4.1 Perceptron

4 (Generalized) Linear Methods

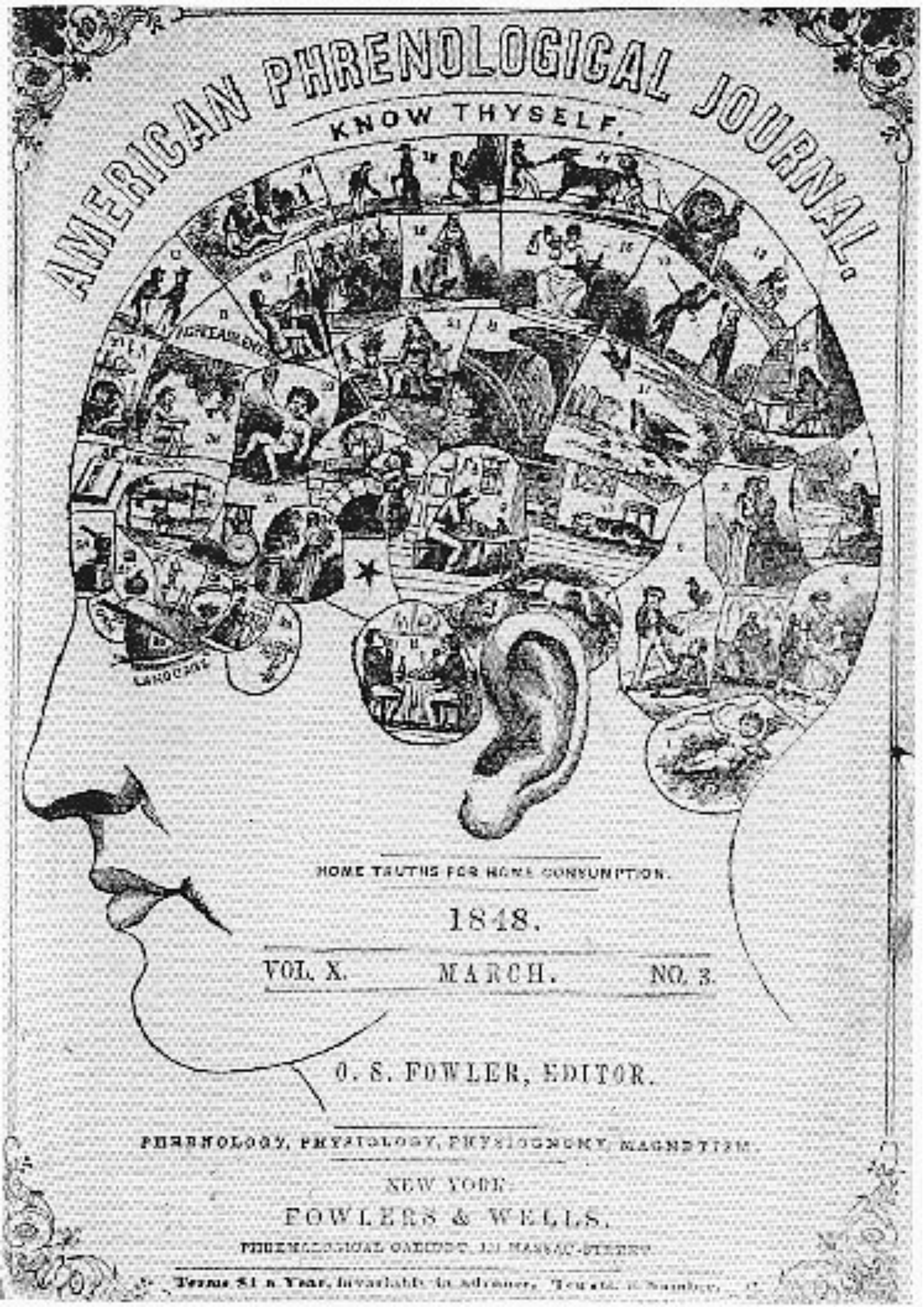
Alexander Smola

Introduction to Machine Learning 10-701

http://alex.smola.org/teaching/10-701-15

MACHINE LEARNING DEPARTMENT

Carnegie Mellon University



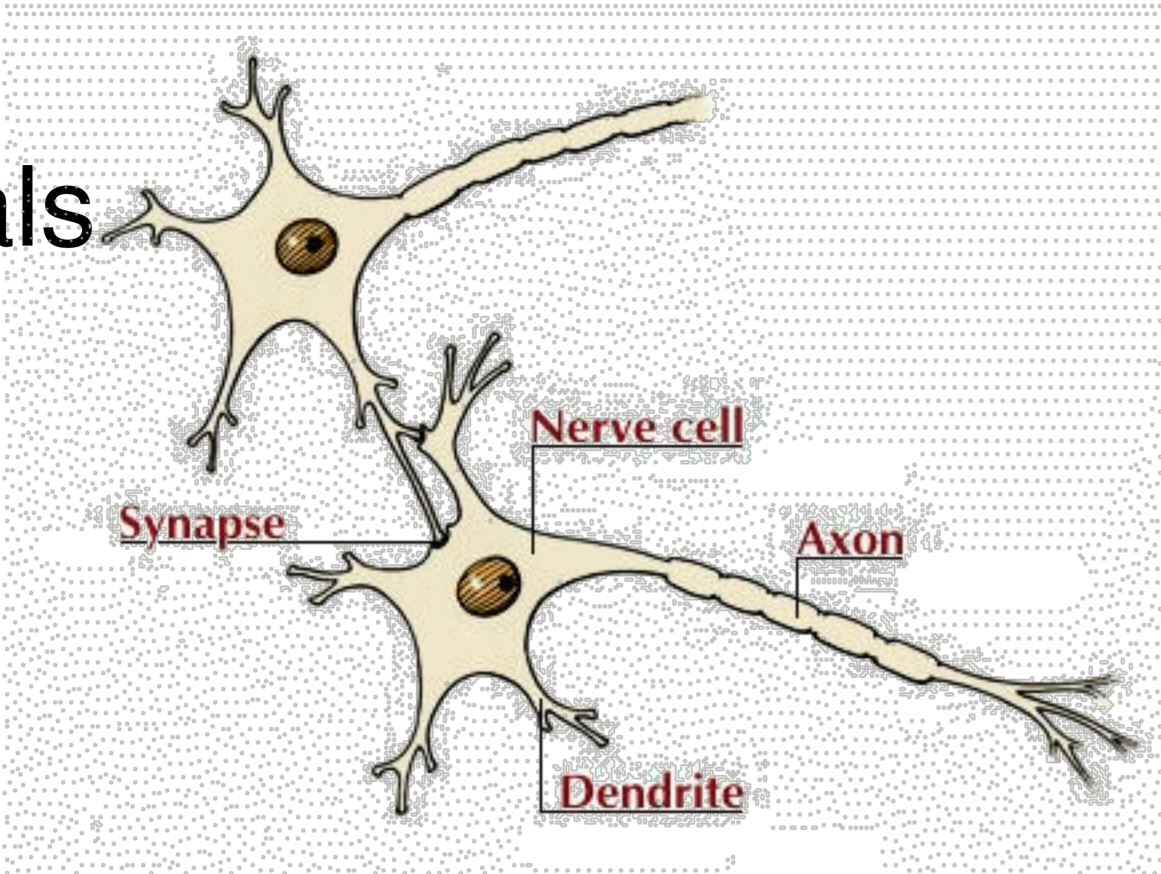
Neurons and Learning

Biology and Learning

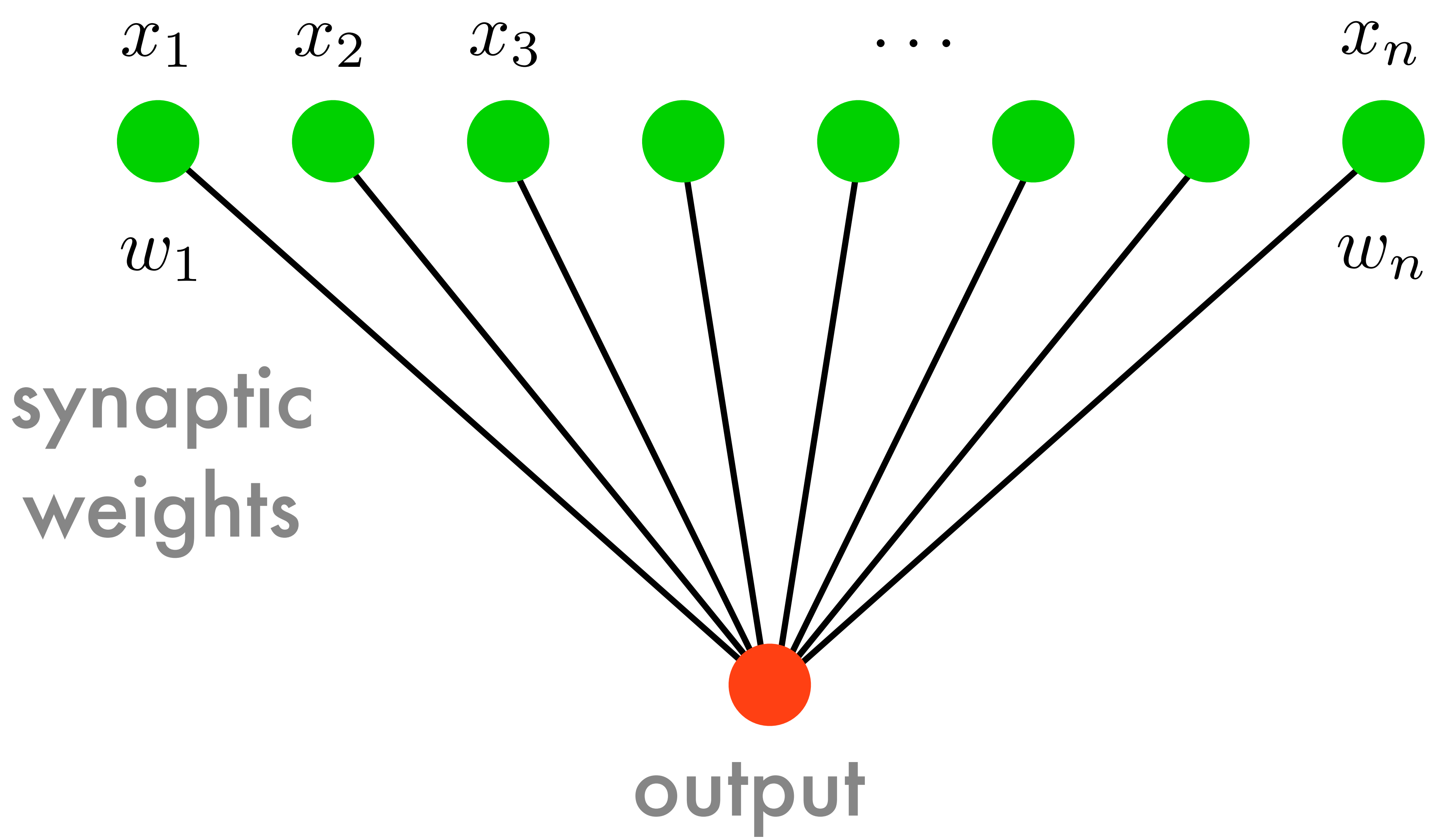
- Basic Idea
  - Good behavior should be rewarded, bad behavior punished (or not rewarded). This improves system fitness.
  - Killing a sabertooth tiger should be rewarded ...
  - Correlated events should be combined.
  - Pavlov’s salivating dog.
- Training mechanisms
  - Behavioral modification of individuals (learning)
    Successful behavior is rewarded (e.g. food).
  - Hard-coded behavior in the genes (instinct)
    The wrongly coded animal does not reproduce.

Neurons

- Soma (CPU)
  Cell body - combines signals
- Dendrite (input bus)
  Combines the inputs from several other nerve cells
- Synapse (interface)
  Interface and parameter store between neurons
- Axon (cable)
  May be up to 1m long and will transport the activation signal to neurons at different locations



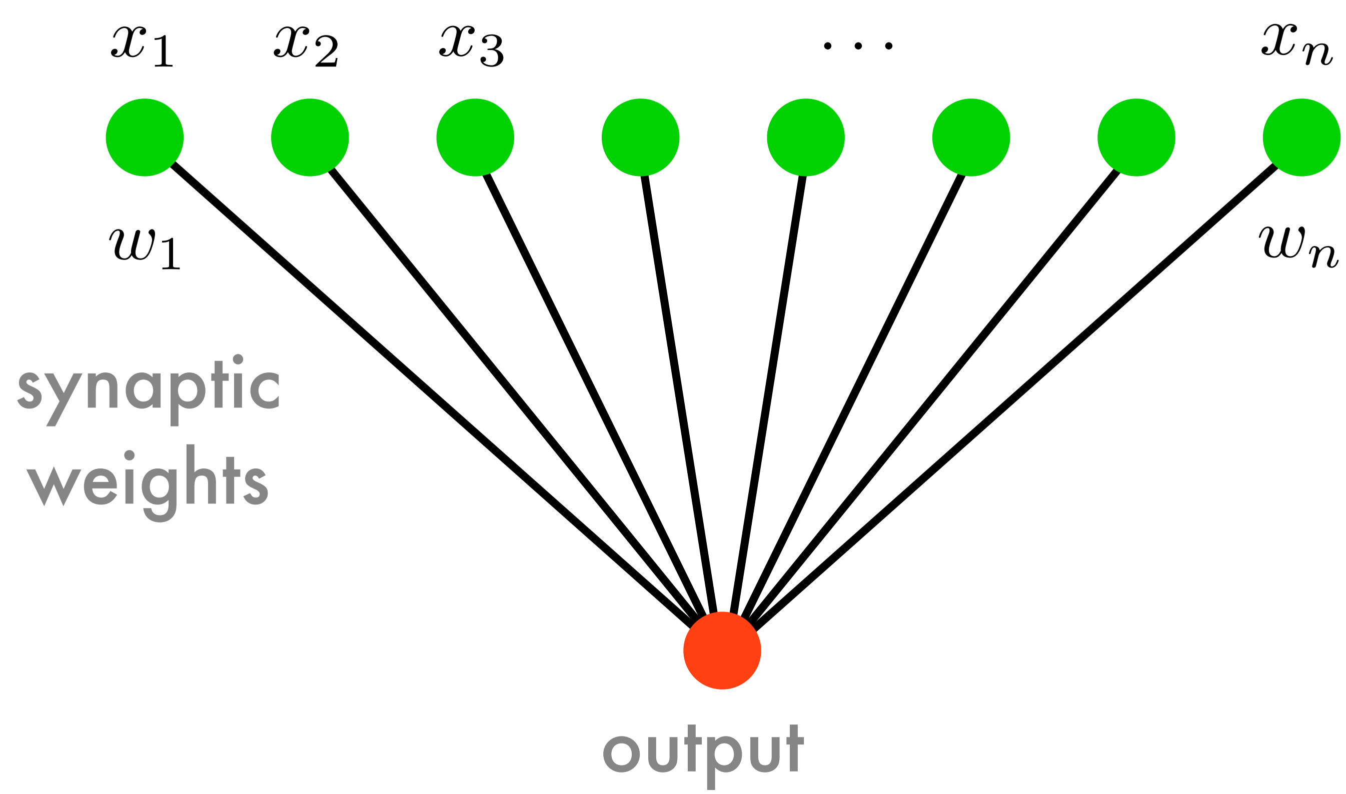
Neurons



$$f(x) = \sum_i w_i x_i = \langle w, x \rangle$$

Perceptron

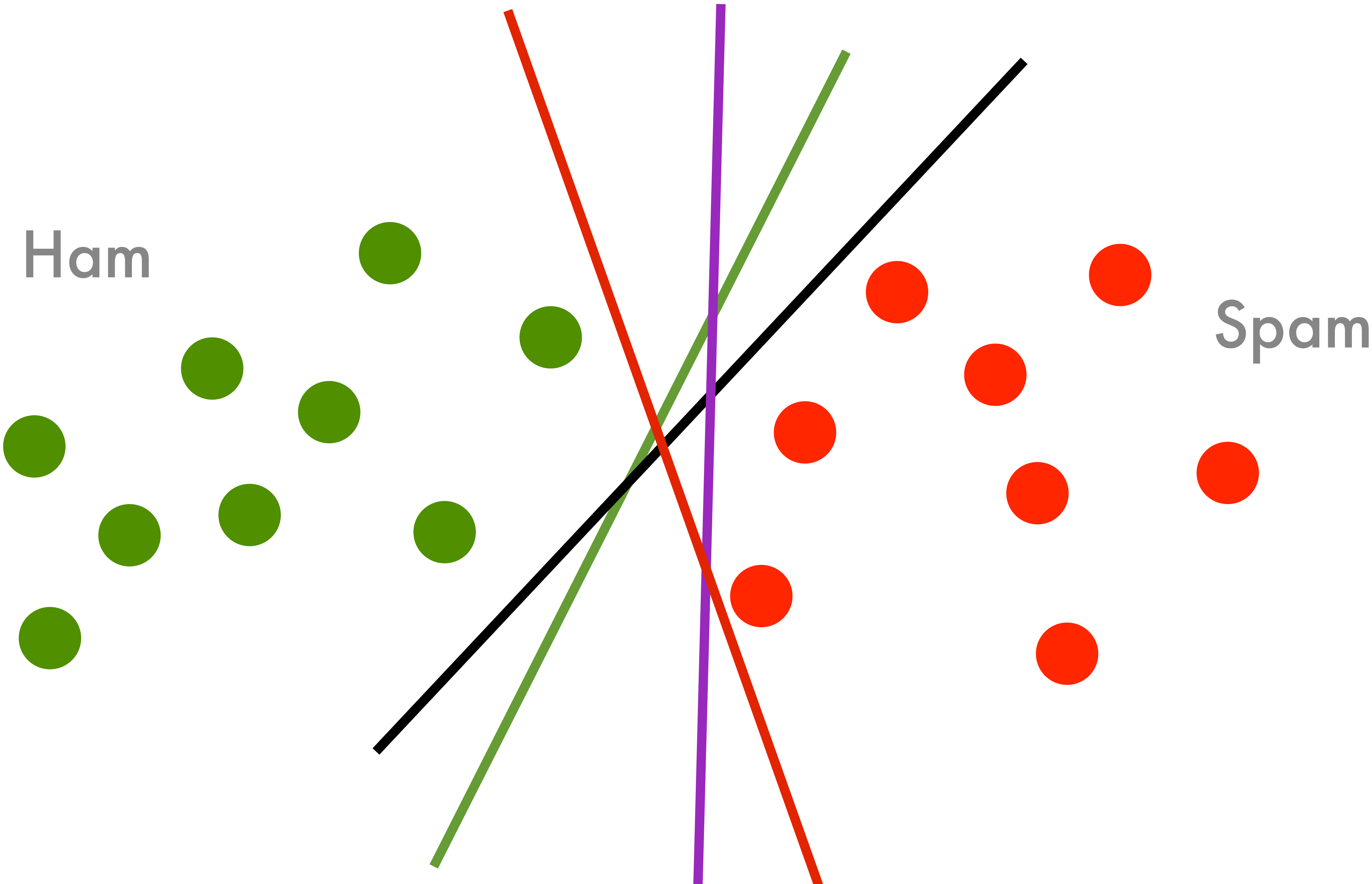
- Weighted linear combination
- Nonlinear decision function
- Linear offset (bias)

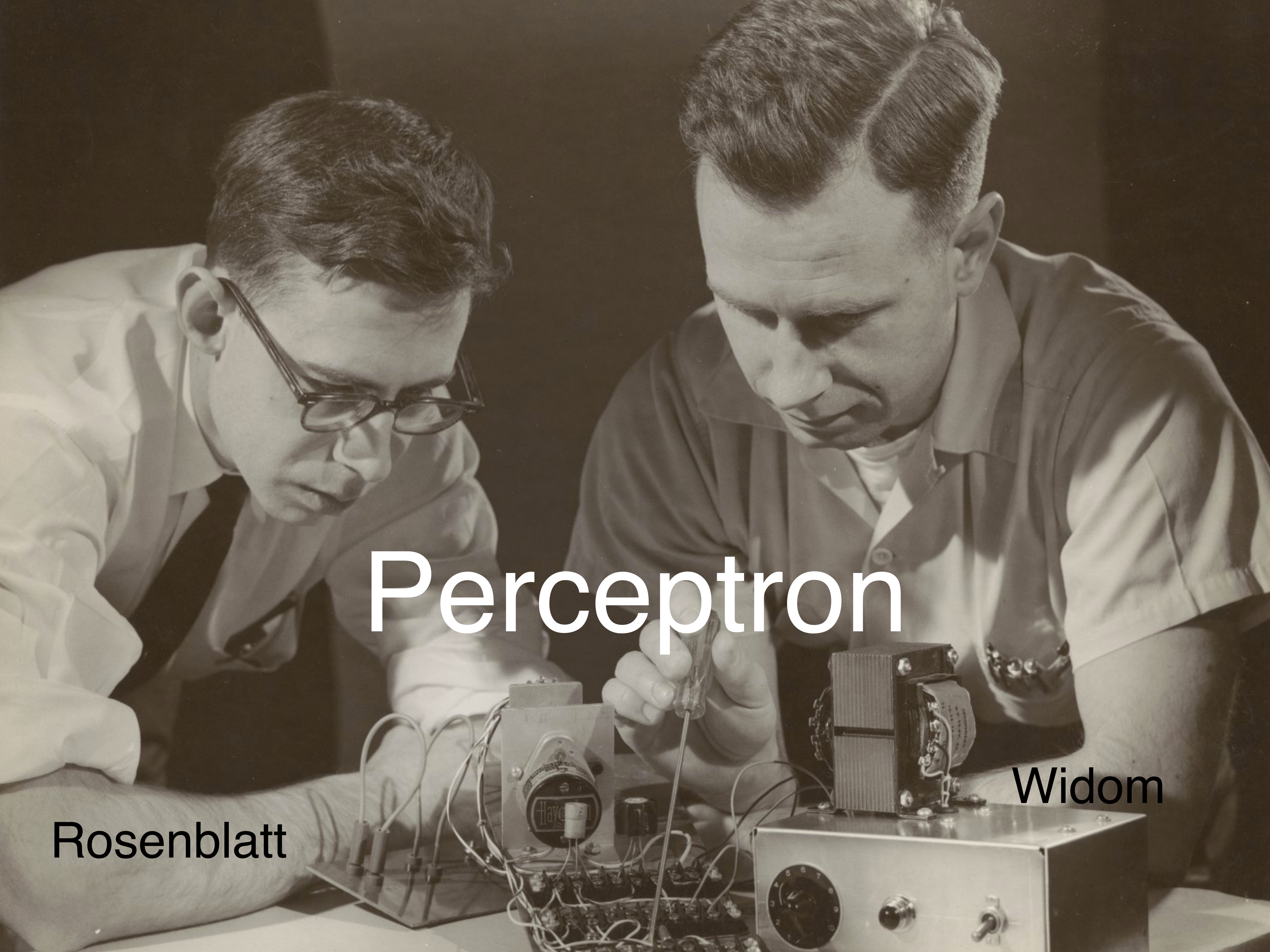


$$f(x) = \sigma\left(\langle w, x \rangle + b\right)$$

- Linear separating hyperplanes (spam/ham, novel/typical, click/no click)
- Learning
  Estimating the parameters w and b

Perceptron




Perceptron
Widom
Rosenblatt

The Perceptron

initialize $w = 0$ and $b = 0$
repeat
  if $y_i \left[\langle w, x_i \rangle + b\right] \leq 0$ **then**
    $w \leftarrow w + y_i x_i$ and $b \leftarrow b + y_i$
  end if
until all classified correctly

- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum_{i \in I} y_i x_i$
- Classifier is linear combination of inner products $f(x) = \sum_{i \in I} y_i \langle x_i, x \rangle + b$

Convergence Theorem

- If there exists some (w^*, b^*) with unit length and

$$y_i \left[\langle x_i, w^* \rangle + b^*\right] \geq \rho \text{ for all } i$$

  then the perceptron converges to a linear separator after a number of steps bounded by

$$\left(b^{*2} + 1\right)\left(r^2 + 1\right)\rho^{-2} \text{ where } \|x_i\| \leq r$$

- Dimensionality independent
- Order independent (i.e. also worst case)
- Scales with ‘difficulty’ of problem

Proof

Starting Point

We start from $w_1 = 0$ and $b_1 = 0$.

Step 1: Bound on the increase of alignment

Denote by w_i the value of w at step i (analogously b_i).

Alignment: $\langle (w_i, b_i), (w^*, b^*) \rangle$

For error in observation (x_i, y_i) we get

$$
\begin{aligned}
& \langle (w_{j+1}, b_{j+1}) \cdot (w^*, b^*) \rangle \\
& = \langle \left[(w_j, b_j) + y_i (x_i, 1) \right], (w^*, b^*) \rangle \\
& = \langle (w_j, b_j), (w^*, b^*) \rangle + y_i \langle (x_i, 1) \cdot (w^*, b^*) \rangle \\
& \geq \langle (w_j, b_j), (w^*, b^*) \rangle + \rho \\
& \geq j\rho.
\end{aligned}
$$

Alignment increases with number of errors.

Proof

Step 2: Cauchy-Schwartz for the Dot Product

$$\begin{aligned}
\langle (w_{j+1}, b_{j+1}) \cdot (w^*, b^*) \rangle &\leq \|(w_{j+1}, b_{j+1})\| \, \|(w^*, b^*)\| \\
&= \sqrt{1 + (b^*)^2} \|(w_{j+1}, b_{j+1})\|
\end{aligned}$$

Step 3: Upper Bound on $\|(w_j, b_j)\|$

If we make a mistake we have

$$\begin{aligned}
\|(w_{j+1}, b_{j+1})\|^2 &= \|(w_j, b_j) + y_i(x_i, 1)\|^2 \\
&= \|(w_j, b_j)\|^2 + 2y_i \langle (x_i, 1), (w_j, b_j) \rangle + \|(x_i, 1)\|^2 \\
&\leq \|(w_j, b_j)\|^2 + \|(x_i, 1)\|^2 \\
&\leq j(R^2 + 1).
\end{aligned}$$

Step 4: Combination of first three steps

$$j\rho \leq \sqrt{1 + (b^*)^2} \|(w_{j+1}, b_{j+1})\| \leq \sqrt{j(R^2 + 1)((b^*)^2 + 1)}$$

Solving for j proves the theorem.

Consequences

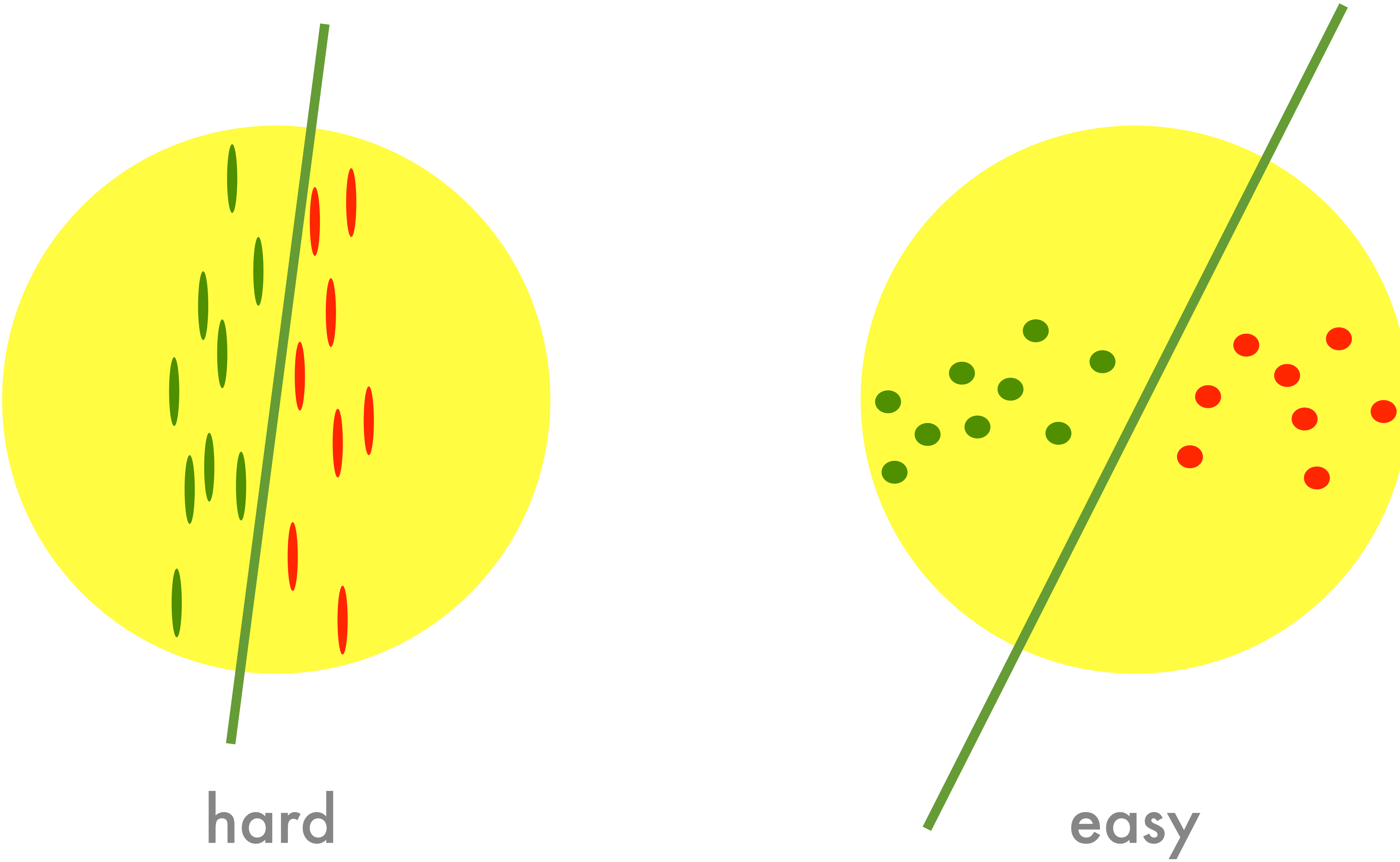
- Only need to store errors.
  This gives a compression bound for perceptron.
- Stochastic gradient descent on hinge loss

$$l(x_i, y_i, w, b) = \max\left(0, 1 - y_i\left[\langle w, x_i \rangle + b\right]\right)$$

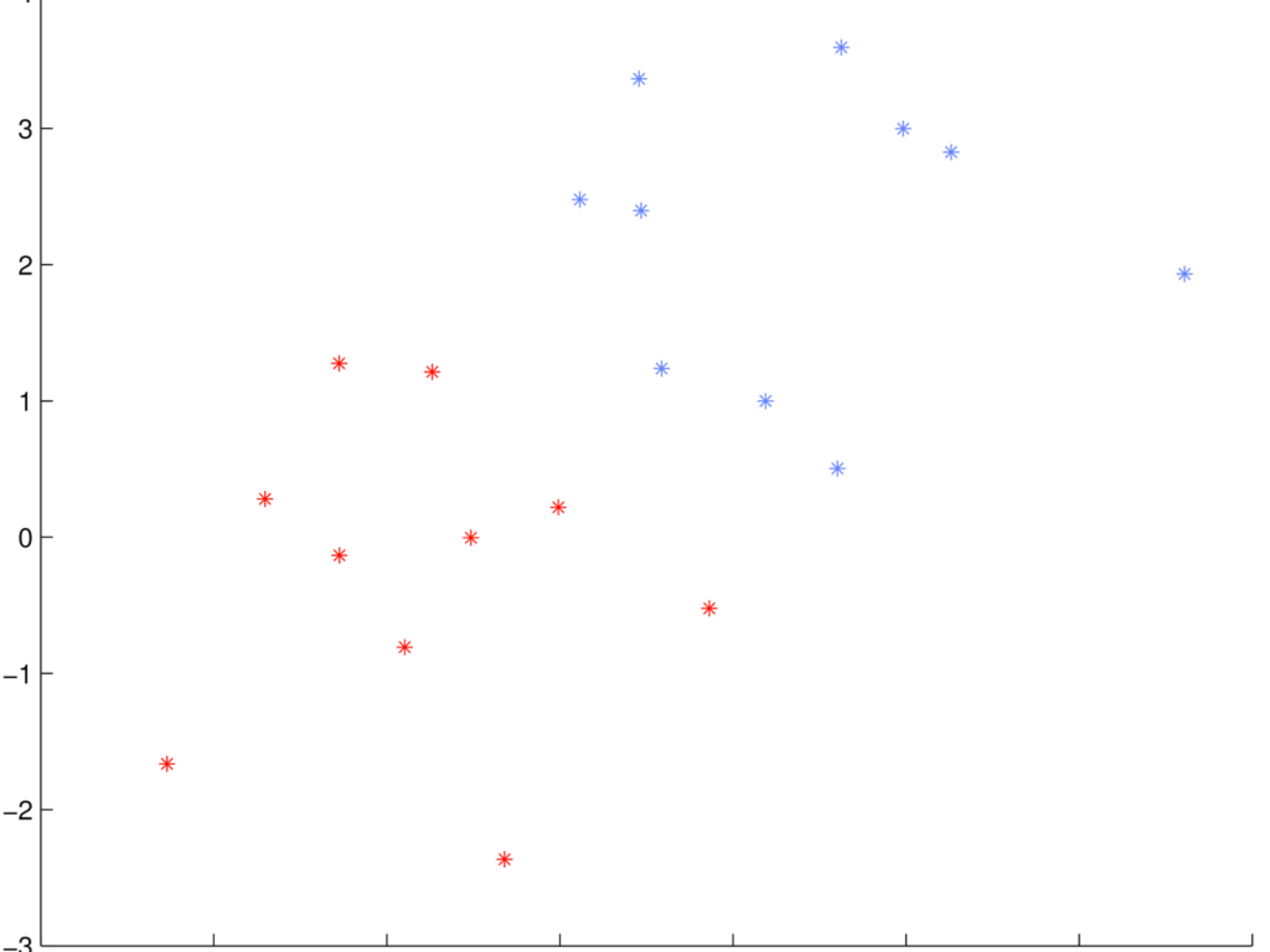
- Fails with noisy data

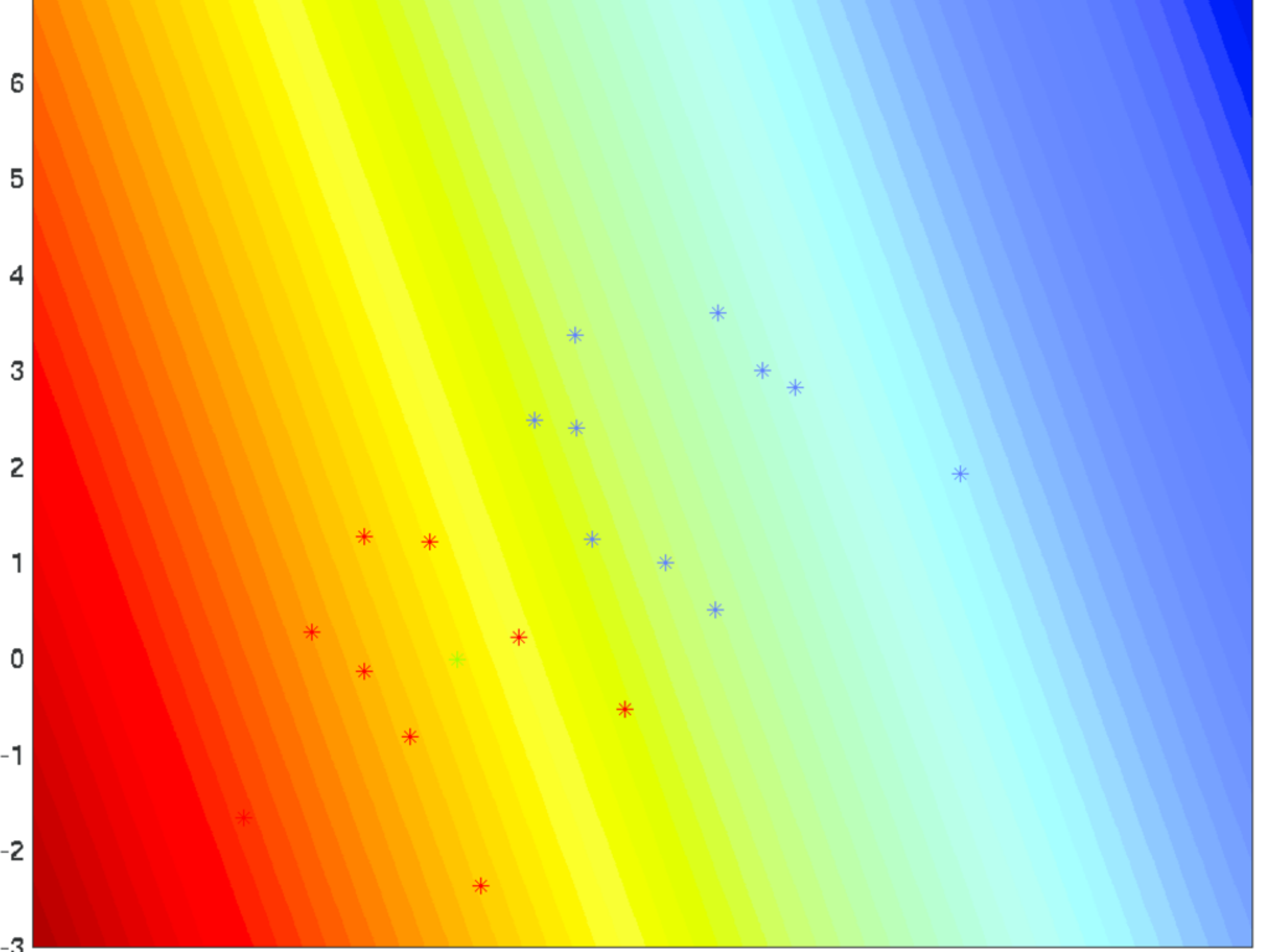
do NOT train your avatar with perceptrons

