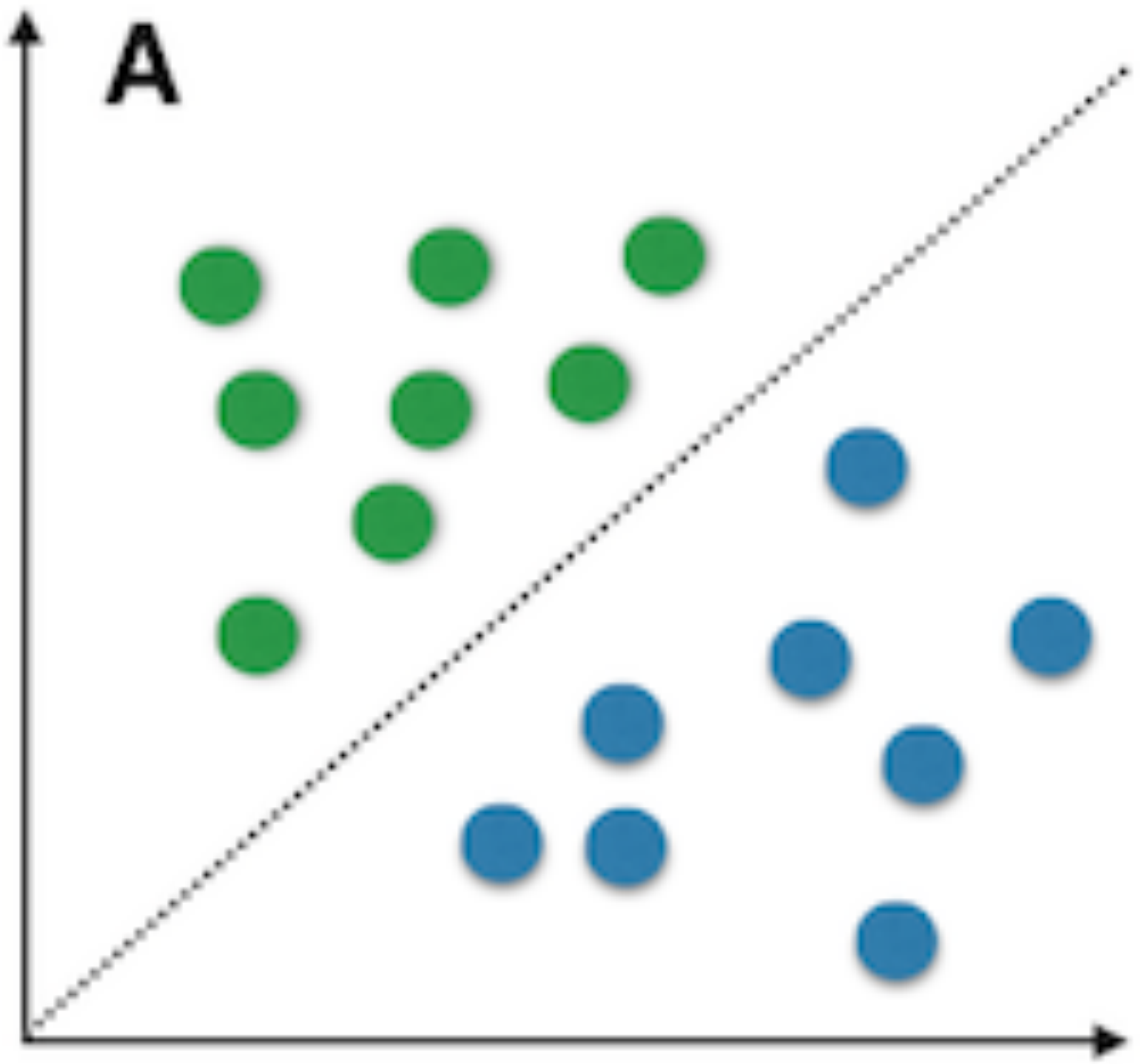
Soft & Kernel SVMs

EECE454 Intro. to Machine Learning Systems

Fall 2024

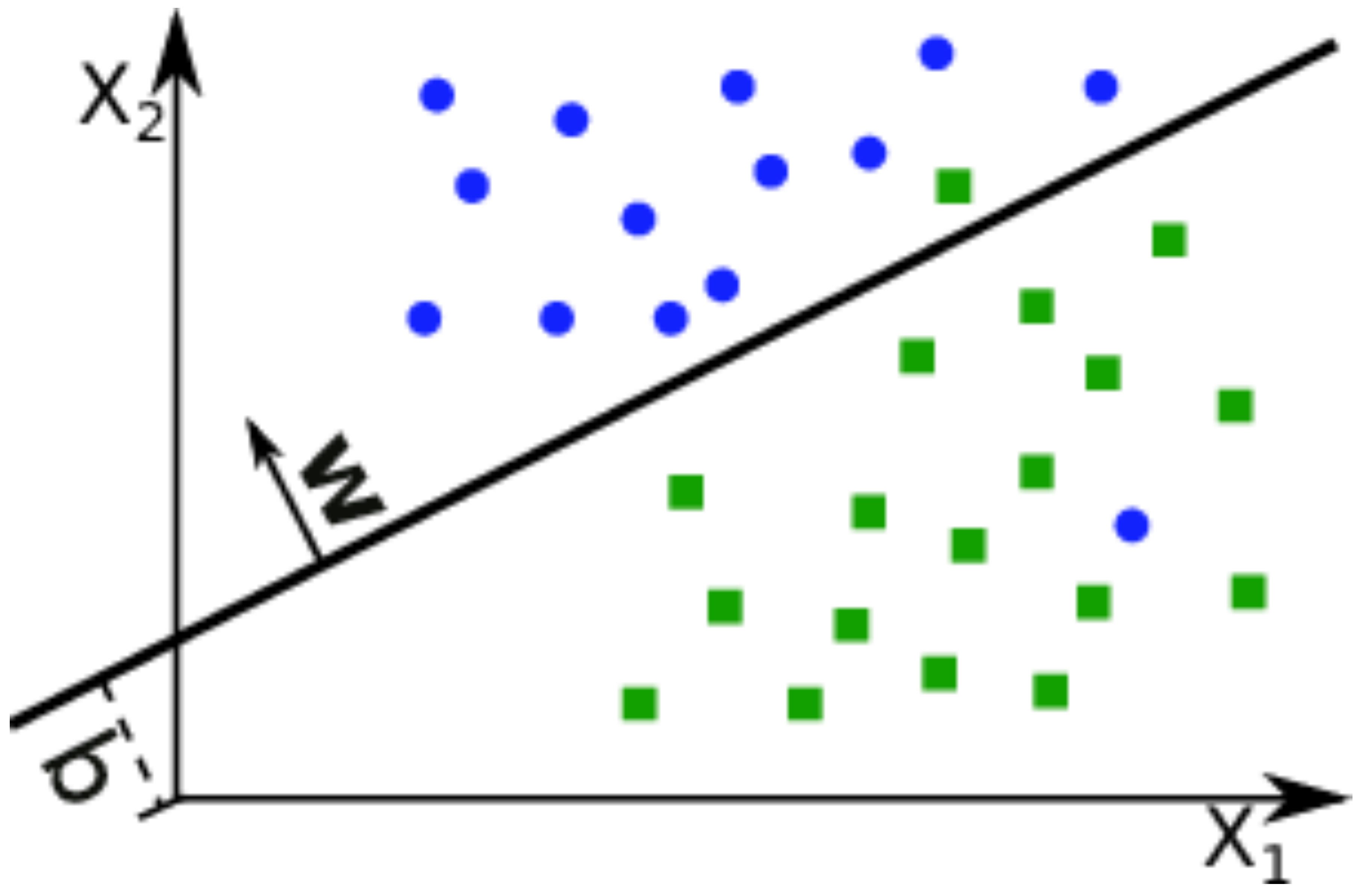
Today

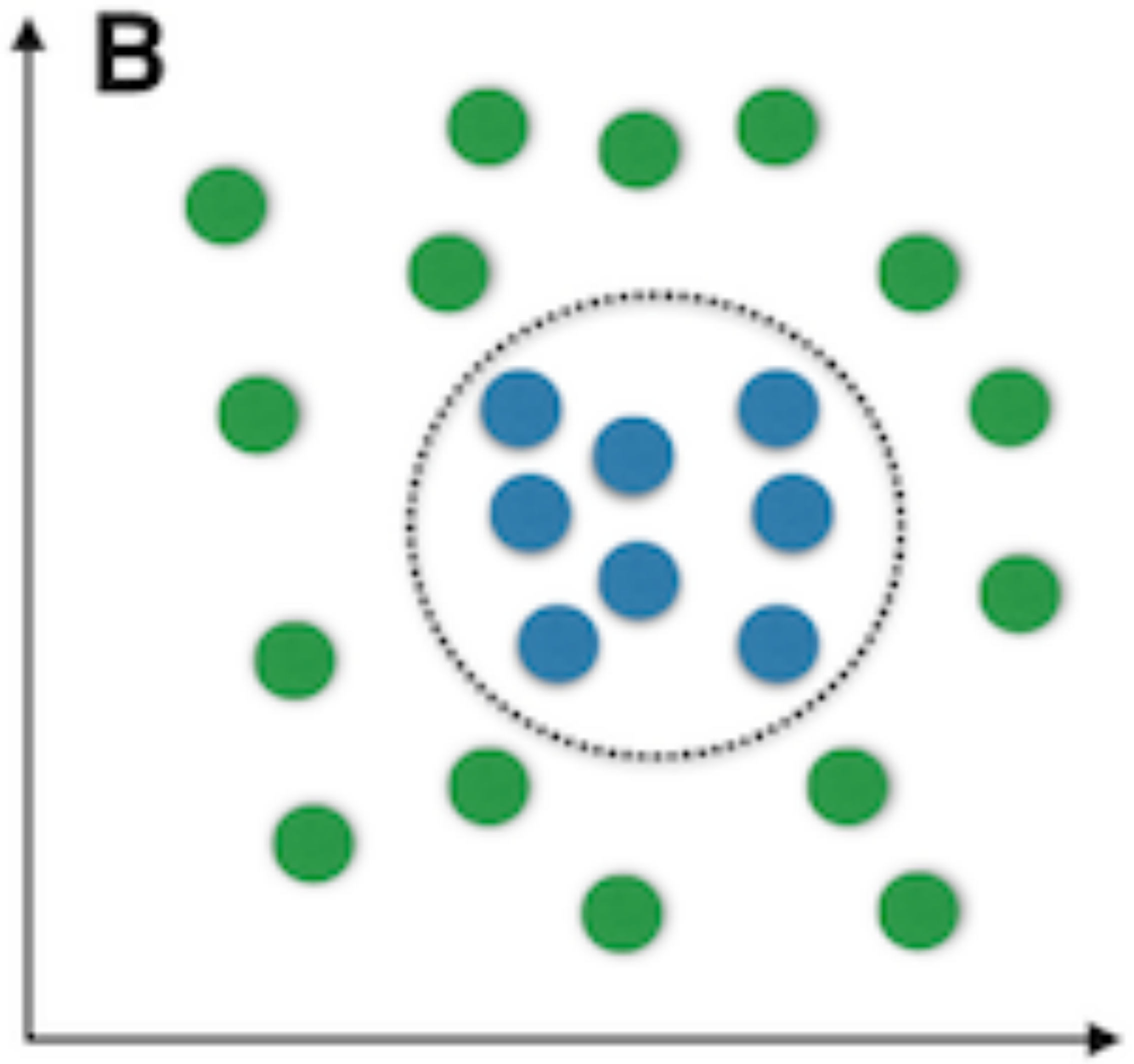
- **Last class.** Support Vector Machine
  - Linear model that maximizes the margin
  - Lagrangian dual —> Quadratic problem
  - Required. Data is linearly separable



Today

- **Last class.** Support Vector Machine
    - Linear model that maximizes the margin
    - Lagrangian dual —> Quadratic problem
    - Required. Data is linearly separable
- **Today.** SVMs that can handle nonseparable data
    - Soft-margin SVM
    - Kernel SVM

