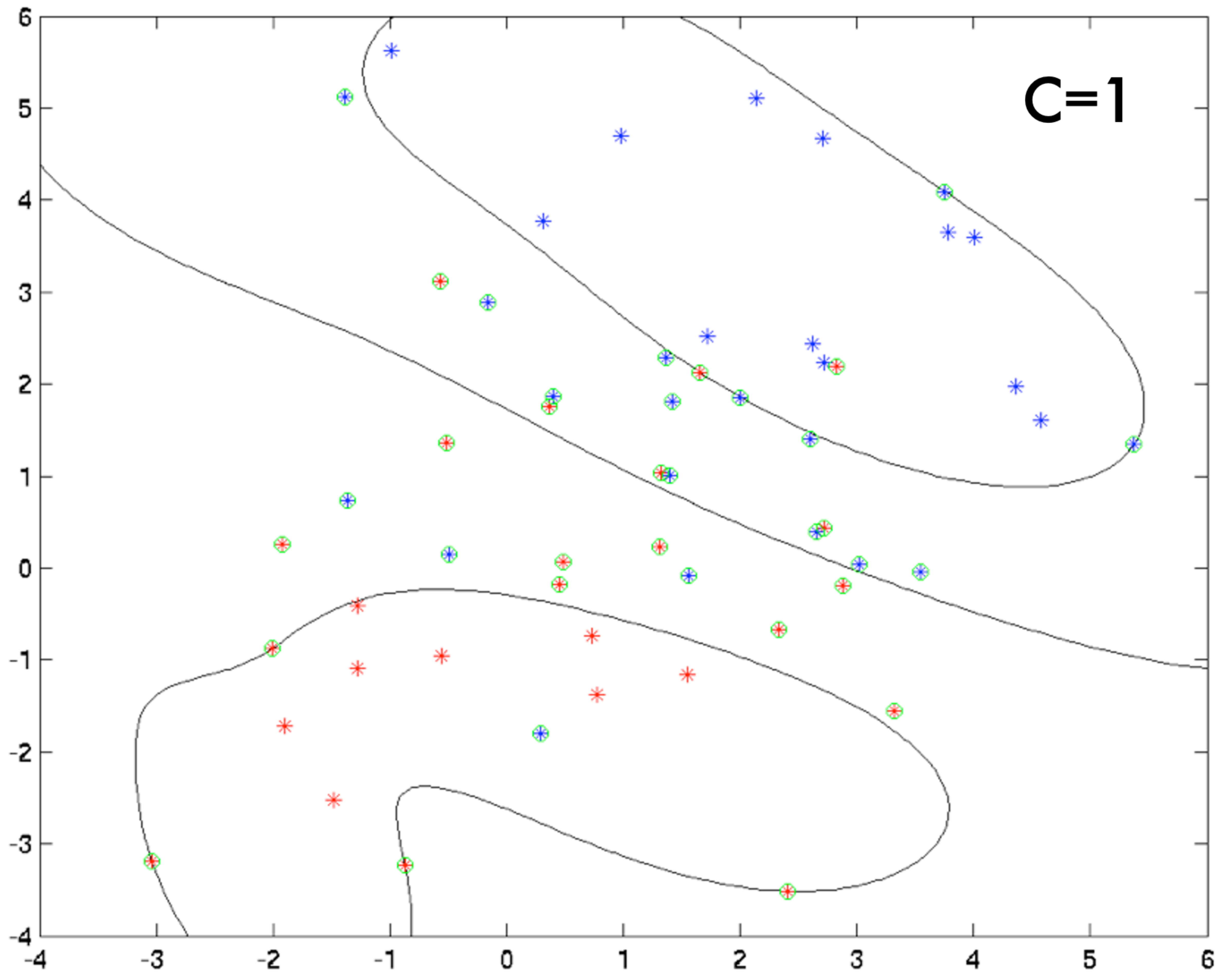
Closer Call



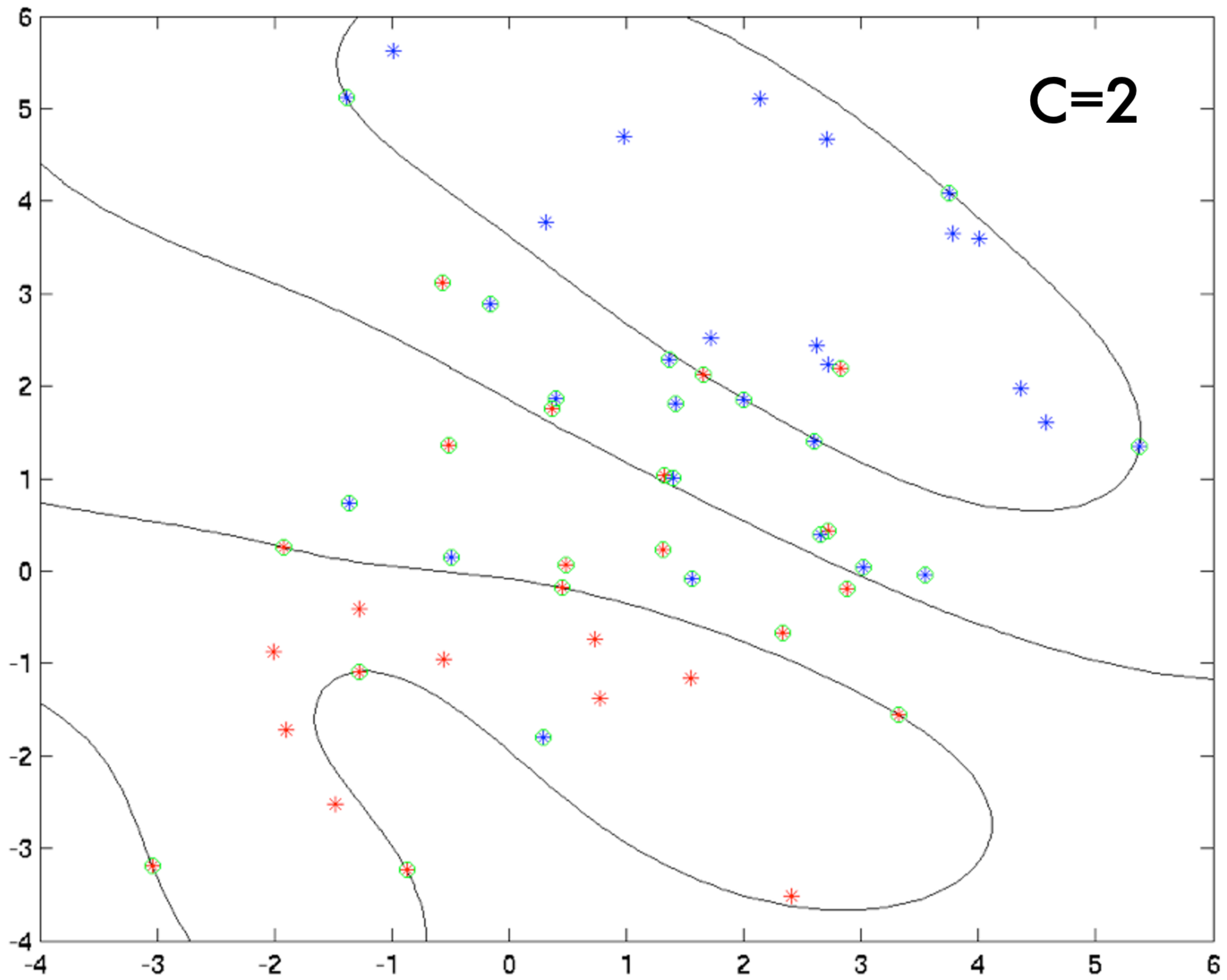
Closer Call