Hardness

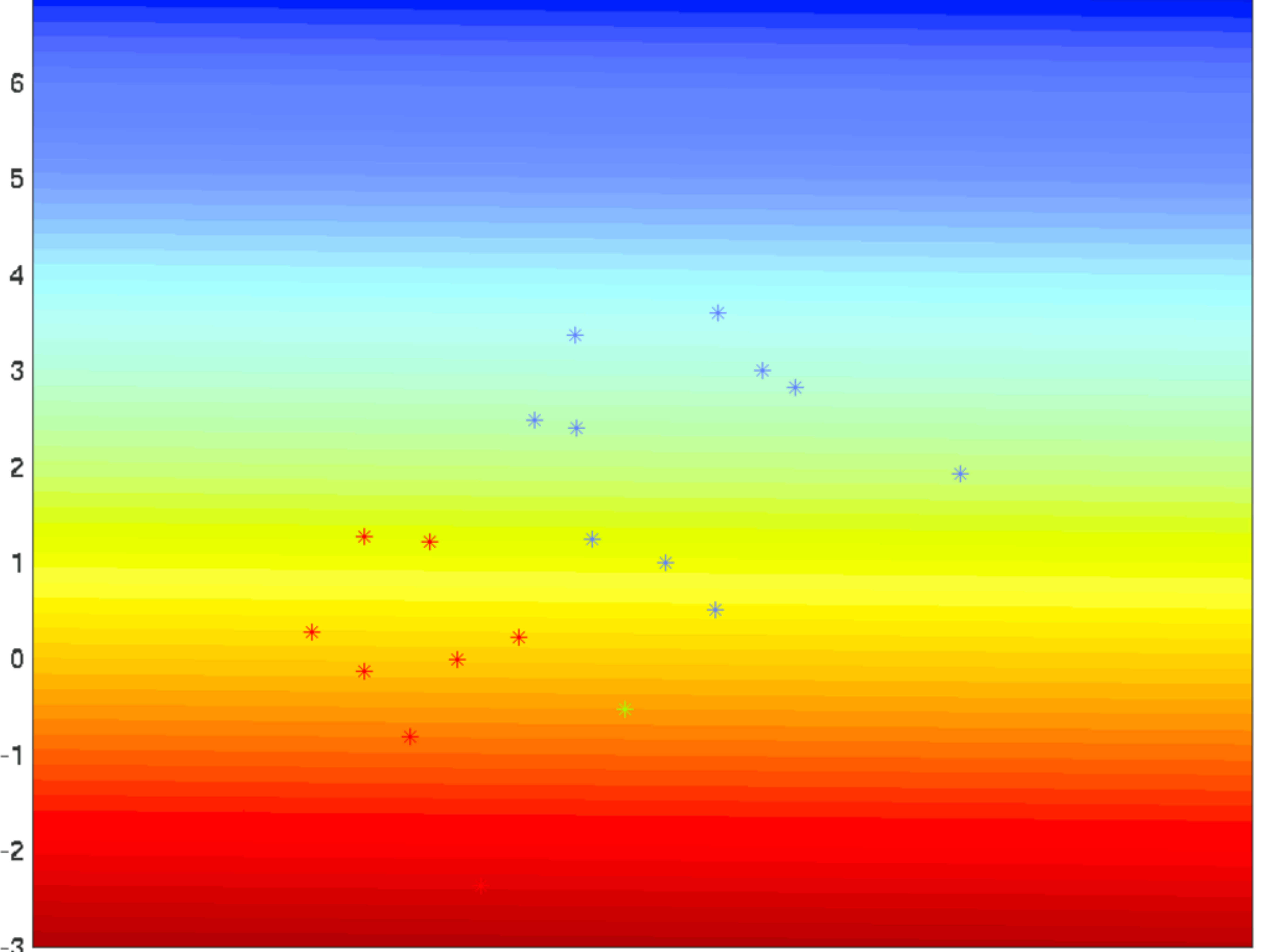


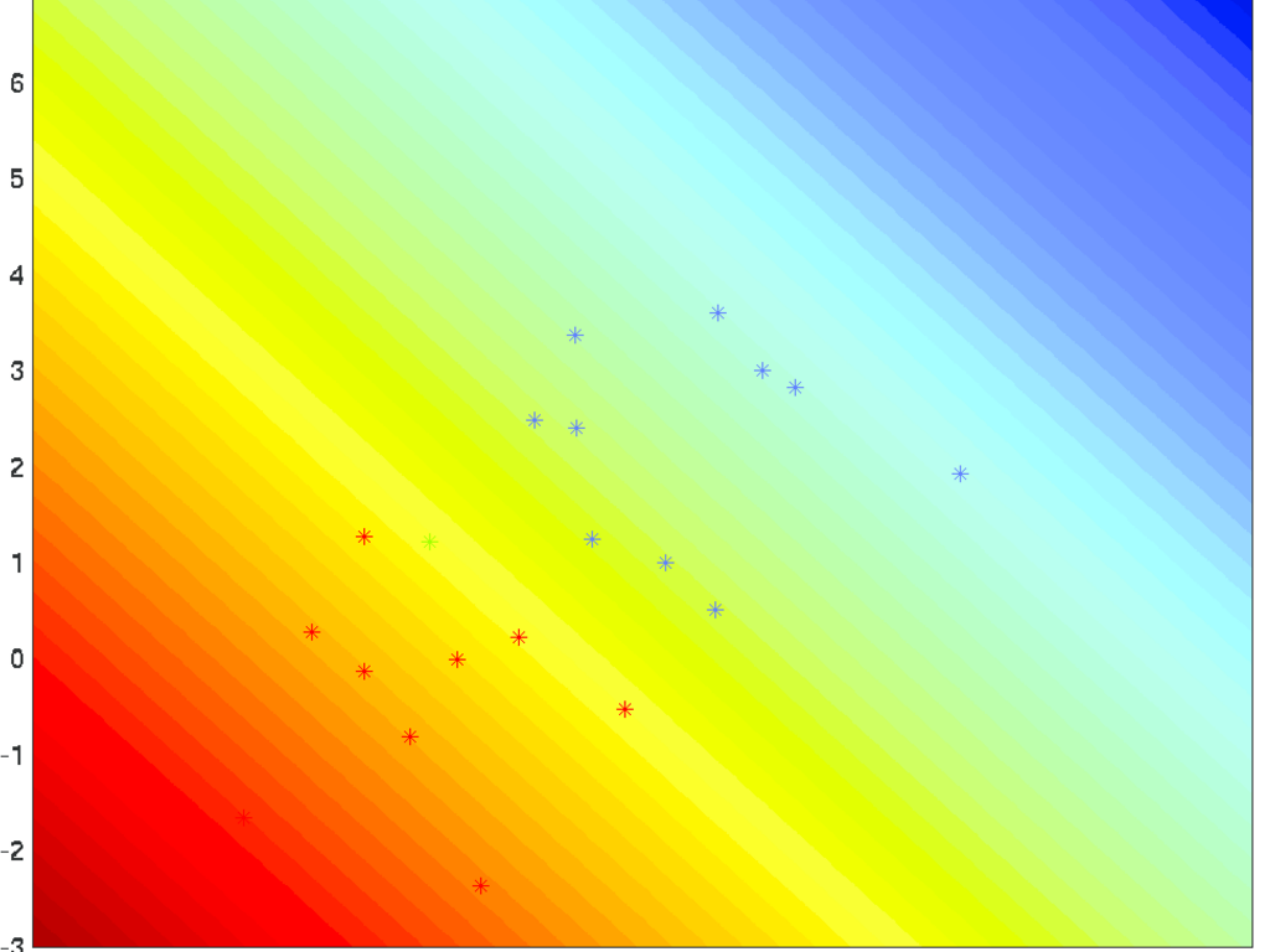
Carnegie Mellon University


3
2
1
0
−1
−2
−3

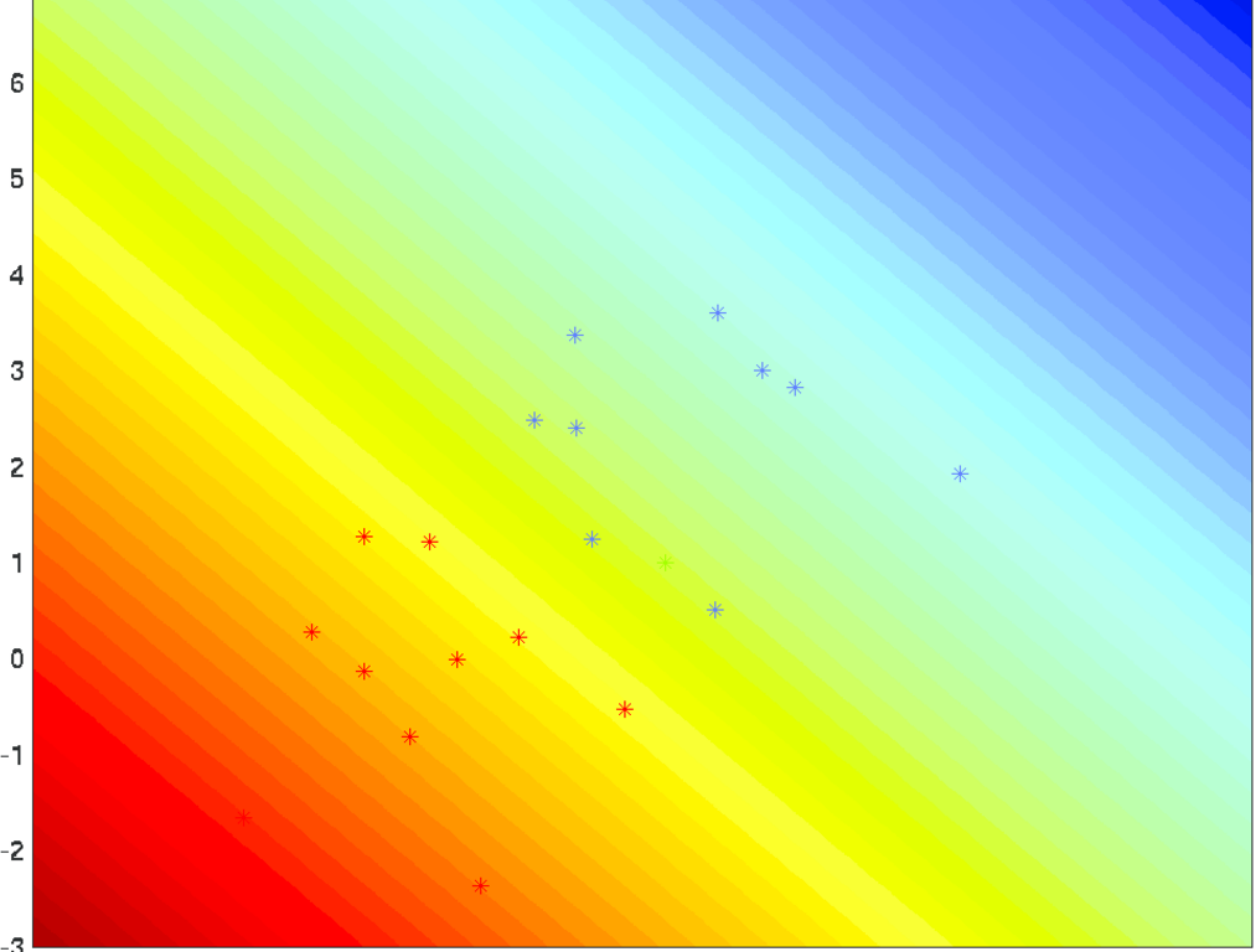


6
5
4
3
2
1
0
-1
-2
-3

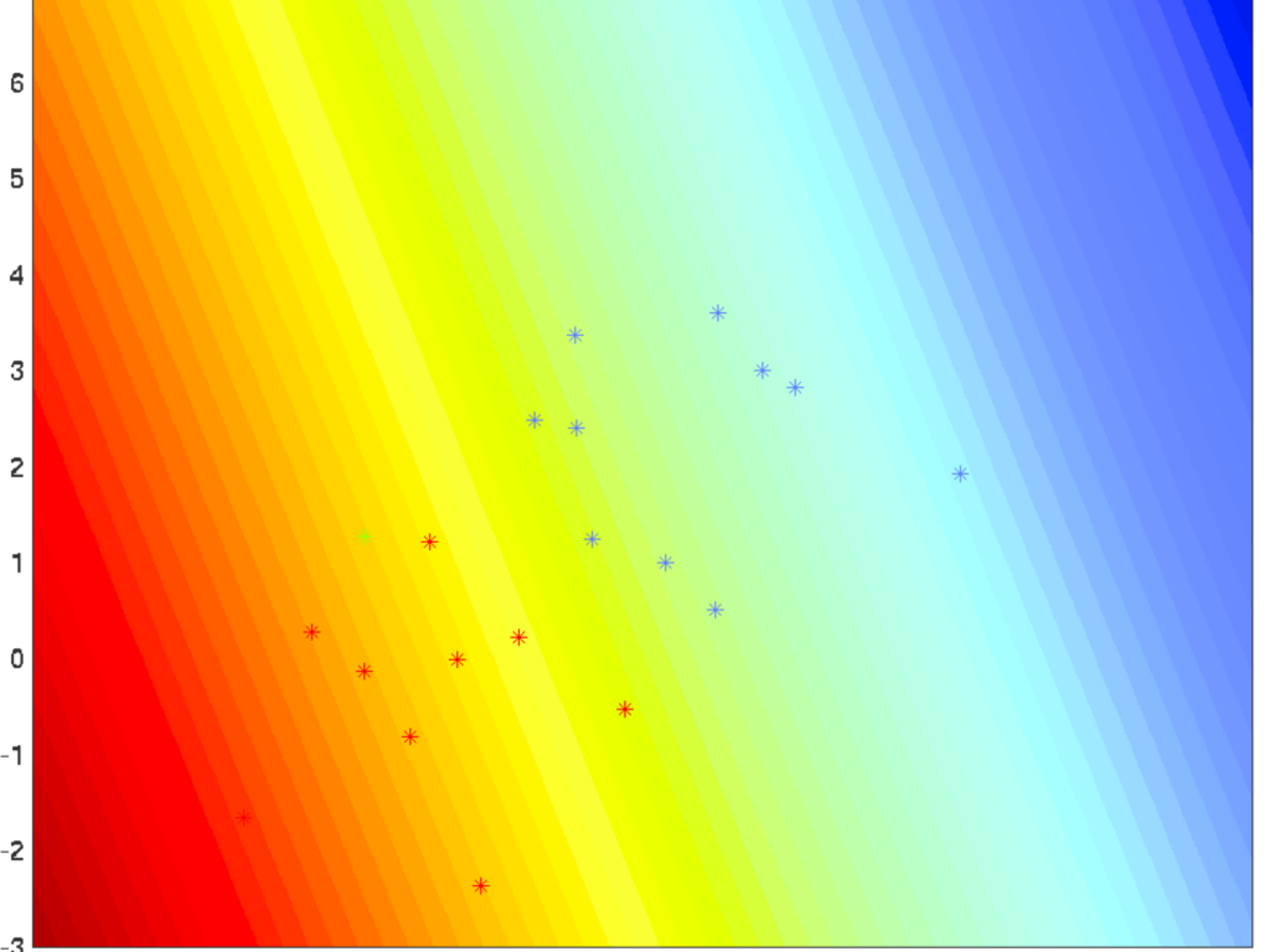

6
5
4
3
2
1
0
-1
-2
-3

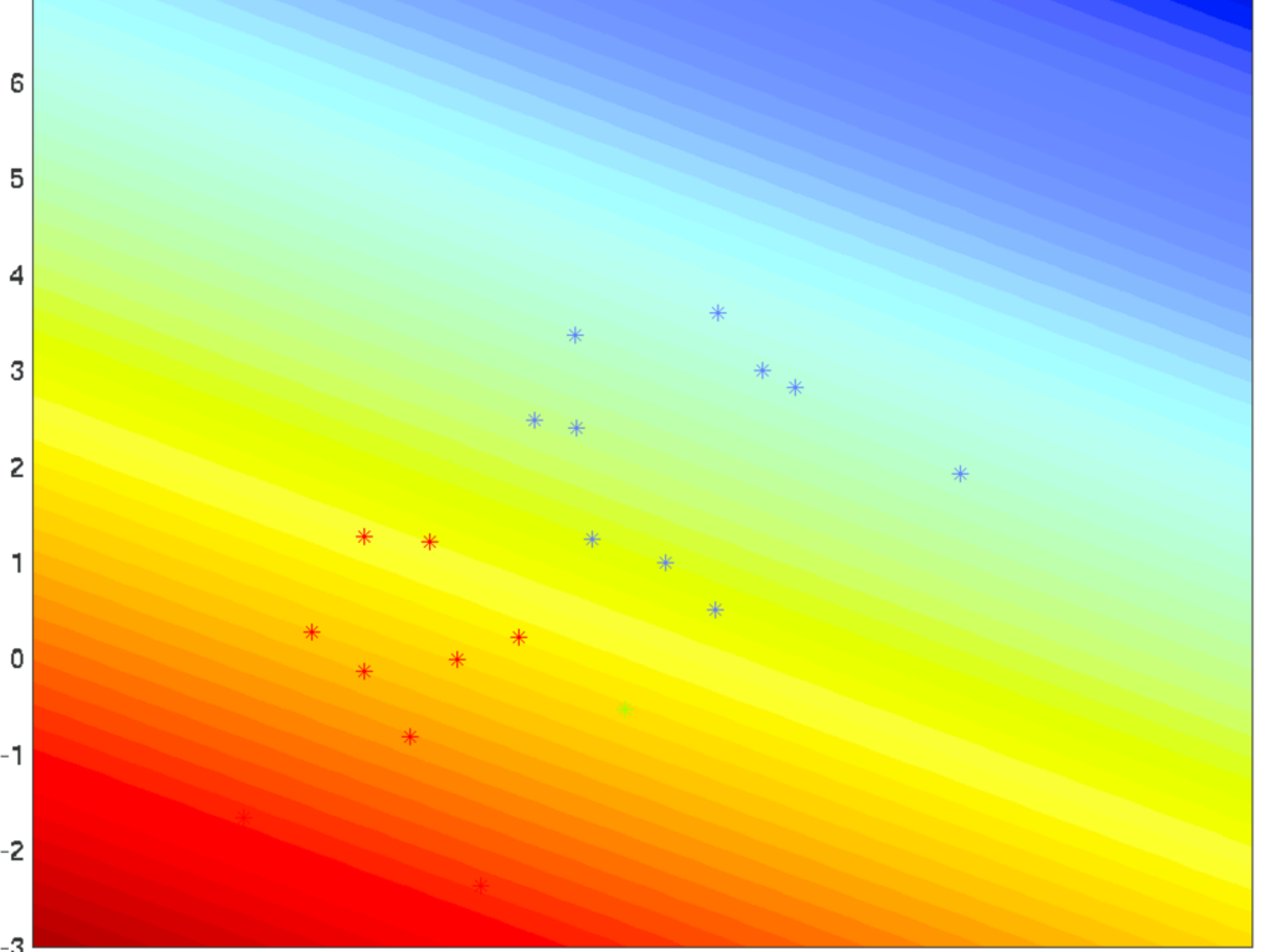


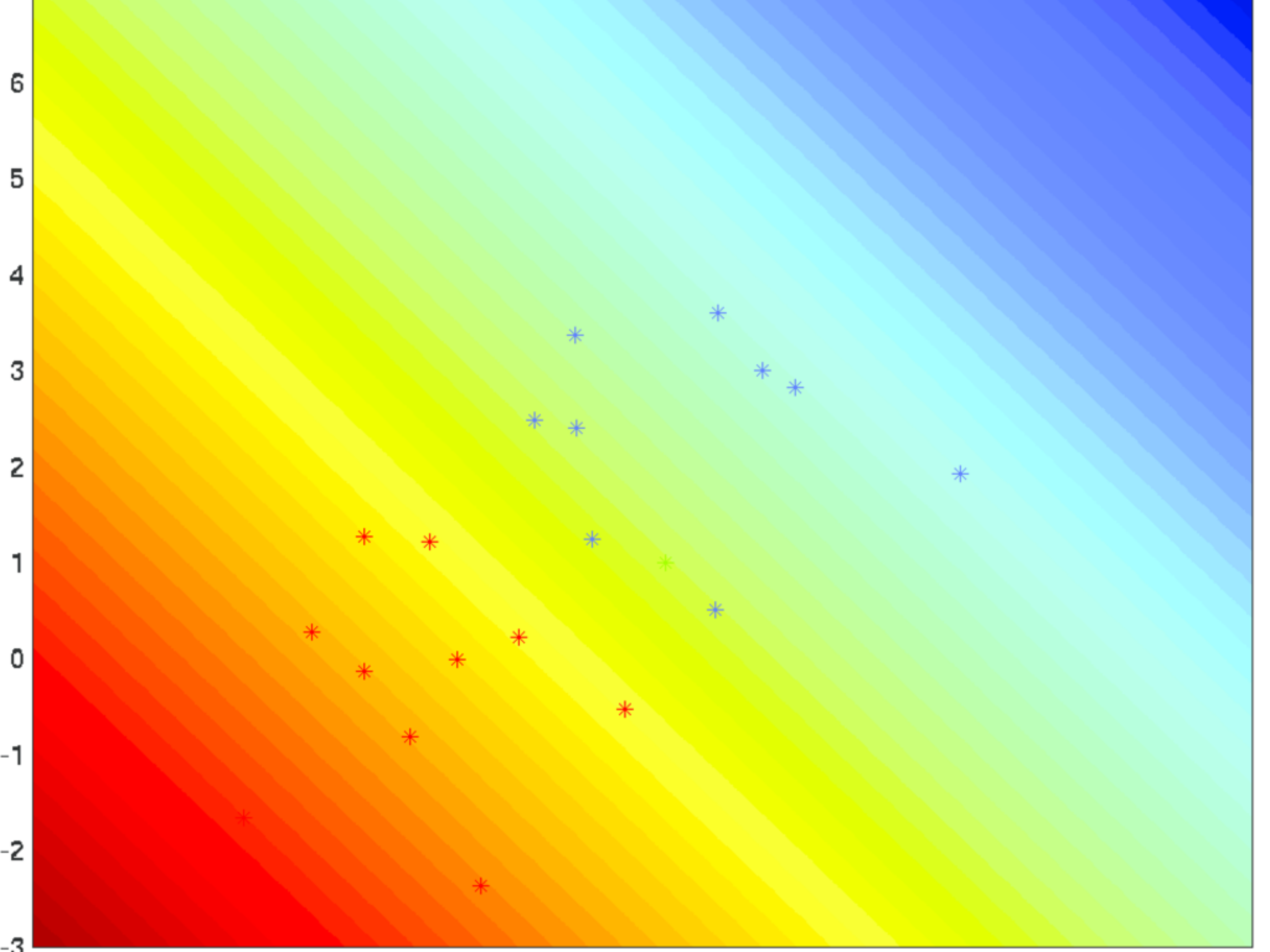
6
5
4
3
2
1
0
-1
-2
-3

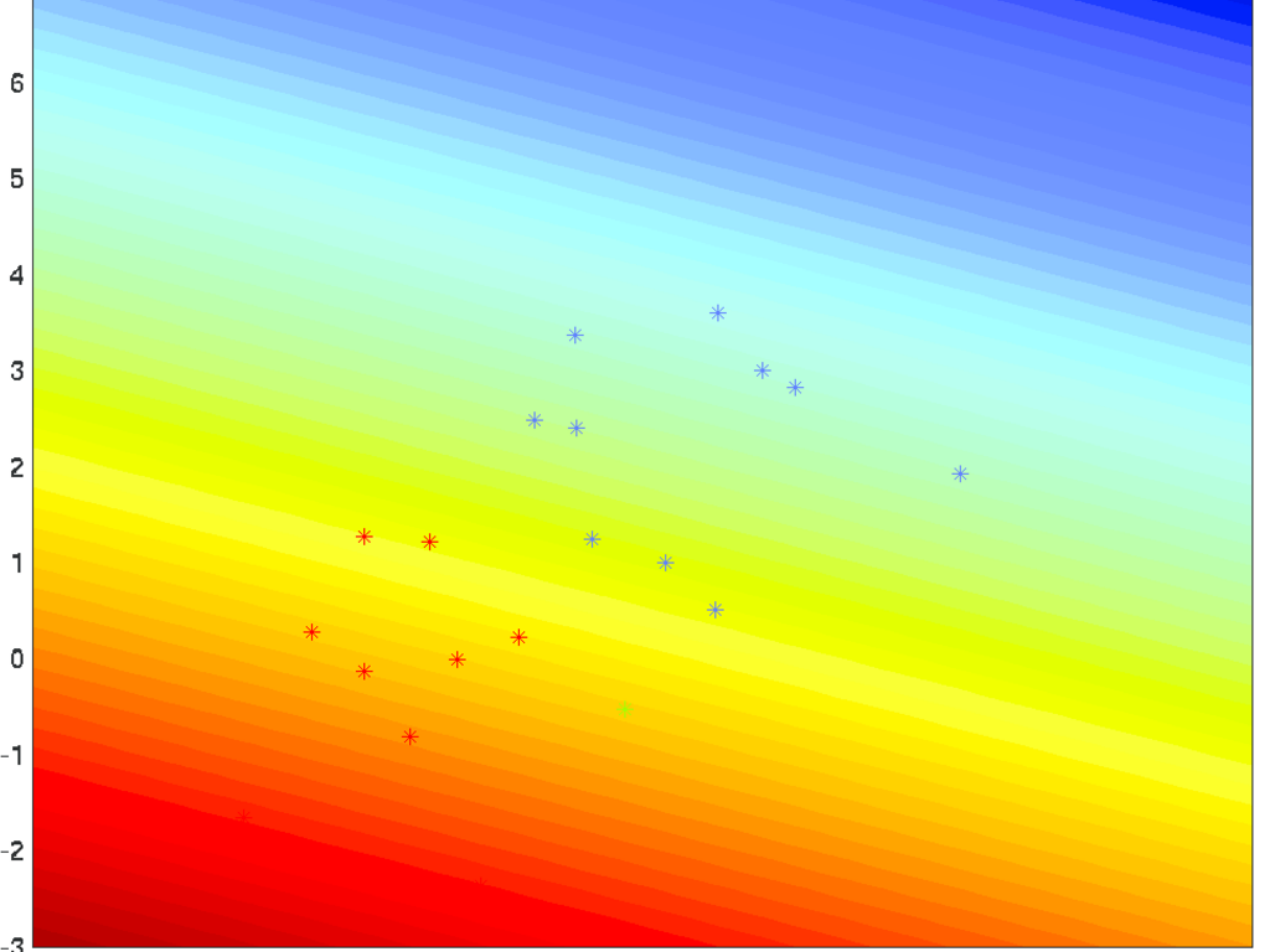


6
5
4
3
2
1
0
-1
-2
-3

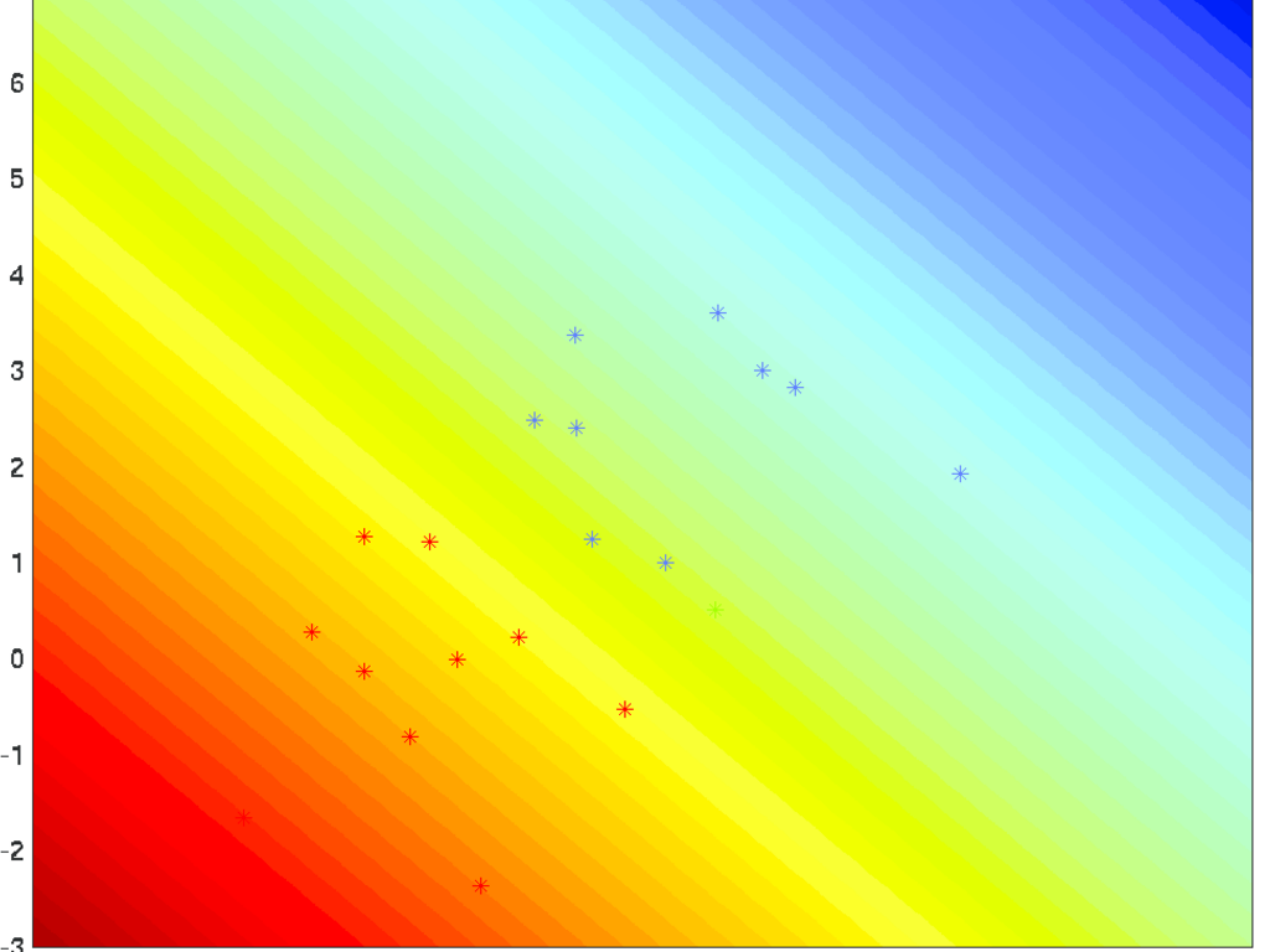

6
5
4
3
2
1
0
-1
-2
-3


6
5
4
3
2
1
0
-1
-2
-3

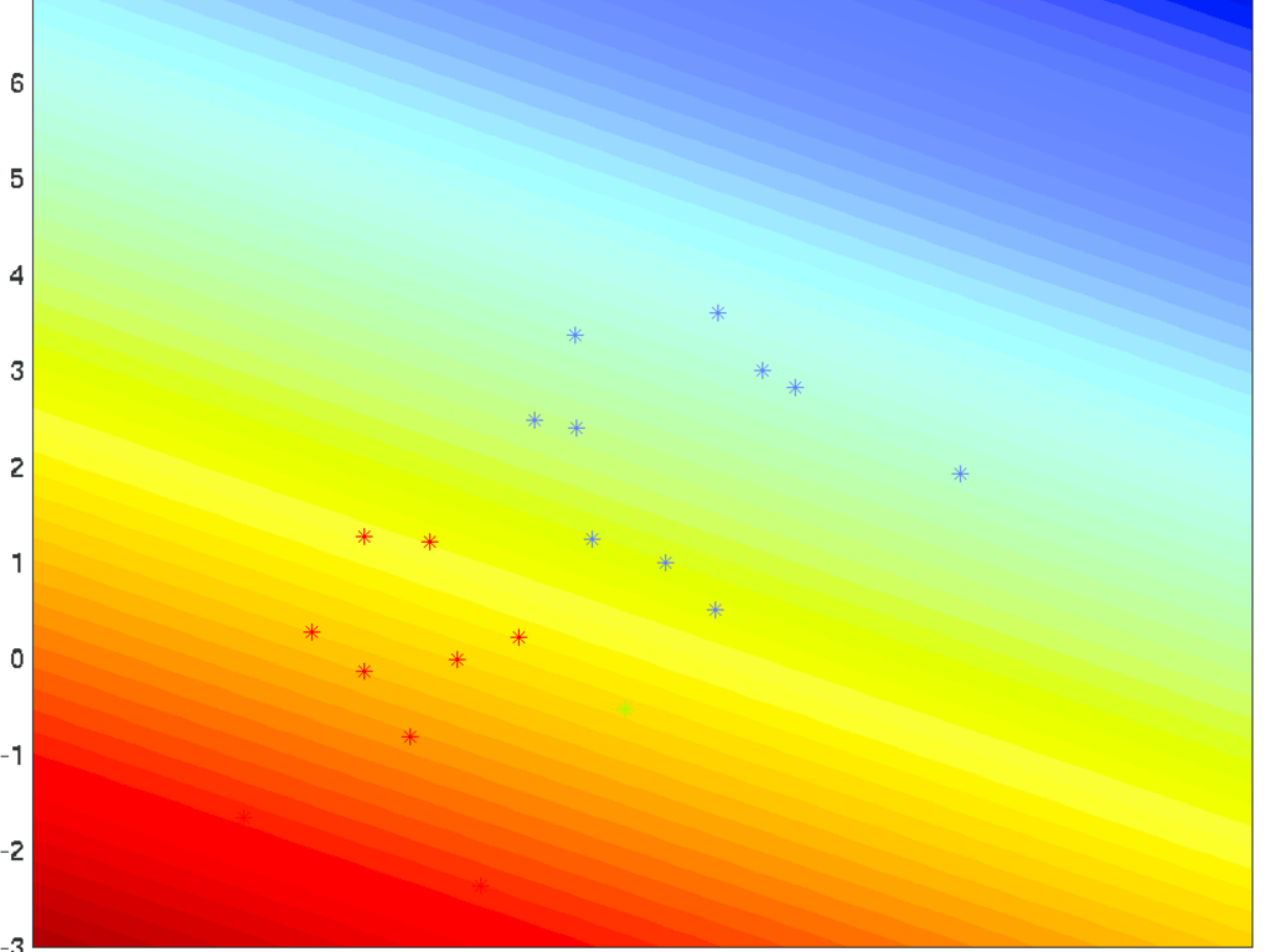

6
5
4
3
2
1
0
-1
-2
-3



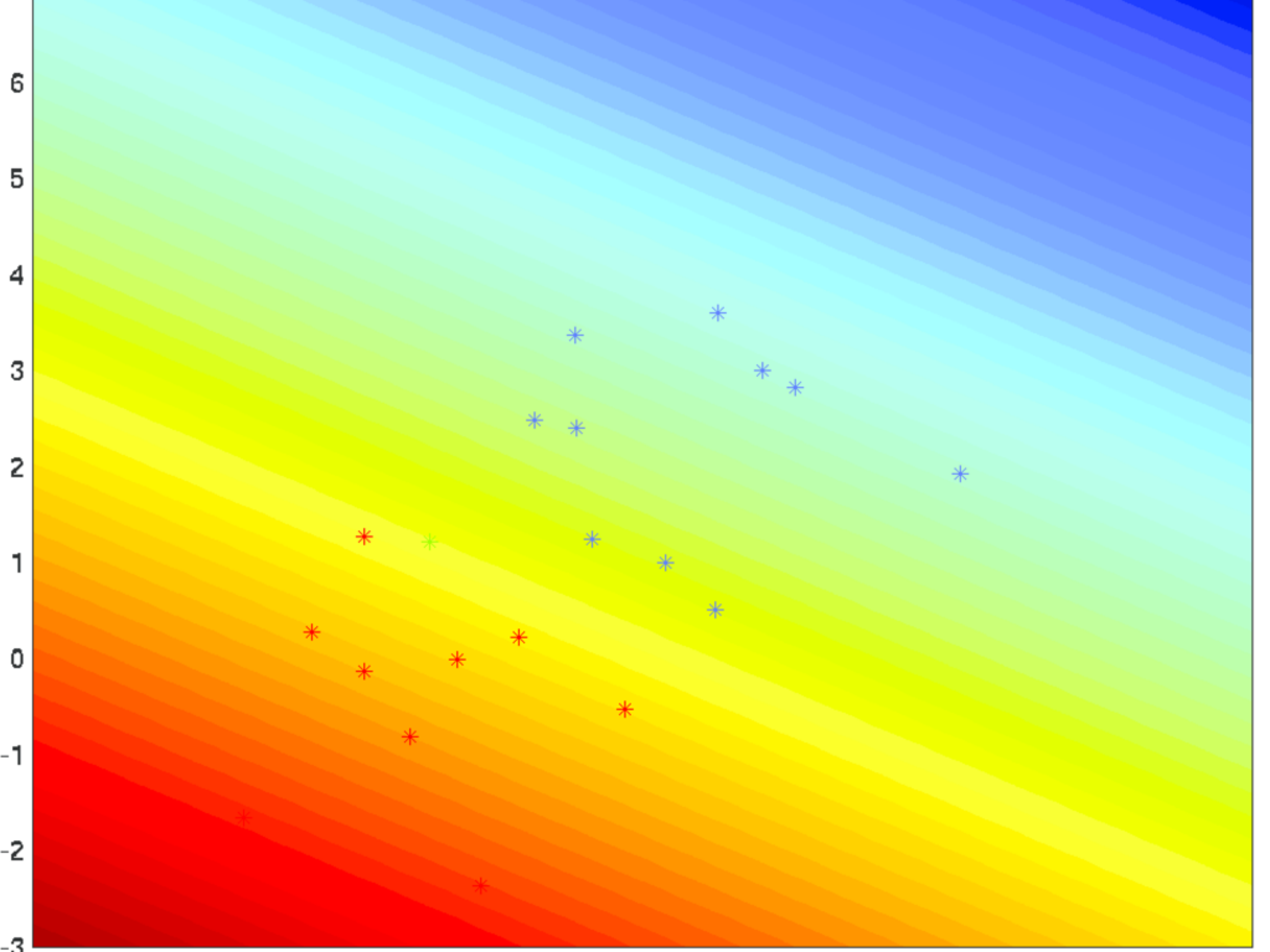
6
5
4
3
2
1
0
-1
-2
-3



6
5
4
3
2
1
0
-1
-2
-3



6
5
4
3
2
1
0
-1
-2
-3



6
5
4
3
2
1
0
-1
-2
-3

4.2 Nonlinearity and Kernels

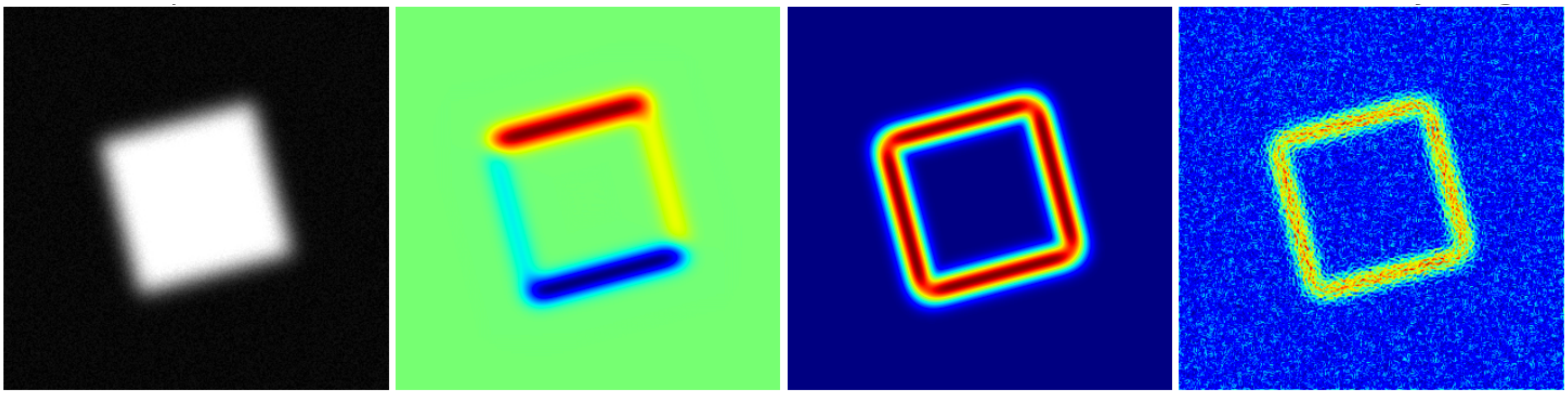
4 (Generalized) Linear Methods

Alexander Smola

Introduction to Machine Learning 10-701

http://alex.smola.org/teaching/10-701-15

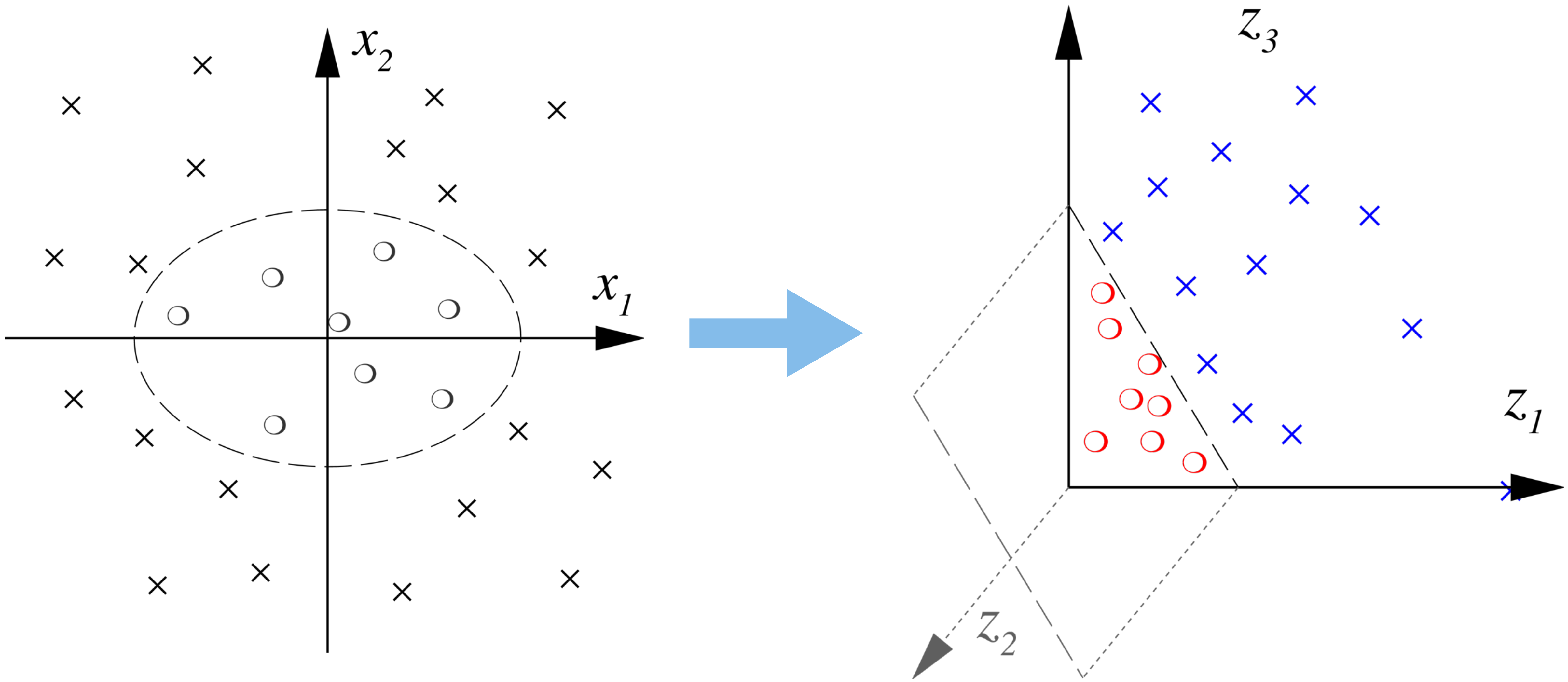
Preprocessing



Nonlinear Features

- Regression
  We got nonlinear functions by preprocessing
- Perceptron
  - Map data into feature space $x \to \phi(x)$
  - Solve problem in this space
  - Query replace $\langle x, x' \rangle$ by $\langle \phi(x), \phi(x') \rangle$ for code
- Feature Perceptron
  - Solution in span of $\phi(x_i)$