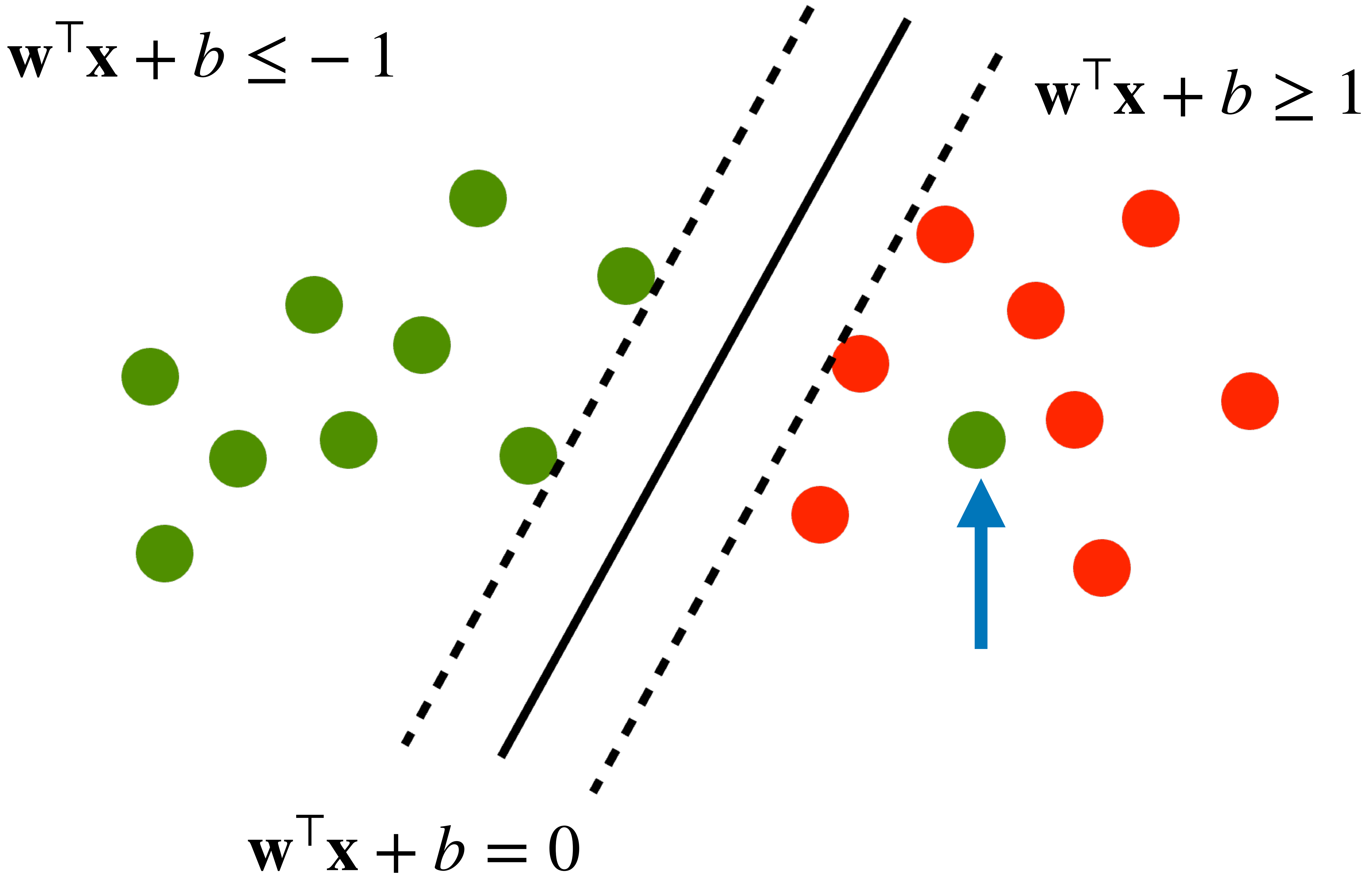




Soft(-Margin) SVM

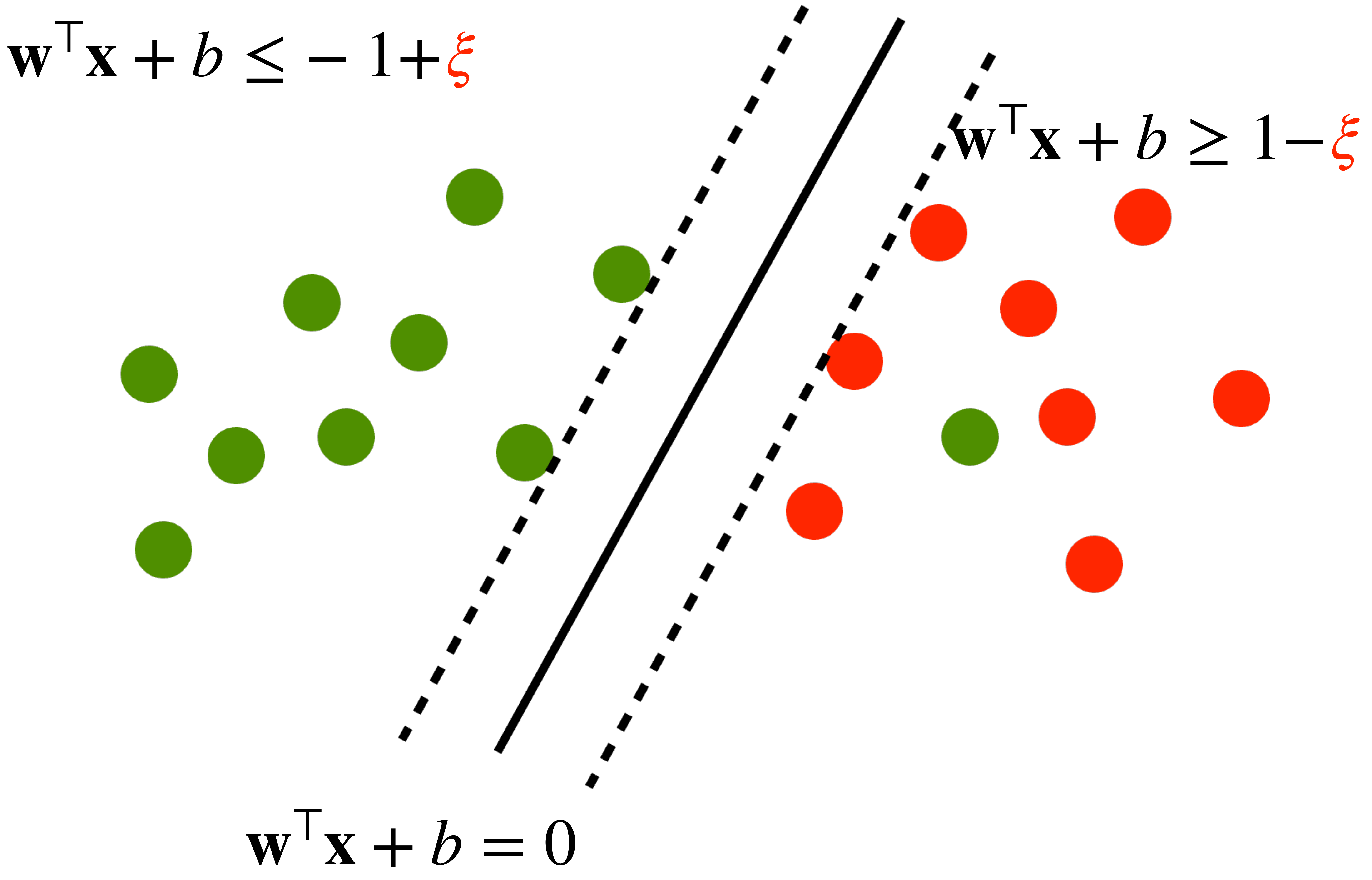
Data with outliers

- Suppose that there exists some **outlier**
    - Then, no linear separator exists
    - Worse. finding a minimum-error separating hyperplane is NP-hard (Minsky & Papert, 1969)
- **Q.** How can we handle this situation?



Data with outliers

- Suppose that there exists some **outlier**
  - Then, no linear separator exists
  - Worse. finding a minimum-error separating hyperplane is NP-hard (Minsky & Papert, 1969)
- **Q.** How can we handle this situation?
  - A. Add some slack variable ξ
    - Then, aim for minimizing the slack as well



Formulation

- We are now solving the optimization problem

$$\ell^* = \min_{\mathbf{w},b,\xi} \frac{\|\mathbf{w}\|^2}{2} + C \cdot \sum_i \xi_i$$

$$\text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0$$

Formulation

- We are now solving the optimization problem

$$\ell^* = \min_{\mathbf{w},b,\xi} \frac{\|\mathbf{w}\|^2}{2} + C \cdot \sum_i \xi_i$$

$$\text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0$$

- Then, we know that the problem is always feasible
  - Constraint can be met in any case
  - For example, let $\mathbf{w} = \mathbf{0}$, $b = 0$, and $\xi_i = 1$.

Dual Formulation

- As a dual, we get

$$\min_{\mathbf{w},b,\xi} \max_{\alpha,\eta} \left(\frac{\|\mathbf{w}\|^2}{2} + C \sum_i \xi_i - \sum_i \alpha_i \big(y_i(\mathbf{x}_i^\top \mathbf{w} + b) + \xi_i - 1 \big) - \sum_i \eta_i \xi_i \right)$$

  - The optimal $(\mathbf{w}, b, \xi)$ is at the saddle point with (α, η)

Dual Formulation

- As a dual, we get

$$\min_{\mathbf{w},b,\xi} \max_{\alpha,\eta} \left(\frac{\|\mathbf{w}\|^2}{2} + C\sum_i \xi_i - \sum_i \alpha_i \big(y_i(\mathbf{x}_i^\top \mathbf{w} + b) + \xi_i - 1\big) - \sum_i \eta_i \xi_i \right)$$

- The optimal $(\mathbf{w}, b, \xi)$ is at the saddle point with (α, η)

- Derivatives for $(\mathbf{w}, b, \xi)$ needs to vanish!
  - $\nabla_{\mathbf{w}} \mathscr{L} = \mathbf{w} - \sum \alpha_i y_i \mathbf{x}_i = 0$
  - $\nabla_b \mathscr{L} = \sum \alpha_i y_i = 0$
  - $\nabla_{\xi_i} \mathscr{L} = C - \alpha_i - \eta_i = 0$

Dual Formulation

- Doing the similar thing, we get the Lagrangian

$$-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j + \sum_i \alpha_i - \sum_i \alpha_i \xi_i + C\sum_i \xi_i - \sum_i \eta_i \xi_i$$

$$= -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j + \sum_i \alpha_i$$

Dual Formulation

- Doing the similar thing, we get the Lagrangian

$$-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j + \sum_i \alpha_i - \sum_i \alpha_i \xi_i + C\sum_i \xi_i - \sum_i \eta_i \xi_i$$