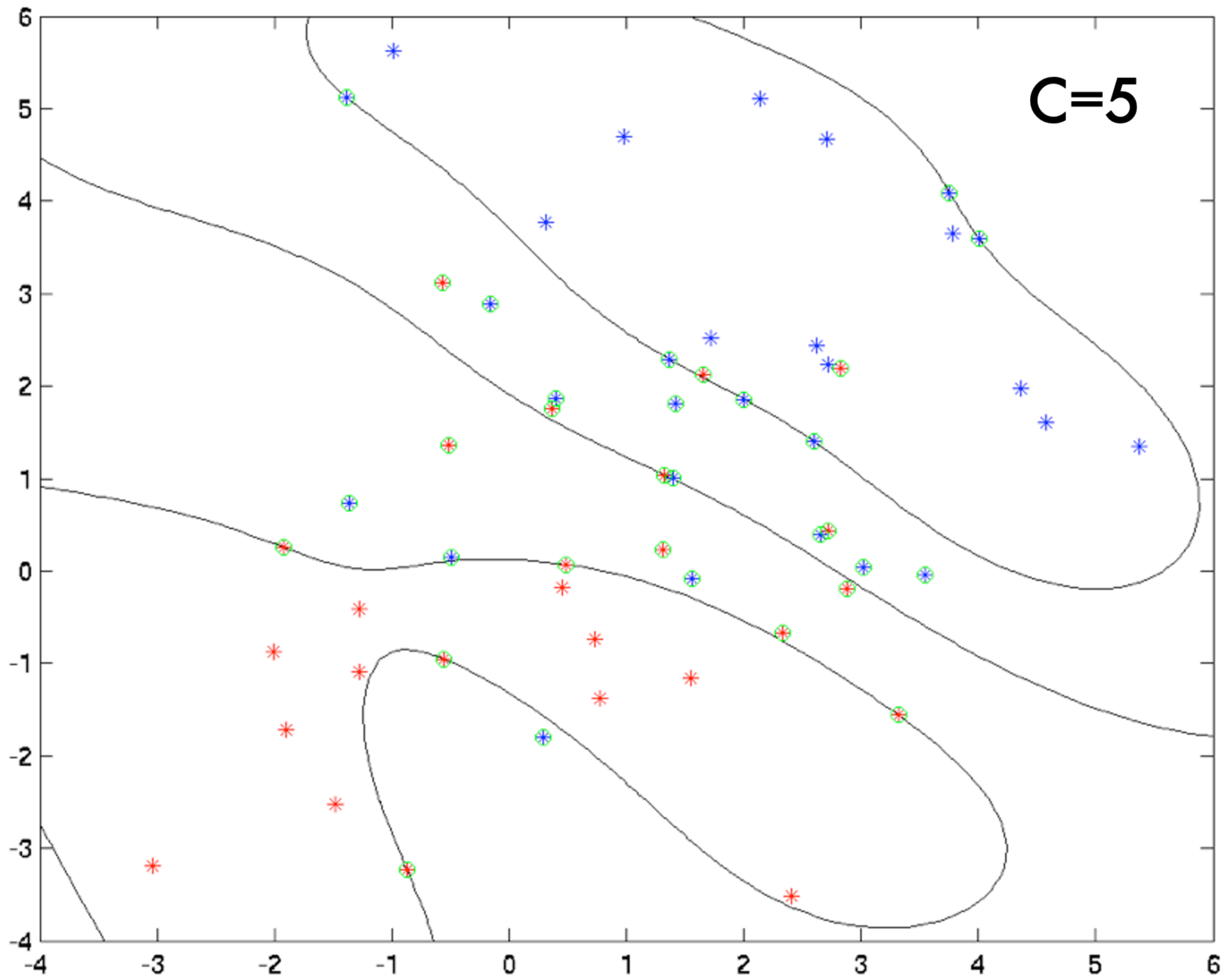
With Outliers



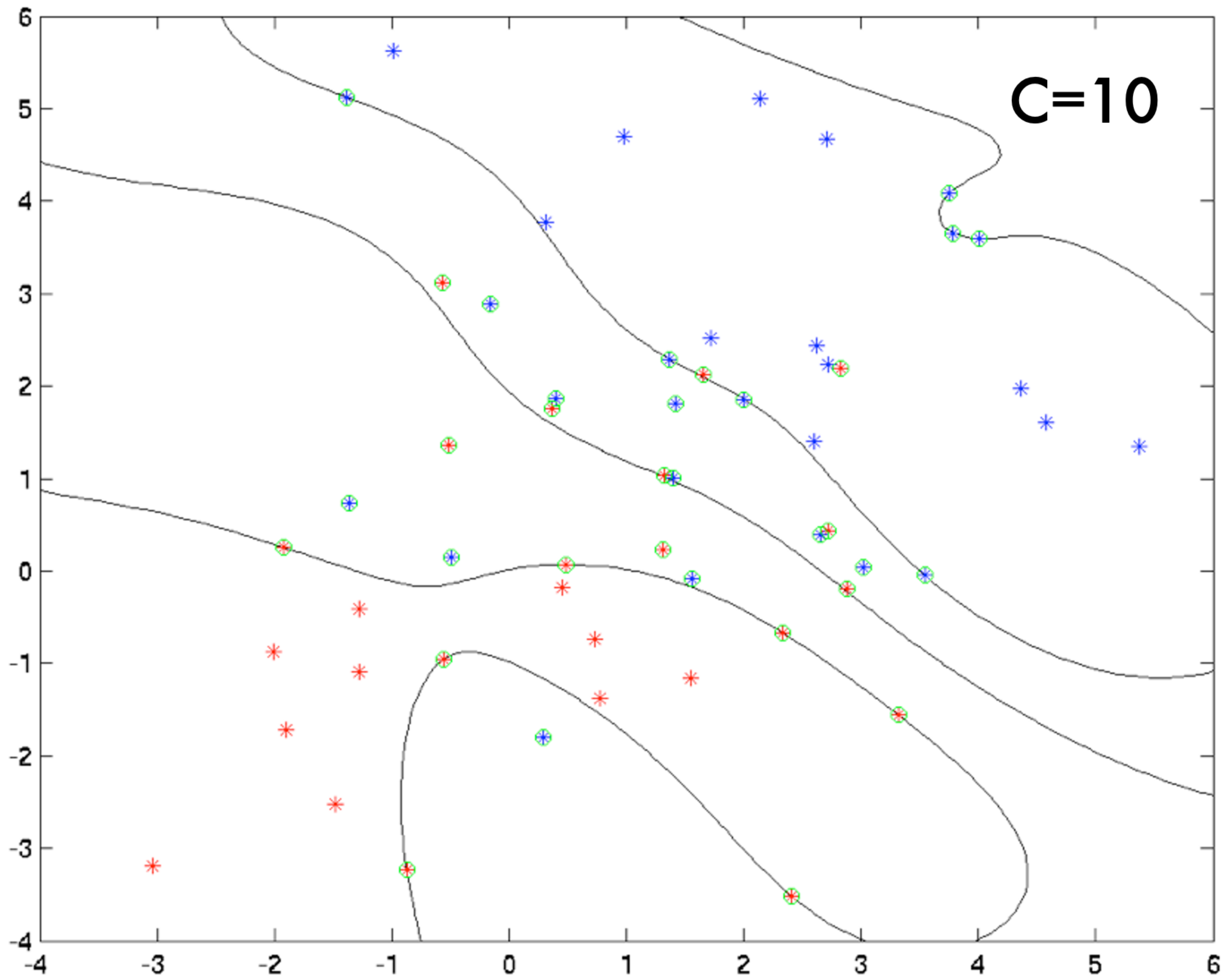
With Outliers