Quadratic Features



- Separating surfaces are
  Circles, hyperbolae, parabolae

Constructing Features

	1	2	3	4	5	6	7	8	9	0
Loops	0	0	0	1	0	1	0	2	1	1
3 Joints	0	0	0	0	0	1	0	0	1	0
4 Joints	0	0	0	1	0	0	0	1	0	0
Angles	0	1	1	1	1	0	1	0	0	0
Ink	1	2	2	2	2	2	1	3	2	2

Feature Engineering for Spam Filtering

```
Delivered-To: alex.smola@gmail.com
Received: by 10.216.47.73 with SMTP id s51cs361171web;
        Tue, 3 Jan 2012 14:17:53 -0800 (PST)
Received: by 10.213.17.145 with SMTP id s17mr2519891eba.147.1325629071725;
        Tue, 03 Jan 2012 14:17:51 -0800 (PST)
Return-Path: <alex+caf_=alex.smola=gmail.com@smola.org>
Received: from mail-ey0-f175.google.com (mail-ey0-f175.google.com [209.85.215.175])
        by mx.google.com with ESMTPS id n4si29264232eef.57.2012.01.03.14.17.51
        (version=TLSv1/SSLv3 cipher=OTHER);
        Tue, 03 Jan 2012 14:17:51 -0800 (PST)
Received-SPF: neutral (google.com: 209.85.215.175 is neither permitted nor denied by best
guess record for domain of alex+caf_=alex.smola=gmail.com@smola.org) client-
ip=209.85.215.175;
Authentication-Results: mx.google.com; spf=neutral (google.com: 209.85.215.175 is neither
permitted nor denied by best guess record for domain of alex
+caf_=alex.smola=gmail.com@smola.org) smtp.mail=alex+caf_=alex.smola=gmail.com@smola.org;
dkim=pass (test mode) header.i=@googlemail.com
Received: by eaal1 with SMTP id l1so15092746eaa.6
        for <alex.smola@gmail.com>; Tue, 03 Jan 2012 14:17:51 -0800 (PST)
Received: by 10.205.135.18 with SMTP id ie18mr5325064bkc.72.1325629071362;
        Tue, 03 Jan 2012 14:17:51 -0800 (PST)
X-Forwarded-To: alex.smola@gmail.com
X-Forwarded-For: alex@smola.org alex.smola@gmail.com
Delivered-To: alex@smola.org
Received: by 10.204.65.198 with SMTP id k6cs206093bki;
        Tue, 3 Jan 2012 14:17:50 -0800 (PST)
Received: by 10.52.88.179 with SMTP id bh19mr10729402vdb.38.1325629068795;
        Tue, 03 Jan 2012 14:17:48 -0800 (PST)
Return-Path: <althoff.tim@googlemail.com>
Received: from mail-vx0-f179.google.com (mail-vx0-f179.google.com [209.85.220.179])
        by mx.google.com with ESMTPS id dt4si11767074vdb.93.2012.01.03.14.17.48
        (version=TLSv1/SSLv3 cipher=OTHER);
        Tue, 03 Jan 2012 14:17:48 -0800 (PST)
Received-SPF: pass (google.com: domain of althoff.tim@googlemail.com designates
209.85.220.179 as permitted sender) client-ip=209.85.220.179;
Received: by vcbf13 with SMTP id f13so11295098vcb.10
        for <alex@smola.org>; Tue, 03 Jan 2012 14:17:48 -0800 (PST)
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
        d=googlemail.com; s=gamma;
        h=mime-version:sender:date:x-google-sender-auth:message-id:subject
         :from:to:content-type;
        bh=WCbdZ5sXac25dpH02XcRyDOdts993hKwsAVXpGrFh0w=;
        b=WK2B2+ExWnf/gvTkw6uUvKuP4XeoKnlJq3USYTm0RARK8dSFjyOQsIHeAP9Yssxp6O
         7ngGoTzYqd+ZsyJfvQcLAWp1PCJhG8AMcnqWkx0NMeoFvIp2HQooZwxSOCx5ZRgY+7qX
         uIbbdna4lUDXj6UFe16SpLDCkptd8OZ3gr7+o=
MIME-Version: 1.0
Received: by 10.220.108.81 with SMTP id e17mr24104004vcp.67.1325629067787;
 Tue, 03 Jan 2012 14:17:47 -0800 (PST)
Sender: althoff.tim@googlemail.com
Received: by 10.220.17.129 with HTTP; Tue, 3 Jan 2012 14:17:47 -0800 (PST)
Date: Tue, 3 Jan 2012 14:17:47 -0800
X-Google-Sender-Auth: 6bwi6D17HjZIkxOEol38NZzyeHs
Message-ID: <CAFJJHDGPBW+SdZg0MdAABiAKydDk9tpeMoDijYGjoGO-WC7osg@mail.gmail.com>
Subject: CS 281B. Advanced Topics in Learning and Decision Making
From: Tim Althoff <althoff@eecs.berkeley.edu>
To: alex@smola.org
Content-Type: multipart/alternative; boundary=f46d043c7af4b07e8d04b5a7113a

--f46d043c7af4b07e8d04b5a7113a
Content-Type: text/plain; charset=ISO-8859-1
```

- bag of words
- pairs of words
- date & time
- recipient path
- IP number
- sender
- encoding
- links
- ... secret sauce ...

More feature engineering

- Two Interlocking Spirals
  Transform the data into a radial and angular part
  $$(x_1, x_2) = (r \sin \phi, r \cos \phi)$$
- Handwritten Japanese Character Recognition
  - Break down the images into strokes and recognize it
  - Lookup based on stroke order
- Medical Diagnosis
  - Physician's comments
  - Blood status / ECG / height / weight / temperature ...
  - Medical knowledge
- Preprocessing
  - Zero mean, unit variance to fix scale issue (e.g. weight vs. income)
  - Probability integral transform (inverse CDF) as alternative

The Perceptron on features

initialize $w, b = 0$
repeat
    Pick (x_i, y_i) from data
    if $y_i(w \cdot \Phi(x_i) + b) \leq 0$ then

$$\begin{aligned} w' &= w + y_i \Phi(x_i) \\ b' &= b + y_i \end{aligned}$$

until $y_i(w \cdot \Phi(x_i) + b) > 0$ for all i

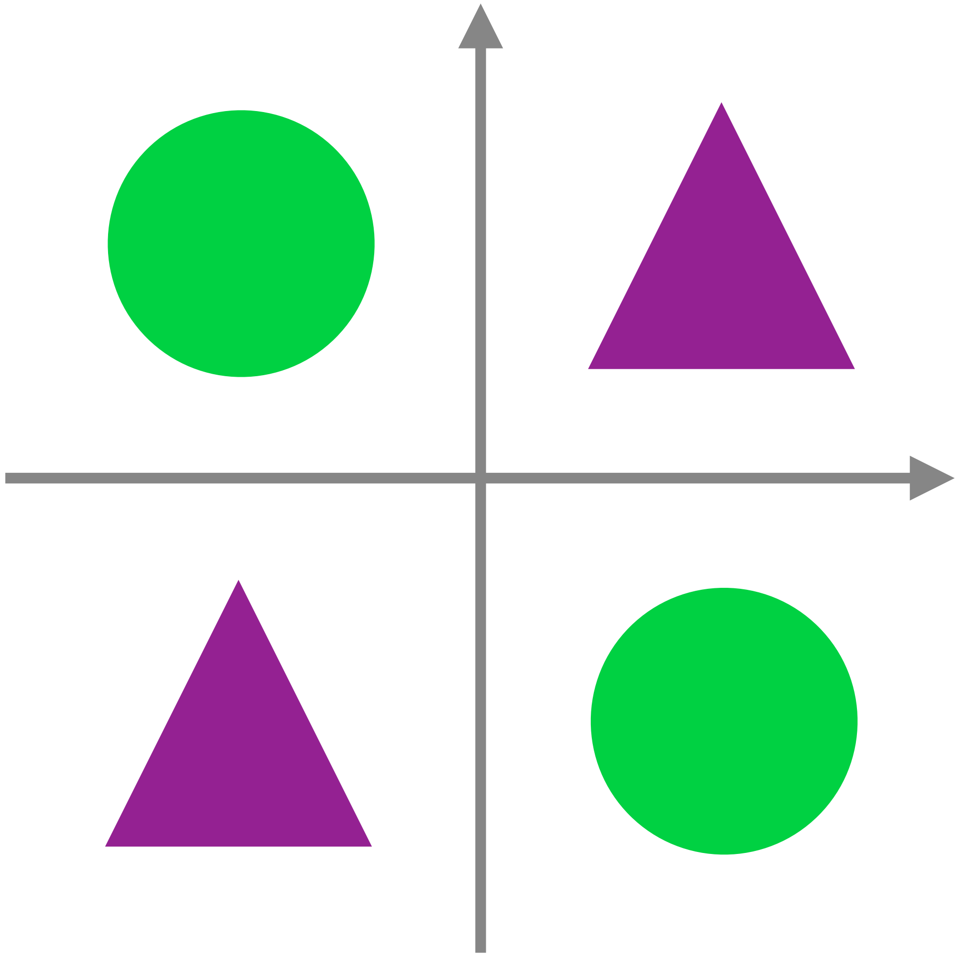
- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum_{i \in I} y_i \phi(x_i)$
- Classifier is linear combination of inner products $f(x) = \sum_{i \in I} y_i \langle \phi(x_i), \phi(x) \rangle + b$

Problems

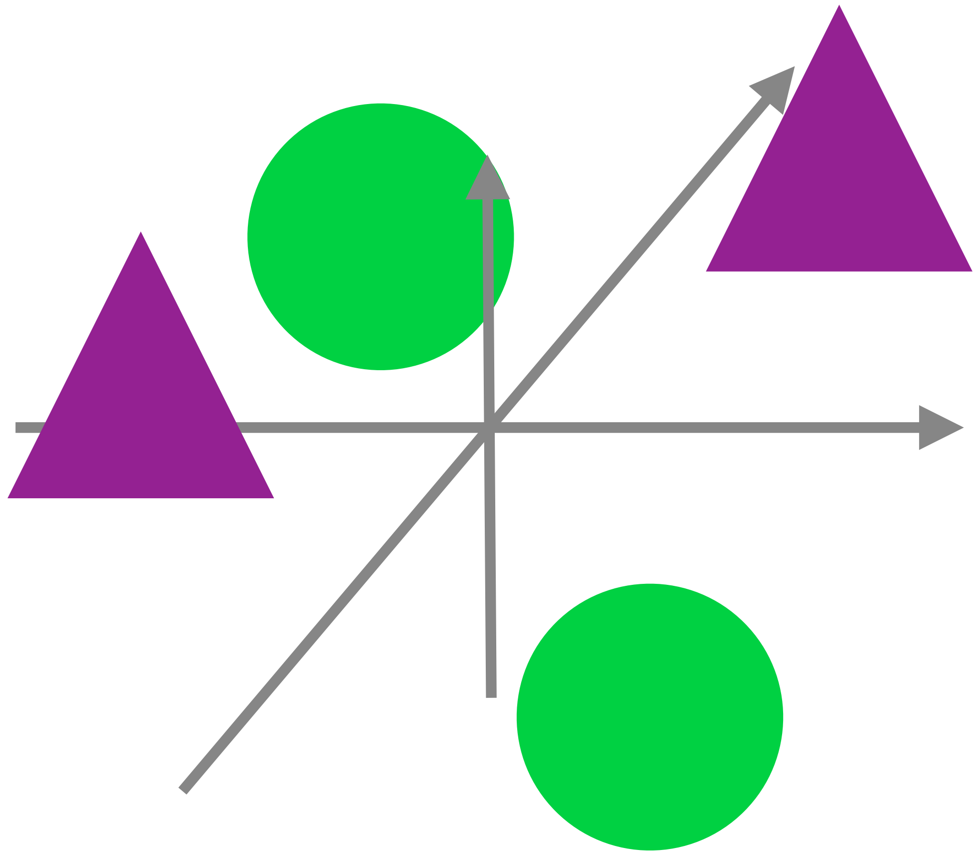
- Problems
  - Need domain expert (e.g. Chinese OCR)
  - Often expensive to compute
  - Difficult to transfer engineering knowledge
- Shotgun Solution
  - Compute many features
  - Hope that this contains good ones
  - Do this efficiently
- Nonlinear methods (needs lots of data & cpu)
  learn the features and the classifier

Kernels

Solving XOR



(x_1, x_2)



$(x_1, x_2, x_1 x_2)$

- XOR not linearly separable
- Mapping into 3 dimensions makes it easily solvable

Quadratic Features

Quadratic Features in $\mathbb{R}^2$

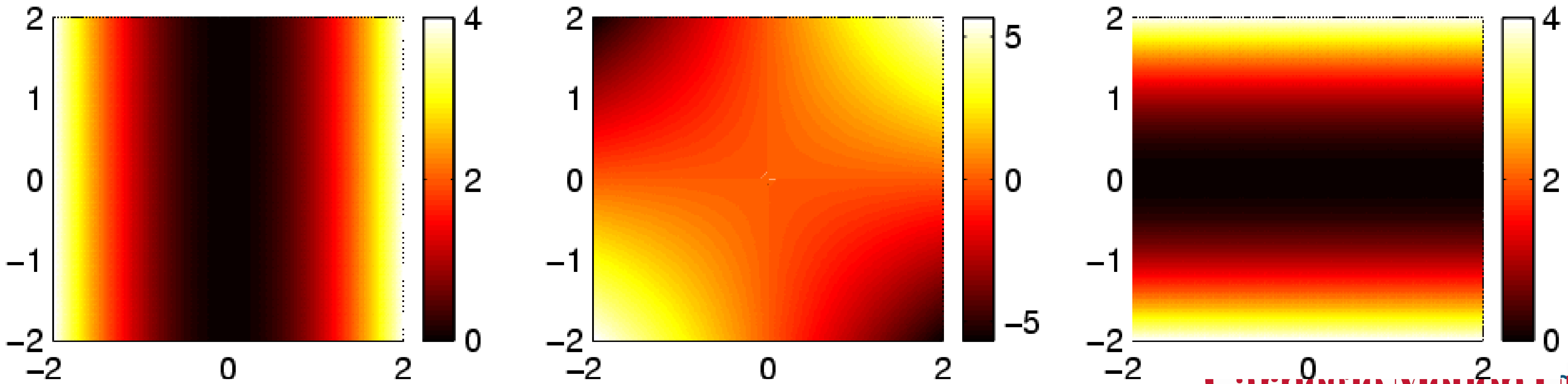
$$\Phi(x) := \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$

Dot Product

$$\begin{aligned}\langle \Phi(x), \Phi(x') \rangle &= \left\langle \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right), \left({x_1'}^2, \sqrt{2}x_1'x_2', {x_2'}^2\right) \right\rangle \\ &= \langle x, x' \rangle^2.\end{aligned}$$

Insight

Trick works for any polynomials of order d via $\langle x, x' \rangle^d$.



SVM with a polynomial Kernel visualization

Created by:
Udi Aharoni

SVM with a polynomial Kernel visualization

Created by:
Udi Aharoni

Computational Efficiency

Problem

- Extracting features can sometimes be very costly.
- Example: second order features in 1000 dimensions. This leads to $5 \cdot 10^5$ numbers. For higher order polynomial features much worse.

Solution

Don't compute the features, try to compute dot products implicitly. For some features this works ...

Definition

A kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric function in its arguments for which the following property holds

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle \text{ for some feature map } \Phi.$$

If $k(x, x')$ is much cheaper to compute than $\Phi(x)$ ...

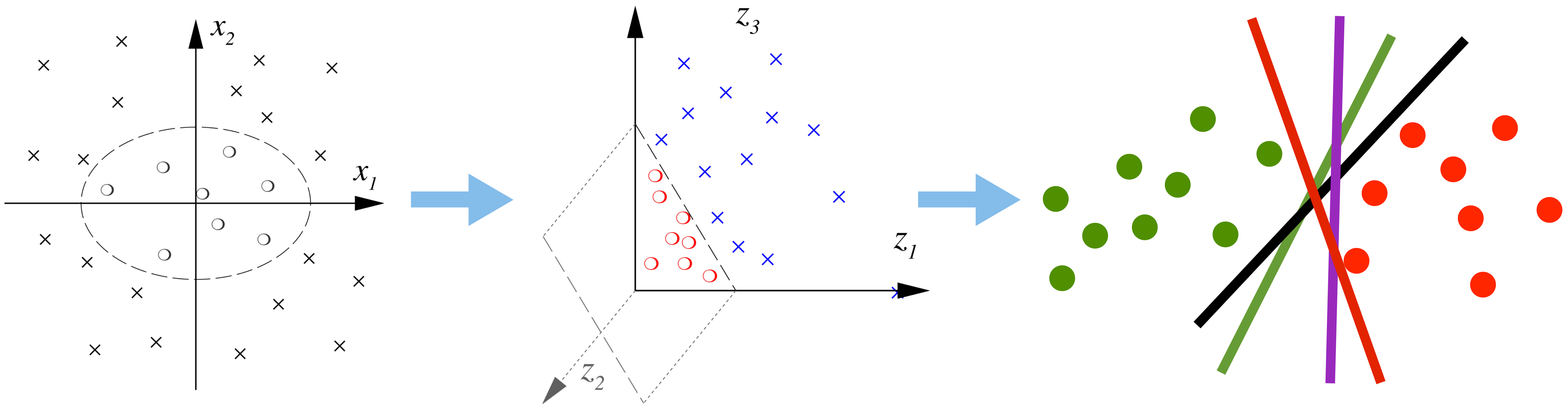
The Kernel Perceptron

initialize $f = 0$
repeat
    Pick (x_i, y_i) from data
    if $y_i f(x_i) \leq 0$ then
        $f(\cdot) \leftarrow f(\cdot) + y_i k(x_i, \cdot) + y_i$
until $y_i f(x_i) > 0$ for all i

- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum_{i \in I} y_i \phi(x_i)$
- Classifier is linear combination of inner products

$$f(x) = \sum_{i \in I} y_i \langle \phi(x_i), \phi(x) \rangle + b = \sum_{i \in I} y_i k(x_i, x) + b$$

Processing Pipeline



- Original data
- Data in feature space (implicit)
- Solve in feature space using kernels

Polynomial Kernels

Idea

- We want to extend $k(x, x') = \langle x, x' \rangle^2$ to

$$k(x, x') = (\langle x, x' \rangle + c)^d \text{ where } c > 0 \text{ and } d \in \mathbb{N}.$$

- Prove that such a kernel corresponds to a dot product.

Proof strategy

Simple and straightforward: compute the explicit sum given by the kernel, i.e.

$$k(x, x') = (\langle x, x' \rangle + c)^d = \sum_{i=0}^{m} \binom{d}{i} (\langle x, x' \rangle)^i c^{d-i}$$

Individual terms $(\langle x, x' \rangle)^i$ are dot products for some $\Phi_i(x)$.

Kernel Conditions

Computability

We have to be able to compute $k(x, x')$ efficiently (much cheaper than dot products themselves).

"Nice and Useful" Functions

The features themselves have to be useful for the learning problem at hand. Quite often this means smooth functions.

Symmetry

Obviously $k(x, x') = k(x', x)$ due to the symmetry of the dot product $\langle \Phi(x), \Phi(x') \rangle = \langle \Phi(x'), \Phi(x) \rangle$.

Dot Product in Feature Space

Is there always a Φ such that k really is a dot product?

Mercer's Theorem

The Theorem

For any symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which is square integrable in $\mathcal{X} \times \mathcal{X}$ and which satisfies

$$\int_{\mathcal{X}\times\mathcal{X}} k(x, x')f(x)f(x')dxdx' \geq 0 \text{ for all } f \in L_2(\mathcal{X})$$

there exist $\phi_i : \mathcal{X} \to \mathbb{R}$ and numbers $\lambda_i \geq 0$ where

$$k(x, x') = \sum_i \lambda_i \phi_i(x)\phi_i(x') \text{ for all } x, x' \in \mathcal{X}.$$

Interpretation

Double integral is the continuous version of a vector-matrix-vector multiplication. For positive semidefinite matrices we have

$$\sum \sum k(x_i, x_j)\alpha_i\alpha_j \geq 0$$

Properties

Distance in Feature Space

Distance between points in feature space via

$$\begin{aligned} d(x, x')^2 :=& \|\Phi(x) - \Phi(x')\|^2 \\ =& \langle \Phi(x), \Phi(x) \rangle - 2\langle \Phi(x), \Phi(x') \rangle + \langle \Phi(x'), \Phi(x') \rangle \\ =& k(x, x) + k(x', x') - 2k(x, x) \end{aligned}$$

Kernel Matrix

To compare observations we compute dot products, so we study the matrix K given by

$$K_{ij} = \langle \Phi(x_i), \Phi(x_j) \rangle = k(x_i, x_j)$$

where x_i are the training patterns.

Similarity Measure

The entries K_{ij} tell us the overlap between $\Phi(x_i)$ and $\Phi(x_j)$, so $k(x_i, x_j)$ is a similarity measure.

Properties

K is Positive Semidefinite

Claim: $\alpha^\top K \alpha \geq 0$ for all $\alpha \in \mathbb{R}^m$ and all kernel matrices $K \in \mathbb{R}^{m \times m}$. Proof:

$$\sum_{i,j}^{m} \alpha_i \alpha_j K_{ij} = \sum_{i,j}^{m} \alpha_i \alpha_j \langle \Phi(x_i), \Phi(x_j) \rangle$$

$$= \left\langle \sum_{i}^{m} \alpha_i \Phi(x_i), \sum_{j}^{m} \alpha_j \Phi(x_j) \right\rangle = \left\| \sum_{i=1}^{m} \alpha_i \Phi(x_i) \right\|^2$$

Kernel Expansion

If w is given by a linear combination of $\Phi(x_i)$ we get

$$\langle w, \Phi(x) \rangle = \left\langle \sum_{i=1}^{m} \alpha_i \Phi(x_i), \Phi(x) \right\rangle = \sum_{i=1}^{m} \alpha_i k(x_i, x).$$

A Counterexample

A Candidate for a Kernel

$$k(x, x') = \begin{cases} 1 & \text{if } \|x - x'\| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

This is symmetric and gives us some information about the proximity of points, yet it is not a proper kernel . . .

Kernel Matrix

We use three points, $x_1 = 1, x_2 = 2, x_3 = 3$ and compute the resulting “kernelmatrix” K. This yields

$$K = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$ and eigenvalues $(\sqrt{2}-1)^{-1}, 1$ and $(1-\sqrt{2})$.

as eigensystem. Hence k is not a kernel.

Examples

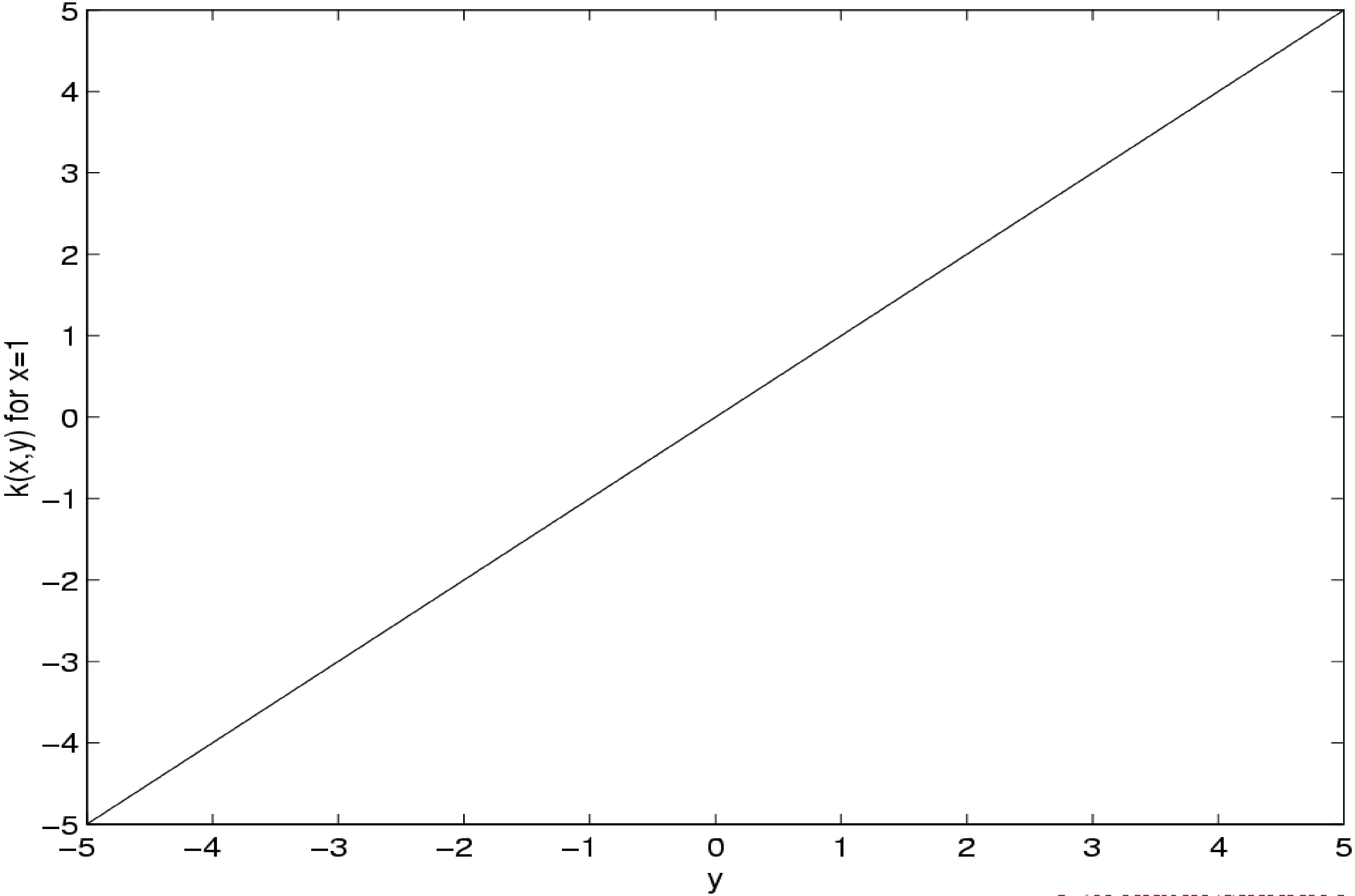
Examples of kernels $k(x, x')$

Linear	$\langle x, x' \rangle$
Laplacian RBF	$\exp\left(-\lambda \|x - x'\|\right)$
Gaussian RBF	$\exp\left(-\lambda \|x - x'\|^2\right)$
Polynomial	$\left(\langle x, x' \rangle + c \rangle\right)^d, c \geq 0,\ d \in \mathbb{N}$
B-Spline	$B_{2n+1}(x - x')$
Cond. Expectation	$\mathbf{E}_c[p(x\|c)p(x'\|c)]$

Simple trick for checking Mercer's condition

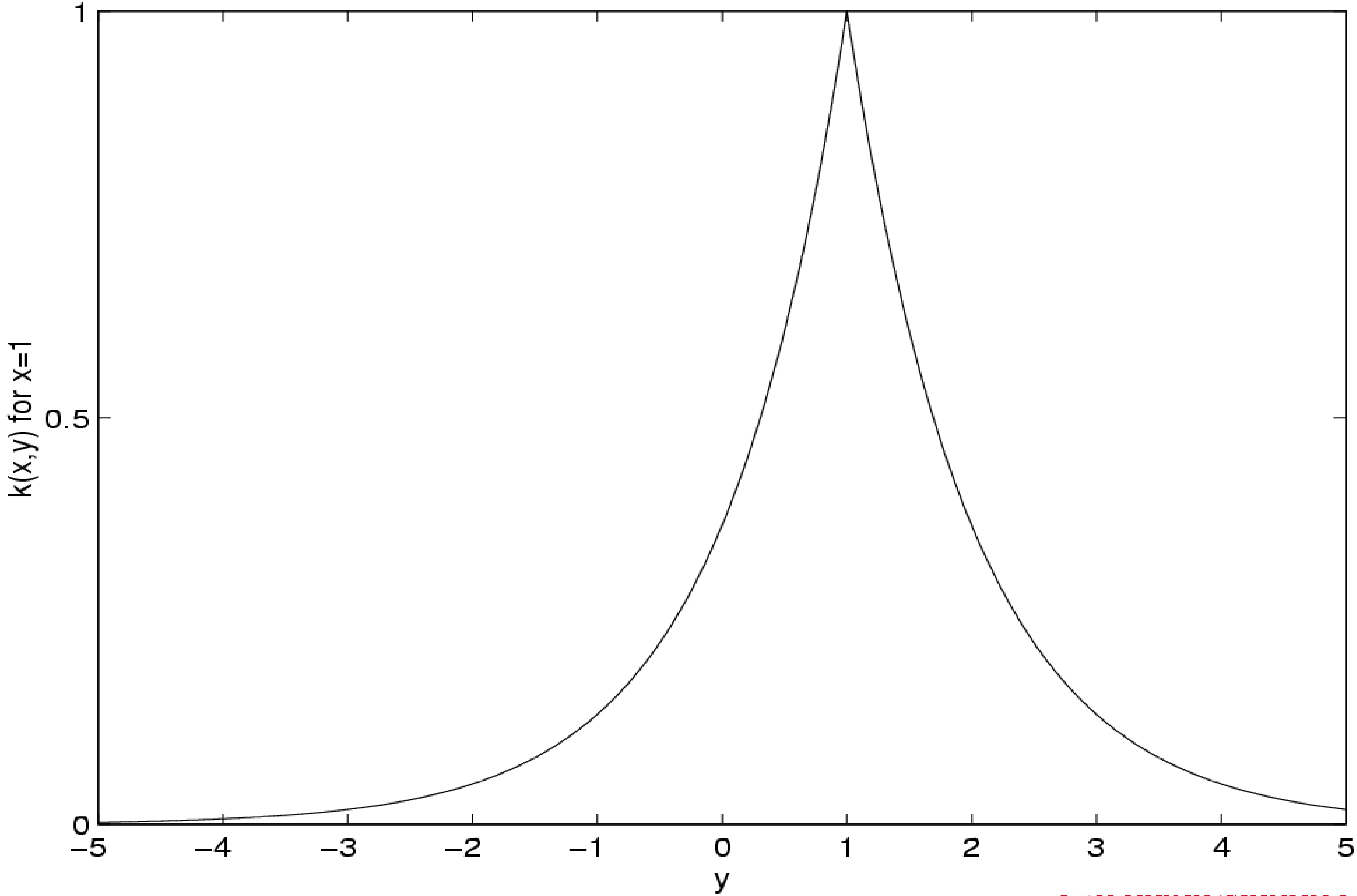
Compute the Fourier transform of the kernel and check that it is nonnegative.