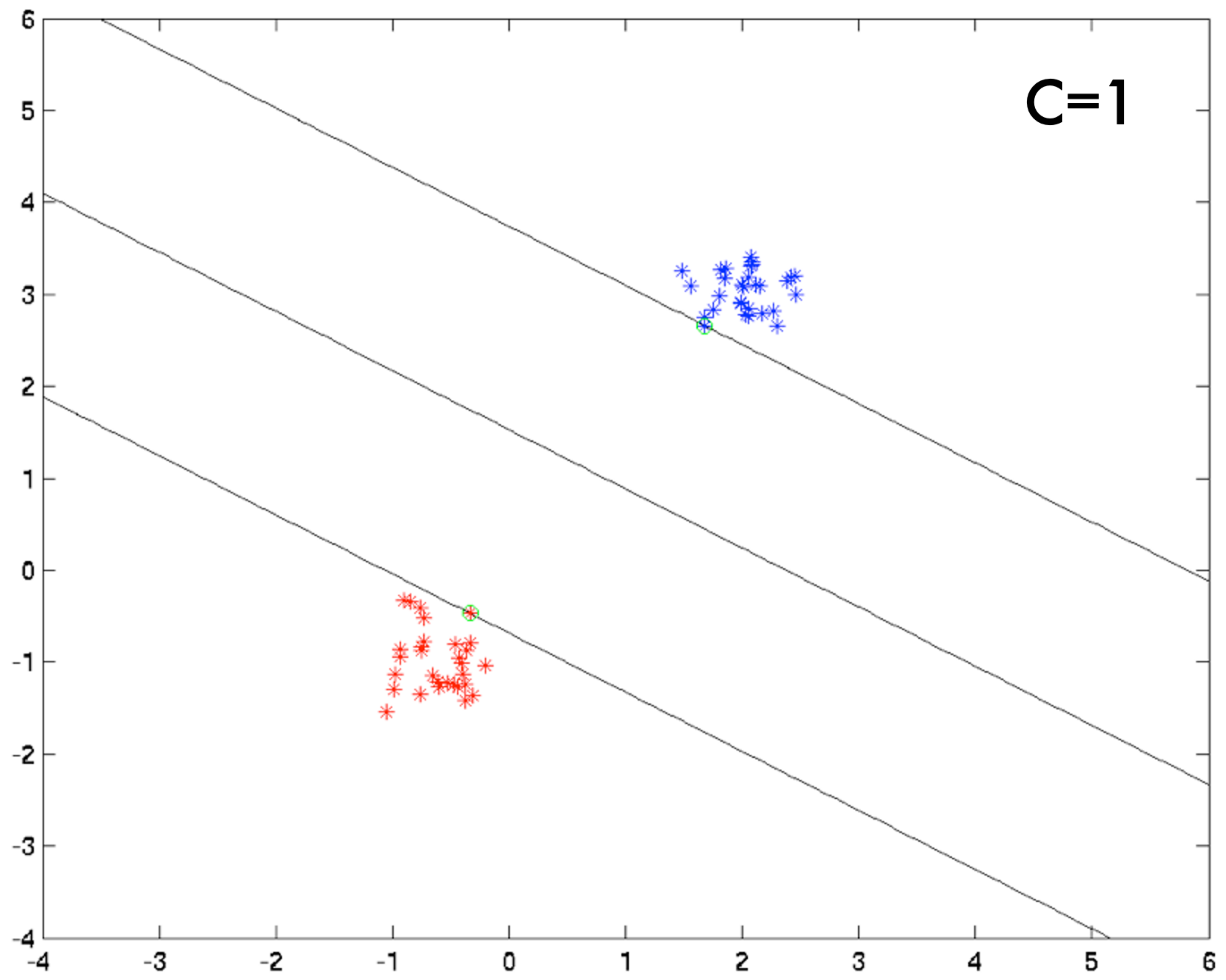
$$= -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j + \sum_i \alpha_i$$

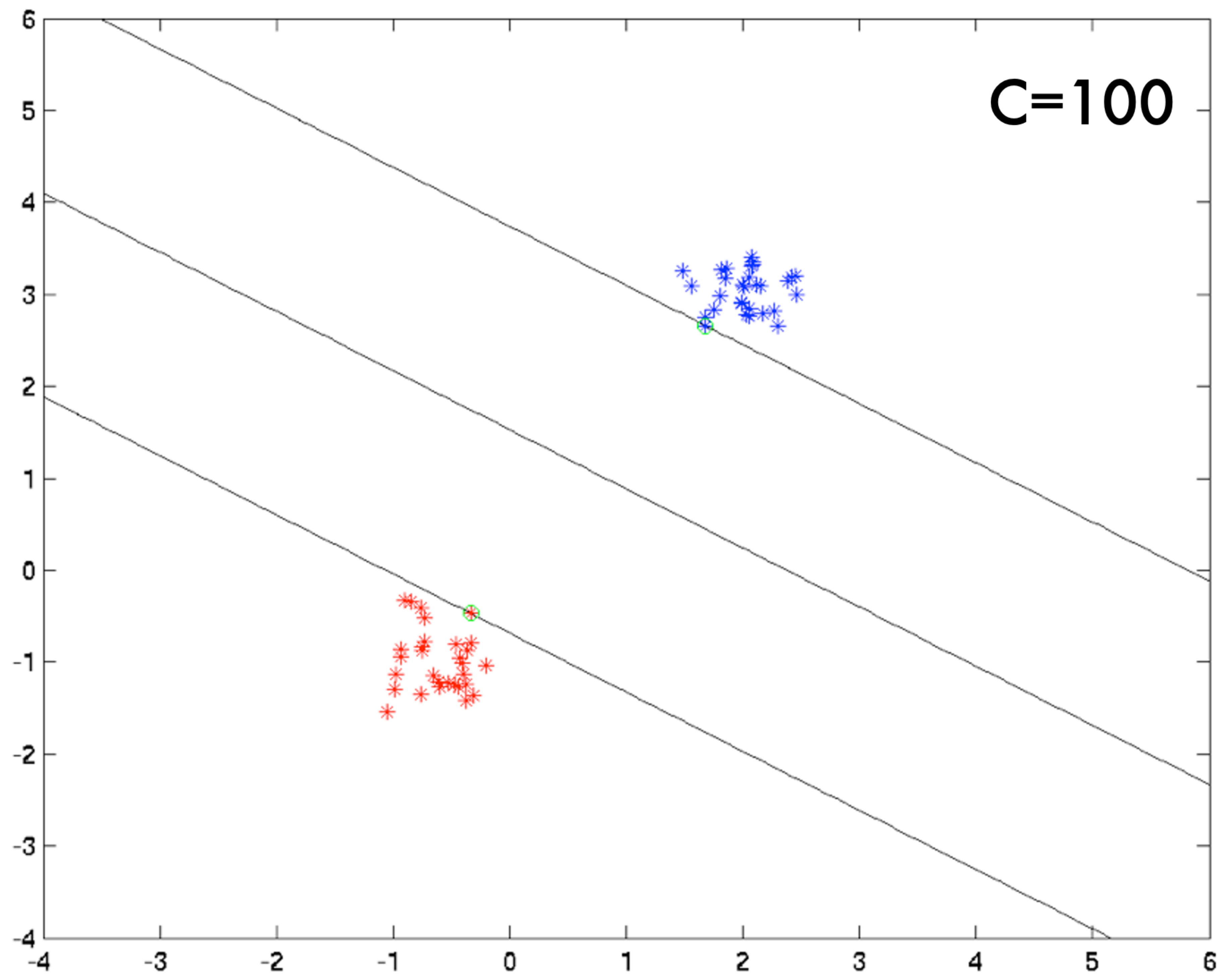
- Summing up, we are solving the optimization

$$\max_{\alpha}\left(-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i\right) \quad \text{subject to} \quad \sum_i \alpha_i y_i = 0 \quad 0 \le \alpha_i \le C$$

Hyperparameter

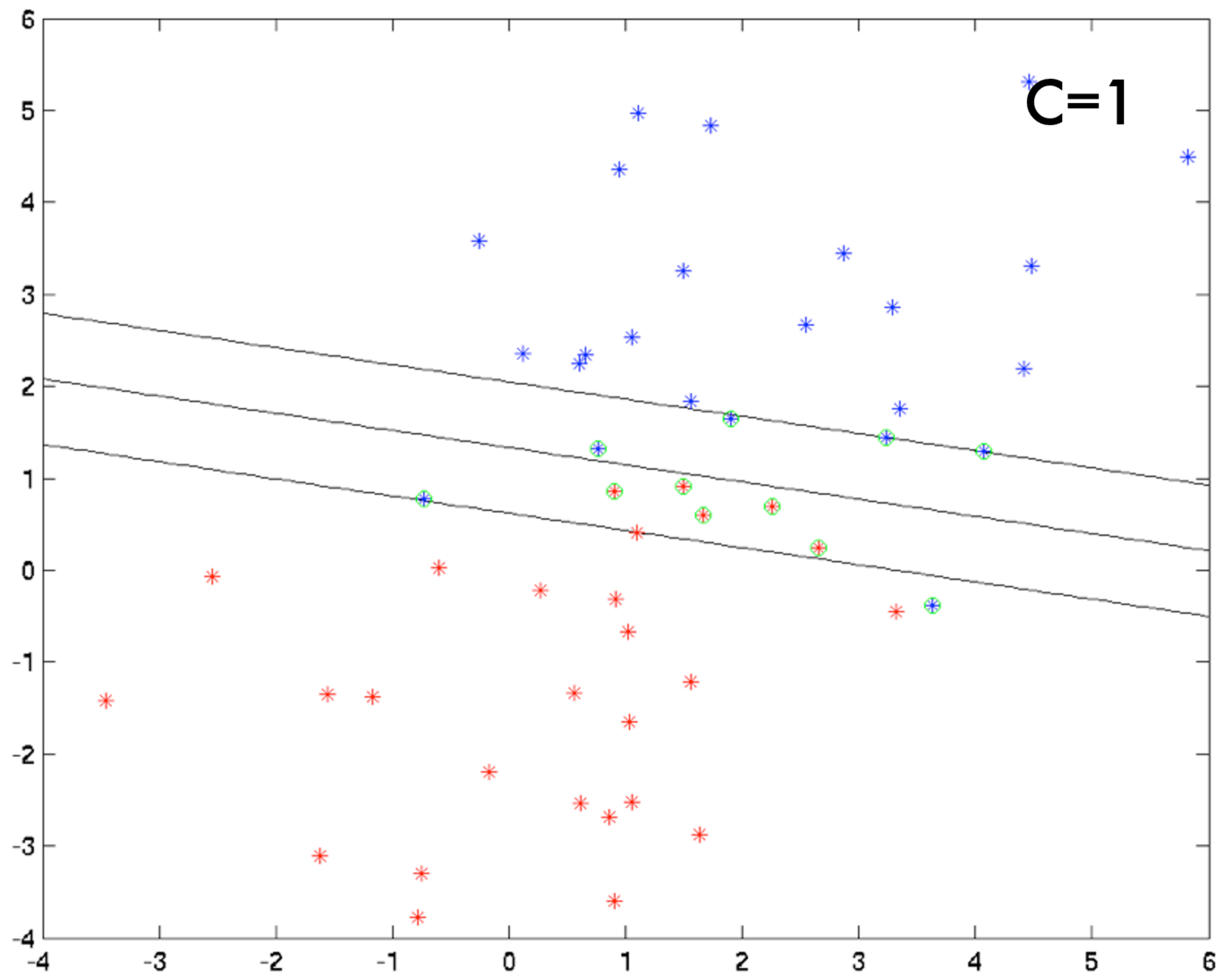
- By increasing the hyperparameter C, we look for a smaller-slack solution
  - No difference when linearly separable…

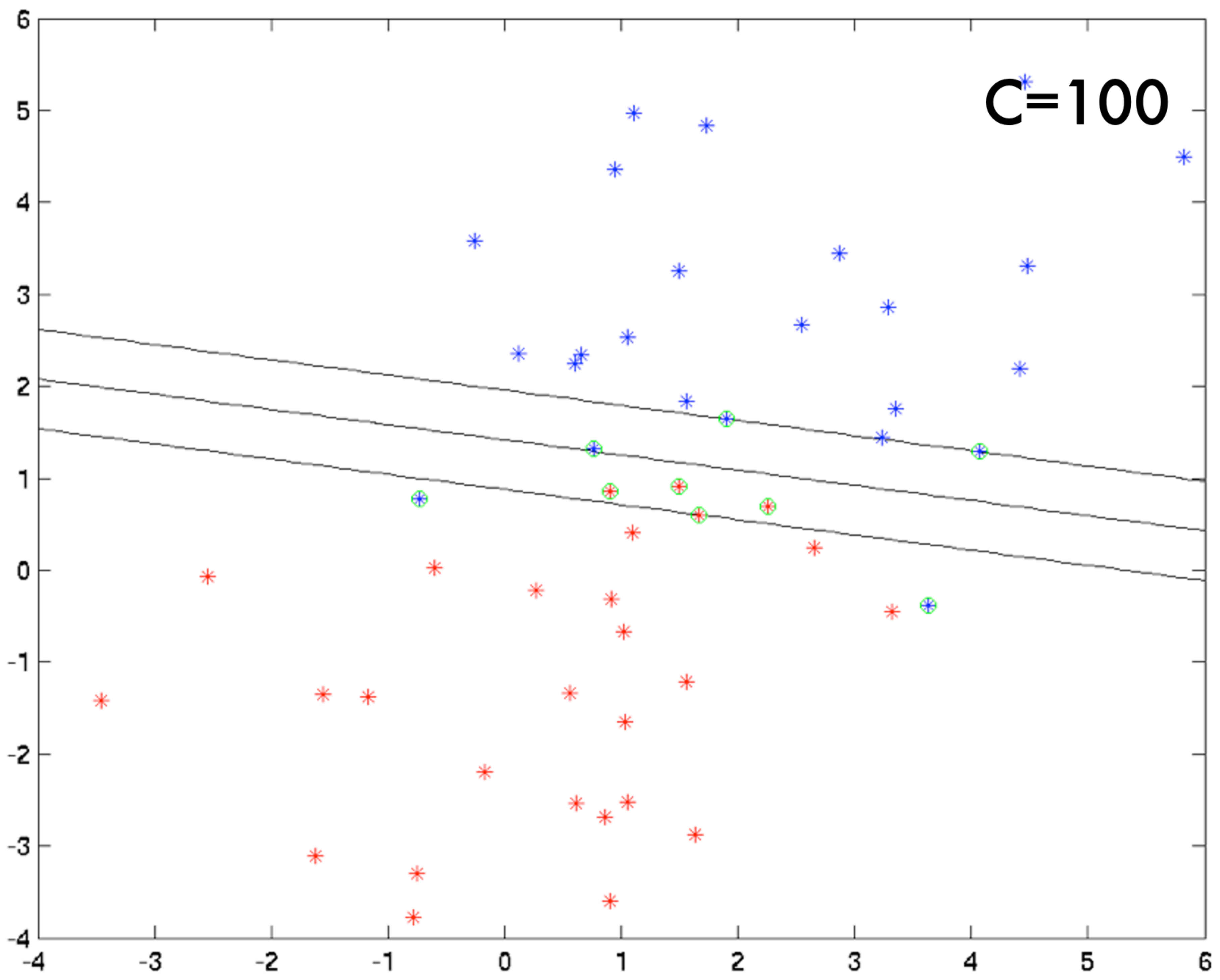




Hyperparameter

- By increasing the hyperparameter C, we look for a smaller-slack solution
  - No difference when linearly separable… but some difference when not