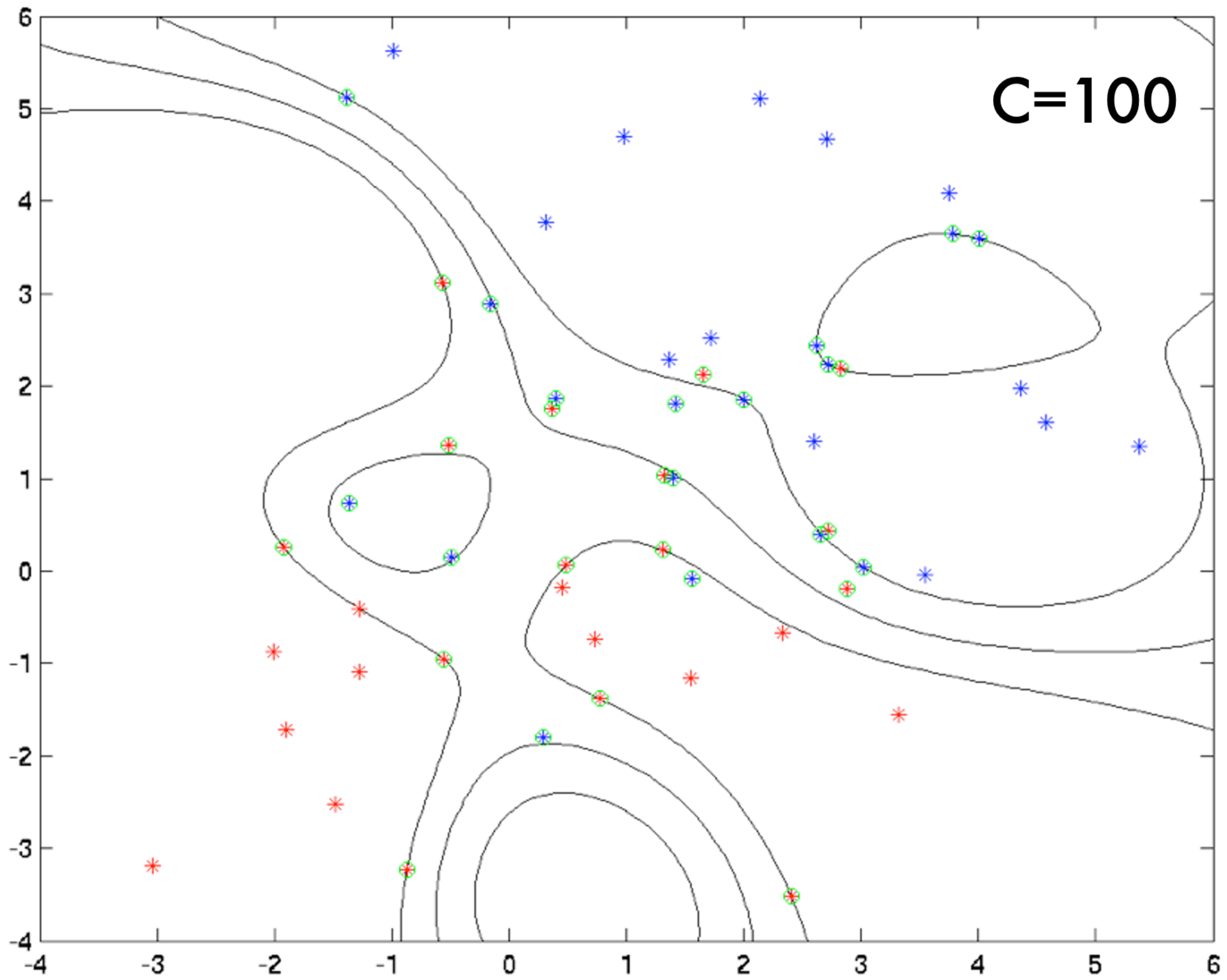
With Outliers



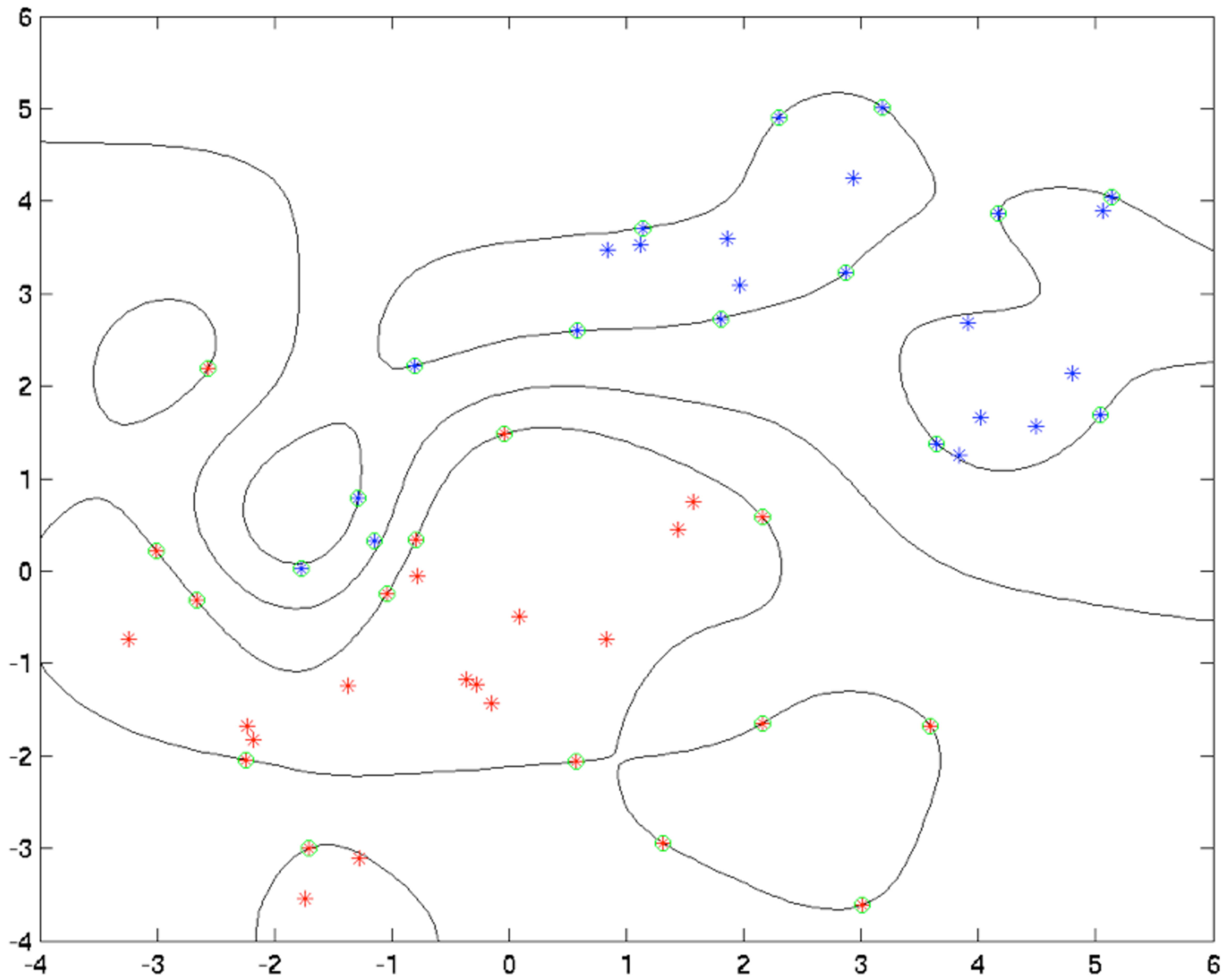
With Outliers