Linear Kernel

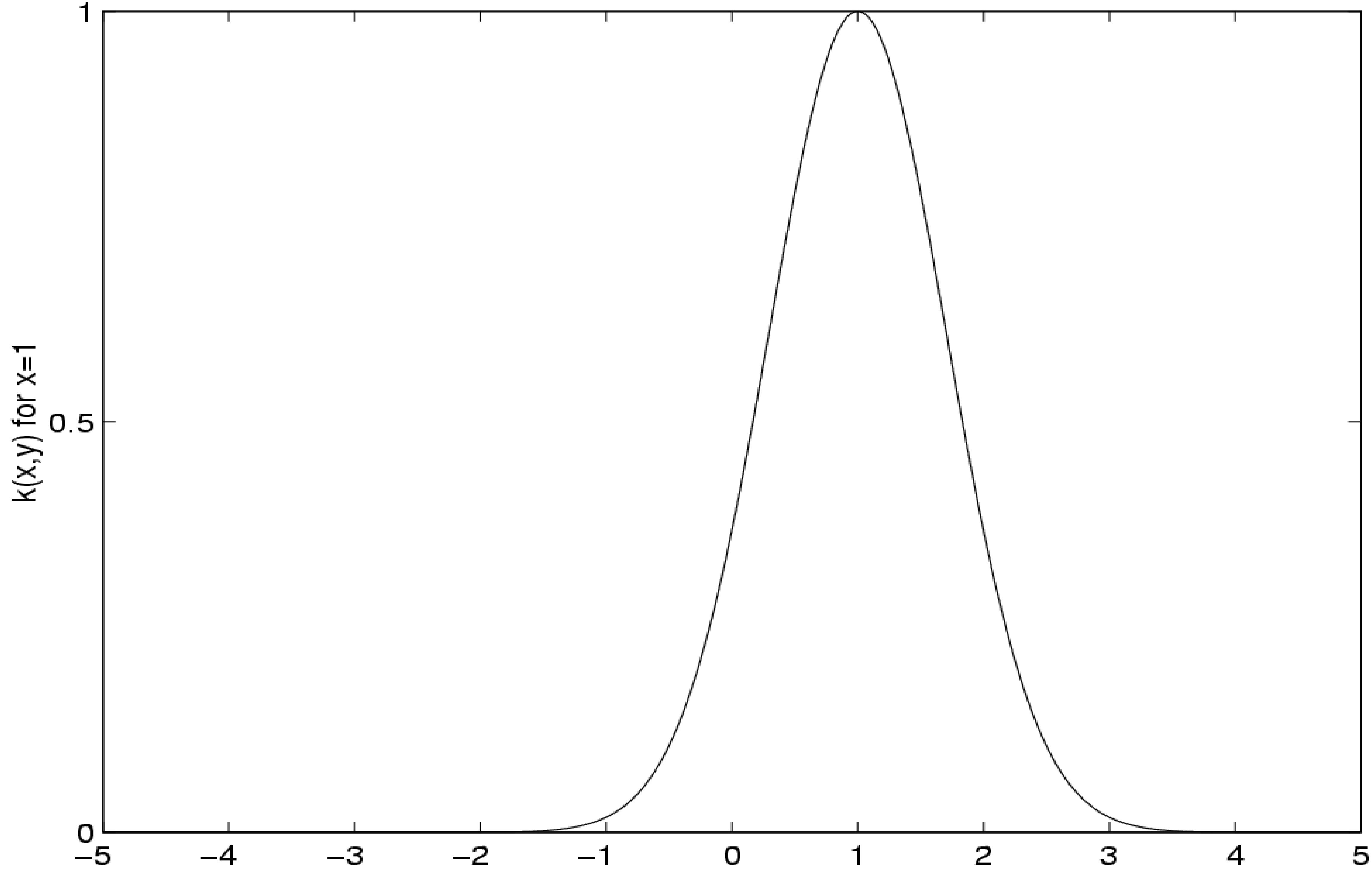


Carnegie Mellon University

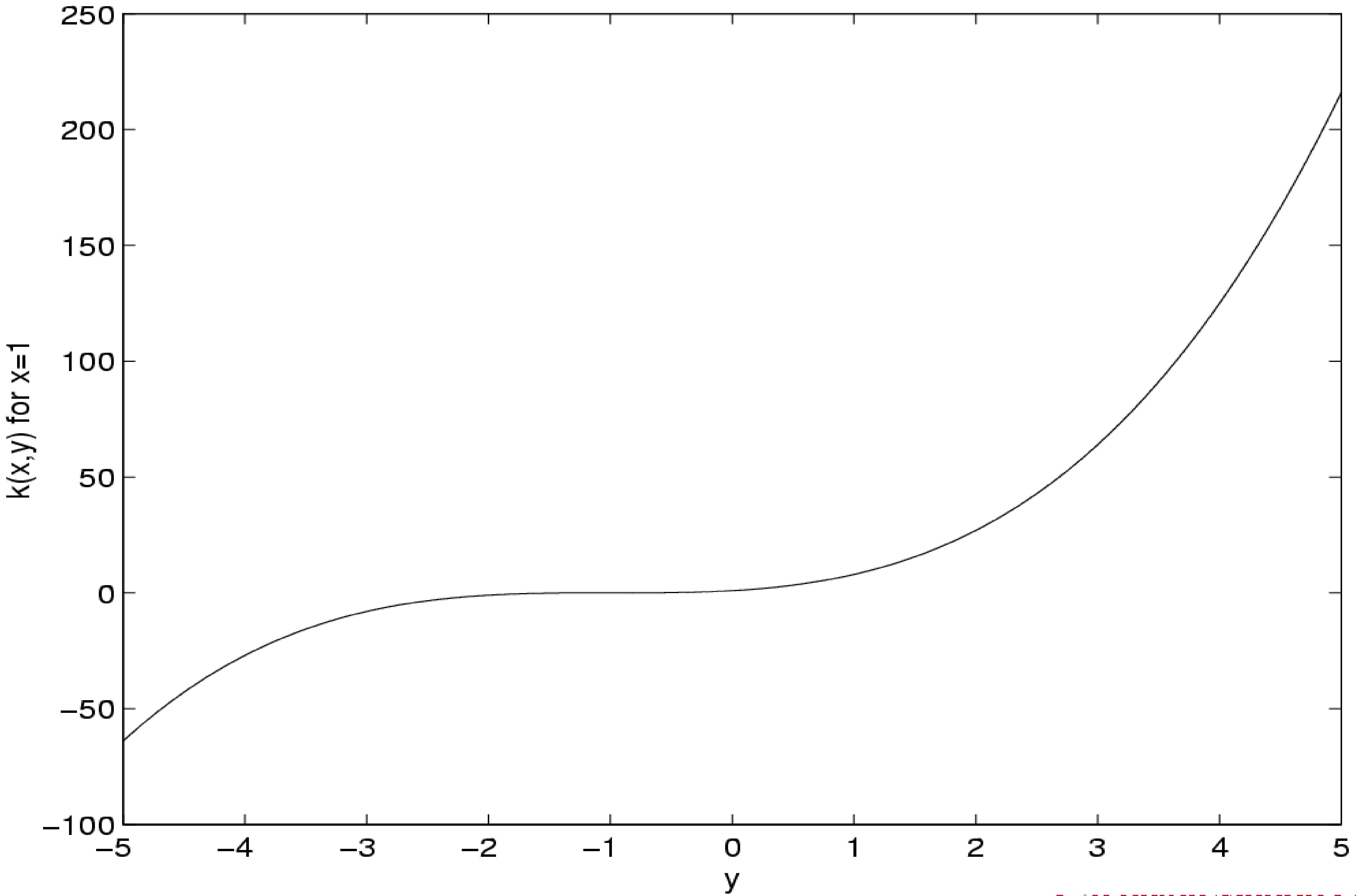
Laplacian Kernel



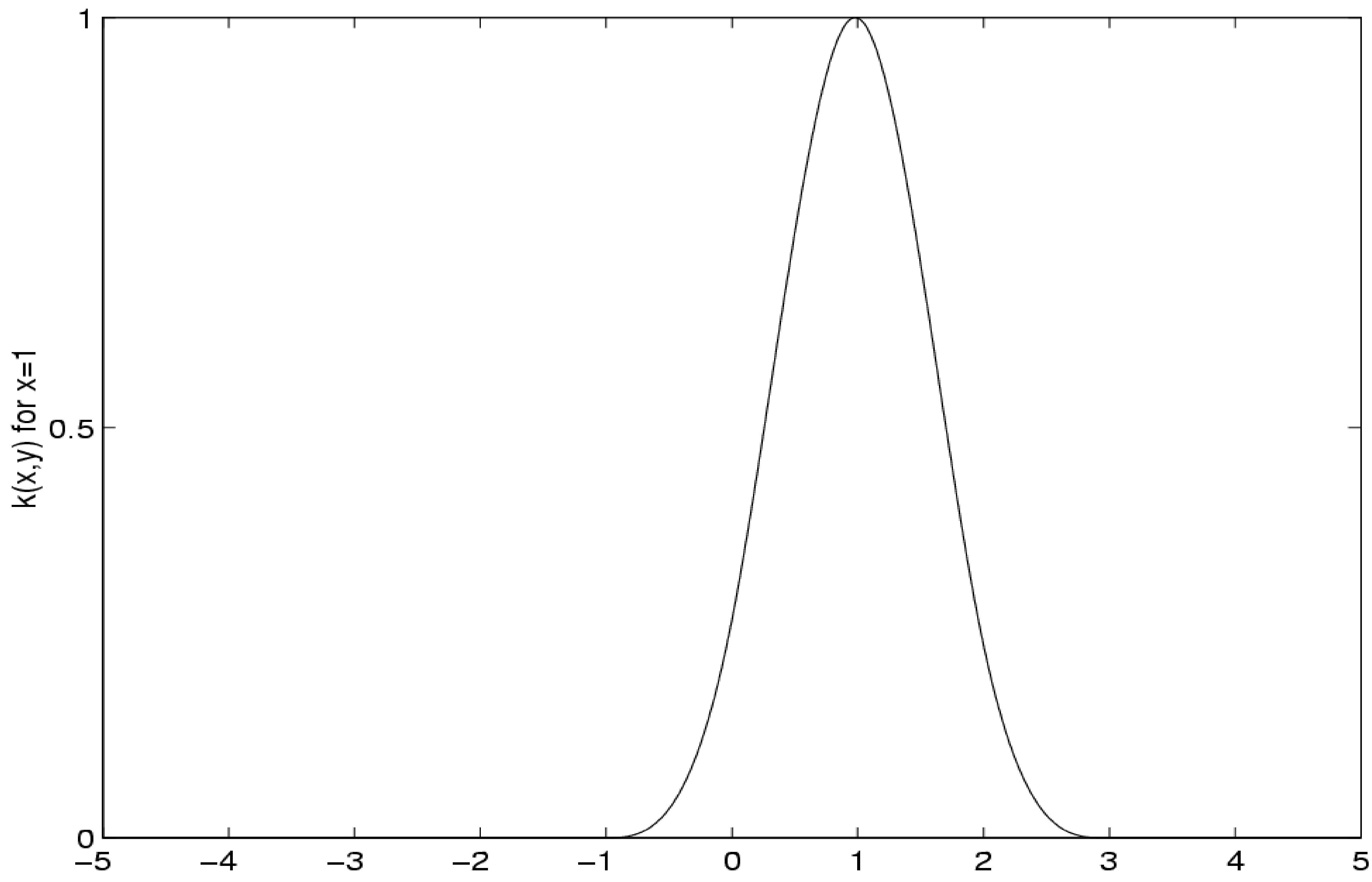
Gaussian Kernel



Polynomial of order 3



B_3 Spline Kernel



Carnegie Mellon University

4.3 Support Vector Machines

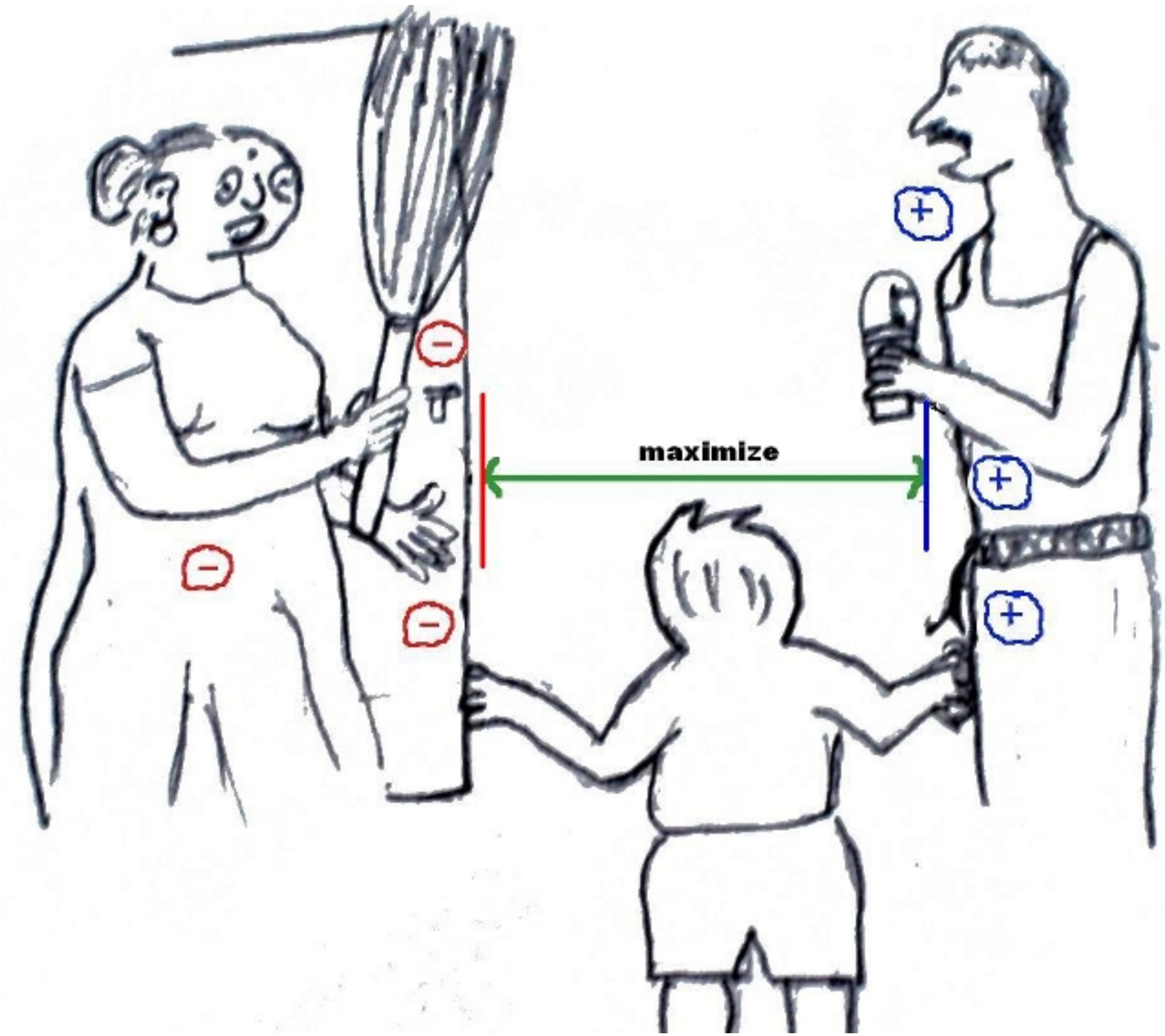
4 (Generalized) Linear Methods

Alexander Smola

Introduction to Machine Learning 10-701

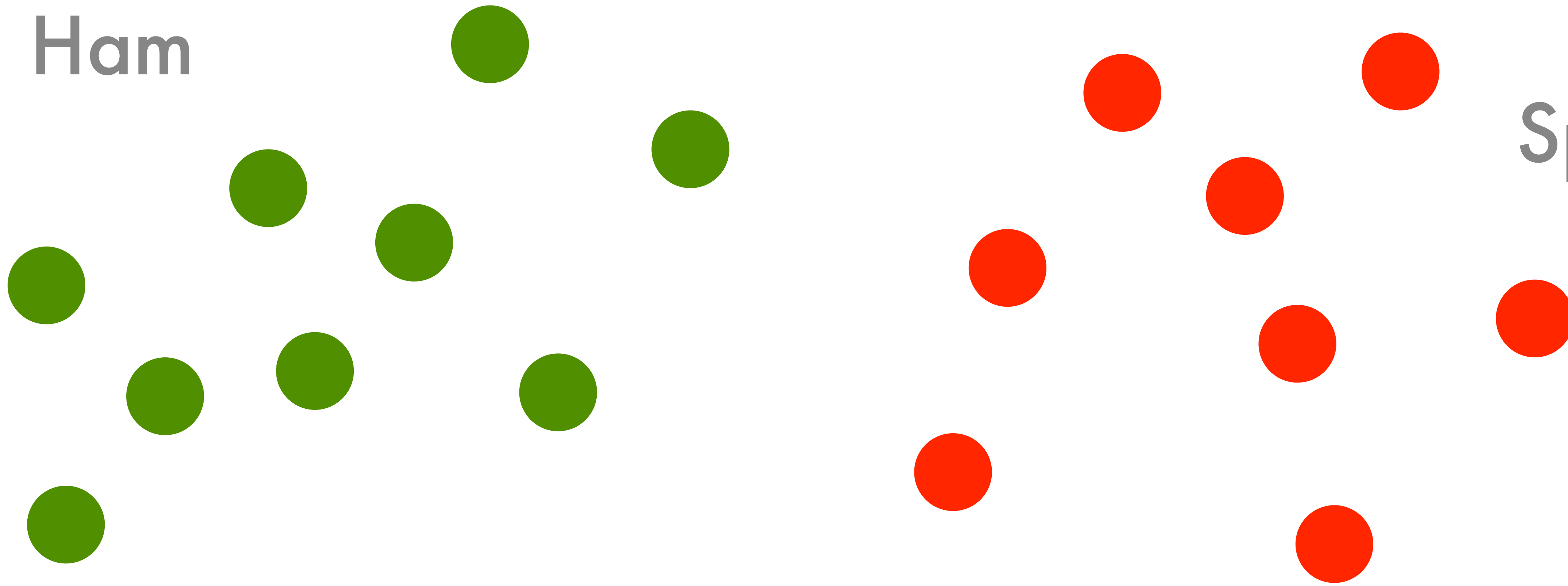
http://alex.smola.org/teaching/10-701-15