Solving the optimization

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

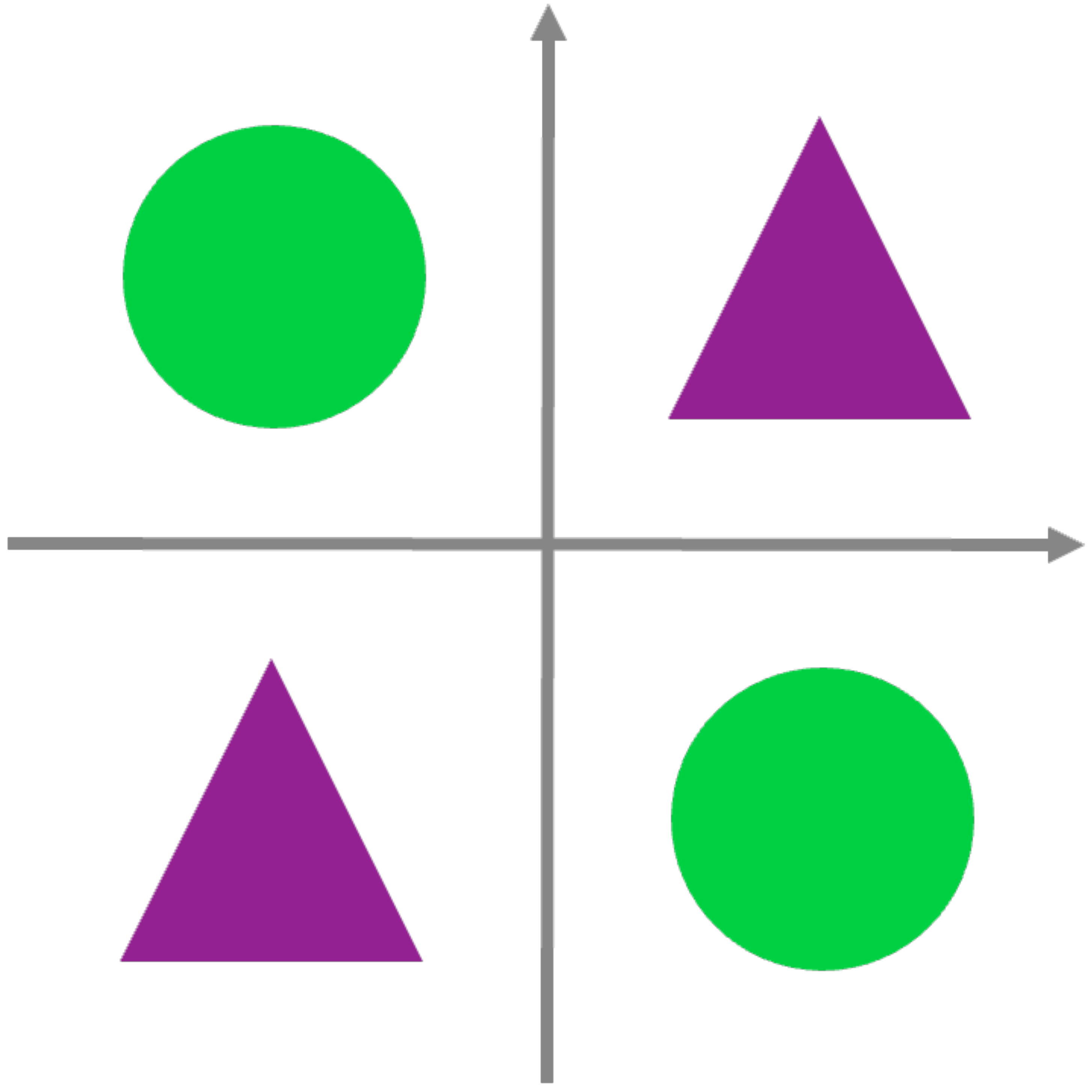
$$\text{subject to} \quad \sum_i \alpha_i y_i = 0 \qquad 0 \le \alpha_i \le C$$

- If the problem is small-scale (e.g., thousands of variables), use off-the-shelf solvers
- If the problem is large-scale, use the fact that only SVs matter, and solve in blocks
    - called “active set method”

Kernel SVM

Nonlinear data

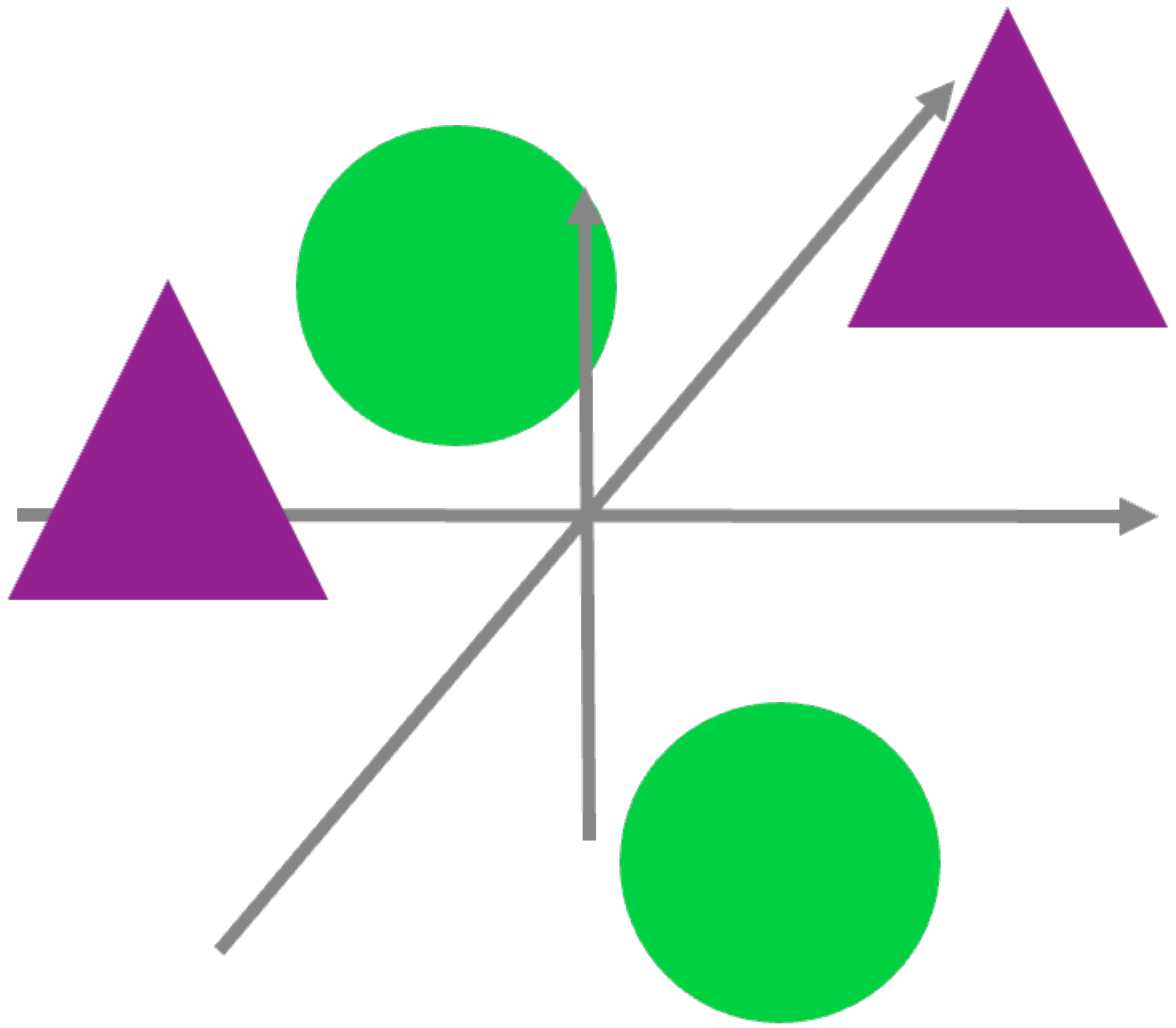
- Suppose that we have a data that looks like **XOR**
  - Not linearly separable
  - Thus no satisfactory linear classifier exists
- **Q.** How to handle these data?



(x_1, x_2)

Nonlinear data

- Suppose that we have a data that looks like **XOR**
  - Not linearly separable
  - Thus no satisfactory linear classifier exists
- **Q.** How to handle these data?
  - A. Map it to a high-dimensional space
    - There exists a clean linear classifier!



$(x_1, x_2, x_1 x_2)$

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

More formally...

- We map the data to a high-dimensional **feature** $\Phi(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^k$
  - Typically, $d < k$ (but not necessarily)

More formally...

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

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

  - This is quite similar to original SVMs, where

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

Choosing the feature

- **Question.** How should we choose $\Phi(\,\cdot\,)$?

Choosing the feature

- **Question.** How should we choose $\Phi(\,\cdot\,)$?