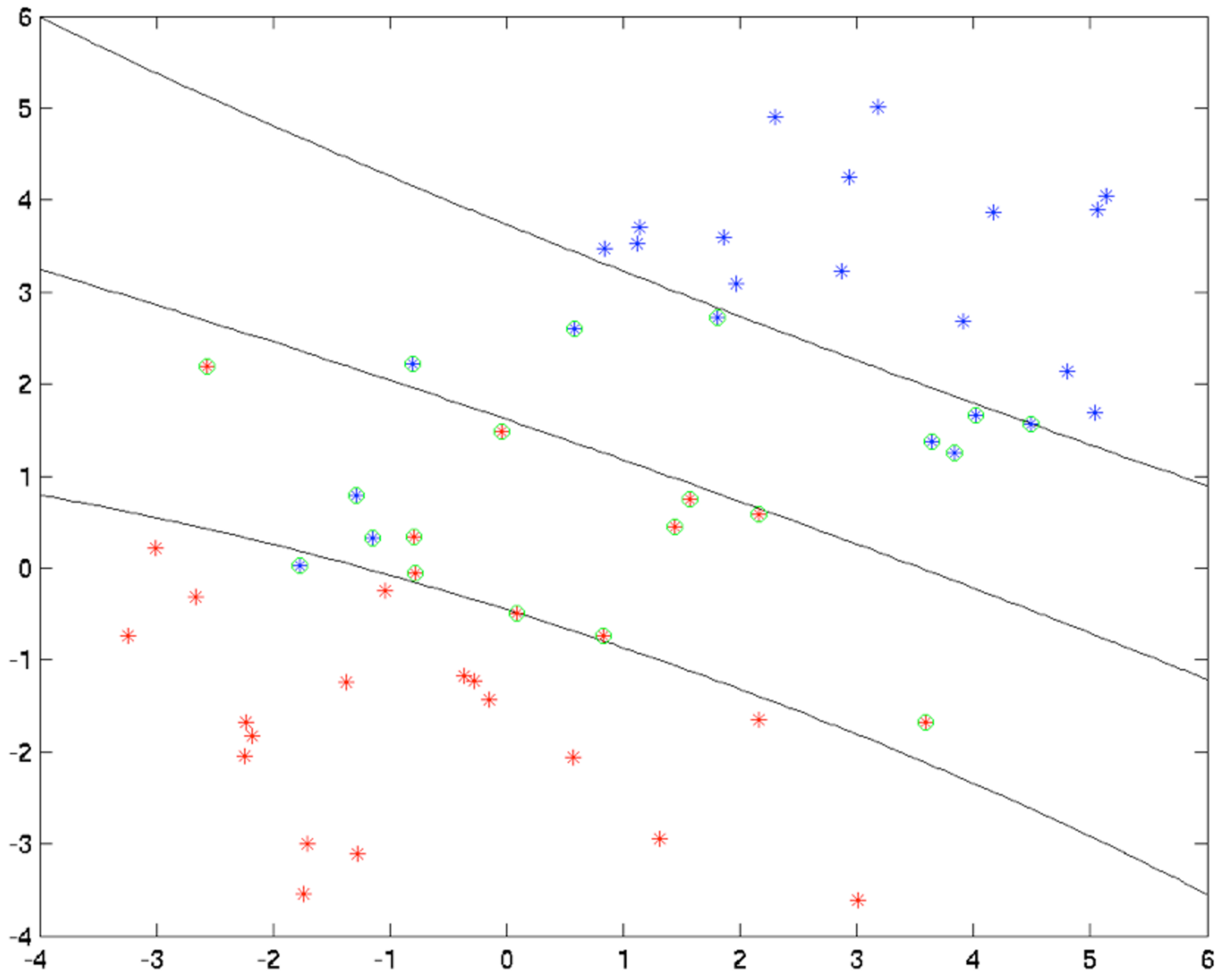
With Outliers



Very Narrow Kernels



Very Wide Kernels



In Deep Learning Era

- In deep learning era, we find nice $\Phi(\cdot)$ using the data.
  - Jointly train with classifier (supervised)
  - Use nice augmentations to find a nice similarity metrics such that
    - $\Phi(\mathbf{x}) - \Phi(\mathbf{x}_{\text{aug}})$ is smaller than $\Phi(\mathbf{x}) - \Phi(\mathbf{x}')$

Cheers

- *Next up.* K-Means