MACHINE LEARNING DEPARTMENT
Carnegie Mellon University

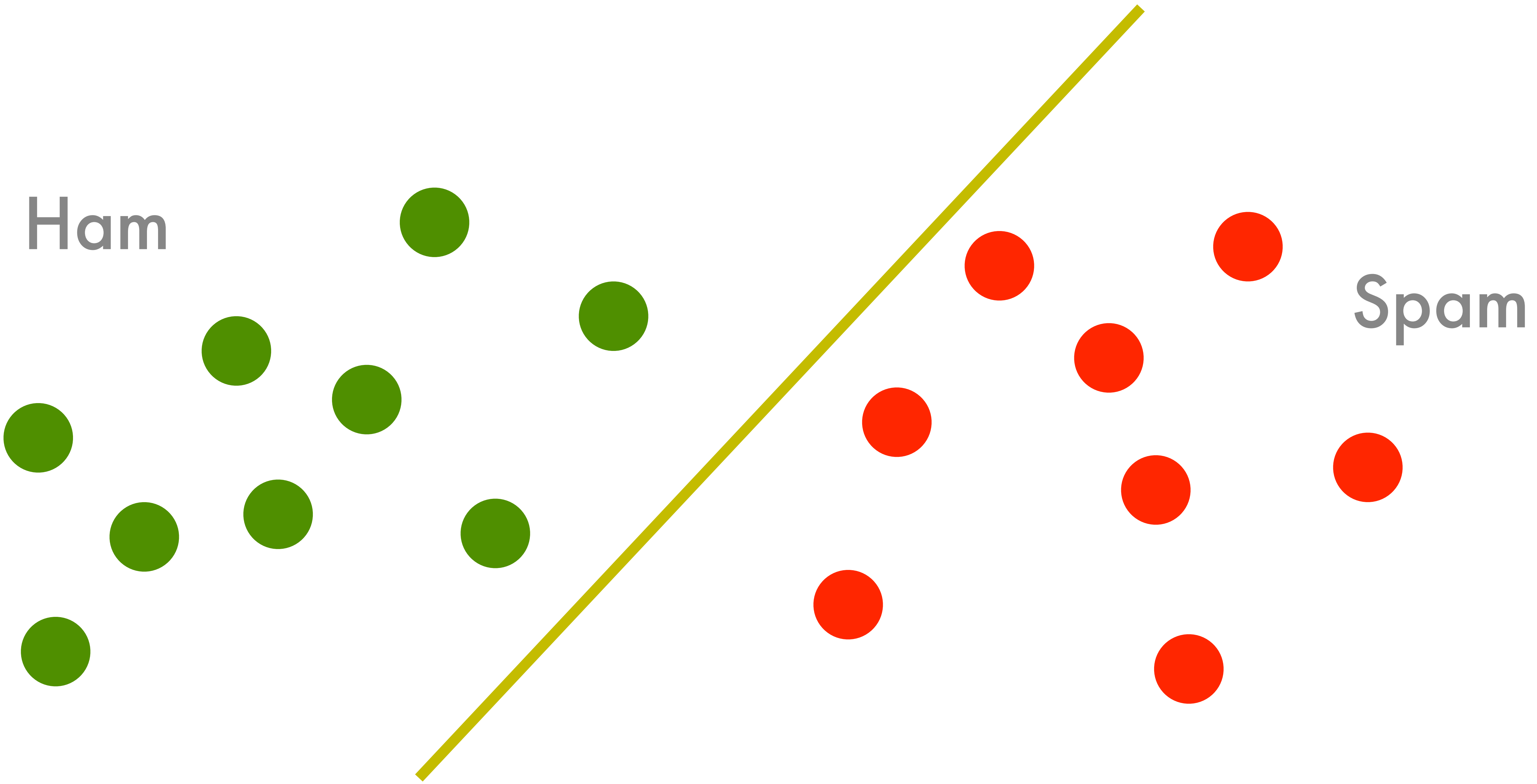


Support Vector Machines

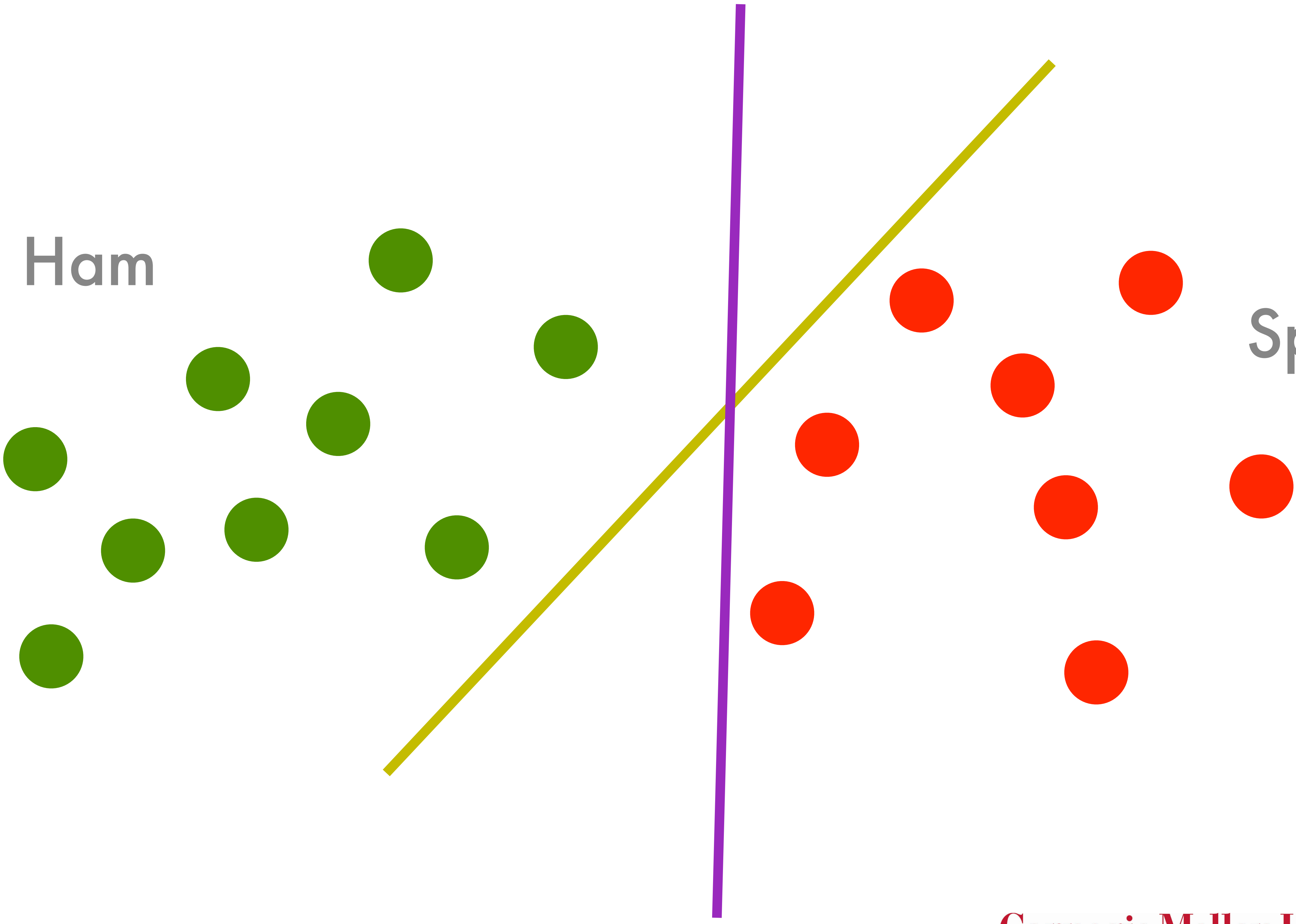
Linear Separator



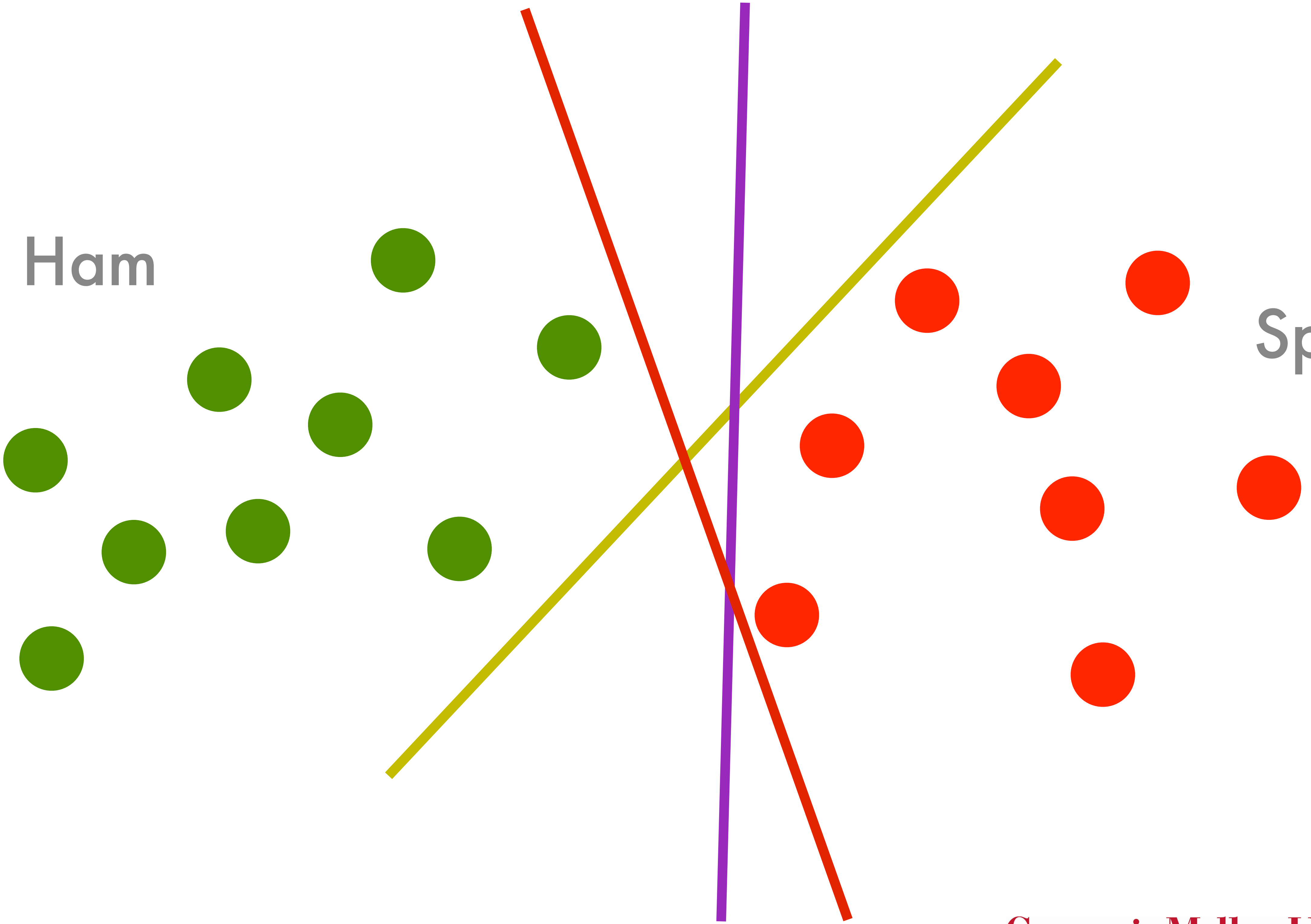
Linear Separator



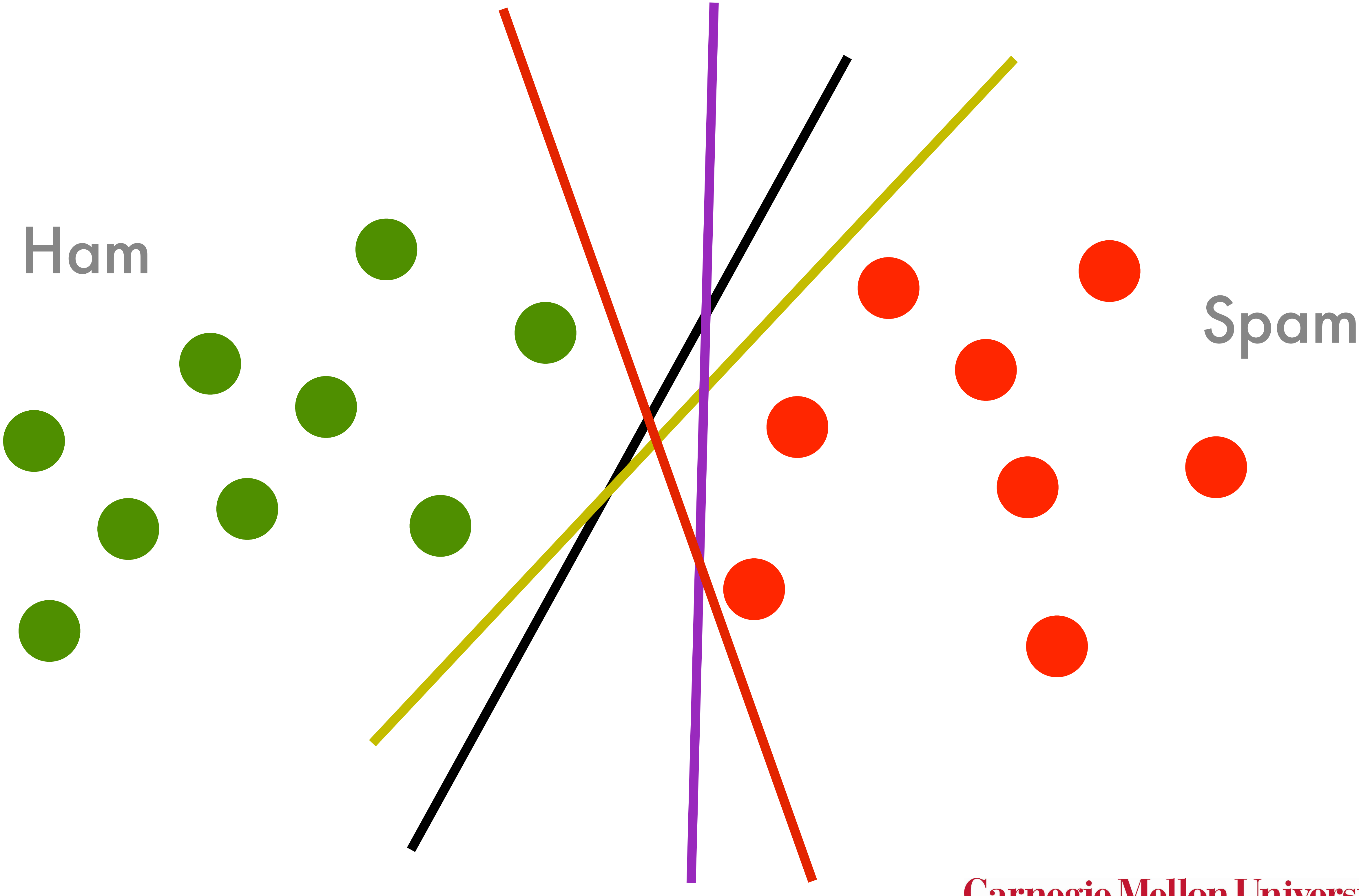
Linear Separator



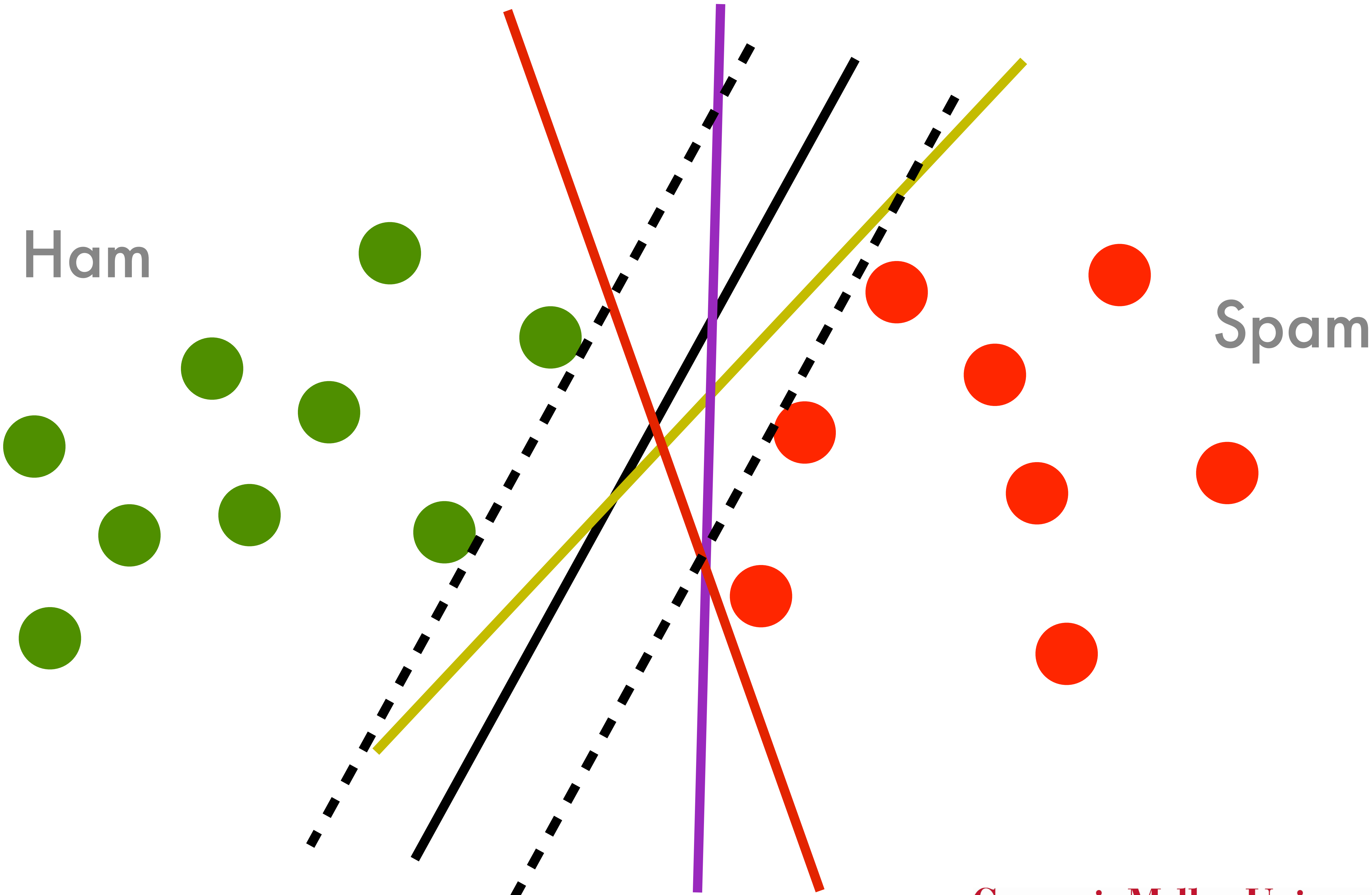
Linear Separator



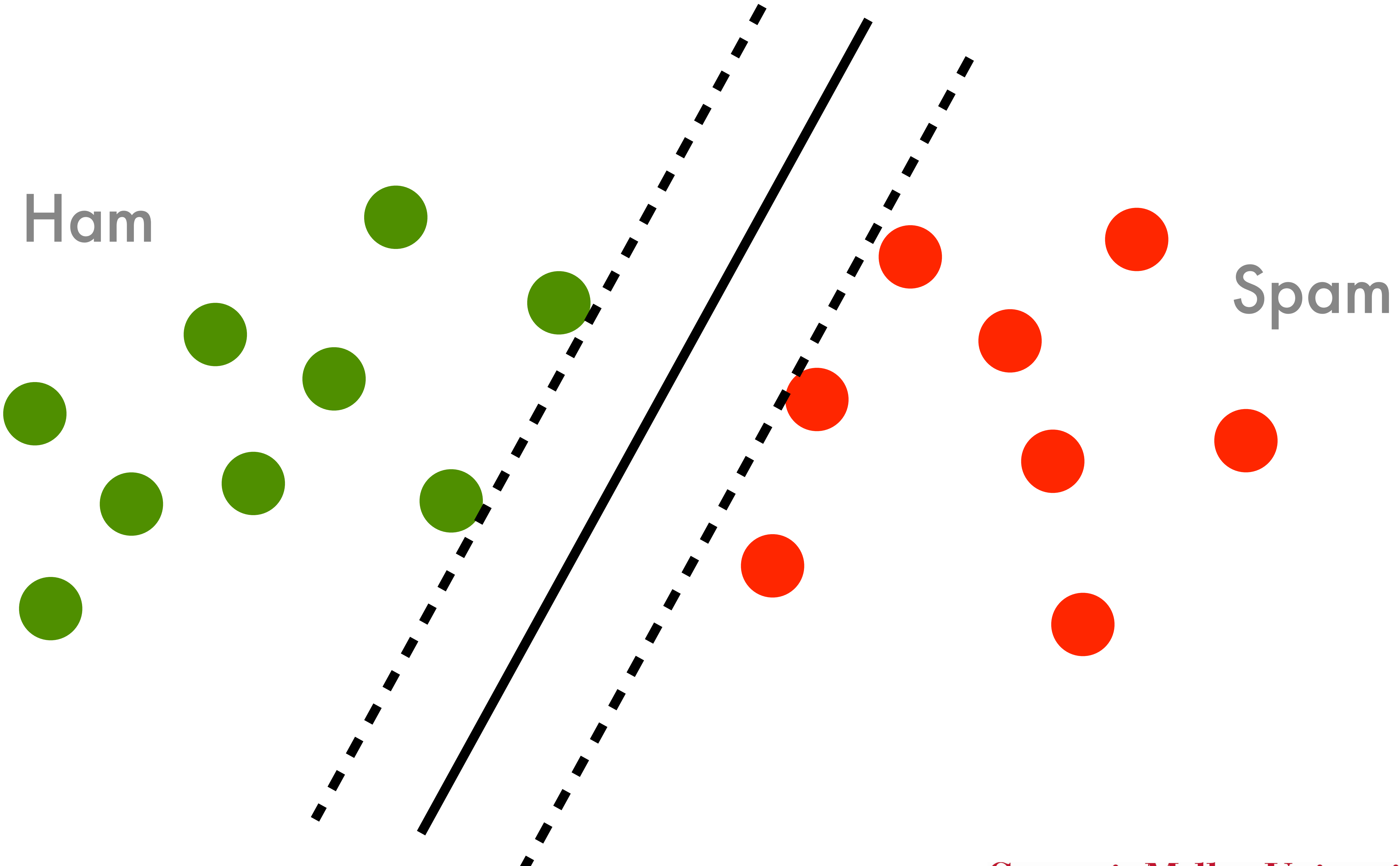
Linear Separator



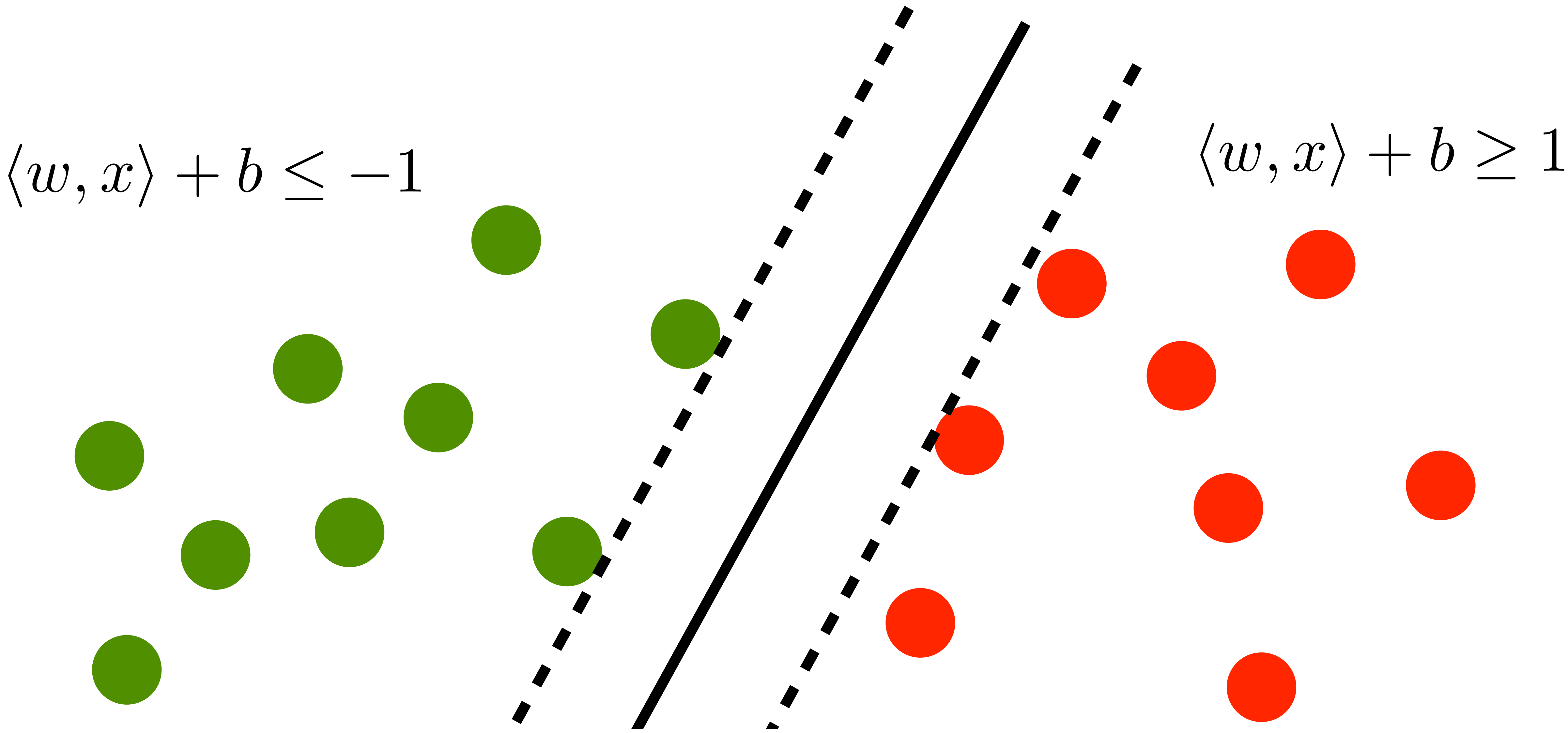
Linear Separator



Linear Separator



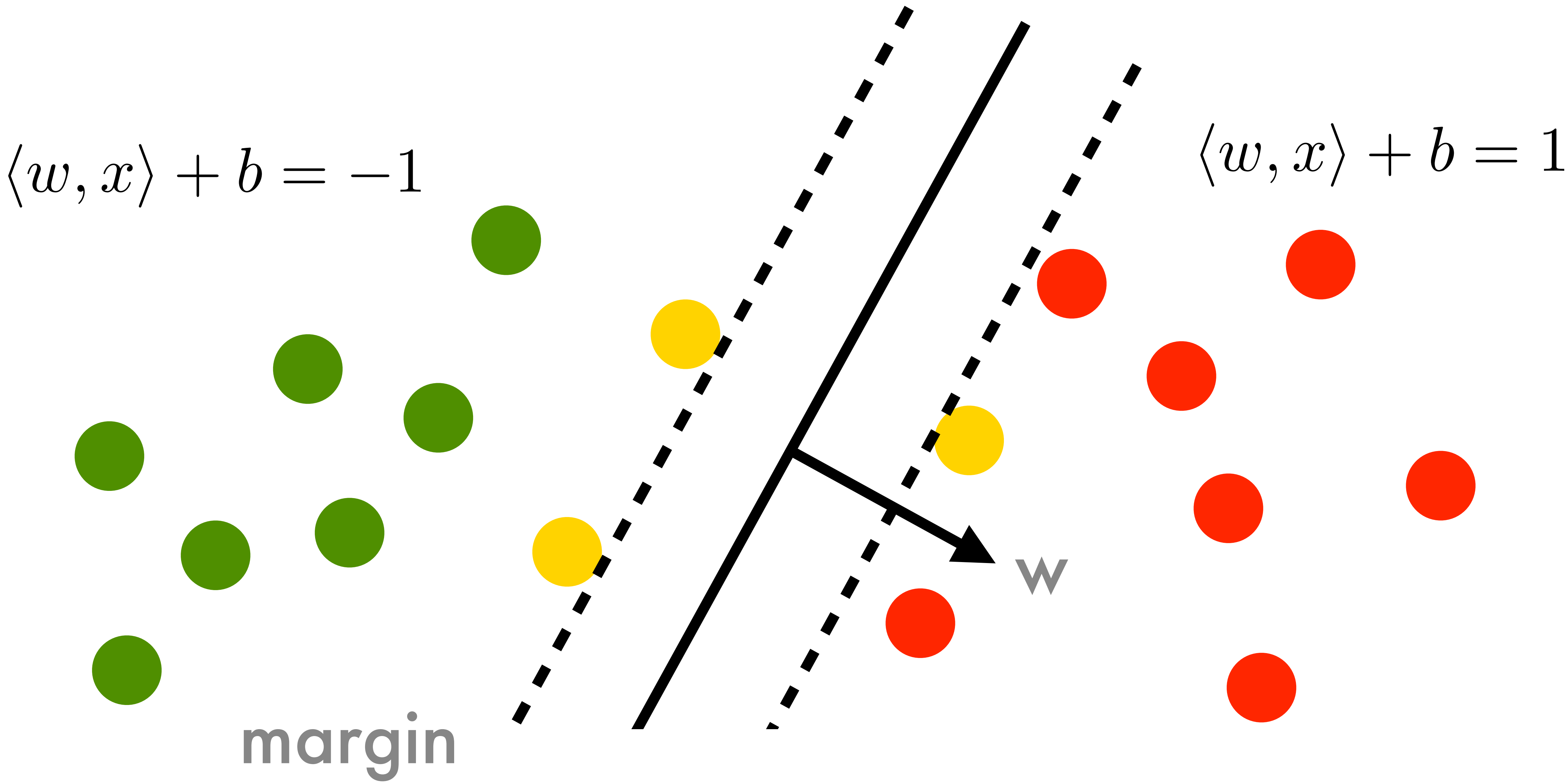
Large Margin Classifier



linear function

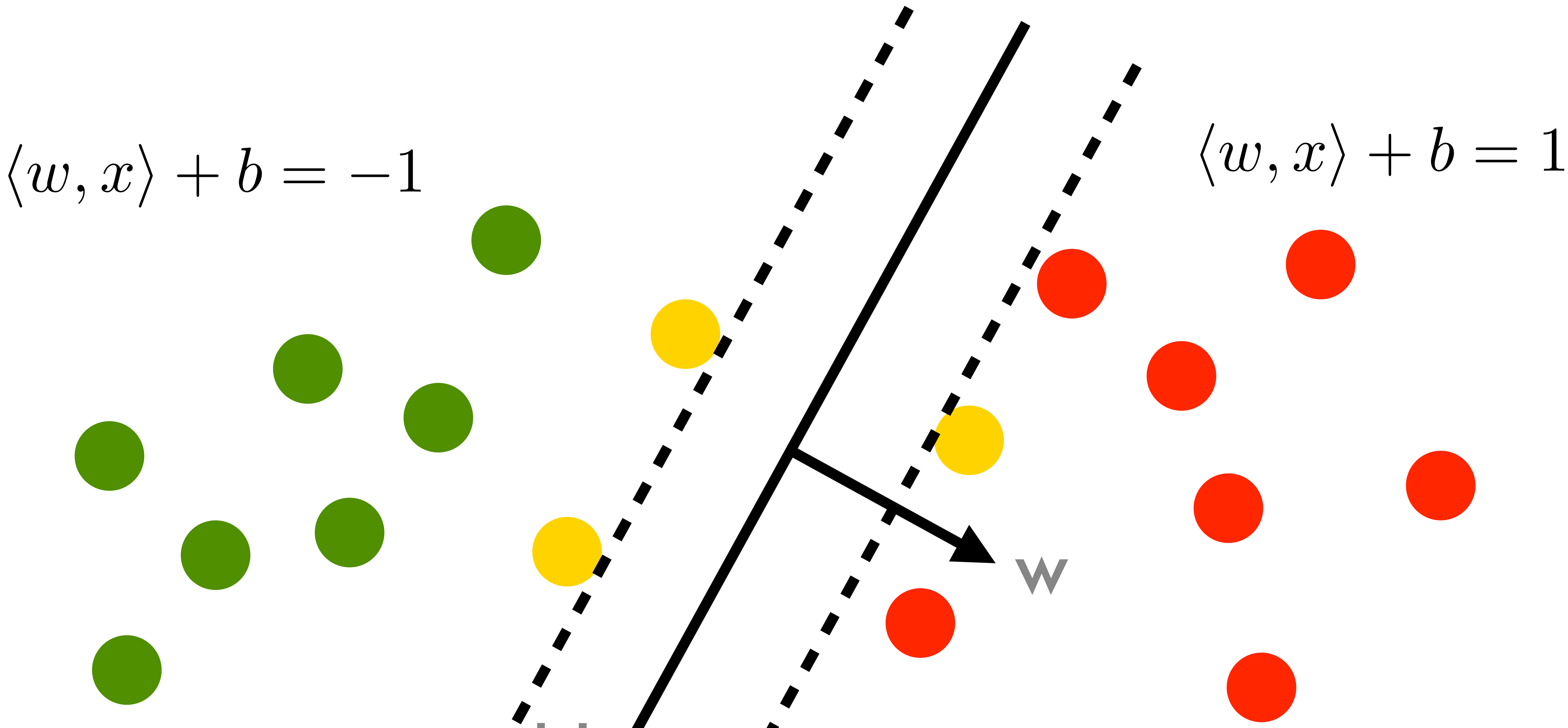
$$f(x) = \langle w, x \rangle + b$$

Large Margin Classifier



$$\frac{\langle x_+ - x_-, w\rangle}{2\|w\|} = \frac{1}{2\|w\|}\left[\left[\langle x_+, w\rangle + b\right] - \left[\langle x_-, w\rangle + b\right]\right] = \frac{1}{\|w\|}$$

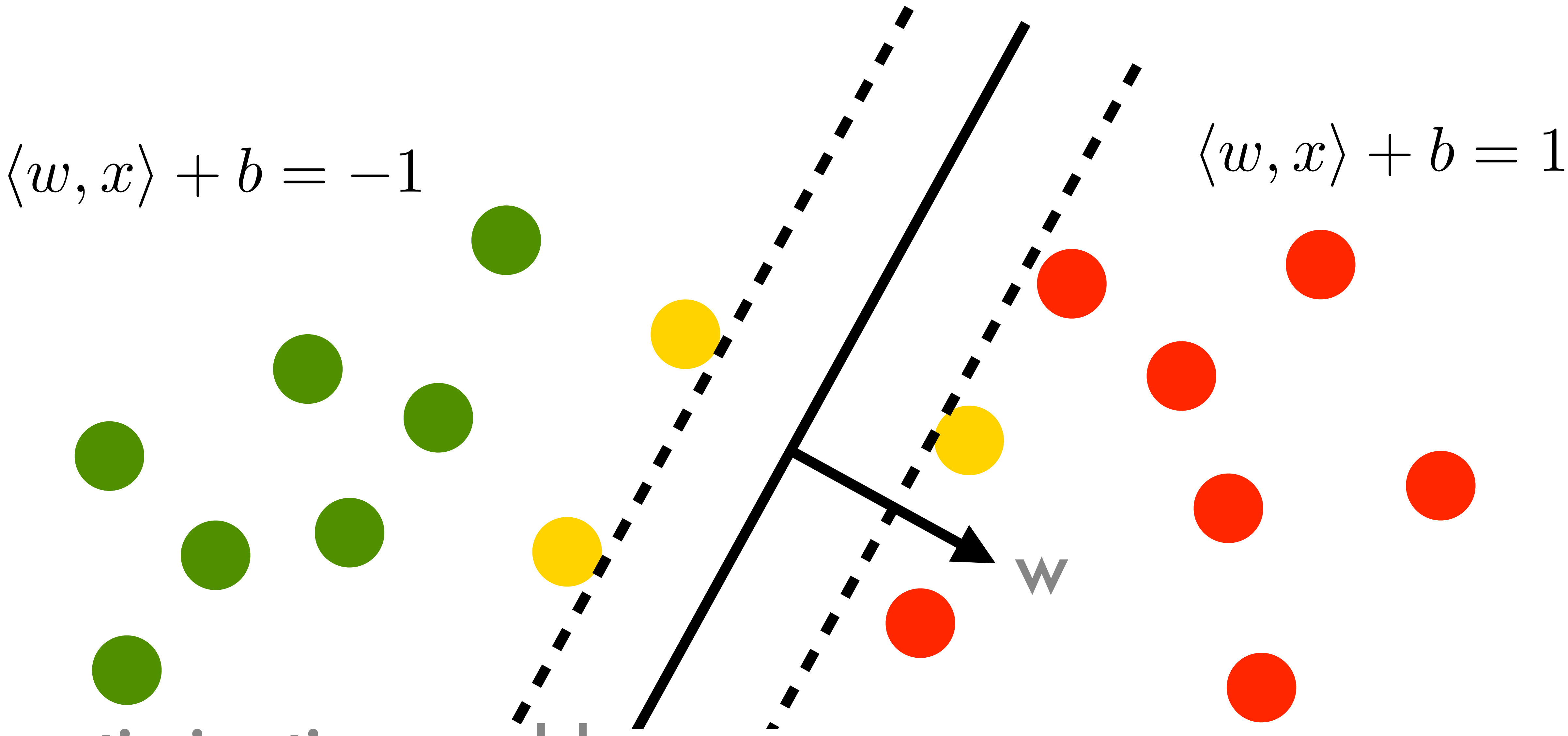
Large Margin Classifier



optimization problem

$$\underset{w,b}{\text{maximize}} \frac{1}{\|w\|} \text{ subject to } y_i \left[\langle x_i, w \rangle + b\right] \geq 1$$

Large Margin Classifier



optimization problem

$$\underset{w,b}{\text{minimize}} \frac{1}{2} \|w\|^2 \text{ subject to } y_i \left[\langle x_i, w \rangle + b \right] \geq 1$$

Dual Problem

- Primal optimization problem

$$\underset{w,b}{\text{minimize}}\ \frac{1}{2}\|w\|^2 \text{ subject to } y_i\left[\langle x_i, w\rangle + b\right] \geq 1$$

- Lagrange function

constraint

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i \left[y_i\left[\langle x_i, w\rangle + b\right] - 1\right]$$

Optimality in w, b is at saddle point with α

- Derivatives in w, b need to vanish

Dual Problem

- Lagrange function

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i \left[y_i \left[\langle x_i, w\rangle + b\right] - 1\right]$$

- Derivatives in w, b need to vanish

$$\partial_w L(w, b, a) = w - \sum_i \alpha_i y_i x_i = 0$$

$$\partial_b L(w, b, a) = \sum_i \alpha_i y_i = 0$$

- Plugging terms back into L yields

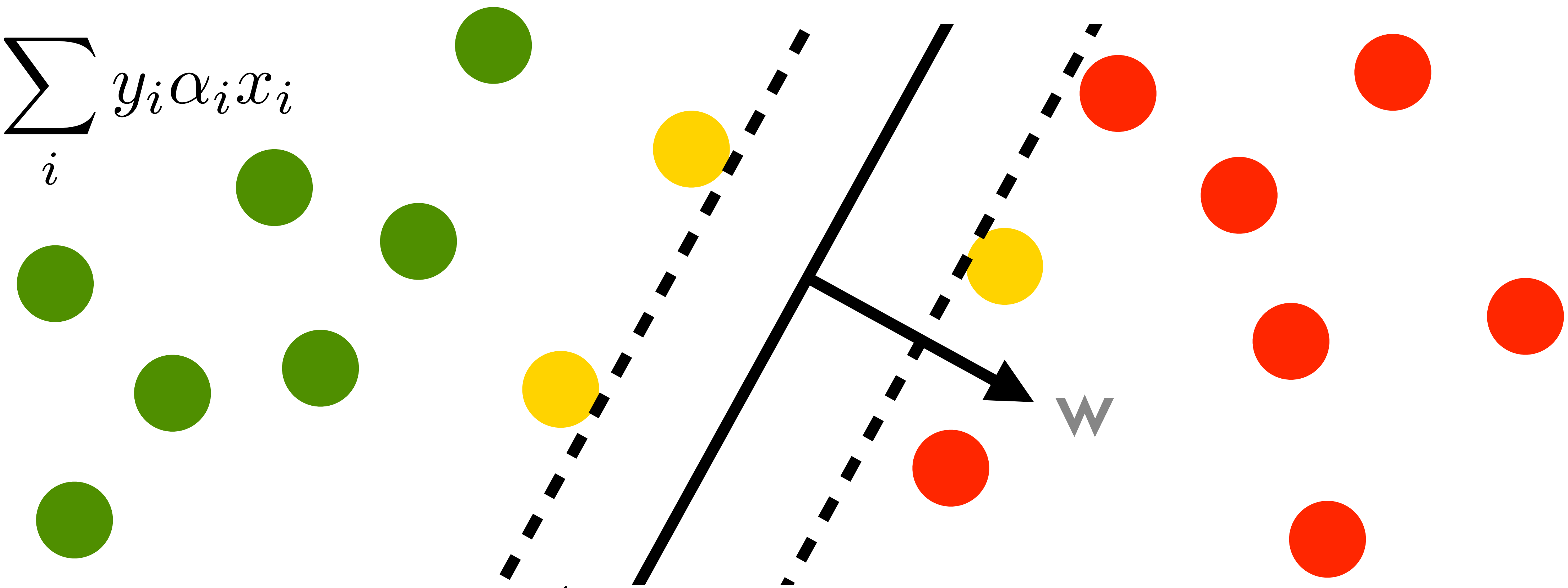
$$\underset{\alpha}{\text{maximize}} - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j\rangle + \sum_i \alpha_i$$

$$\text{subject to } \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0$$

Support Vector Machines

$$\underset{w,b}{\text{minimize}}\ \frac{1}{2}\|w\|^2 \text{ subject to } y_i\left[\langle x_i, w\rangle + b\right] \geq 1$$

$$w = \sum_i y_i \alpha_i x_i$$



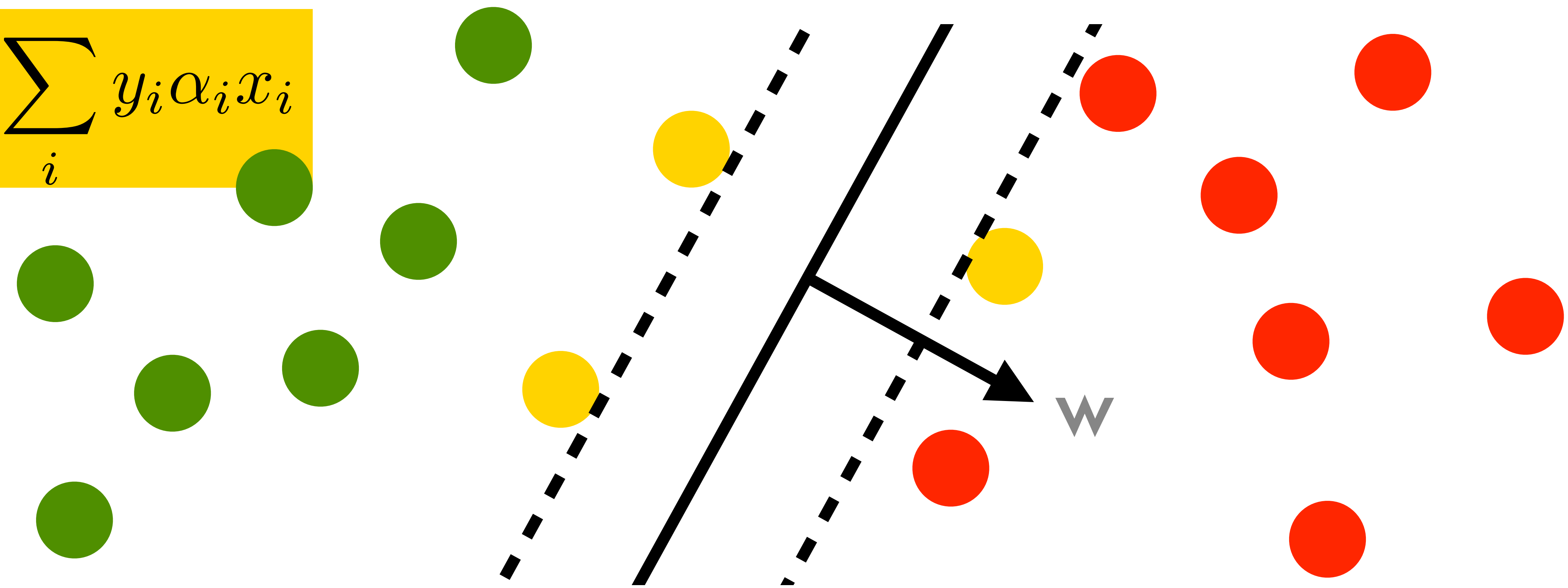
$$\underset{\alpha}{\text{maximize}} - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle + \sum_i \alpha_i$$

$$\text{subject to } \sum \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0$$

Support Vectors

$$\underset{w,b}{\text{minimize}} \; \frac{1}{2}\|w\|^2 \text{ subject to } y_i\left[\langle x_i, w\rangle + b\right] \geq 1$$

$$w = \sum_i y_i \alpha_i x_i$$



Karush Kuhn Tucker
Optimality condition

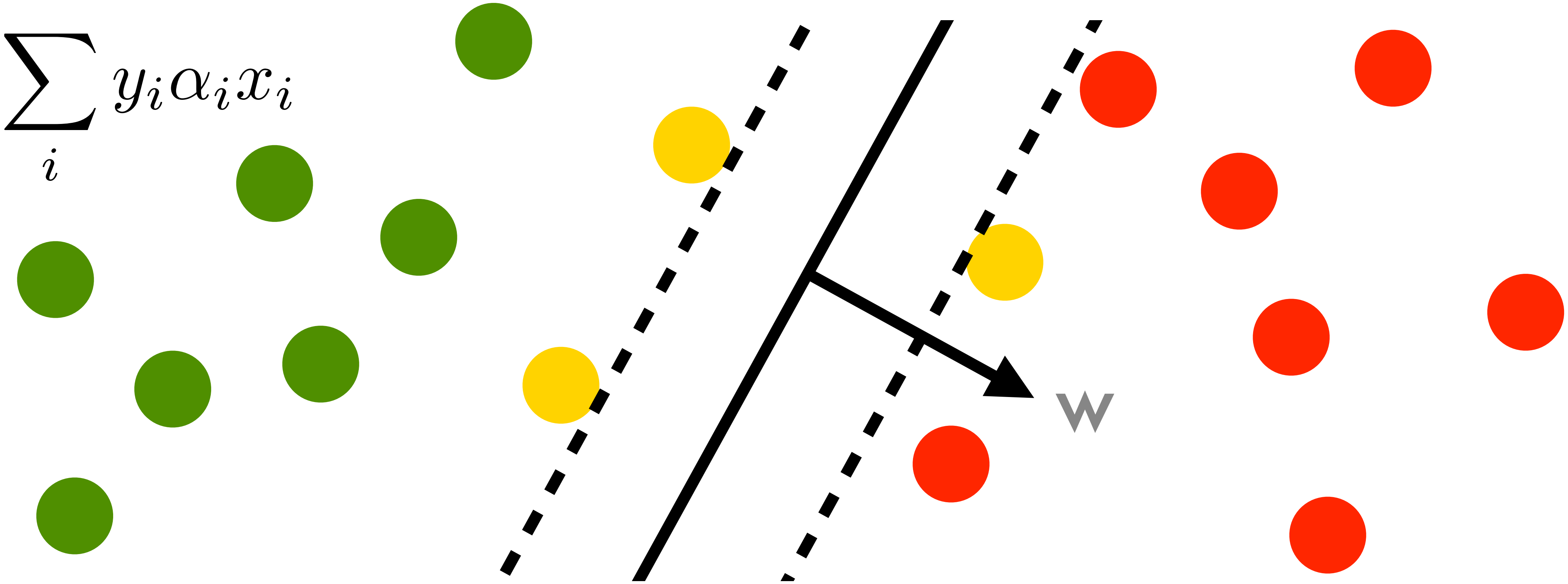
$$\alpha_i\left[y_i\left[\langle w, x_i\rangle + b\right] - 1\right] = 0$$

$$\alpha_i = 0$$

$$\alpha_i > 0 \Longrightarrow y_i\left[\langle w, x_i\rangle + b\right] = 1$$

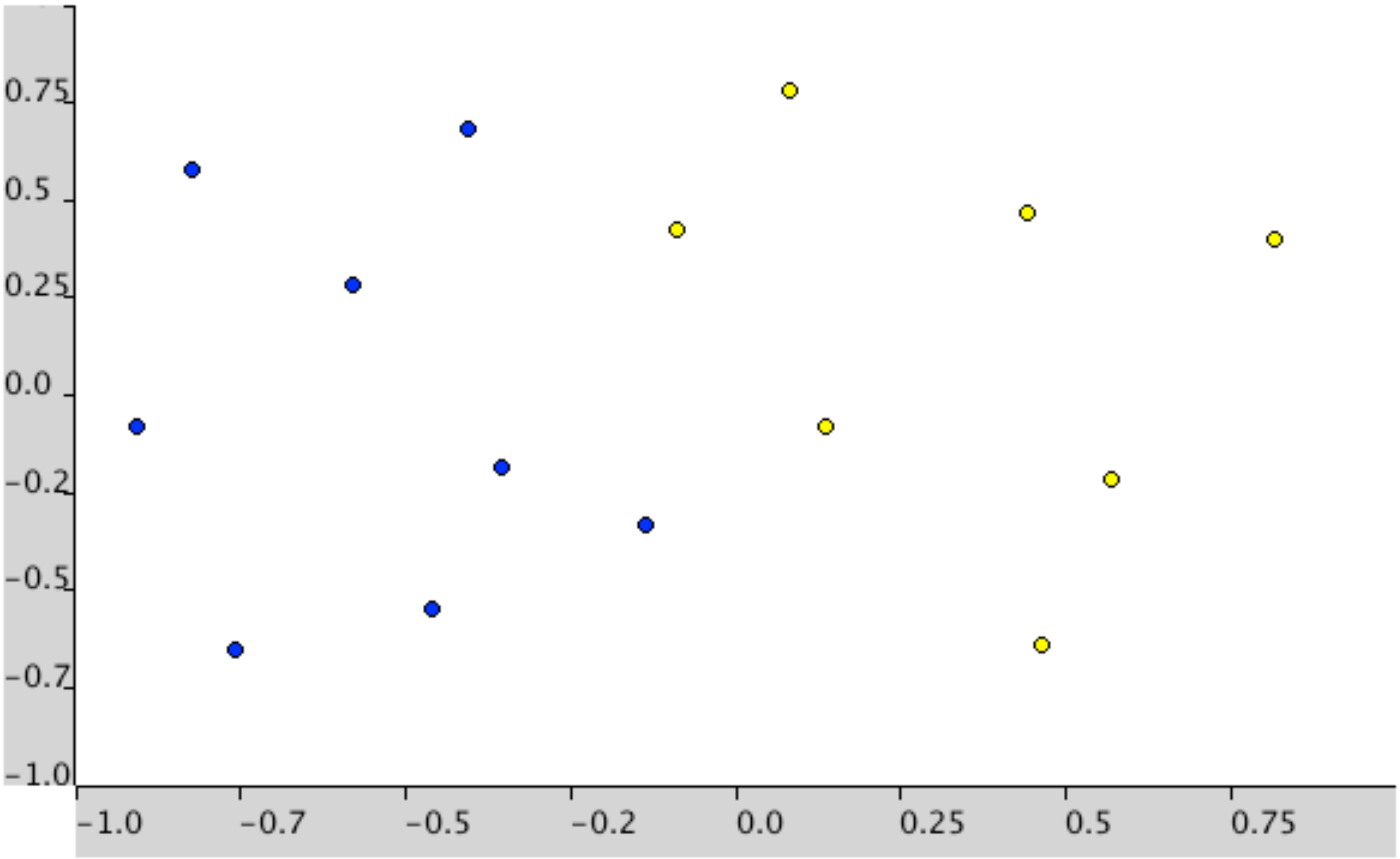
Properties

$$w = \sum_i y_i \alpha_i x_i$$



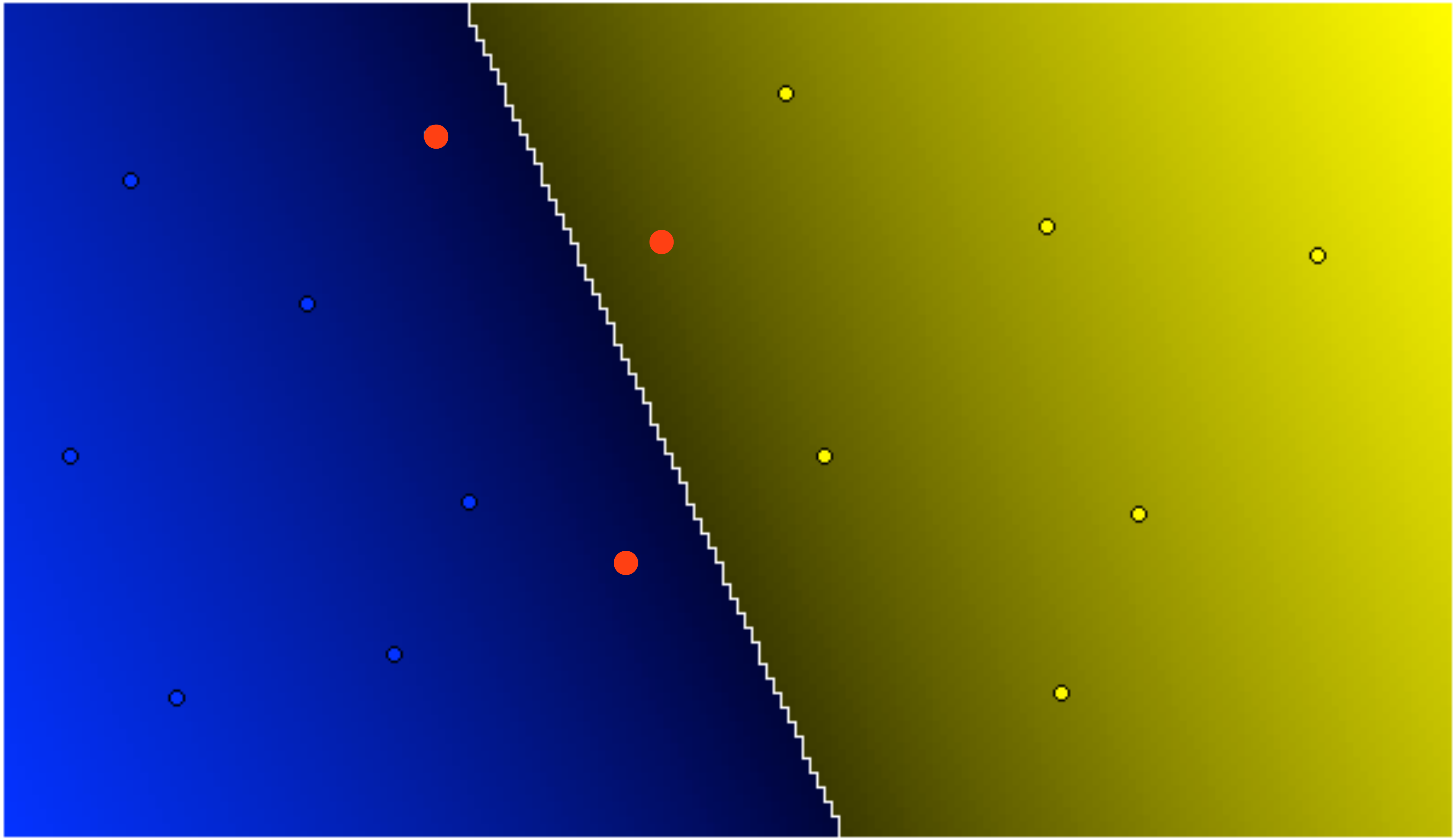
- Weight vector w as weighted linear combination of instances
- Only points on margin matter (ignore the rest and get same solution)
- Only inner products matter
  - Quadratic program
  - We can replace the inner product by a kernel
- Keeps instances away from the margin

Example



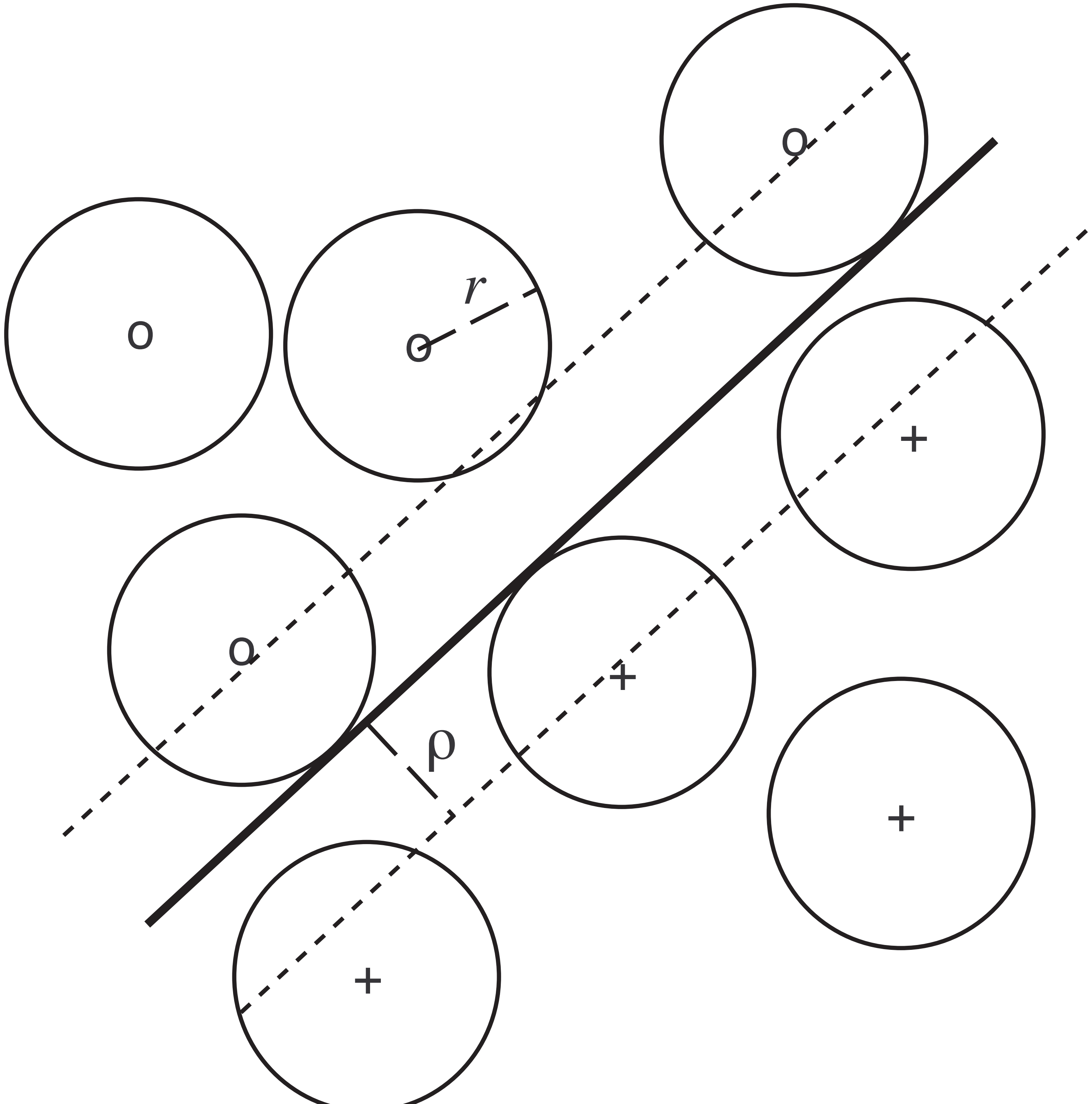
Example

Number of Support Vectors: **3** (-ve: 2, +ve: 1) Total number of points: 15



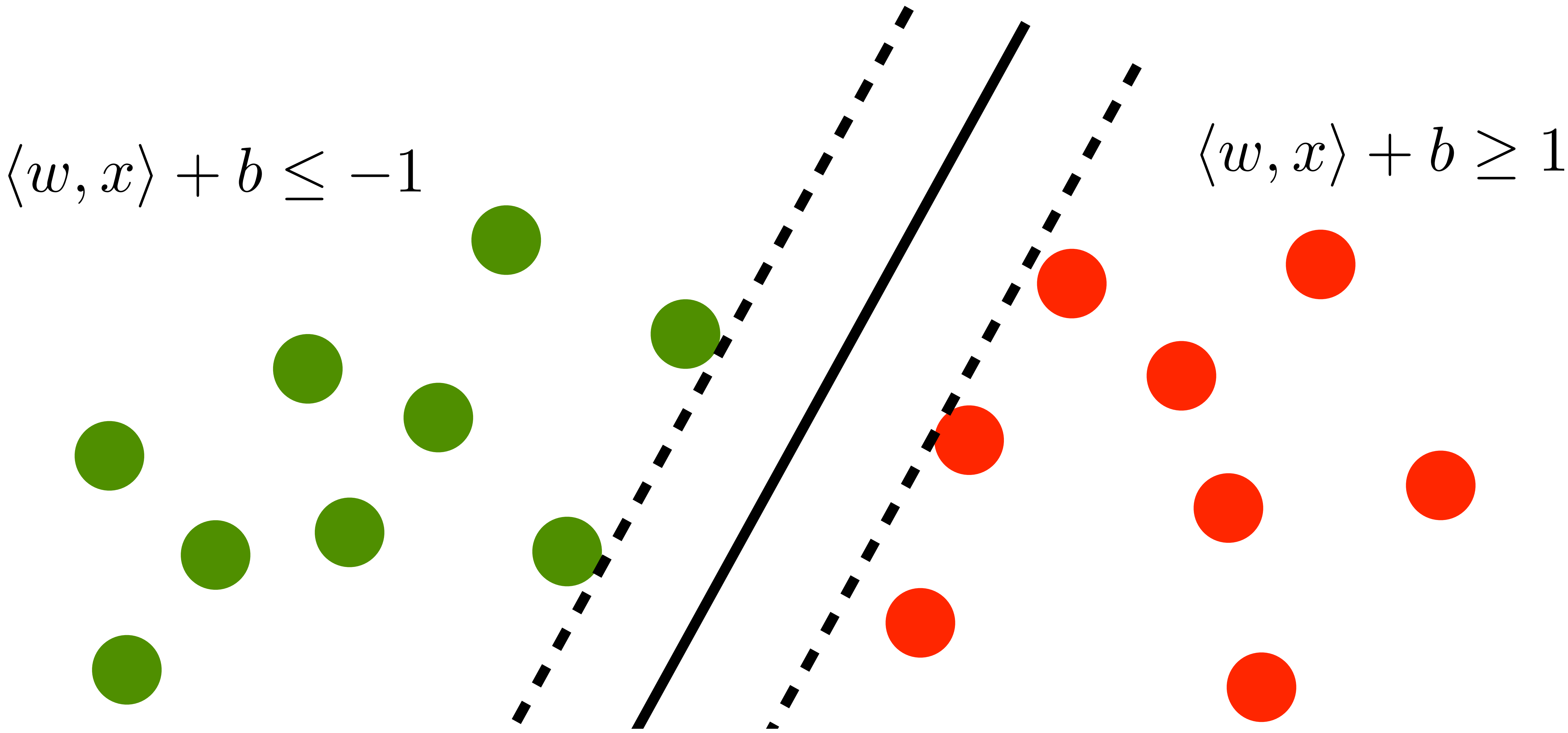
ersity

Why large margins?