  - Naïve way. Simply throw in many features, and let SVM choose

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

Choosing the feature

- **Question.** How should we choose $\Phi(\,\cdot\,)$?
  - Naïve way. Simply throw in many features, and let SVM choose

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

- This is **bad**!
  - overfitting
  - computation
    - computing features
    - computing inner products

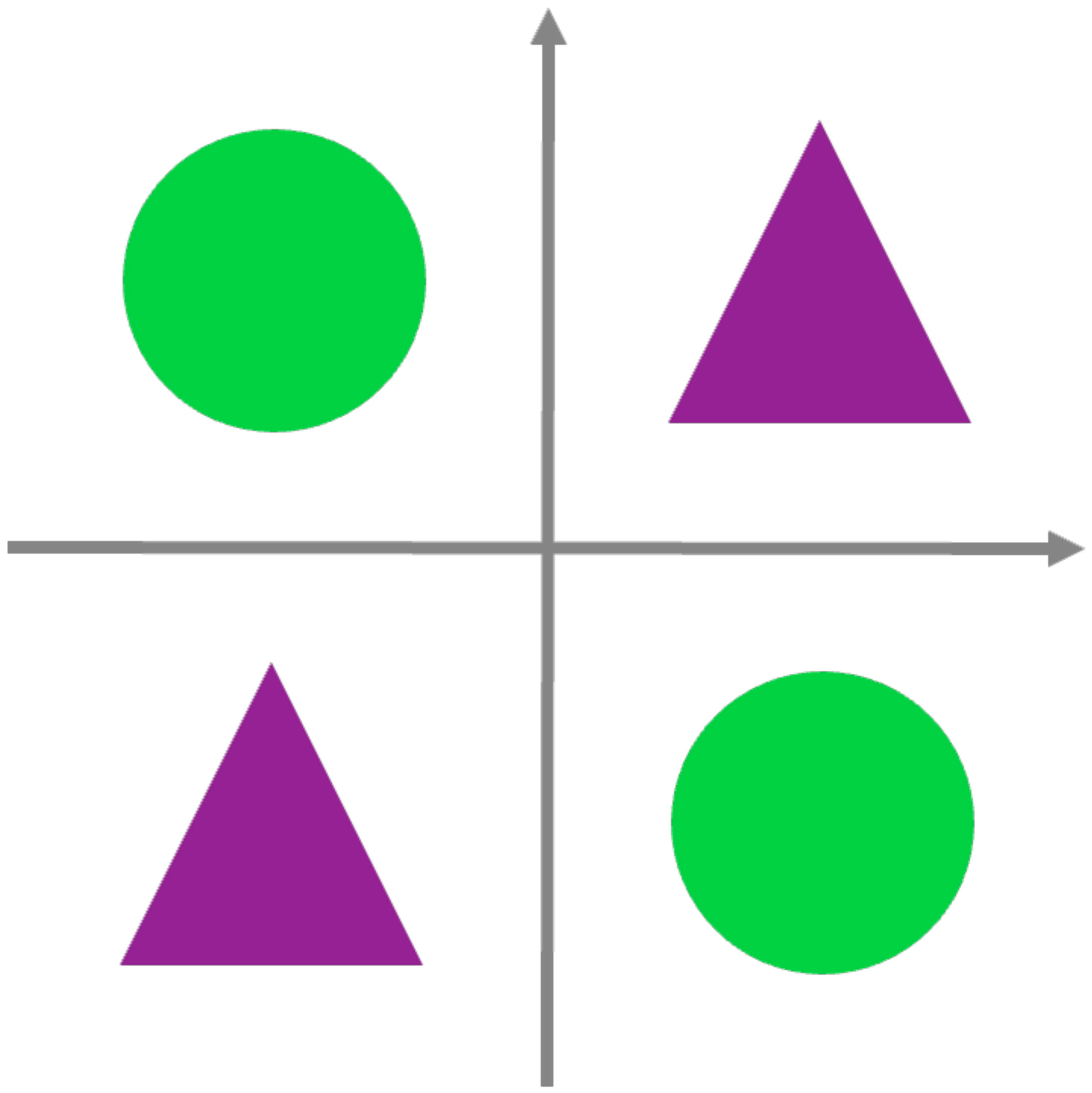
Choosing the feature

- Interestingly, some features admit **computational shortcuts**

Choosing the feature

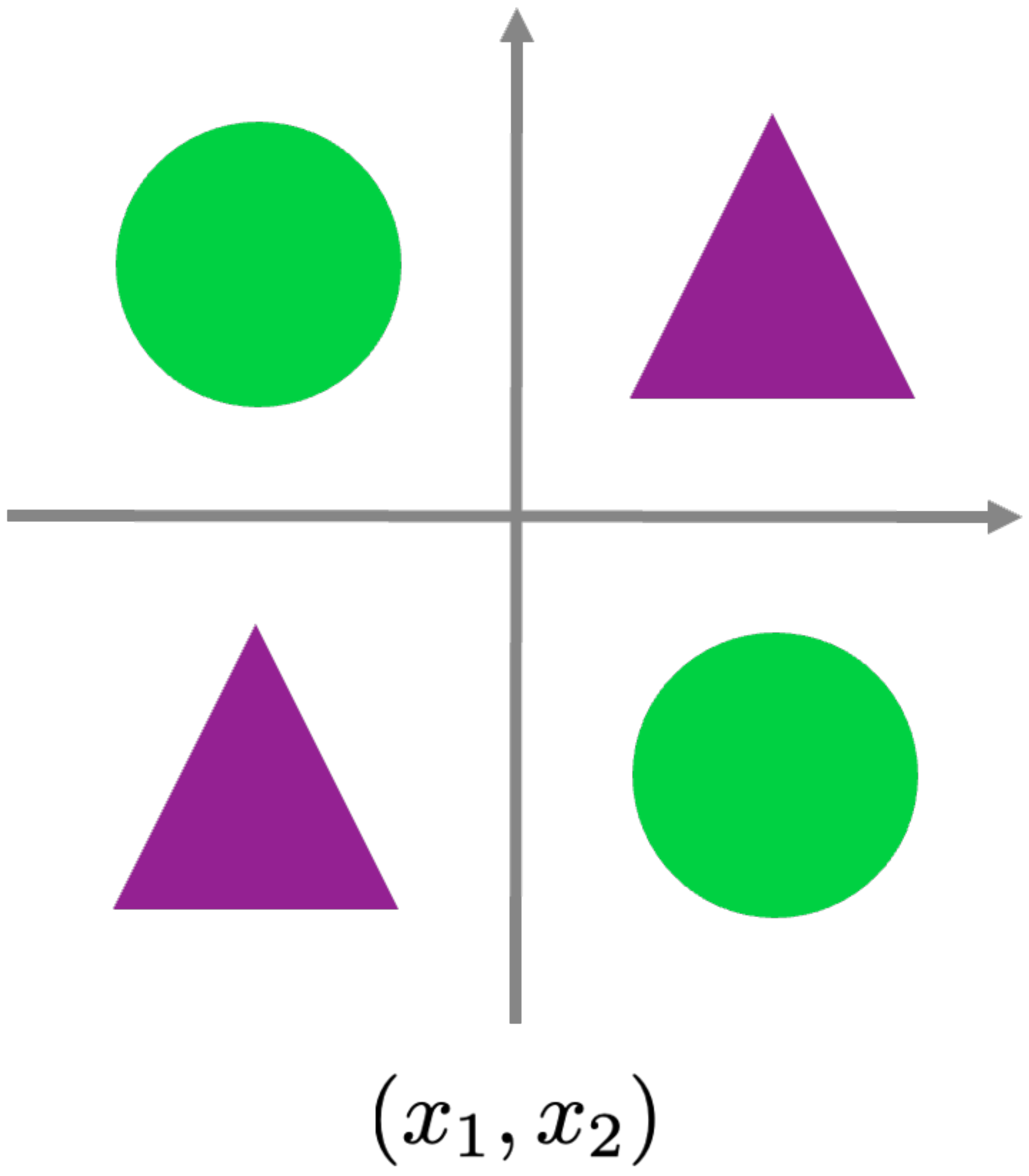
- Interestingly, some features admit **computational shortcuts**

- Example. Recall the **XOR**, 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
- **Question.** So which one is better?



Choosing the feature

- Interestingly, some features admit **computational shortcuts**
  - Example. Recall the **XOR**, and think of two features.
    - $\Phi_a((x_1, x_2)) = (x_1, x_2, x_1x_2)$
    - $\Phi_b((x_1, x_2)) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$
  - Looks similar, but one is better than the other
  - **Question.** So which one is better?
    - Answer. Φ_b, for computational reasons



Choosing the feature

- Compare the computations:
  - $\langle \Phi_a(\mathbf{x}), \Phi_a(\mathbf{y}) \rangle = x_1y_1 + x_2y_2 + x_1x_2y_1y_2$
    - Compute 3D features $\phi_{\mathbf{x}} = \Phi_a(\mathbf{x})$, $\phi_{\mathbf{y}} = \Phi_a(\mathbf{y})$
    - Compute 3D inner prod $\langle \phi_{\mathbf{x}}, \phi_{\mathbf{y}} \rangle$

Choosing the feature

- Compare the computations:
  - $\langle \Phi_a(\mathbf{x}), \Phi_a(\mathbf{y}) \rangle = x_1y_1 + x_2y_2 + x_1x_2y_1y_2$
    - Compute 3D features $\phi_{\mathbf{x}} = \Phi_a(\mathbf{x})$, $\phi_{\mathbf{y}} = \Phi_a(\mathbf{y})$
    - Compute 3D inner prod $\langle \phi_{\mathbf{x}}, \phi_{\mathbf{y}} \rangle$
  - $\langle \Phi_b(\mathbf{x}), \Phi_b(\mathbf{y}) \rangle = x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 = (\langle \mathbf{x}, \mathbf{y} \rangle)^2$
    - Compute 2D inner prod $\langle \mathbf{x}, \mathbf{y} \rangle$
    - Take a square
      - Less memory & computation

Kernel SVM

- **Idea.** Follow these steps.
  - Choose an easy-to-compute **similarity metric** $K(\cdot\,,\cdot)$
  - Construct predictors of form

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

  - Fit a_i, b

Kernel SVM

- **Idea.** Follow these steps.
  - Choose an easy-to-compute **similarity metric** $K(\cdot\,,\cdot)$
  - Construct predictors of form

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

  - Fit a_i, b
- **Question.** Is this equivalent to doing SVM with features?
  (i.e., does there always exist a $\mathbf{\Phi}$ corresponding to K?)

Kernel SVM

- **Idea.** Follow these steps.
  - Choose an easy-to-compute **similarity metric** $K(\cdot, \cdot)$
  - Construct predictors of form

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

  - Fit a_i, b
- **Question.** Is this equivalent to doing SVM with features?
  (i.e., does there always exist a Φ corresponding to K?)
  - Answer. Yes if K is a **Mercer kernel**

Kernel SVM

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

Kernel SVM

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

Optimizing Kernel SVM

- In kernel SVM, we solve

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

  - Plug in $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$ to recover the original SVM