SOFT
CLASSIFIERS

Large Margin Classifier



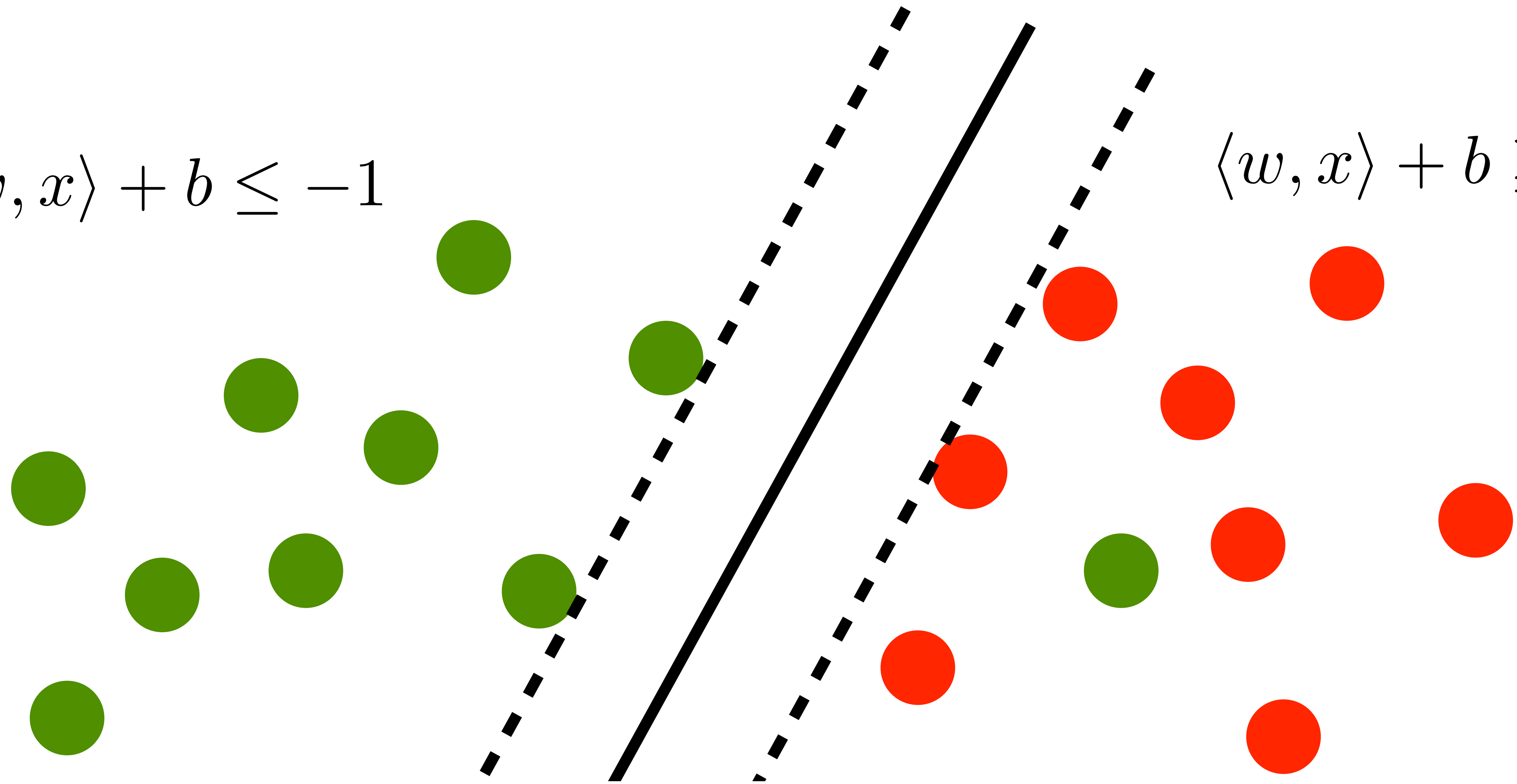
linear function

$$f(x) = \langle w, x \rangle + b$$

Large Margin Classifier

$\langle w, x \rangle + b \leq -1$

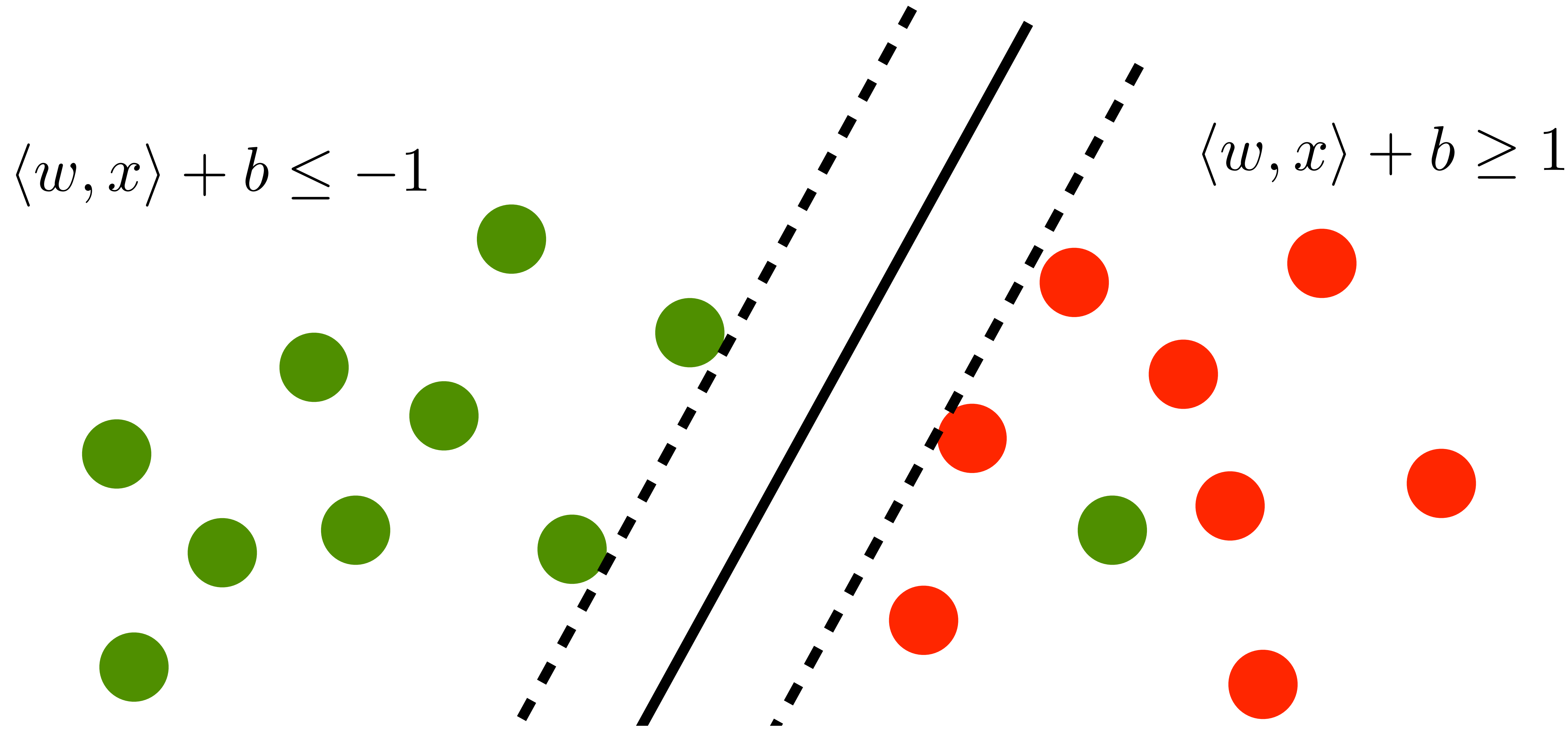
$\langle w, x \rangle + b \geq 1$



linear function

$$f(x) = \langle w, x \rangle + b$$

Large Margin Classifier

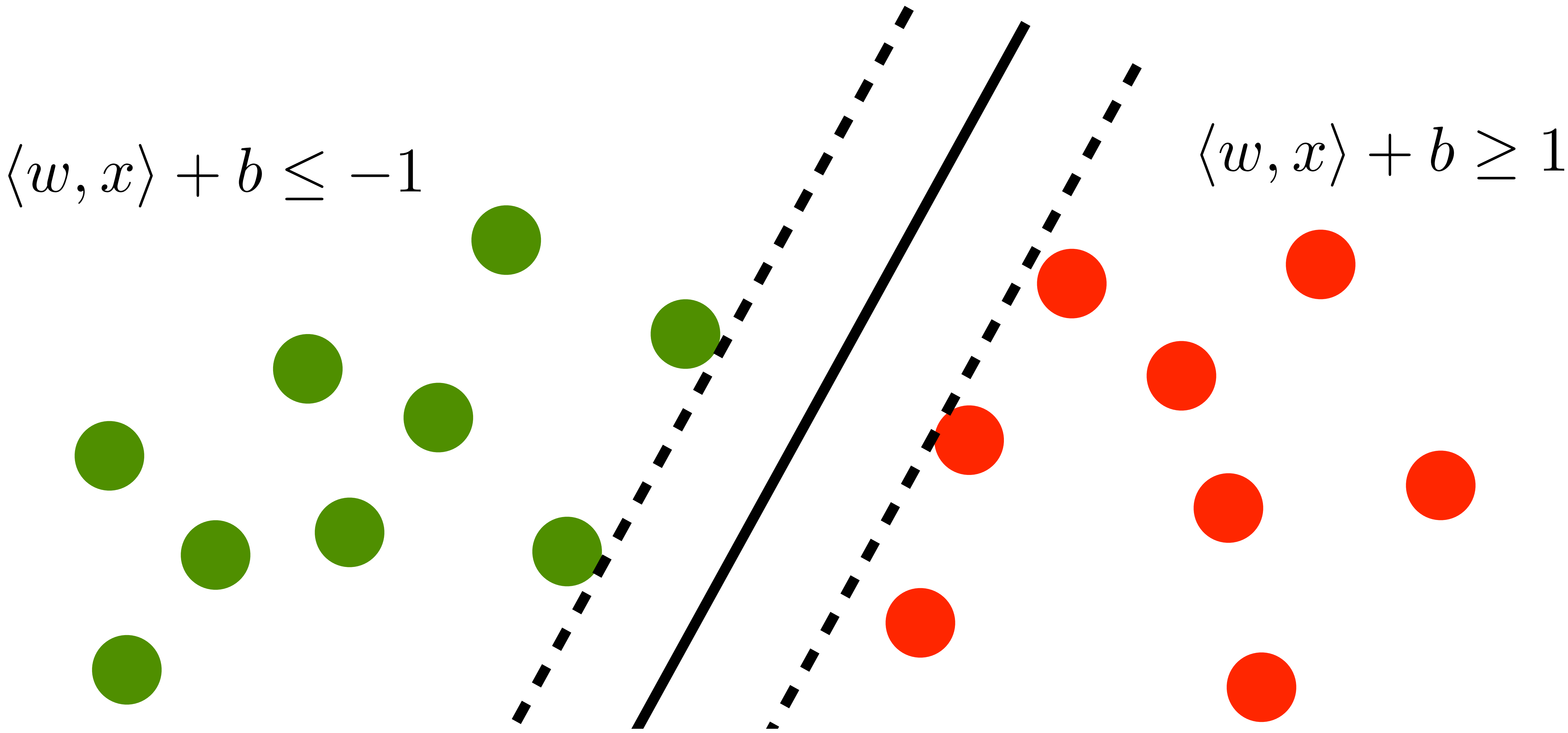


linear function
$f(x) = \langle w, x \rangle + b$

linear separator is impossible

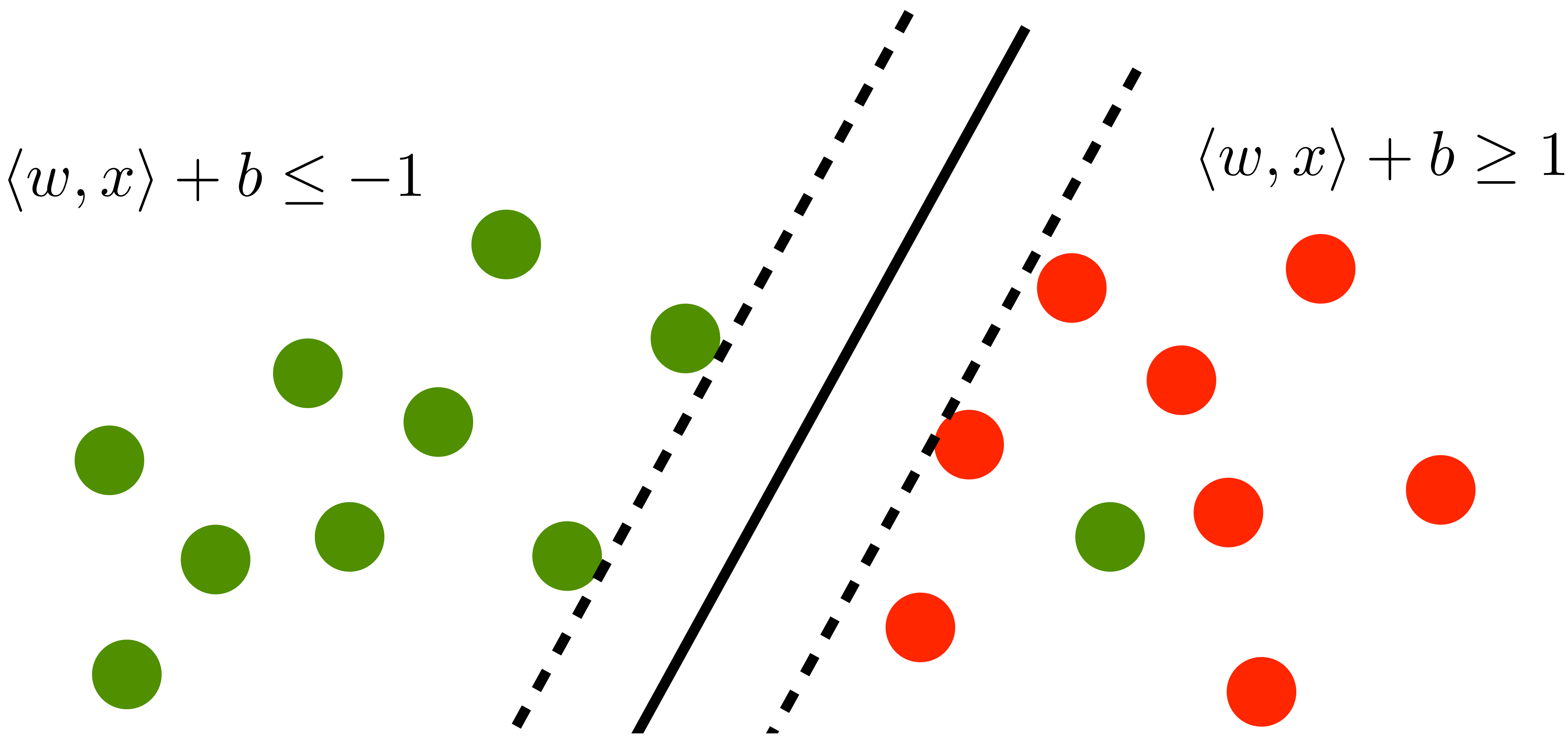
Carnegie Mellon University

Large Margin Classifier



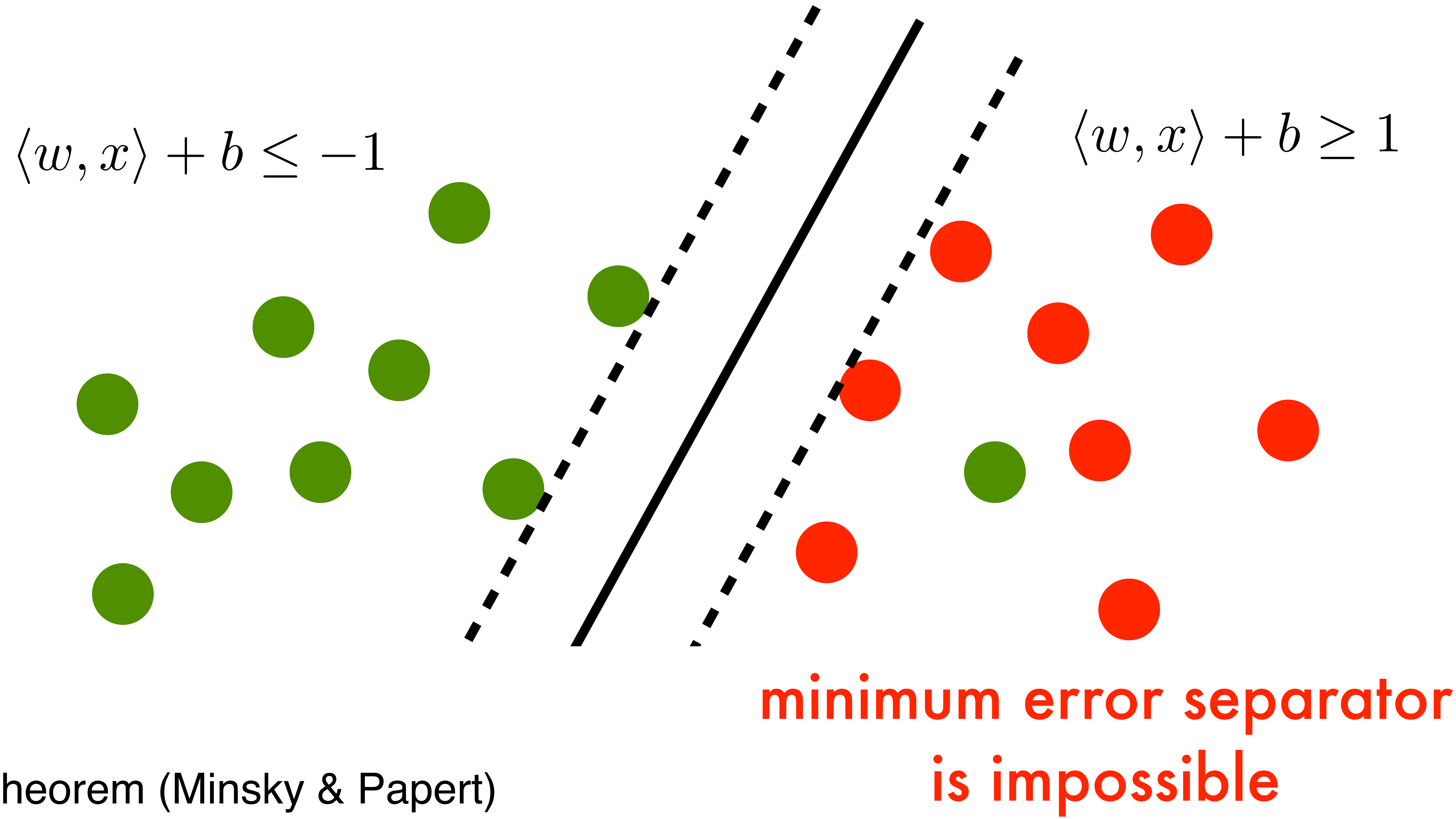
Theorem (Minsky & Papert)
Finding the minimum error separating hyperplane is NP hard

Large Margin Classifier



Theorem (Minsky & Papert)
Finding the minimum error separating hyperplane is NP hard

Large Margin Classifier

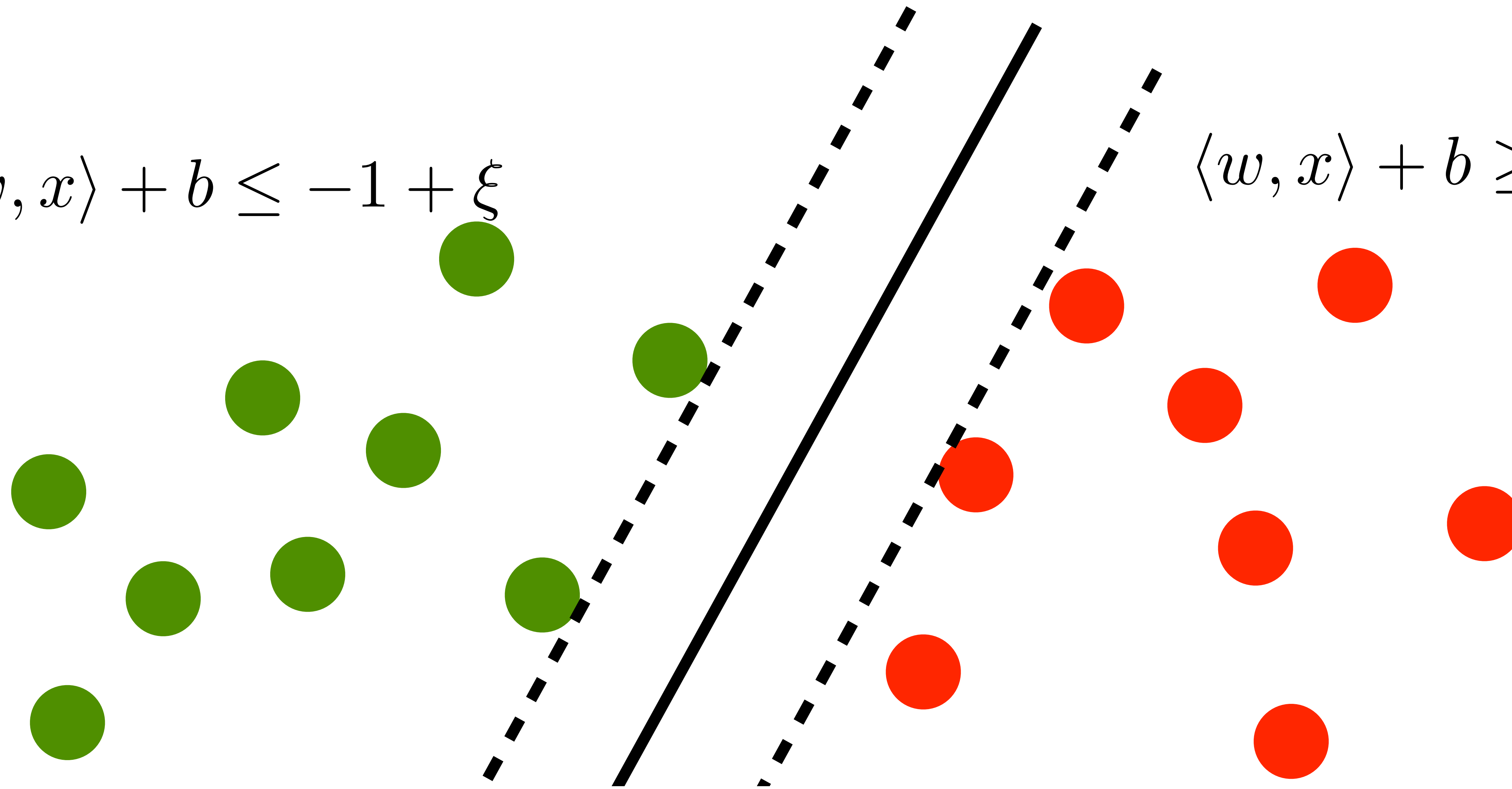


Theorem (Minsky & Papert)
Finding the minimum error separating hyperplane is NP hard

Adding slack variables

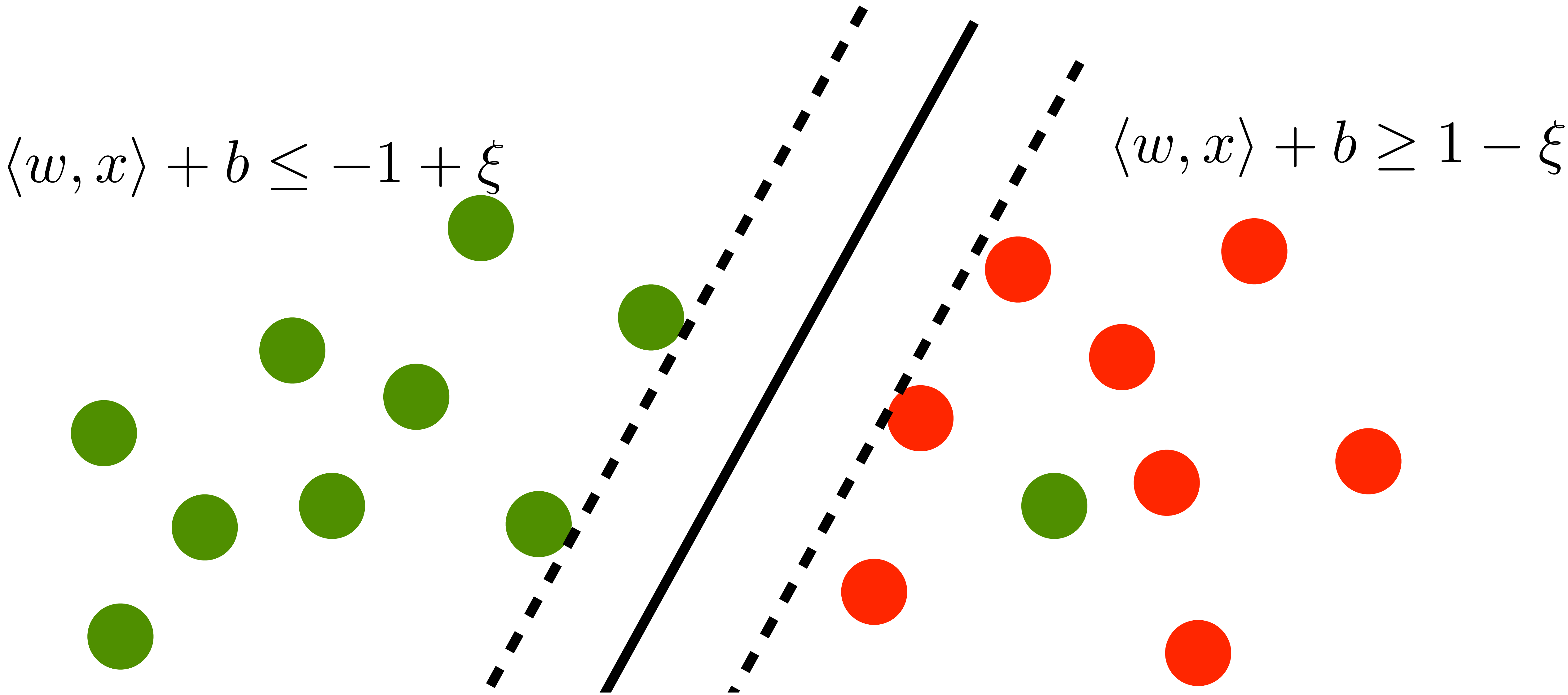
$$\langle w, x \rangle + b \leq -1 + \xi$$

$$\langle w, x \rangle + b \geq 1 - \xi$$



Convex optimization problem

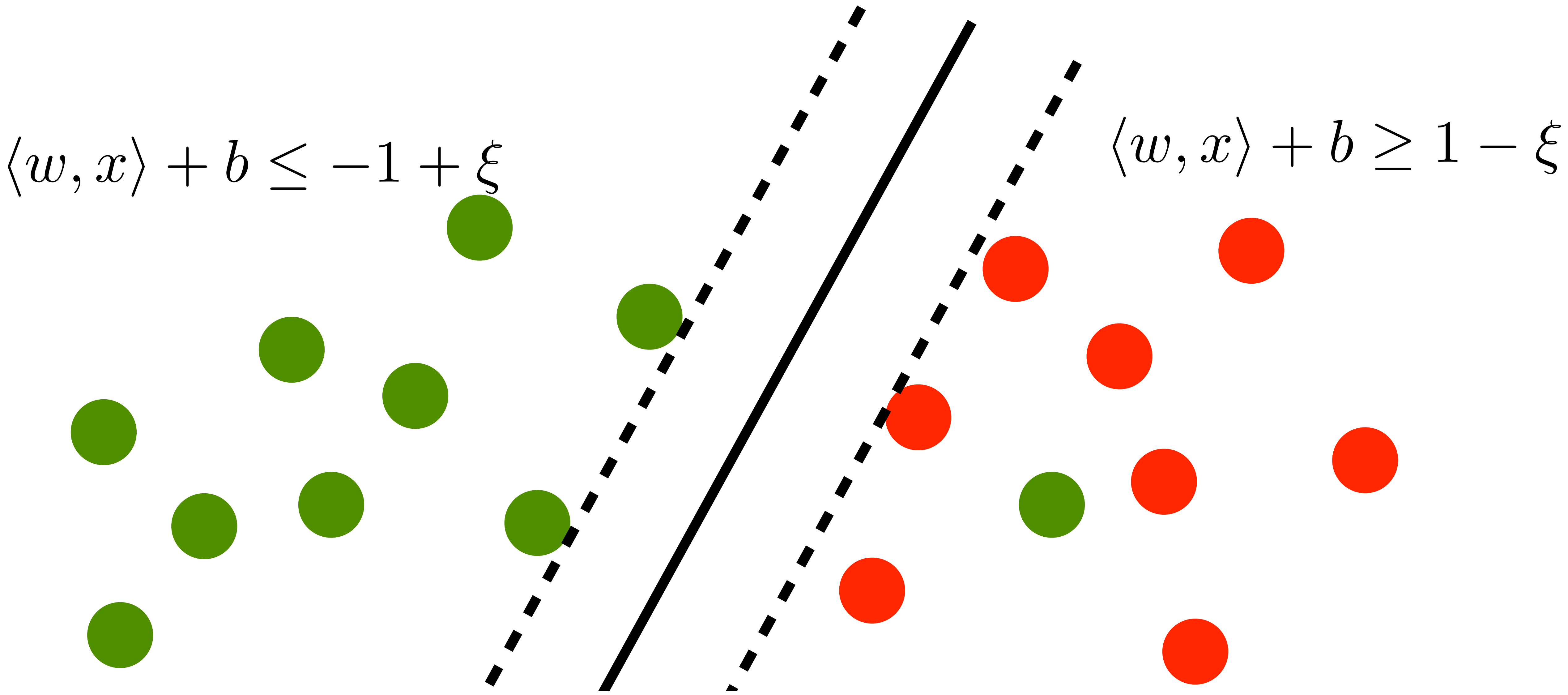
Adding slack variables



Convex optimization problem

Carnegie Mellon University

Adding slack variables



minimize amount of slack

Convex optimization problem

Intermezzo
Convex Programs for Dummies

- Primal optimization problem

$$\underset{x}{\text{minimize}}\, f(x) \text{ subject to } c_i(x) \leq 0$$

- Lagrange function

$$L(x, \alpha) = f(x) + \sum_i \alpha_i c_i(x)$$

- First order optimality conditions in x

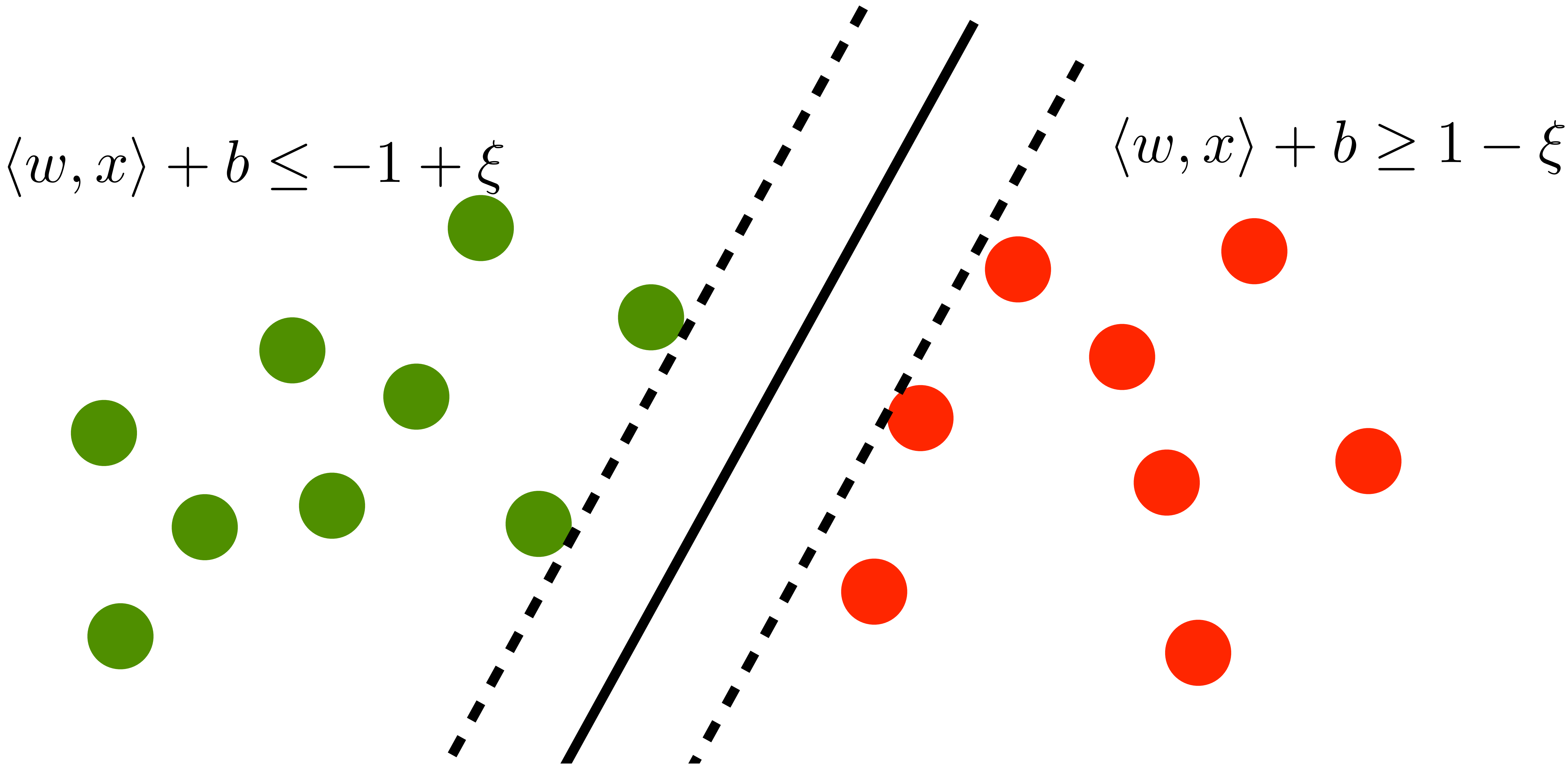
$$\partial_x L(x, \alpha) = \partial_x f(x) + \sum_i \alpha_i \partial_x c_i(x) = 0$$

- Solve for x and plug it back into L

$$\underset{\alpha}{\text{maximize}}\, L(x(\alpha), \alpha)$$

(keep explicit constraints)

Adding slack variables

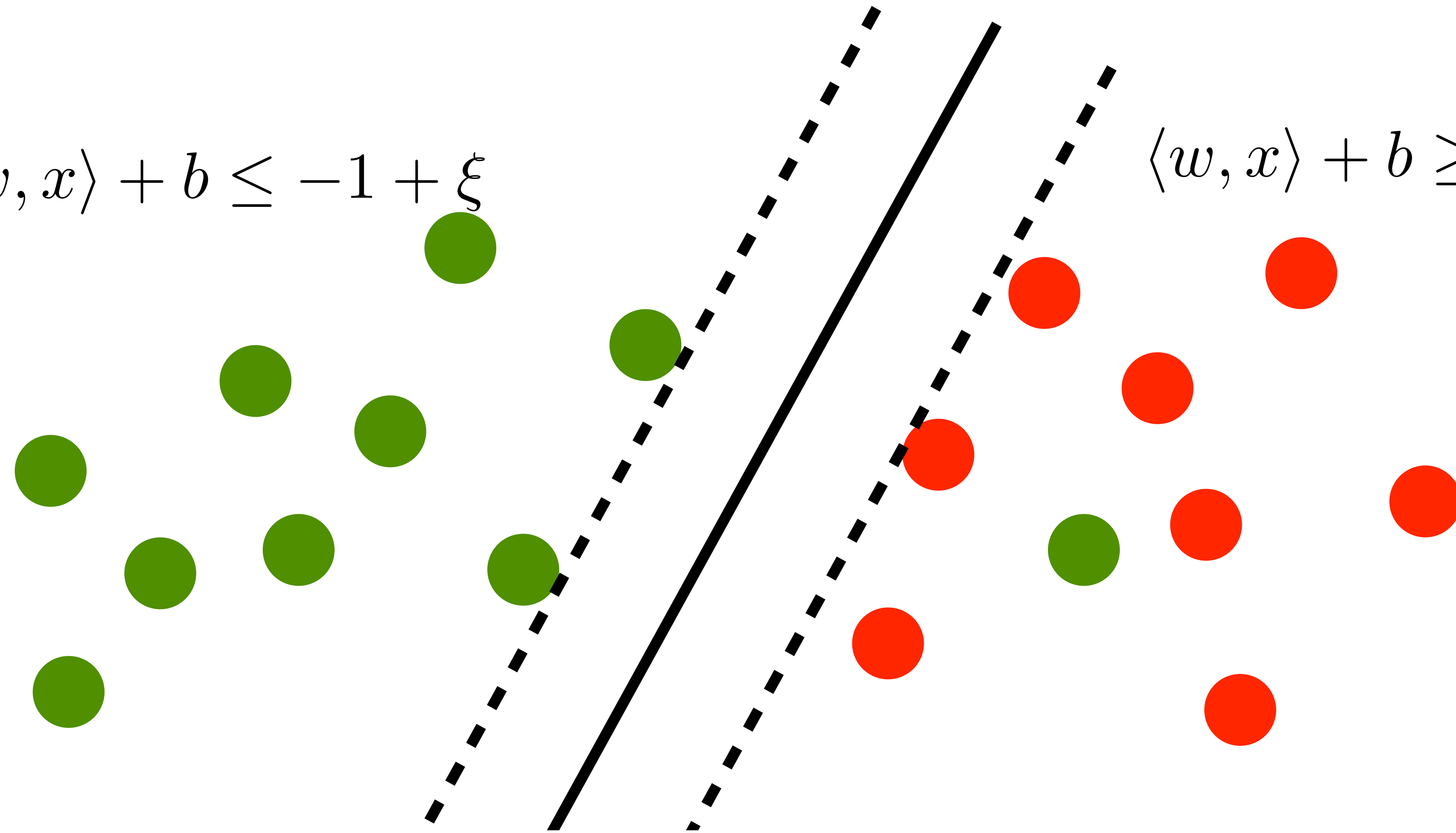


Convex optimization problem

Adding slack variables

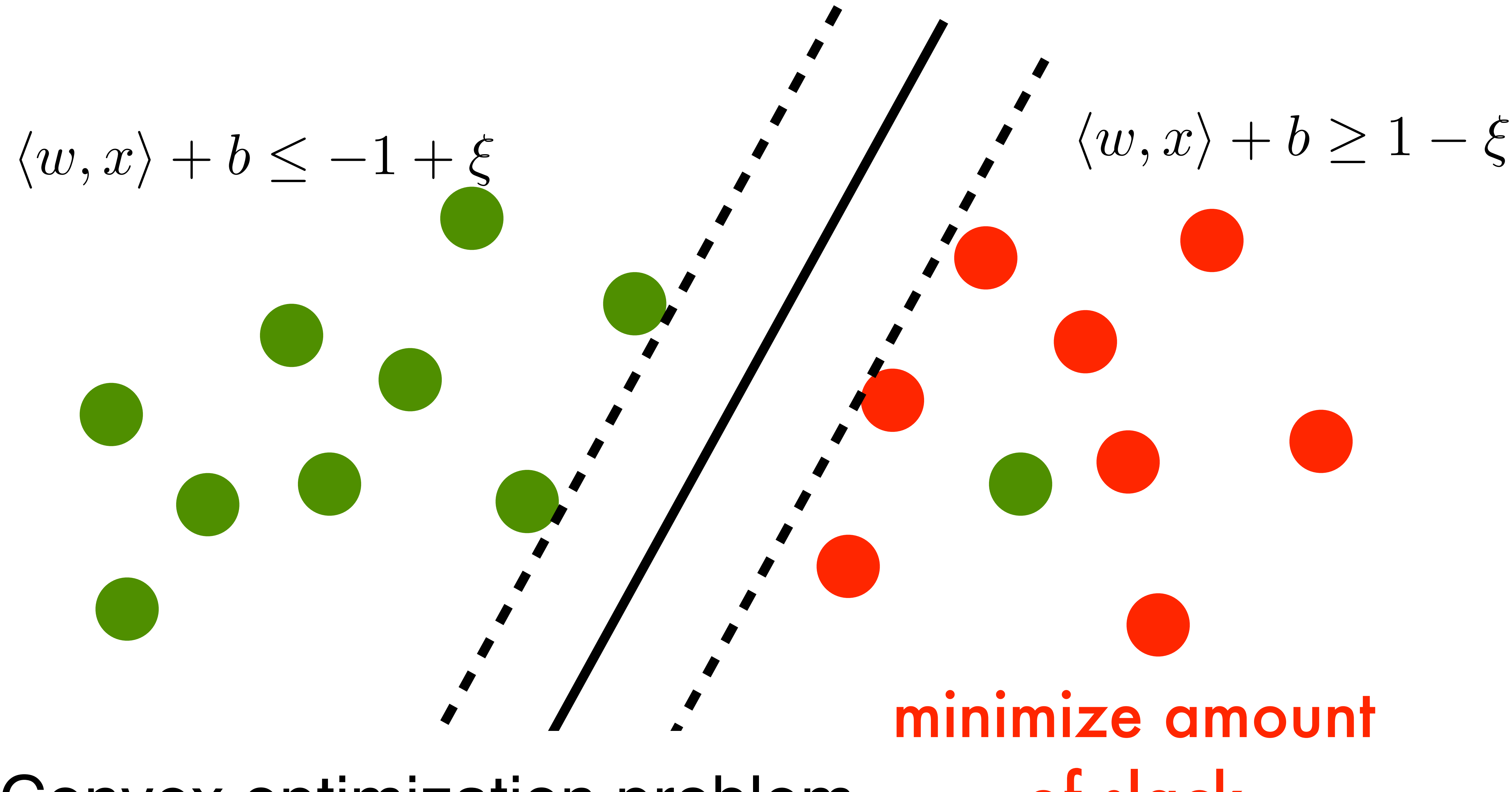
$$\langle w, x \rangle + b \leq -1 + \xi$$

$$\langle w, x \rangle + b \geq 1 - \xi$$



Convex optimization problem

Adding slack variables



Convex optimization problem

Adding slack variables

- Hard margin problem

$$\underset{w,b}{\text{minimize}} \ \frac{1}{2} \|w\|^2 \ \text{ subject to } y_i \left[\langle w, x_i \rangle + b\right] \geq 1$$