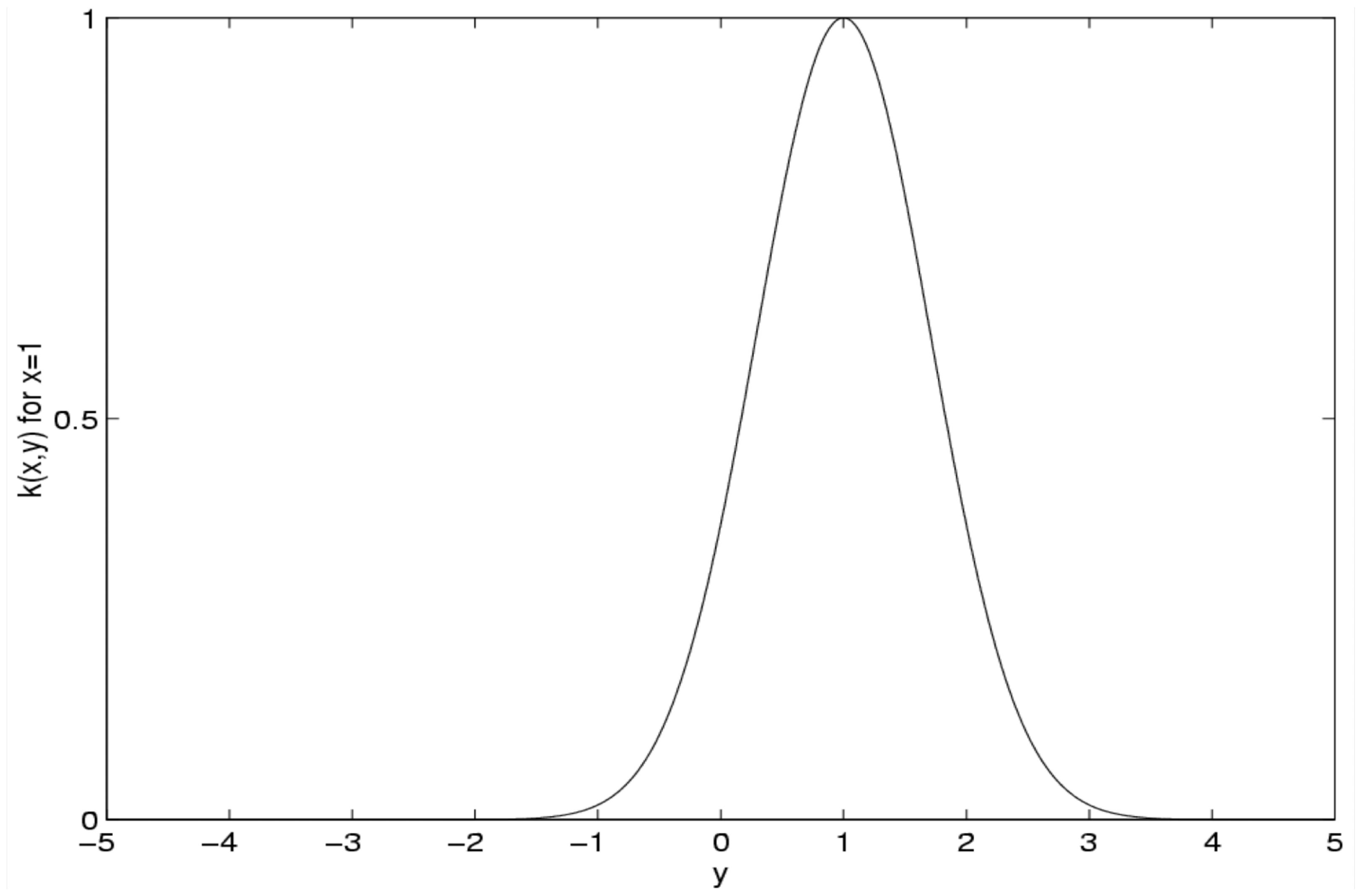
Optimizing Kernel SVM

- In kernel SVM, we solve

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

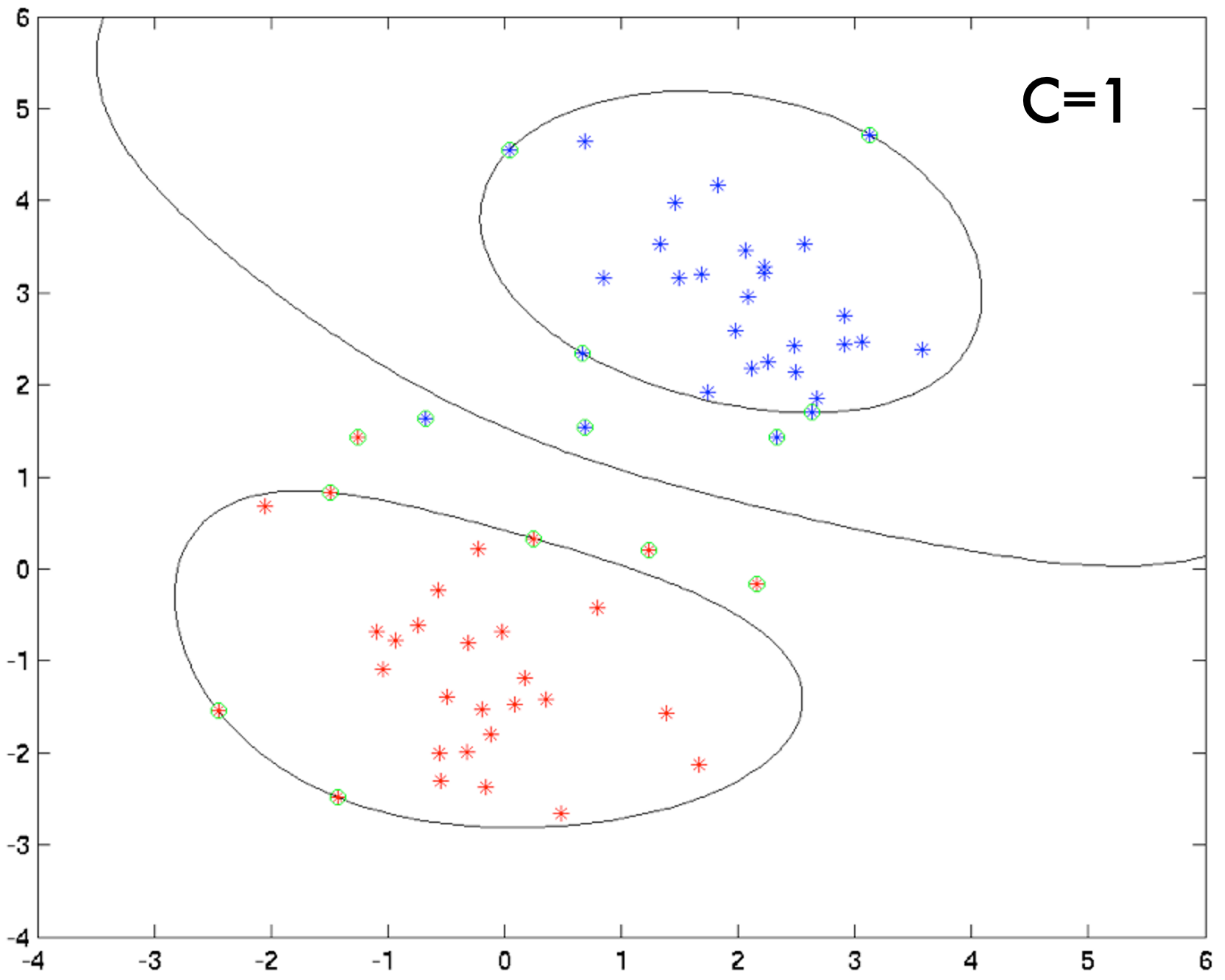
- Plug in $K(\mathbf{x},\mathbf{y})=\mathbf{x}^{\top}\mathbf{y}$ to recover the original SVM

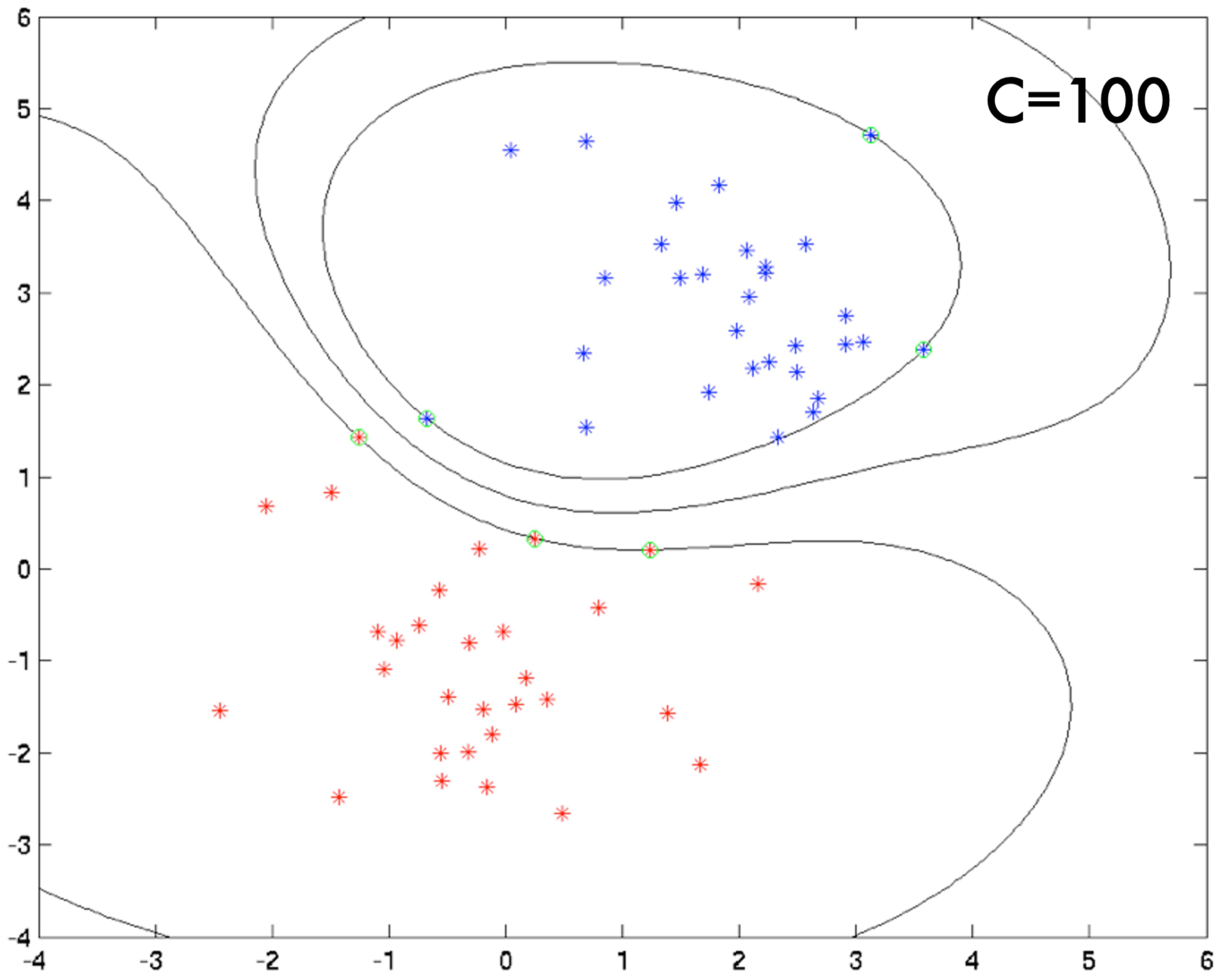
- Other choices
  - Laplacian RBF $\exp(-\lambda\|\mathbf{x}-\mathbf{x}'\|_2)$
  - Gaussian RBF $\exp(-\lambda\|\mathbf{x}-\mathbf{x}'\|_2^2)$ —>
  - Polynomial $(\langle\mathbf{x},\mathbf{x}'\rangle+c)^d$
  - B-Spline (look it up)