- With slack variables

$$\underset{w,b}{\text{minimize}} \ \ \frac{1}{2} \|w\|^2 + C \sum_i \xi_i$$

$$\text{subject to } y_i \left[\langle w, x_i \rangle + b\right] \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

Problem is always feasible. $w = 0 \text{ and } b = 0 \text{ and } \xi_i = 1$

Dual Problem

- Primal optimization problem

$$\underset{w,b}{\text{minimize}} \ \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

$$\text{subject to } y_i\left[\langle w, x_i\rangle + b\right] \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

- Lagrange function

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 + C\sum_i \xi_i - \sum_i \alpha_i\left[y_i\left[\langle x_i, w\rangle + b\right] + \xi_i - 1\right] - \sum_i \eta_i \xi_i$$

Optimality in w,b,ξ is at saddle point with α,η

- Derivatives in w,b,ξ need to vanish

Dual Problem

- Lagrange function

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 + C\sum_i \xi_i - \sum_i \alpha_i \left[y_i \left[\langle x_i, w\rangle + b\right] + \xi_i - 1\right] - \sum_i \eta_i \xi_i$$

- Derivatives in w, b need to vanish

$$\partial_w L(w, b, \xi, \alpha, \eta) = w - \sum_i \alpha_i y_i x_i = 0$$

$$\partial_b L(w, b, \xi, \alpha, \eta) = \sum_i \alpha_i y_i = 0$$

$$\partial_{\xi_i} L(w, b, \xi, \alpha, \eta) = C - \alpha_i - \eta_i = 0$$

- Plugging terms back into L yields

$$\underset{\alpha}{\text{maximize}} - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j\rangle + \sum_i \alpha_i$$

$$\text{subject to } \sum_i \alpha_i y_i = 0 \text{ and } \boxed{\alpha_i \in [0, C]}$$

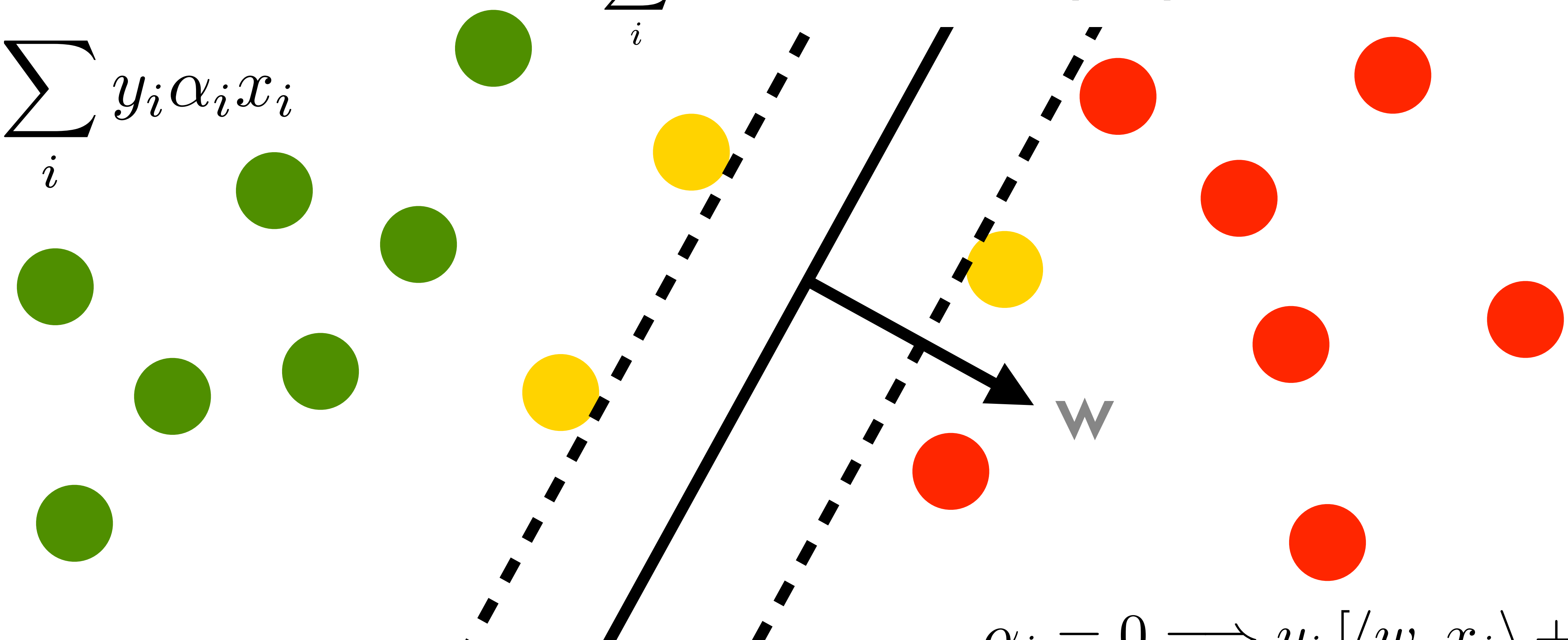
bound influence

Karush Kuhn Tucker Conditions

$$\underset{\alpha}{\text{maximize}} - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle + \sum_i \alpha_i$$

$$\text{subject to } \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, C]$$

$$w = \sum_i y_i \alpha_i x_i$$



$$\alpha_i = 0 \implies y_i\left[\langle w, x_i\rangle + b\right] \geq 1$$

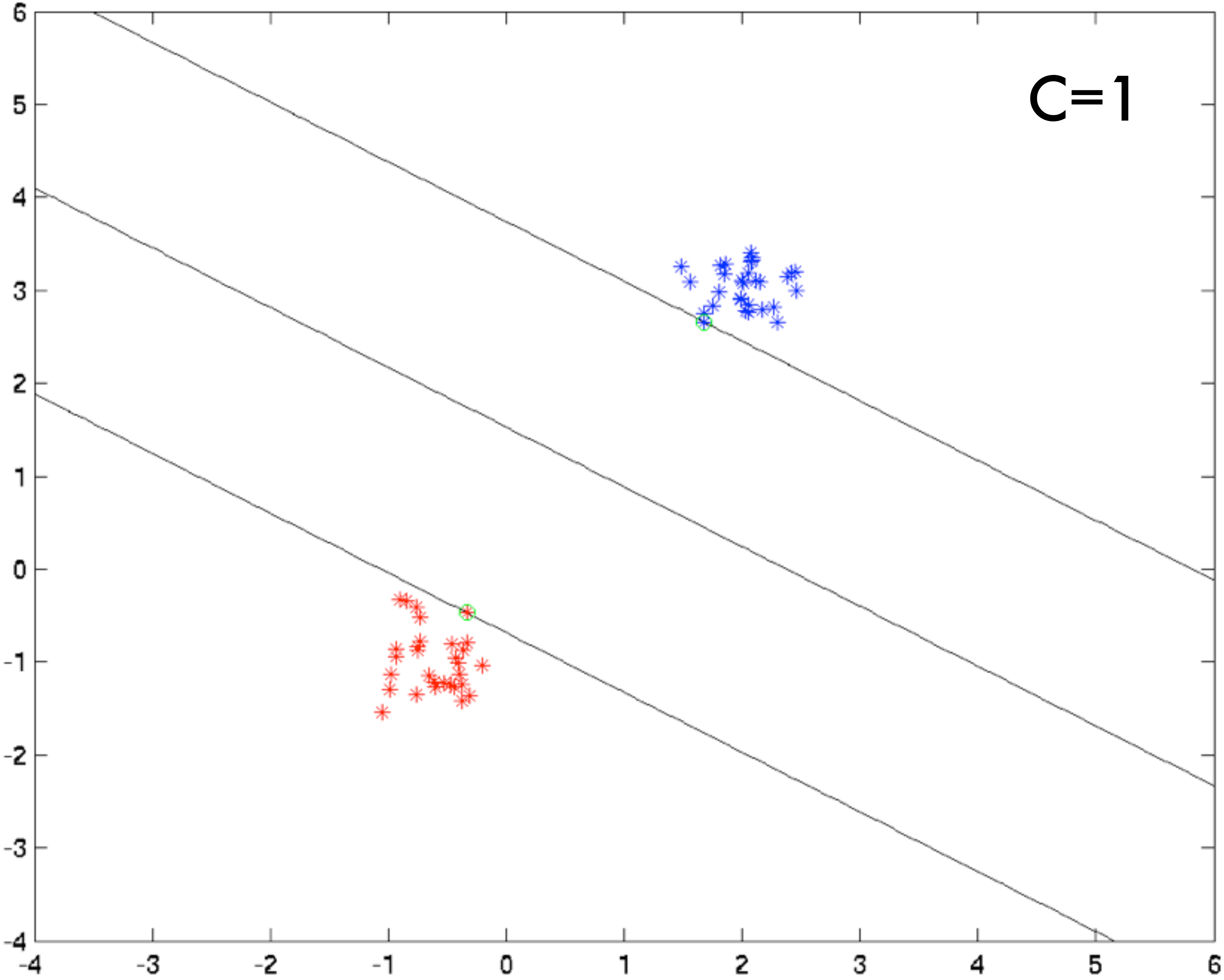
$$\alpha_i\left[y_i\left[\langle w, x_i\rangle + b\right] + \xi_i - 1\right] = 0$$

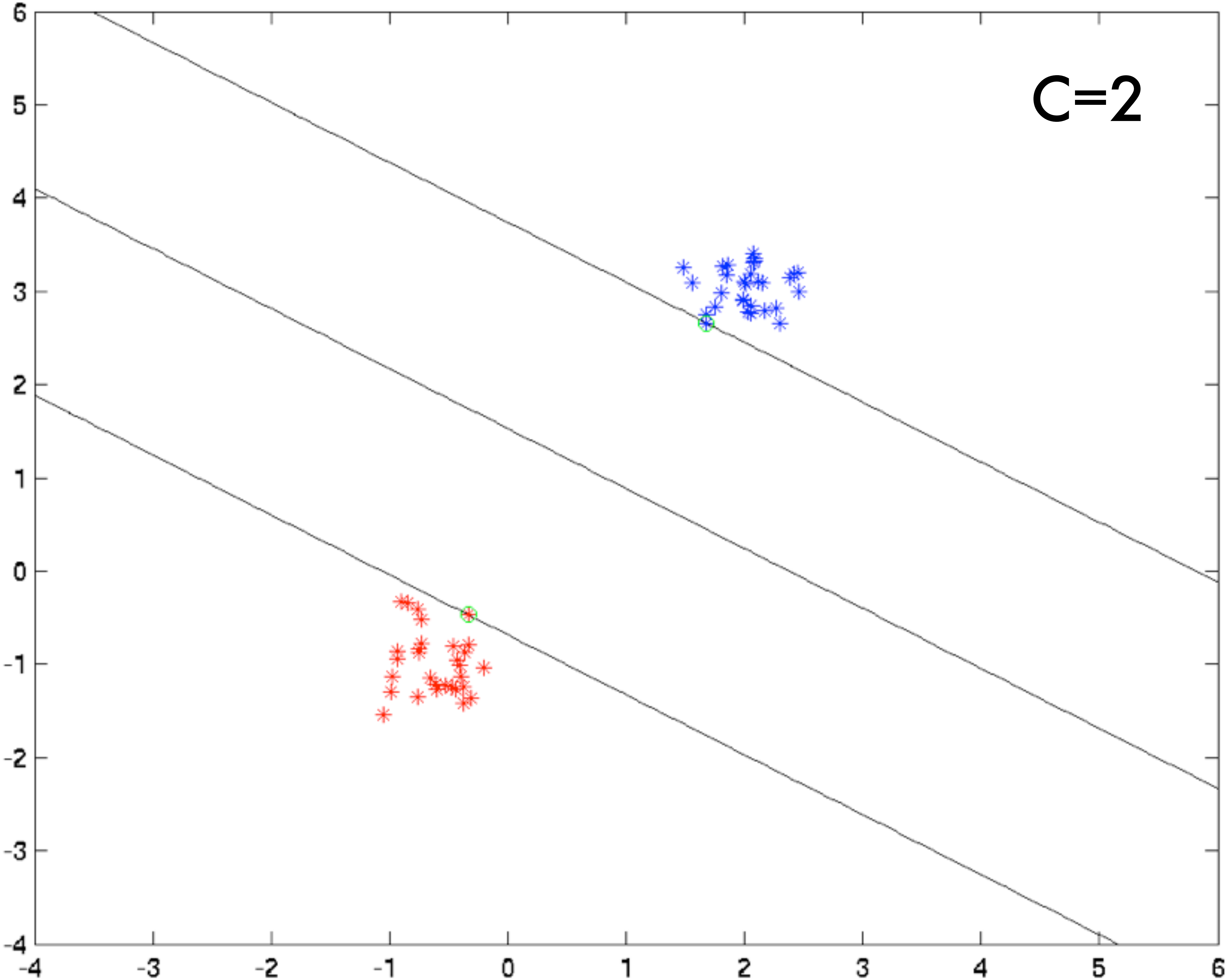
$$0 < \alpha_i < C \implies y_i\left[\langle w, x_i\rangle + b\right] = 1$$

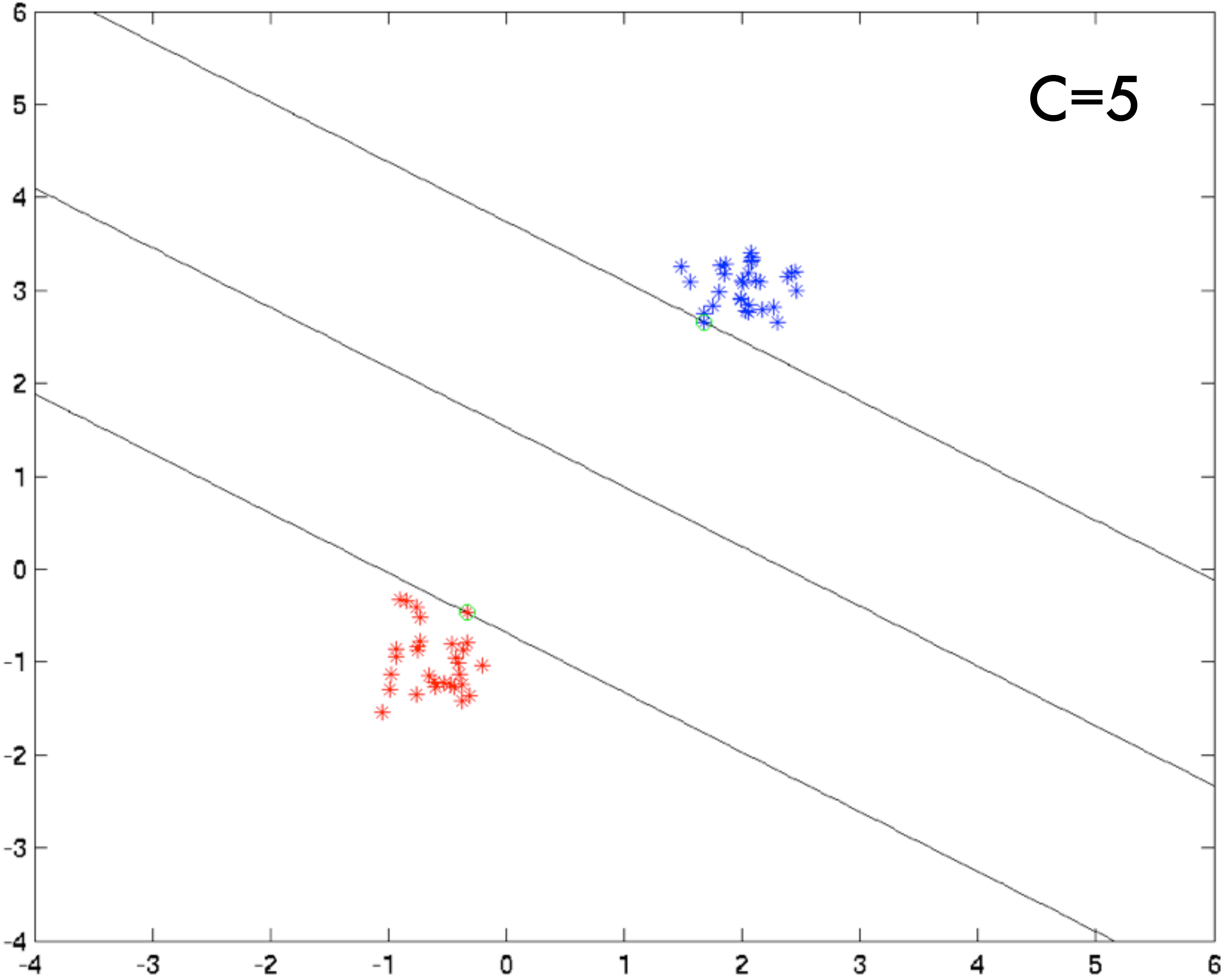
$$\eta_i \xi_i = 0$$

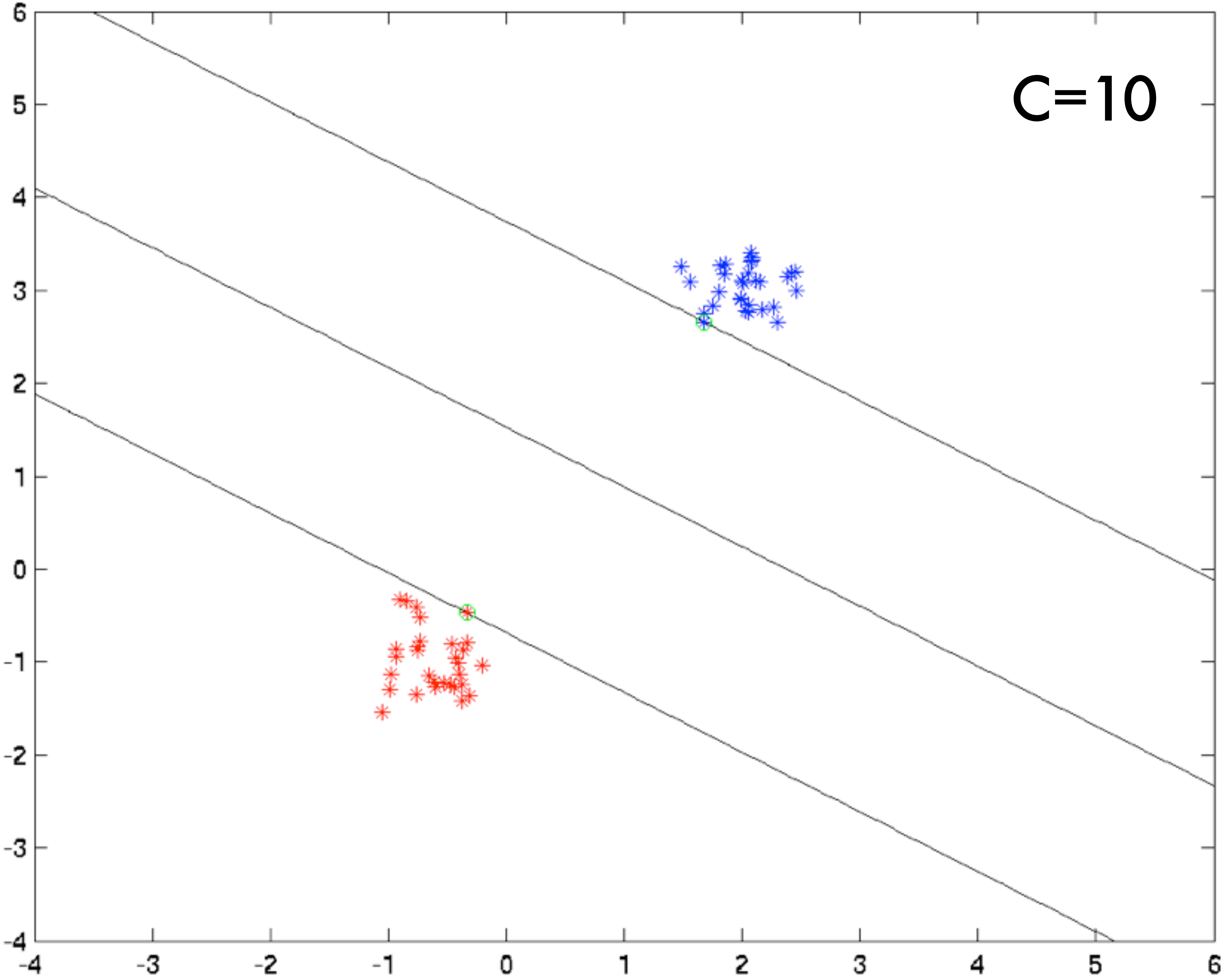
$$\alpha_i = C \implies y_i\left[\langle w, x_i\rangle + b\right] \leq 1$$