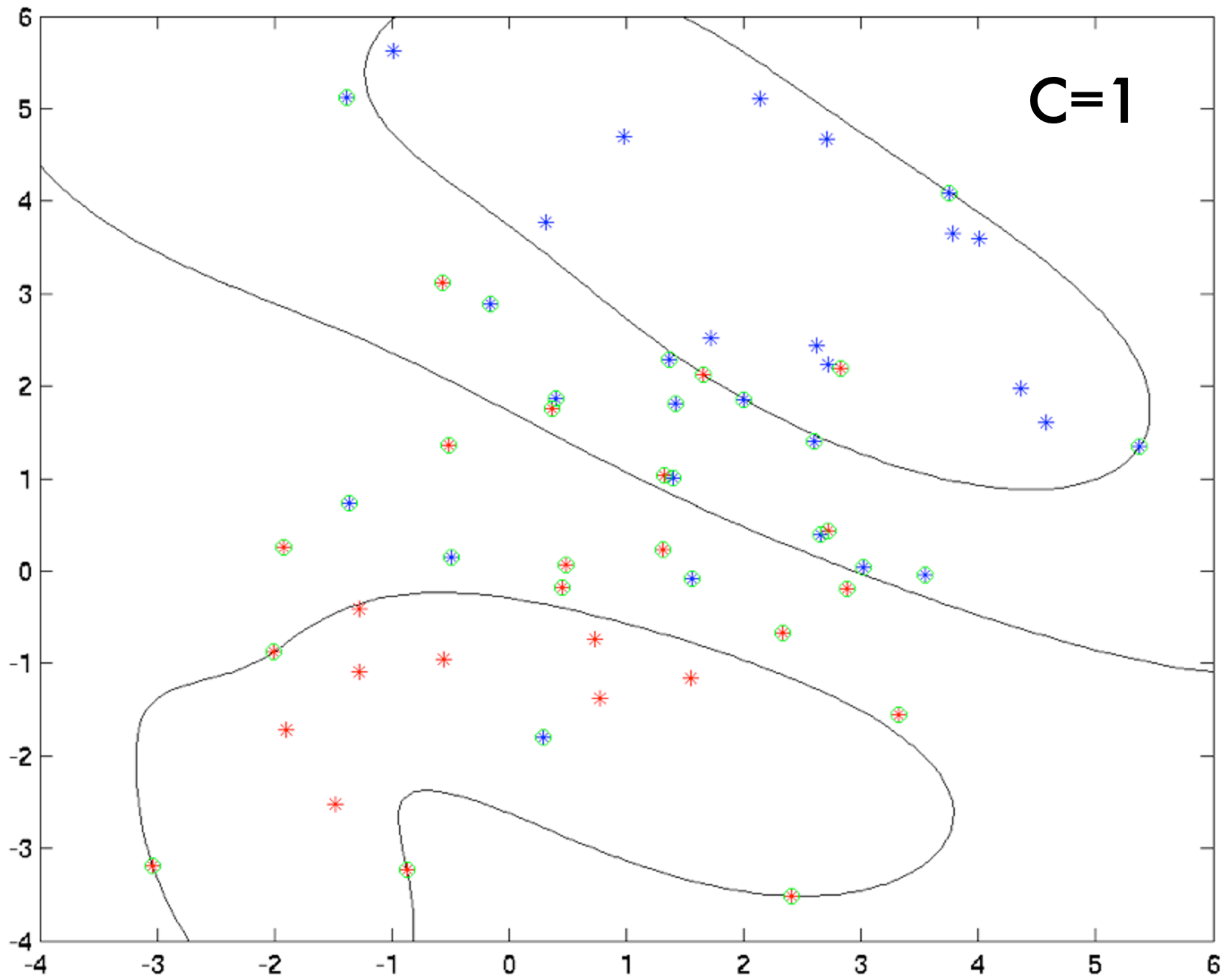
Tuning Kernel SVM

- Again, we can tune hyperparameters to play with the margin

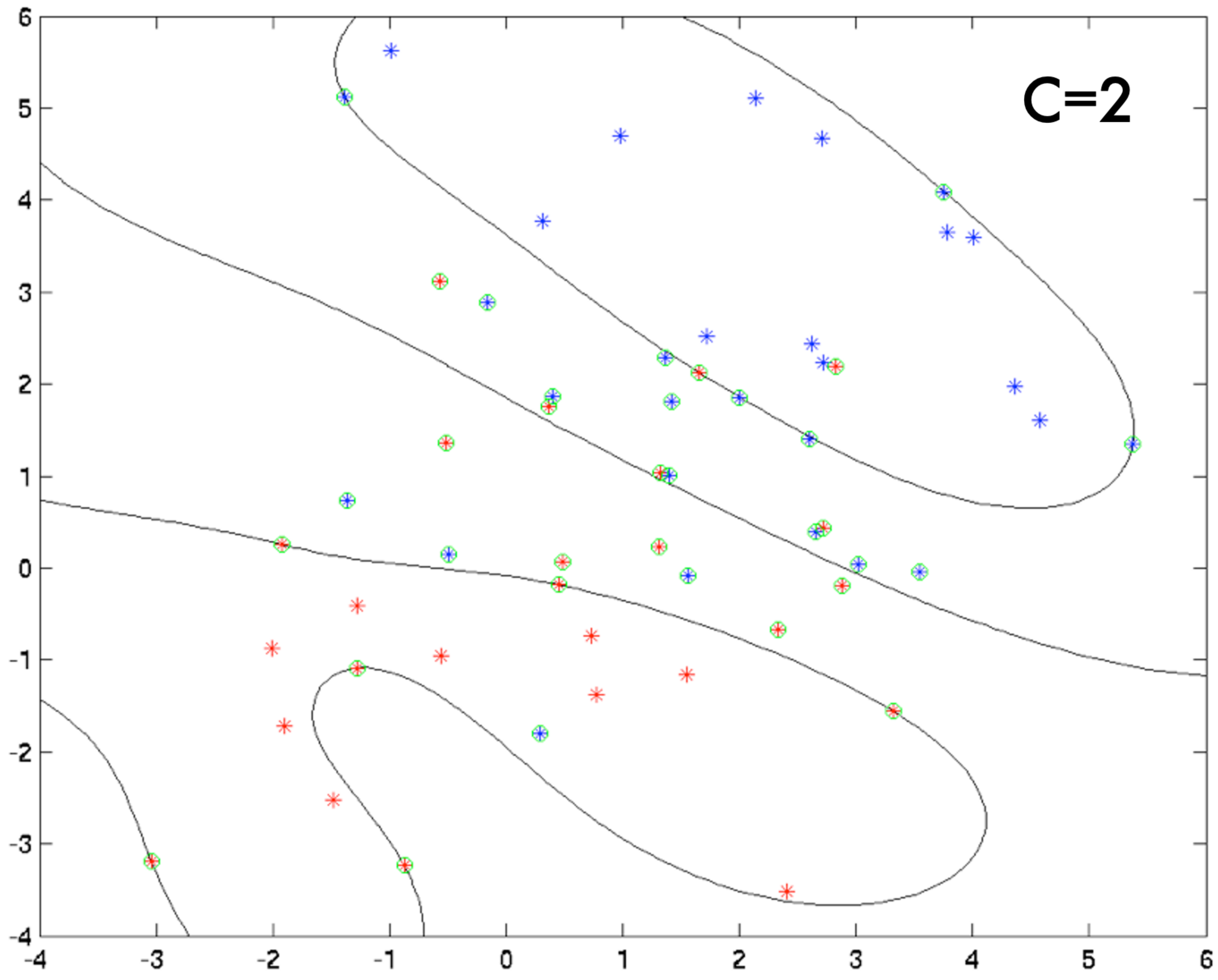




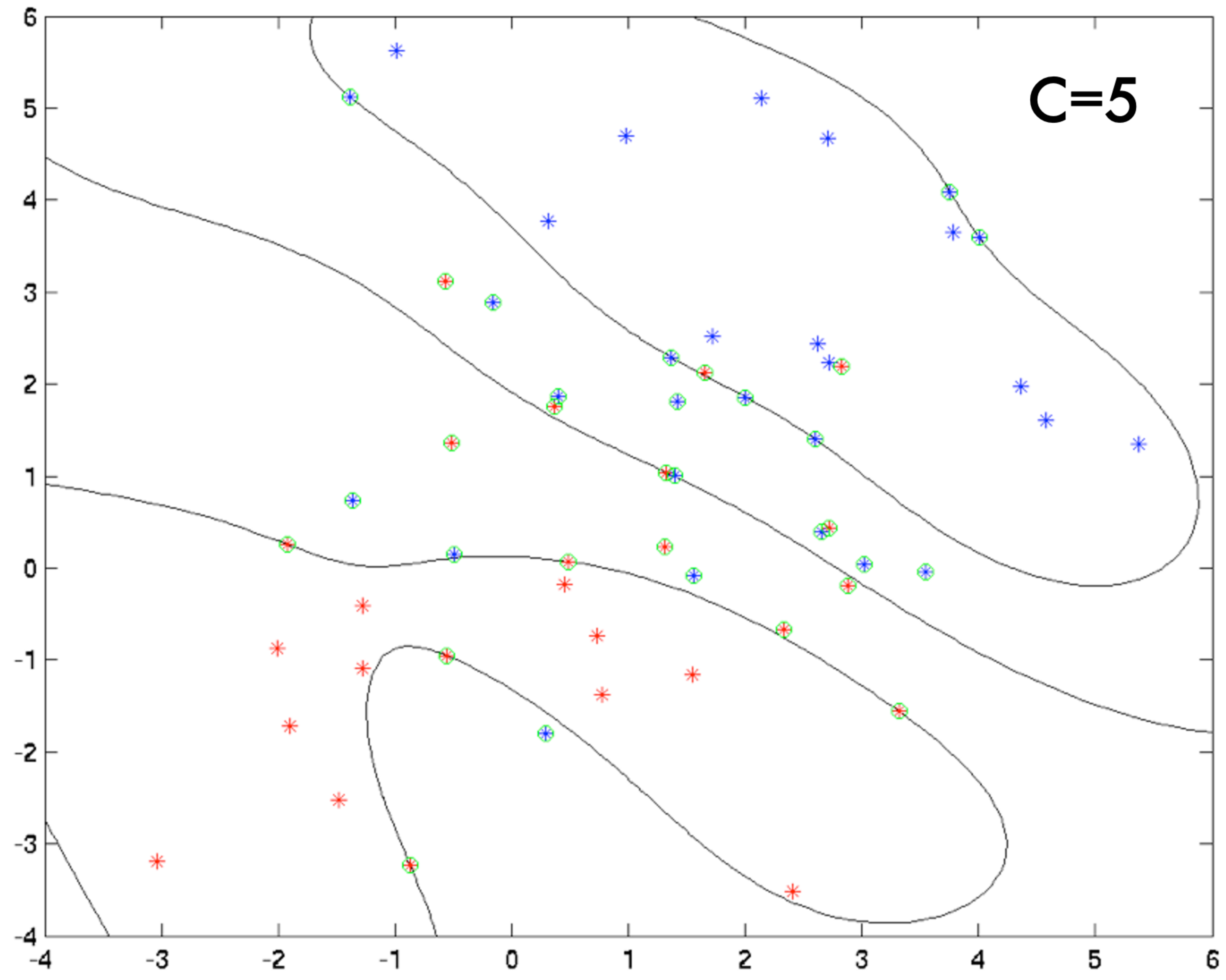
Tuning Kernel SVM: Outliers



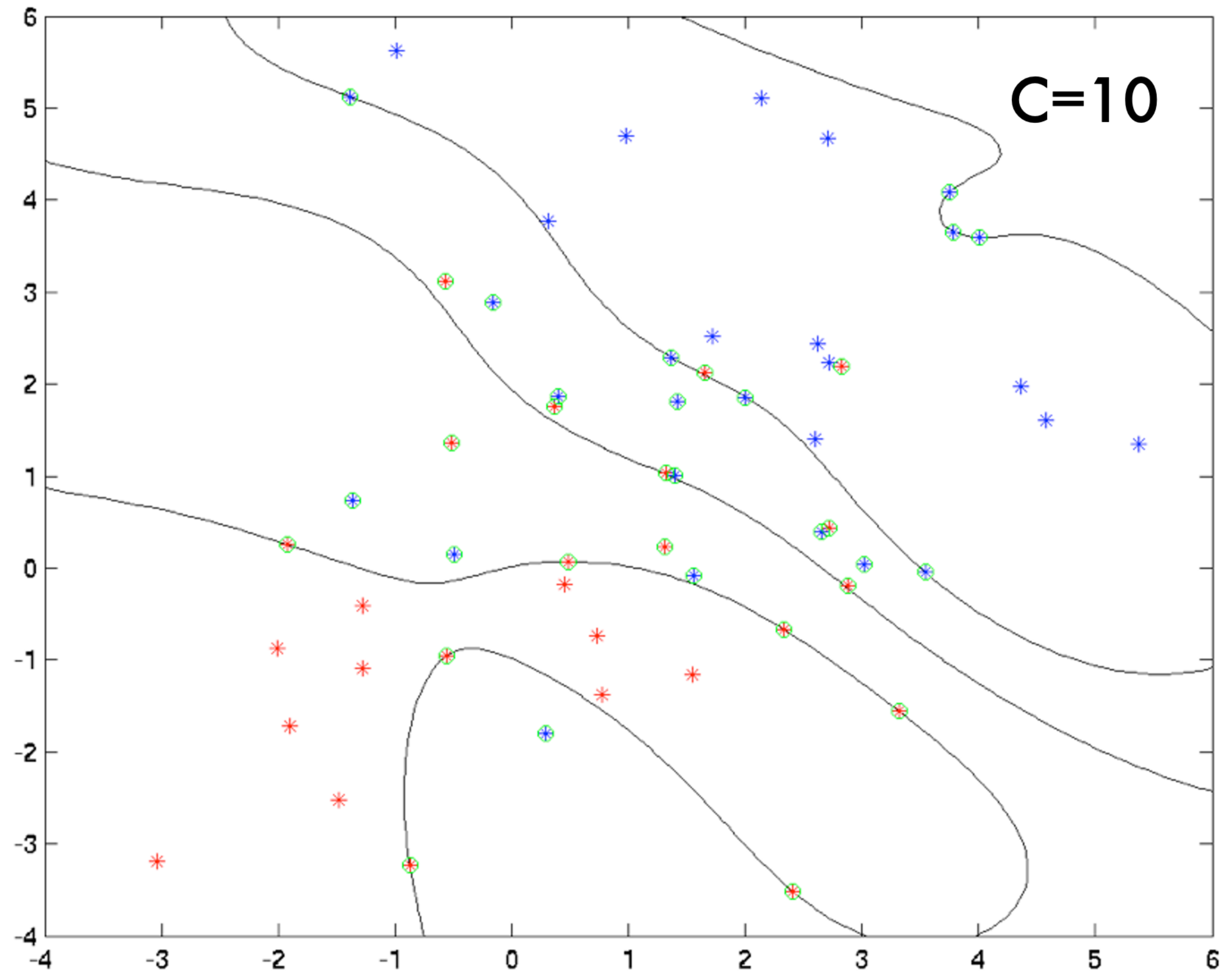
Tuning Kernel SVM: Outliers



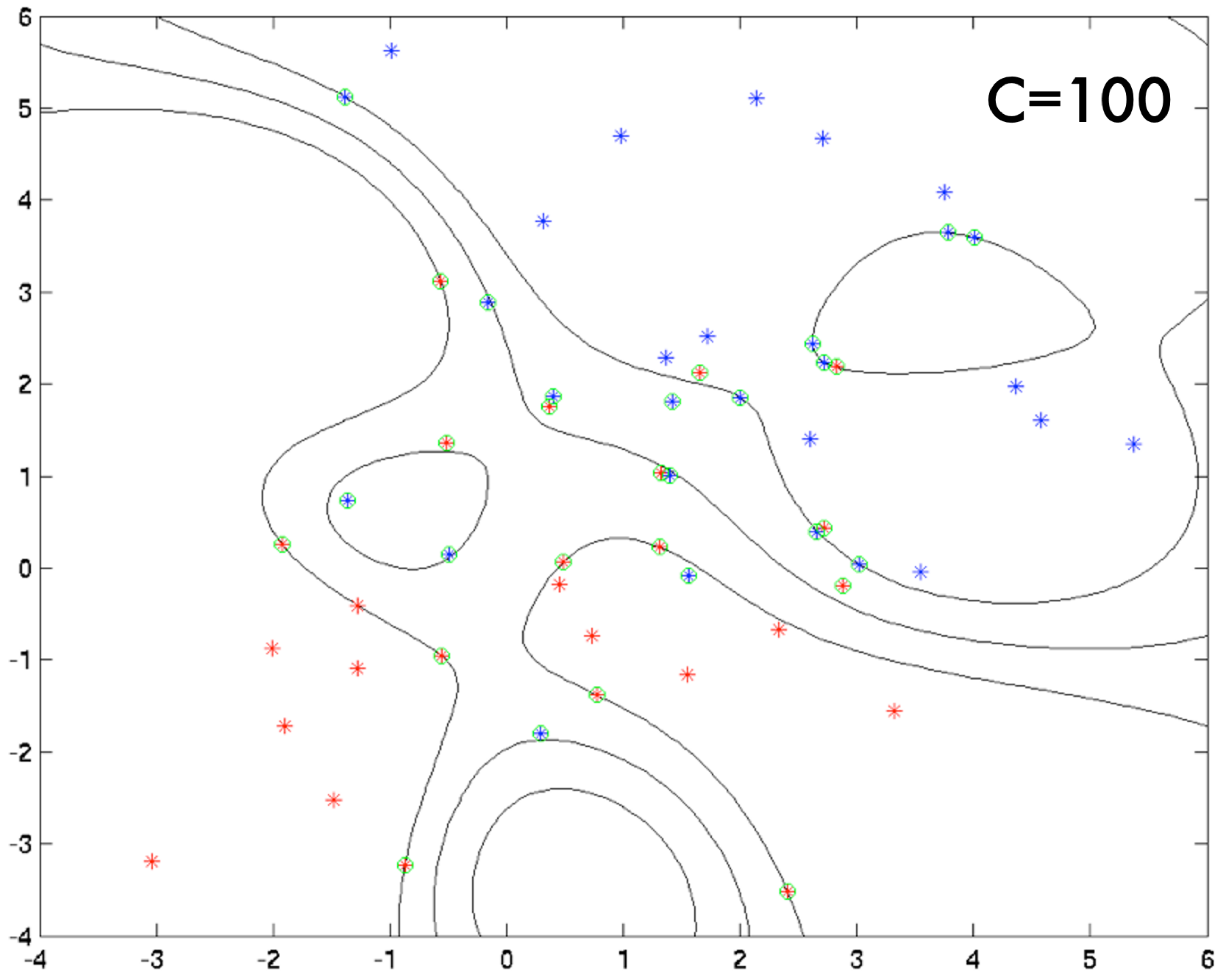
Tuning Kernel SVM: Outliers



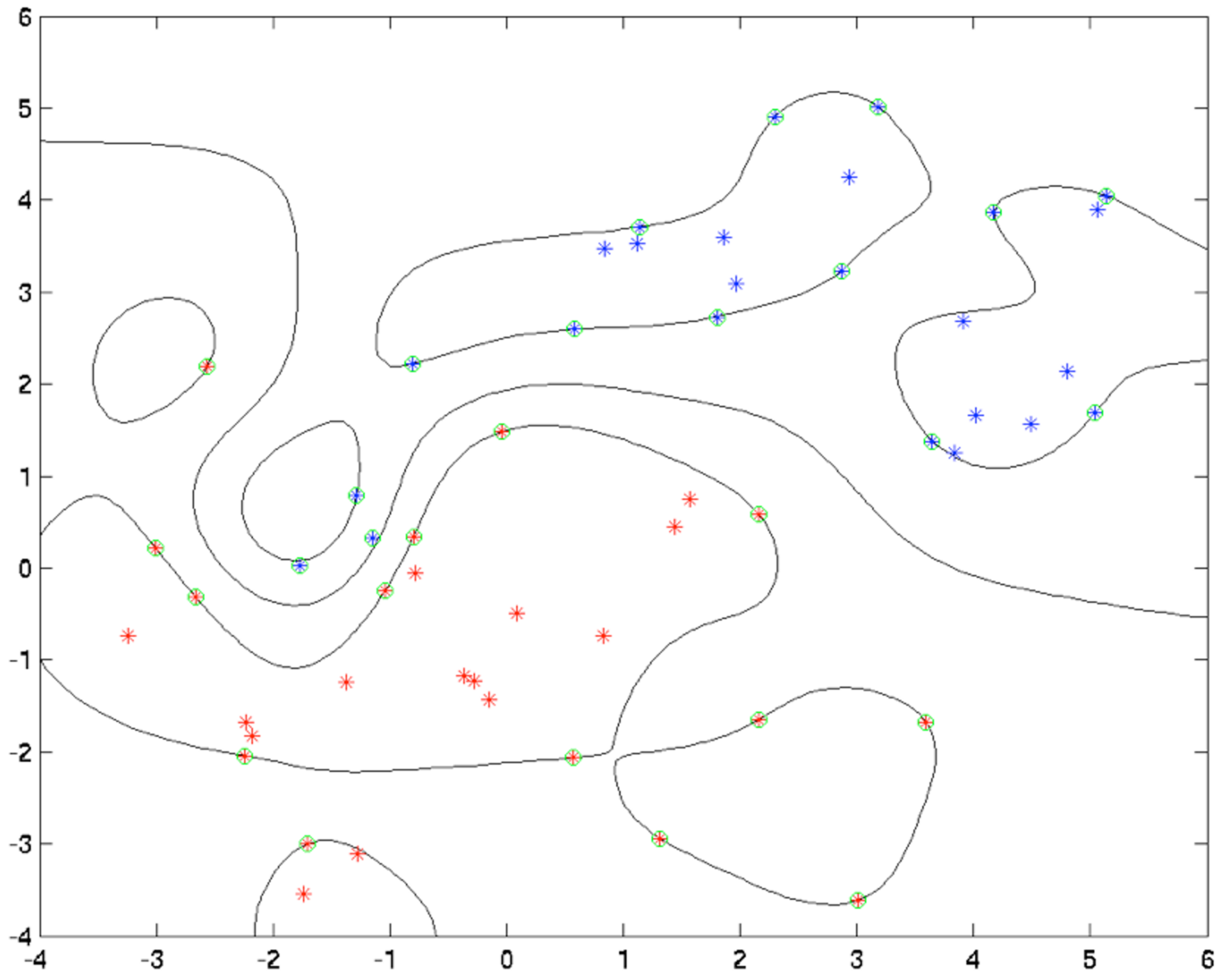
Tuning Kernel SVM: Outliers



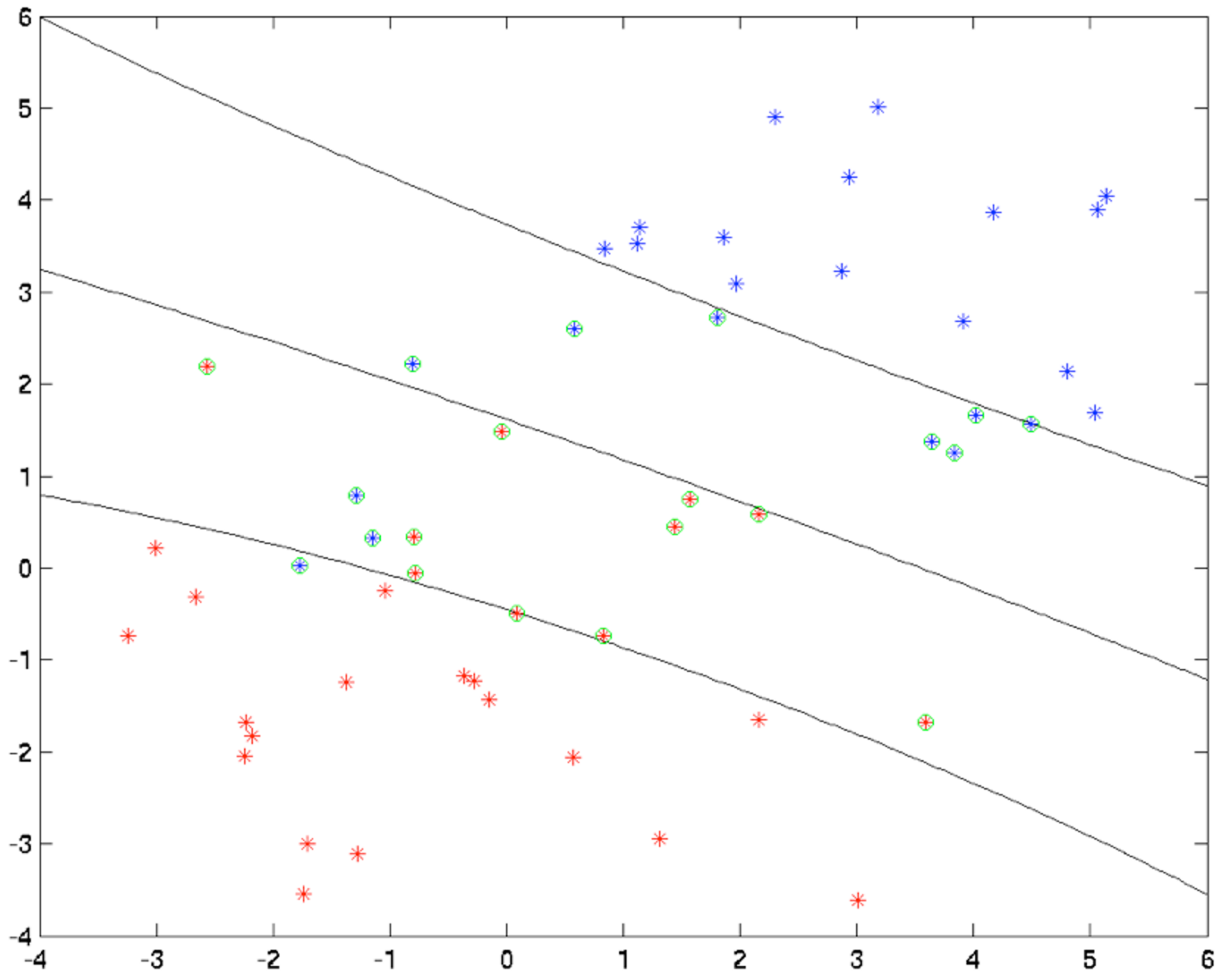
Tuning Kernel SVM: Outliers



Tuning Kernel SVM: Narrow Kernels



Tuning Kernel SVM: Wide Kernels



In deep learning era…

- In modern ML, we find a nice $\mathbf{\Phi}(\,\cdot\,)$ using data + neural nets
  - Expensive, but we can afford them
  - Conduct logistic regression, instead of SVD
    - Ease of joint training
    - Also margin-maximizer (sometimes)
  - Use nice augmentations to find good similarity metric such that
    - $\mathbf{\Phi}(\mathbf{x}) - \mathbf{\Phi}(\mathbf{x}_{\text{aug}})$ is smaller than $\mathbf{\Phi}(\mathbf{x}) - \mathbf{\Phi}(\mathbf{x}')$

Next up

- K-Means

Cheers