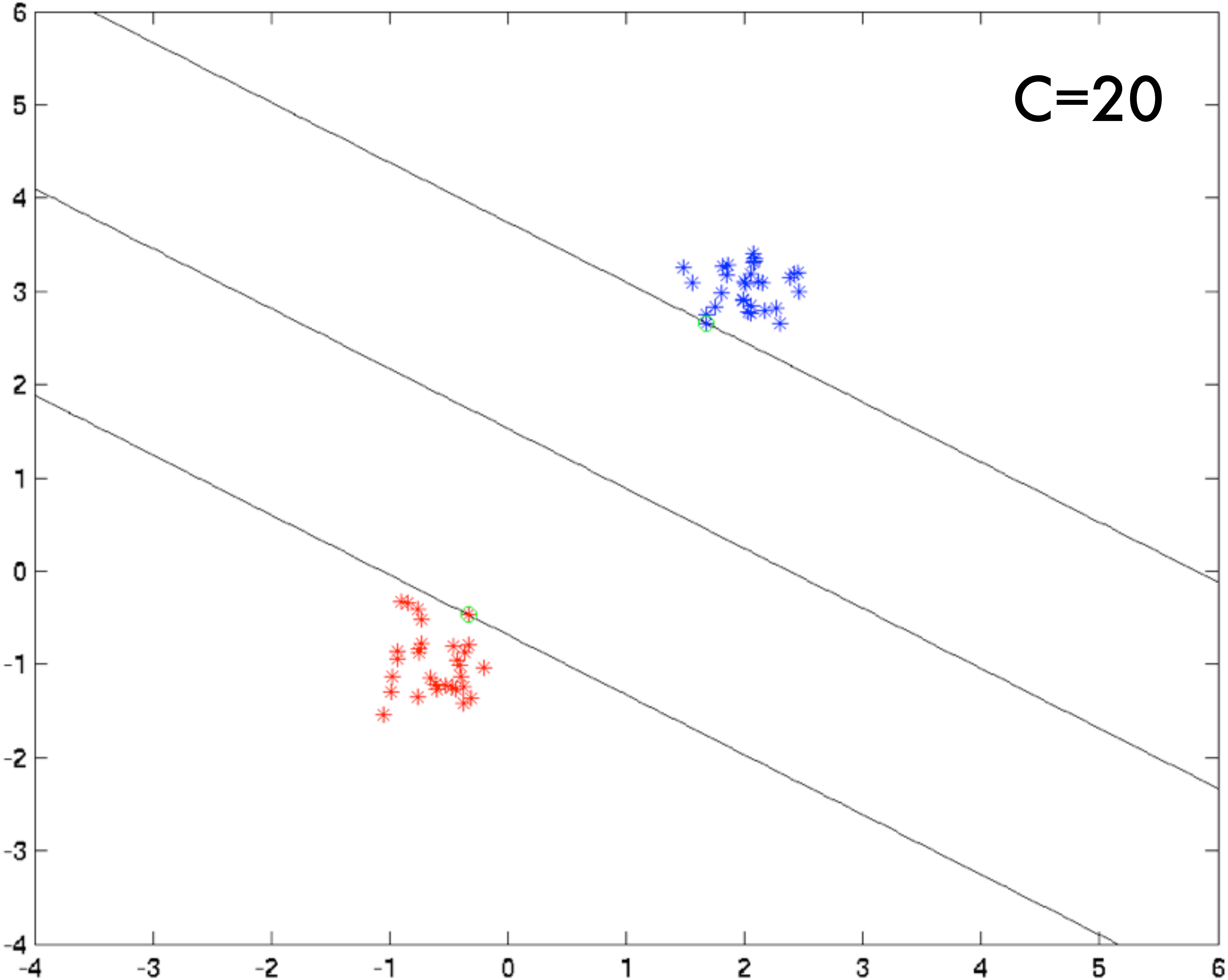
Carnegie Mellon University

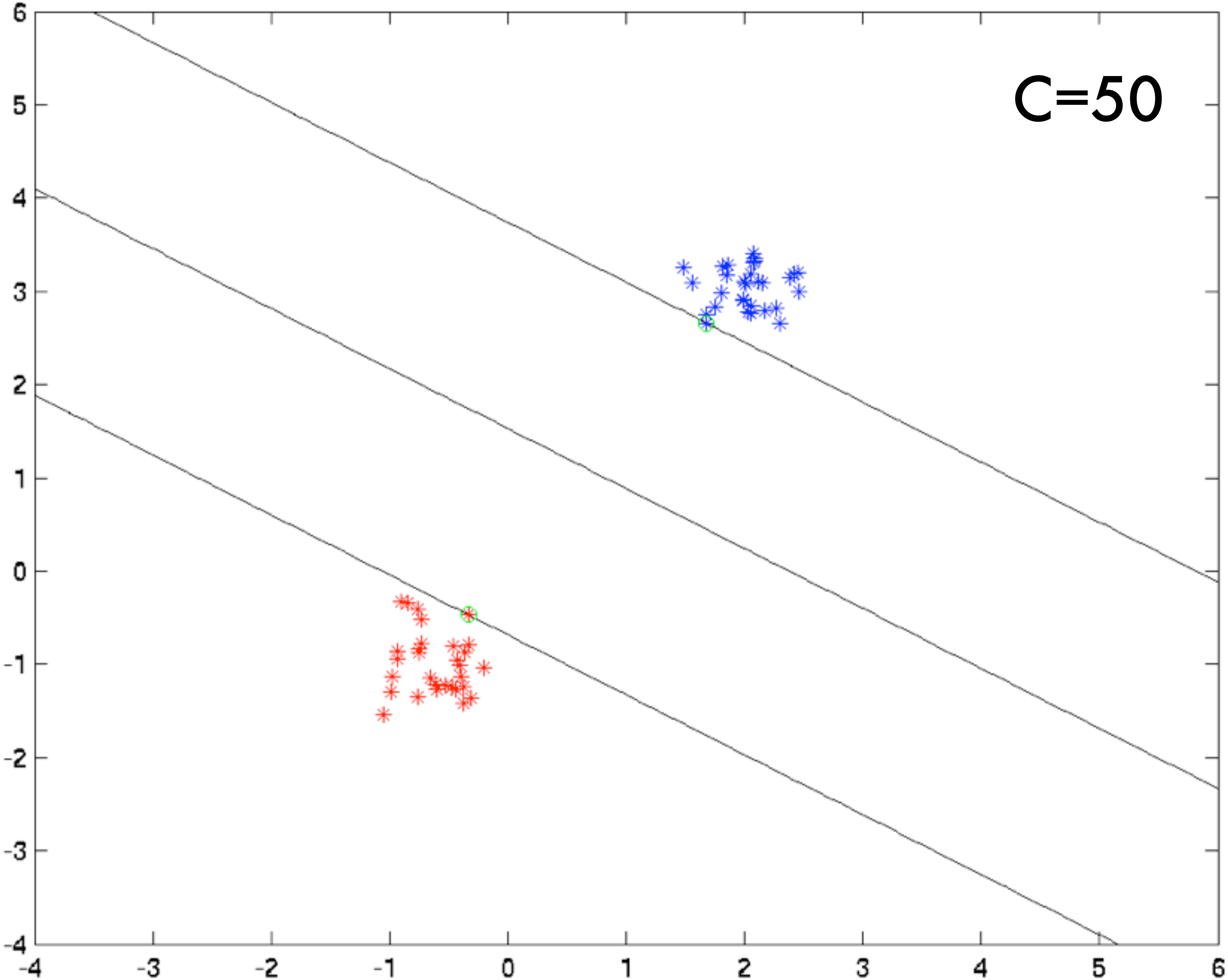

C=1
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

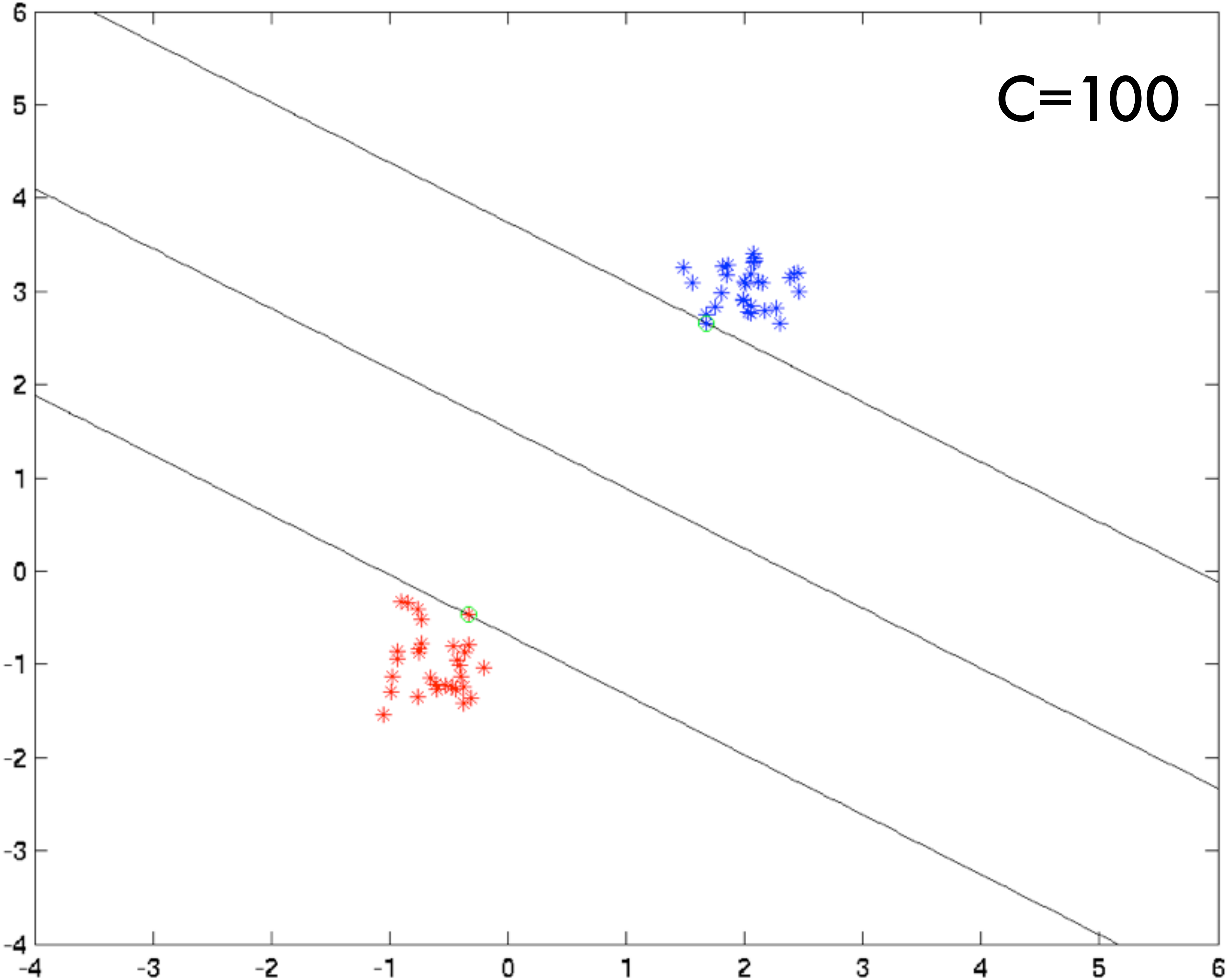

C=2
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


C=5
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

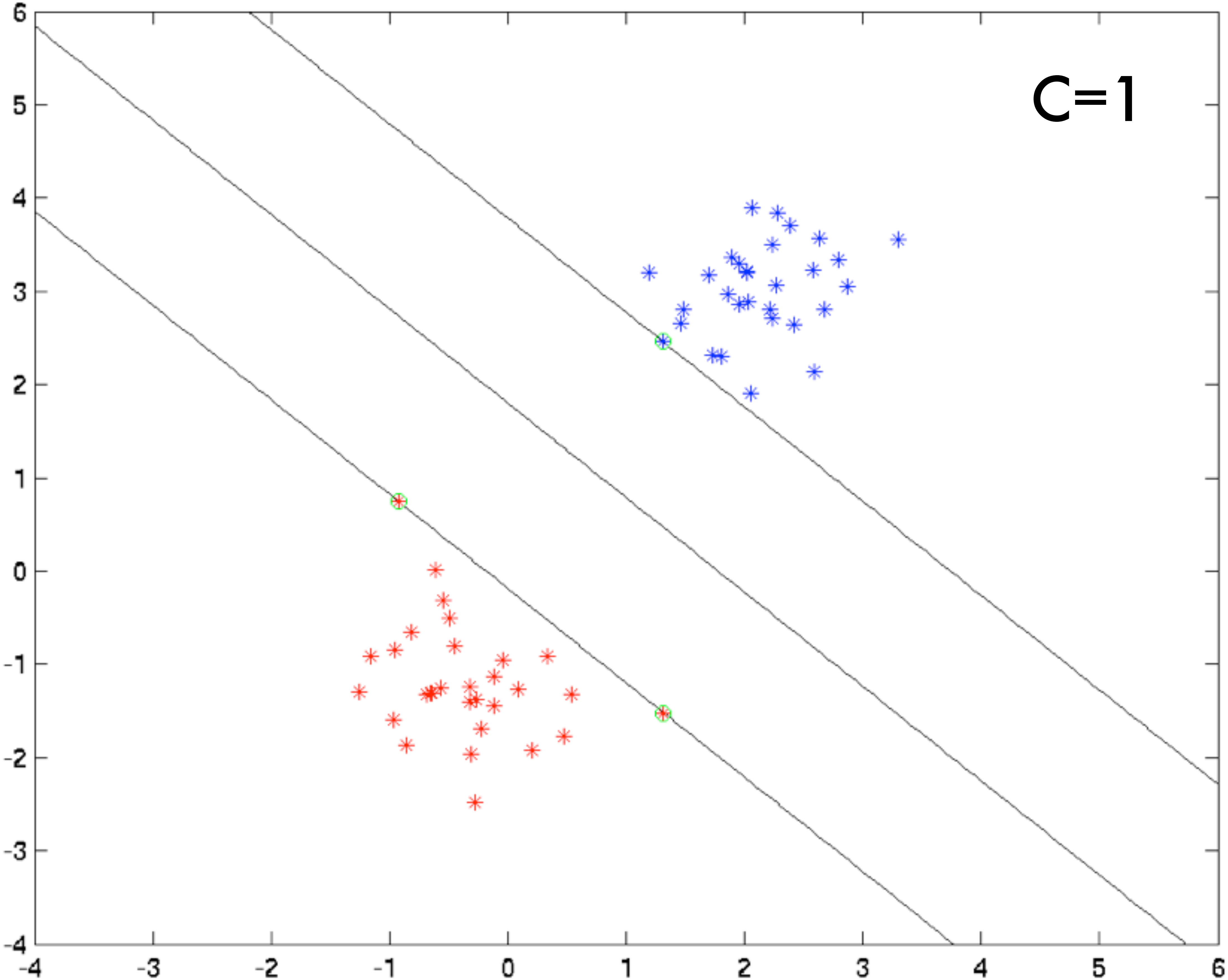

C=10
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


C=20
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

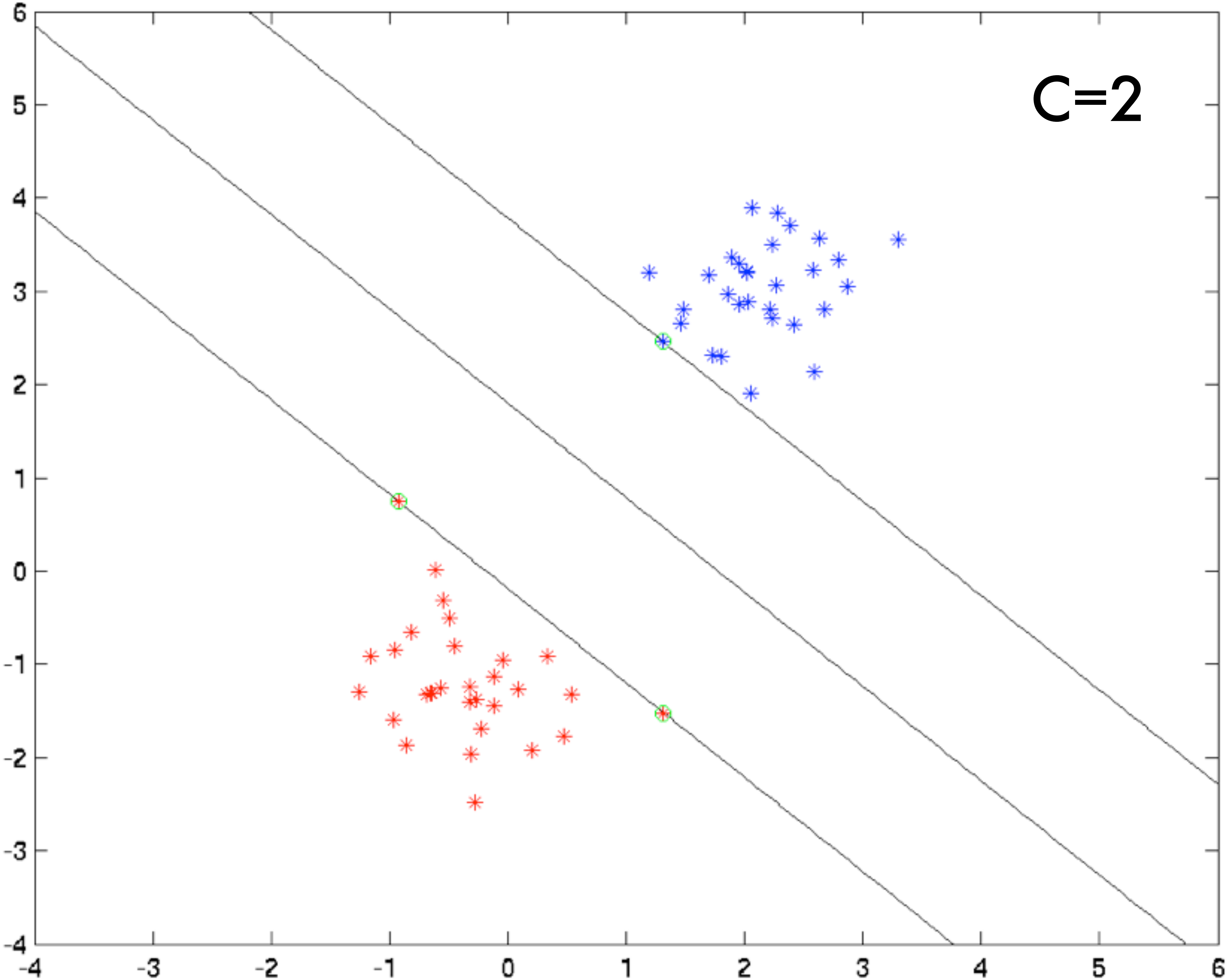

C=50
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

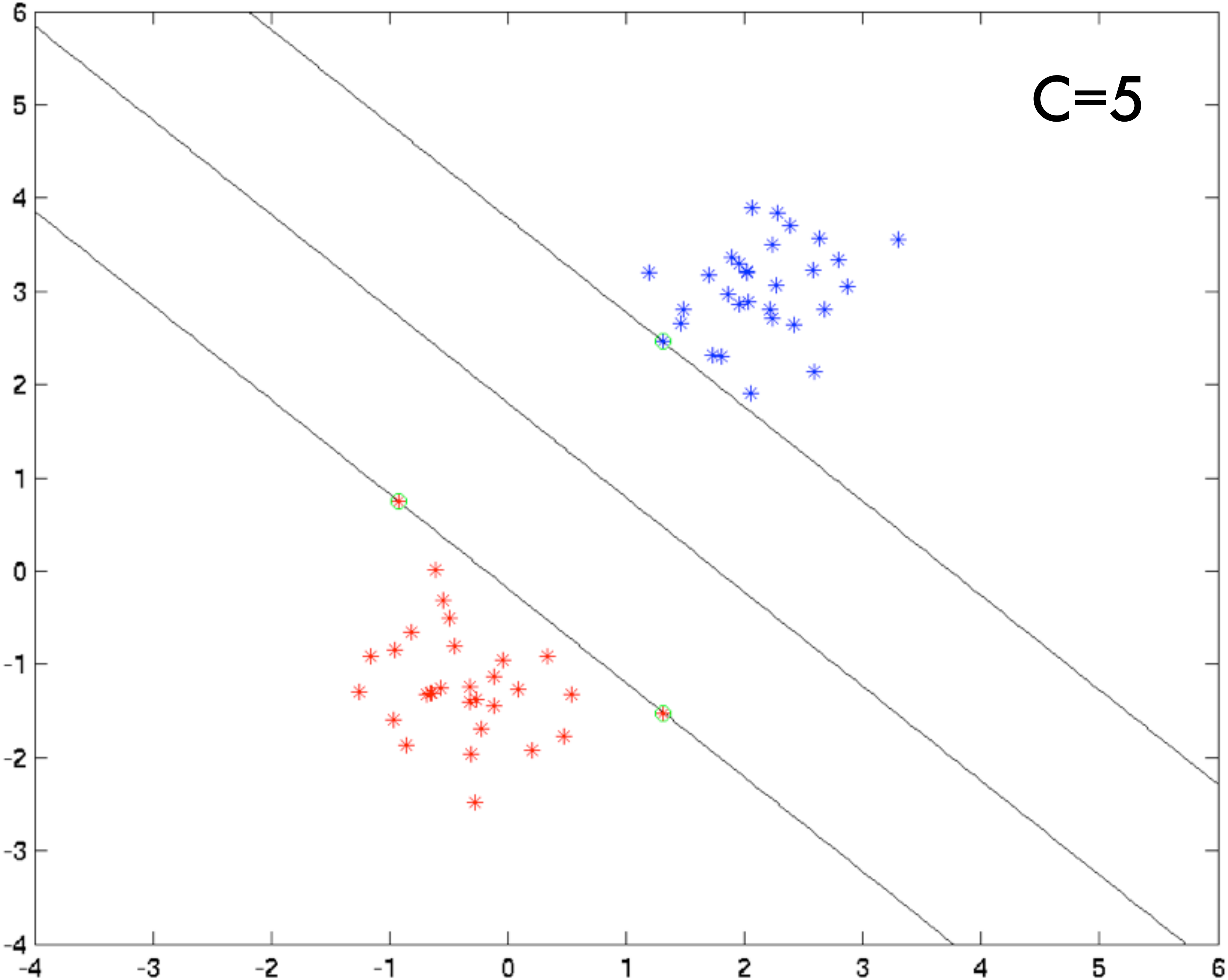


C=100
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

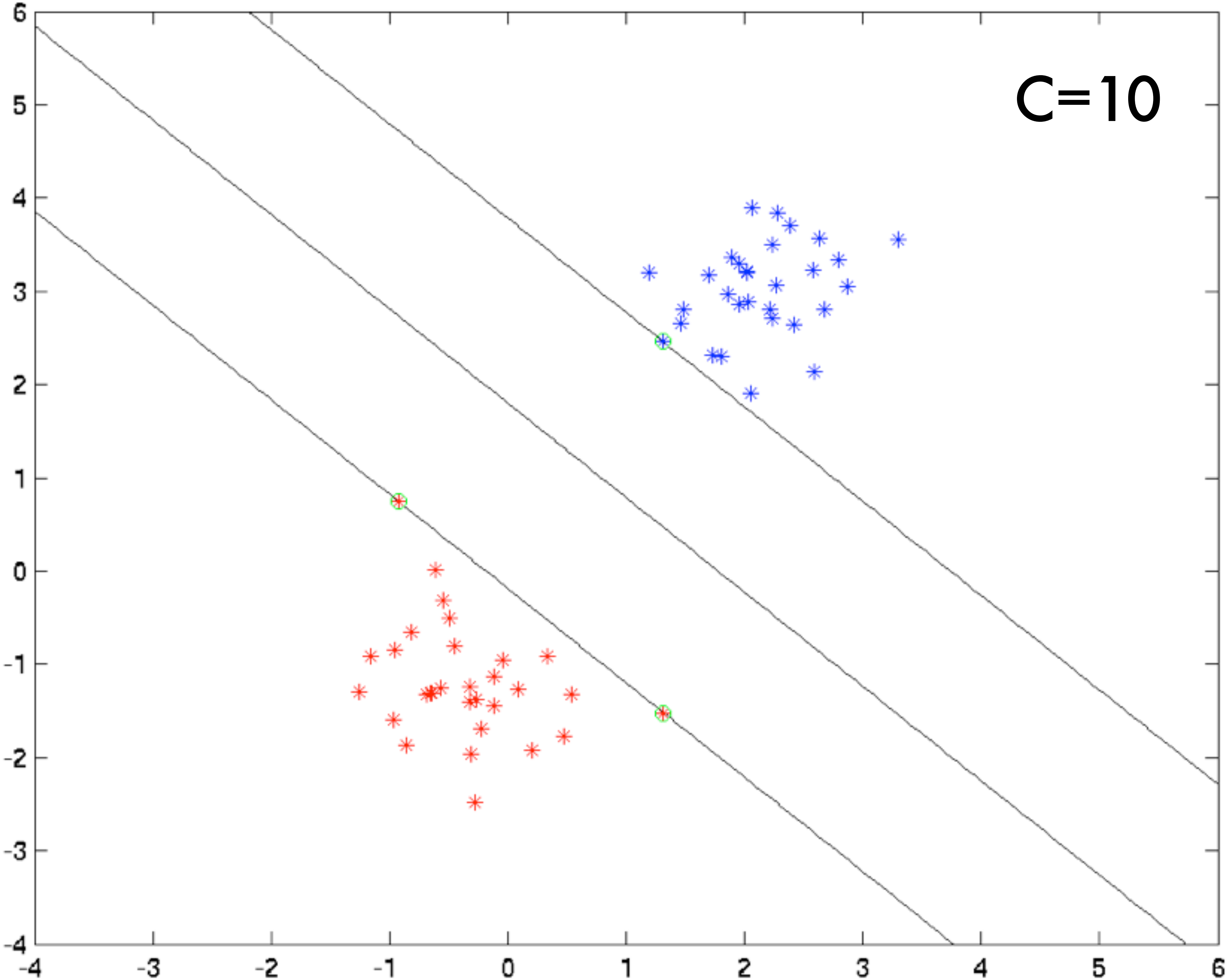


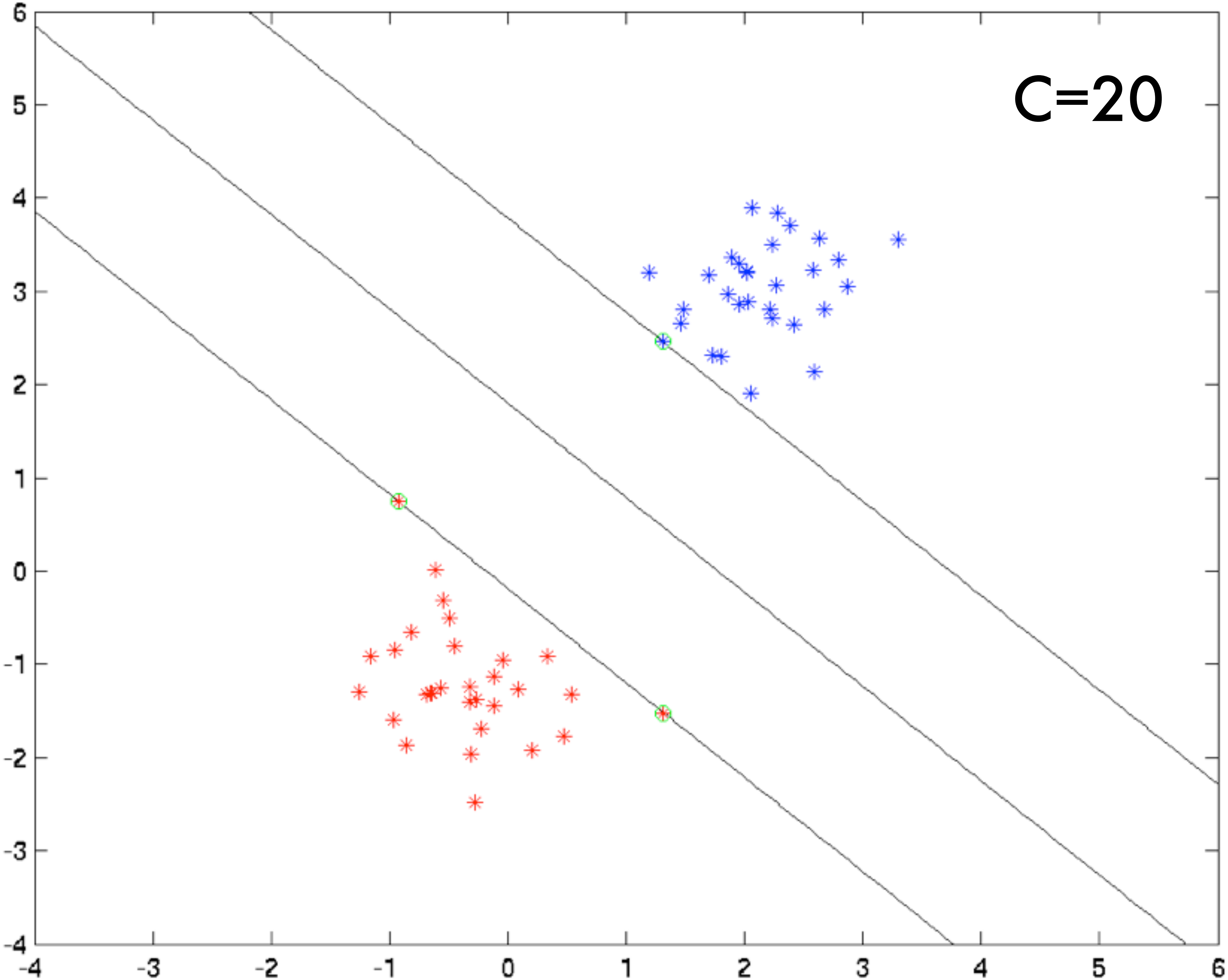
C=1
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

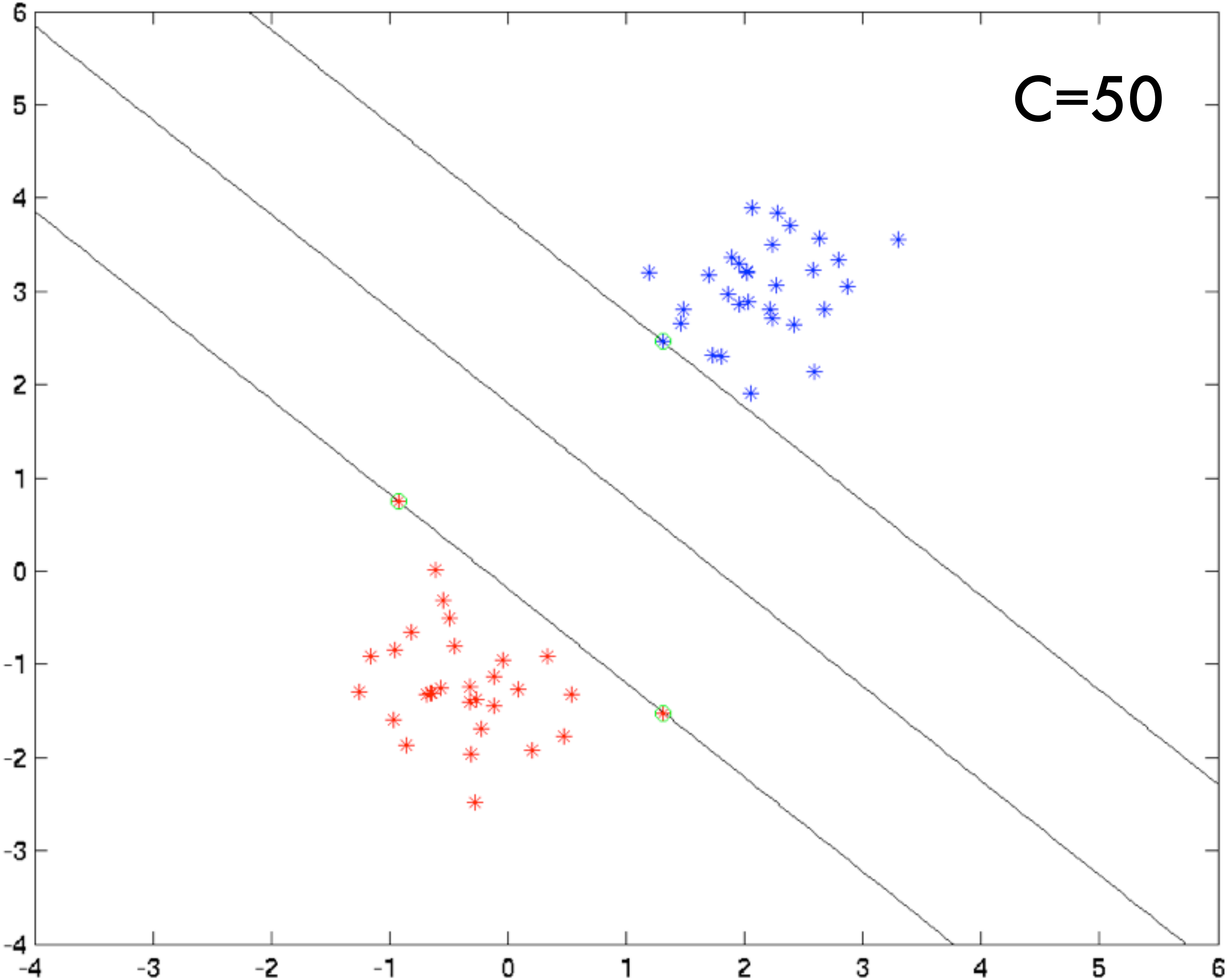

C=2
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

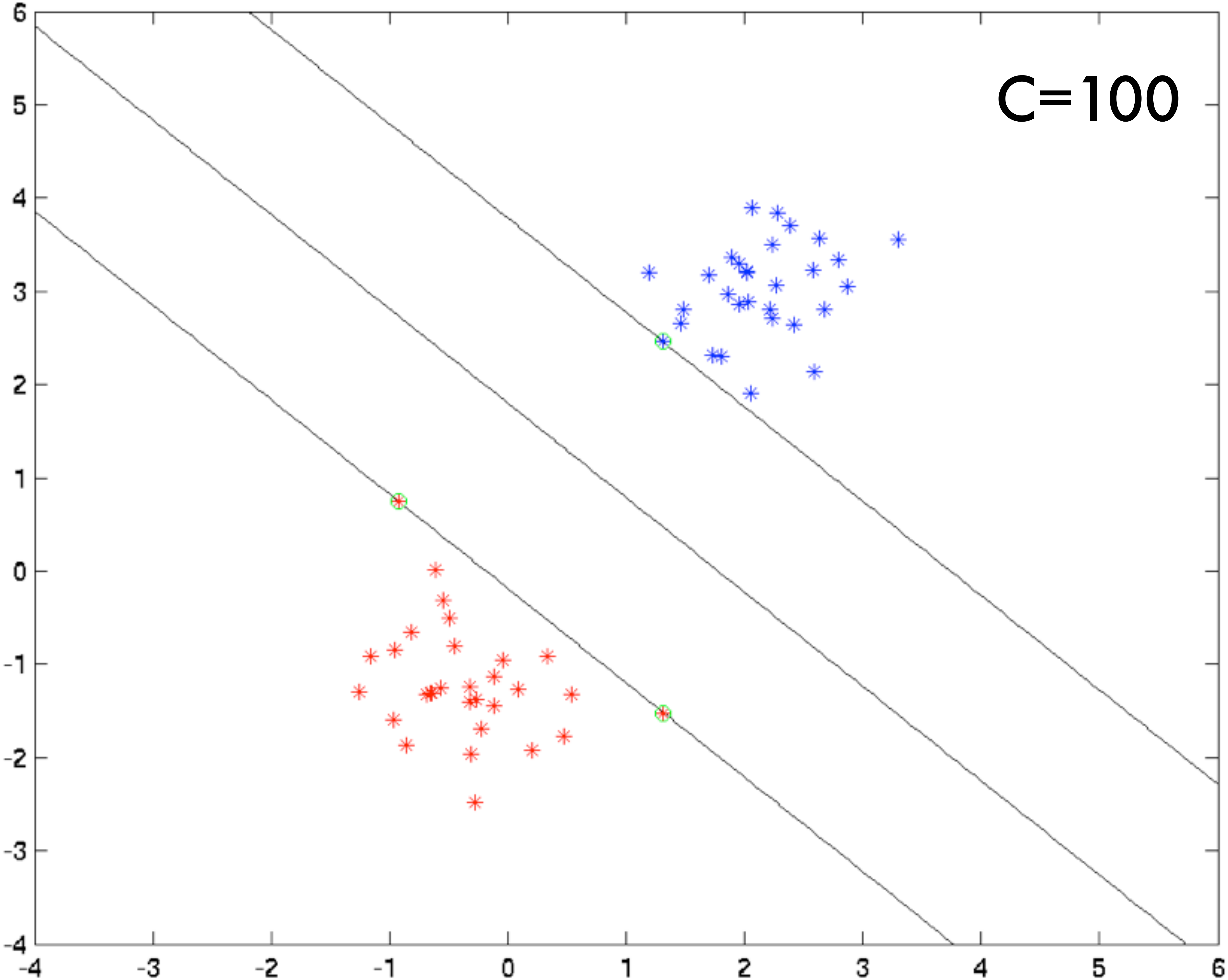


C=5

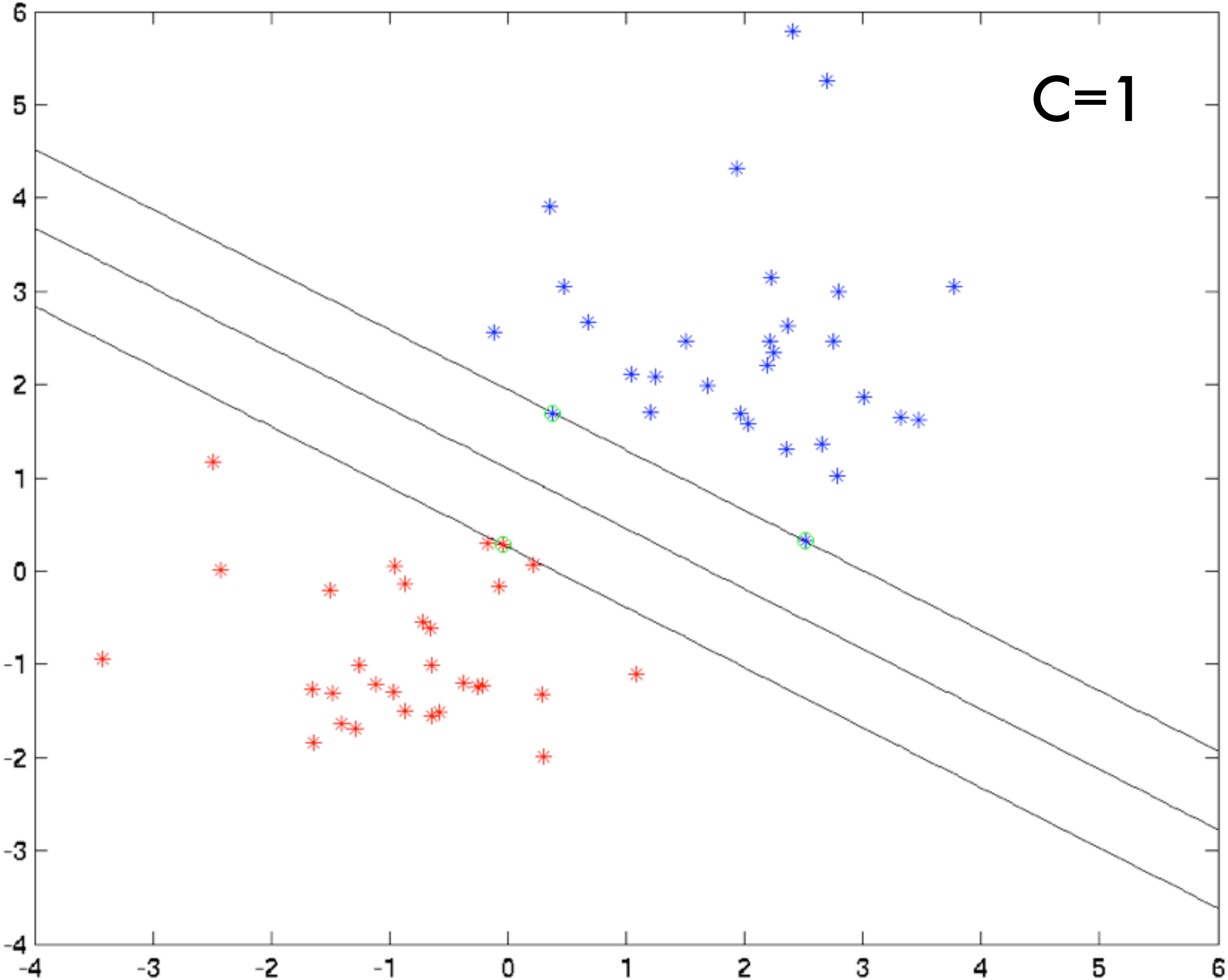

C=10
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


C=20
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

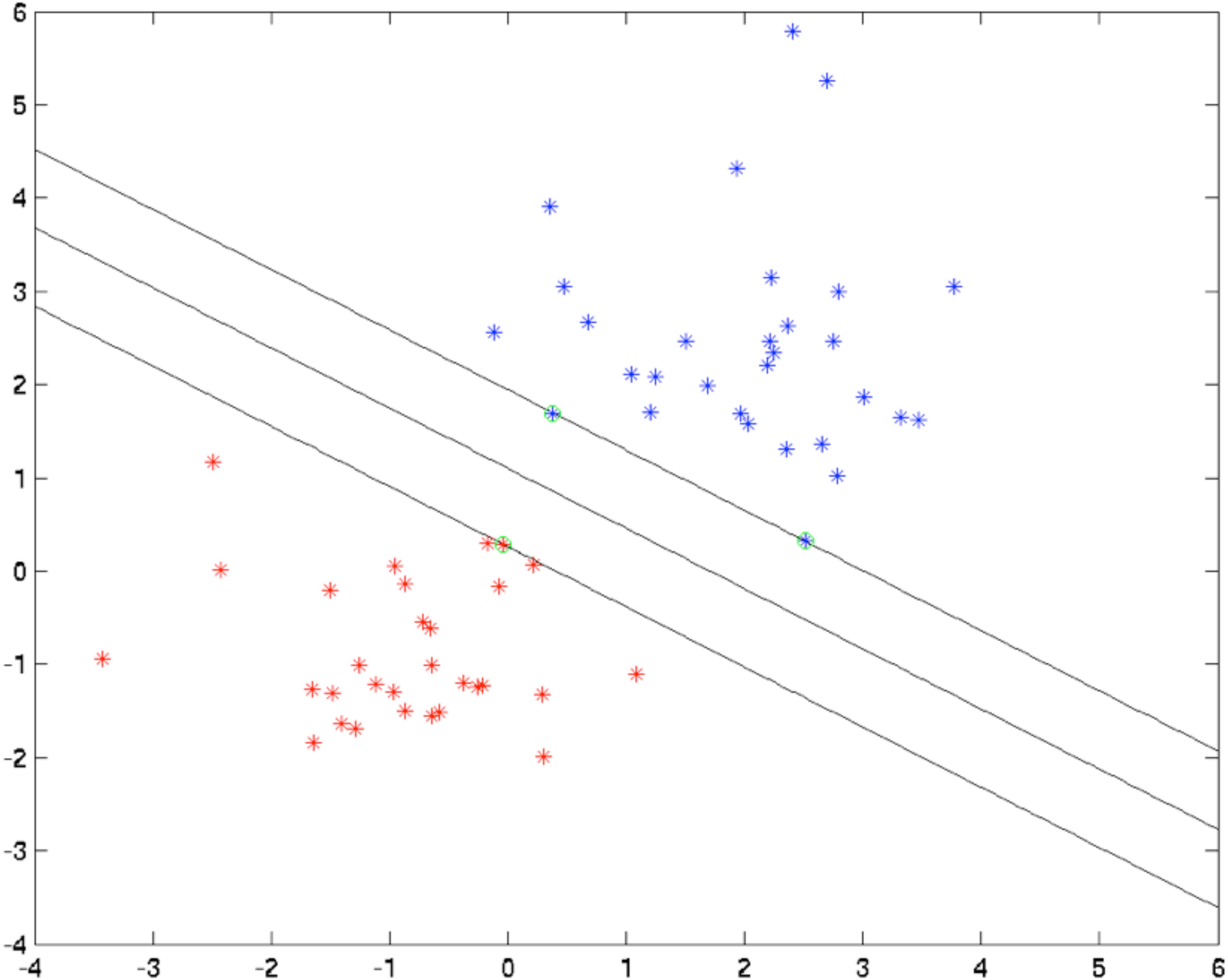

C=50
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

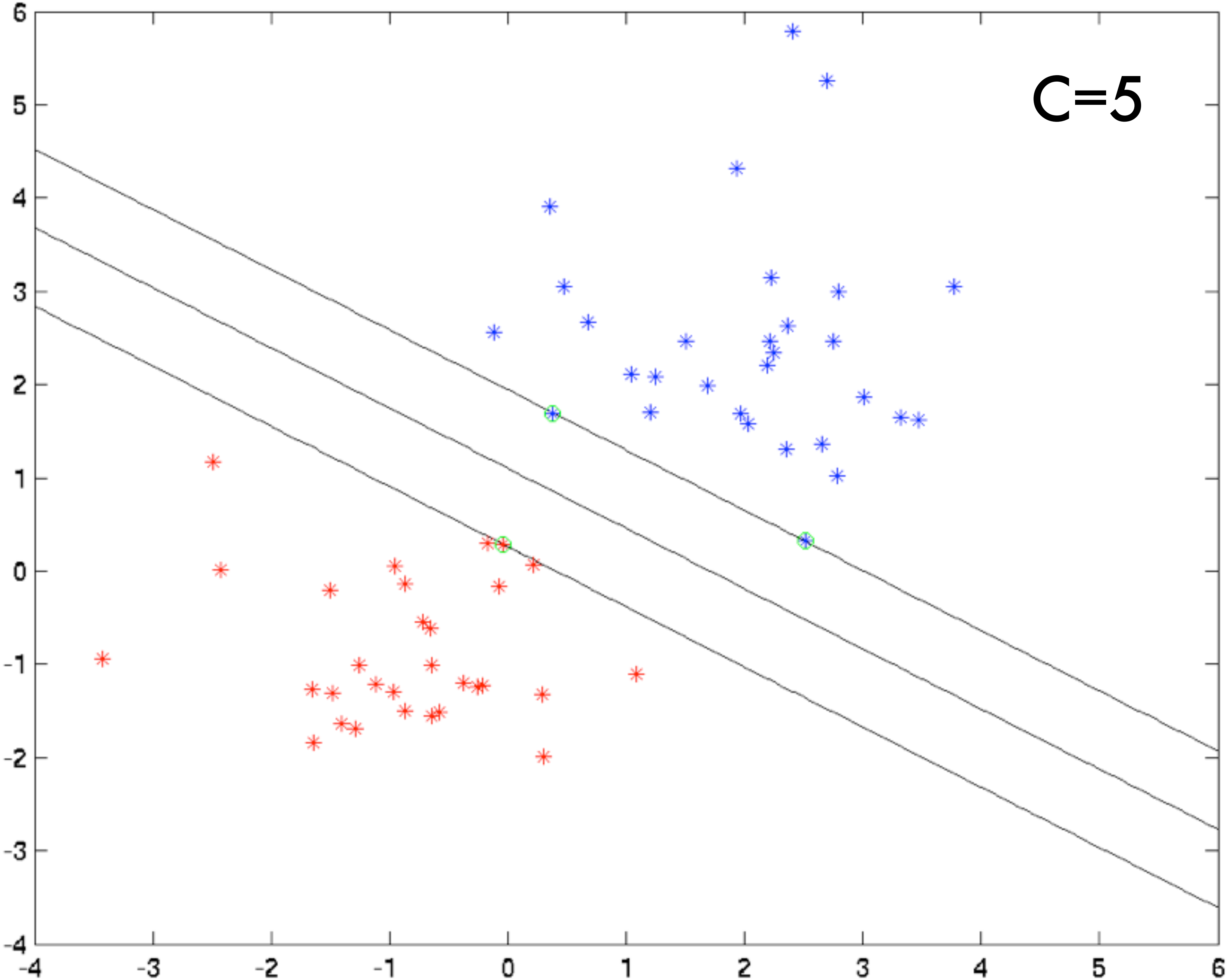


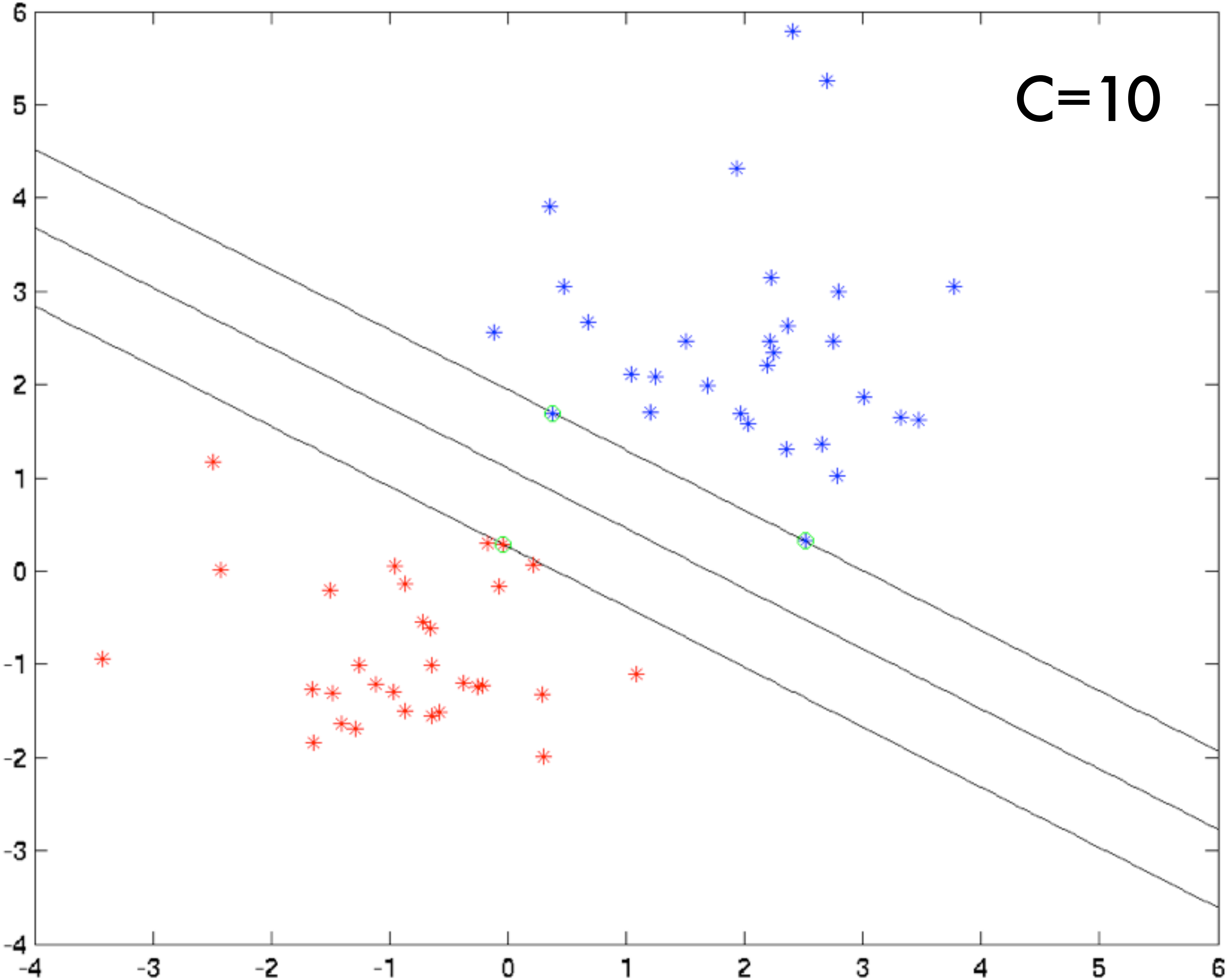
C=100

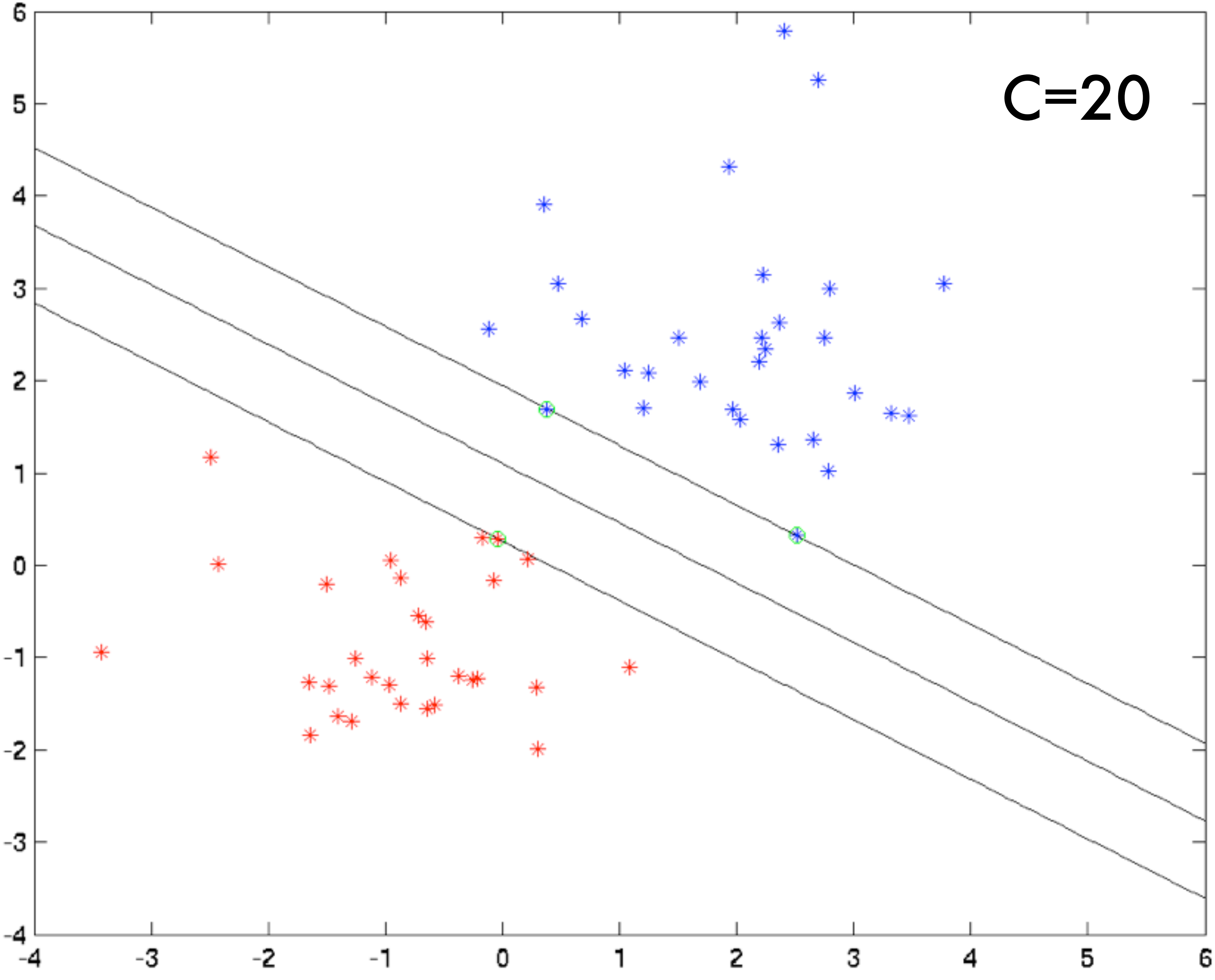


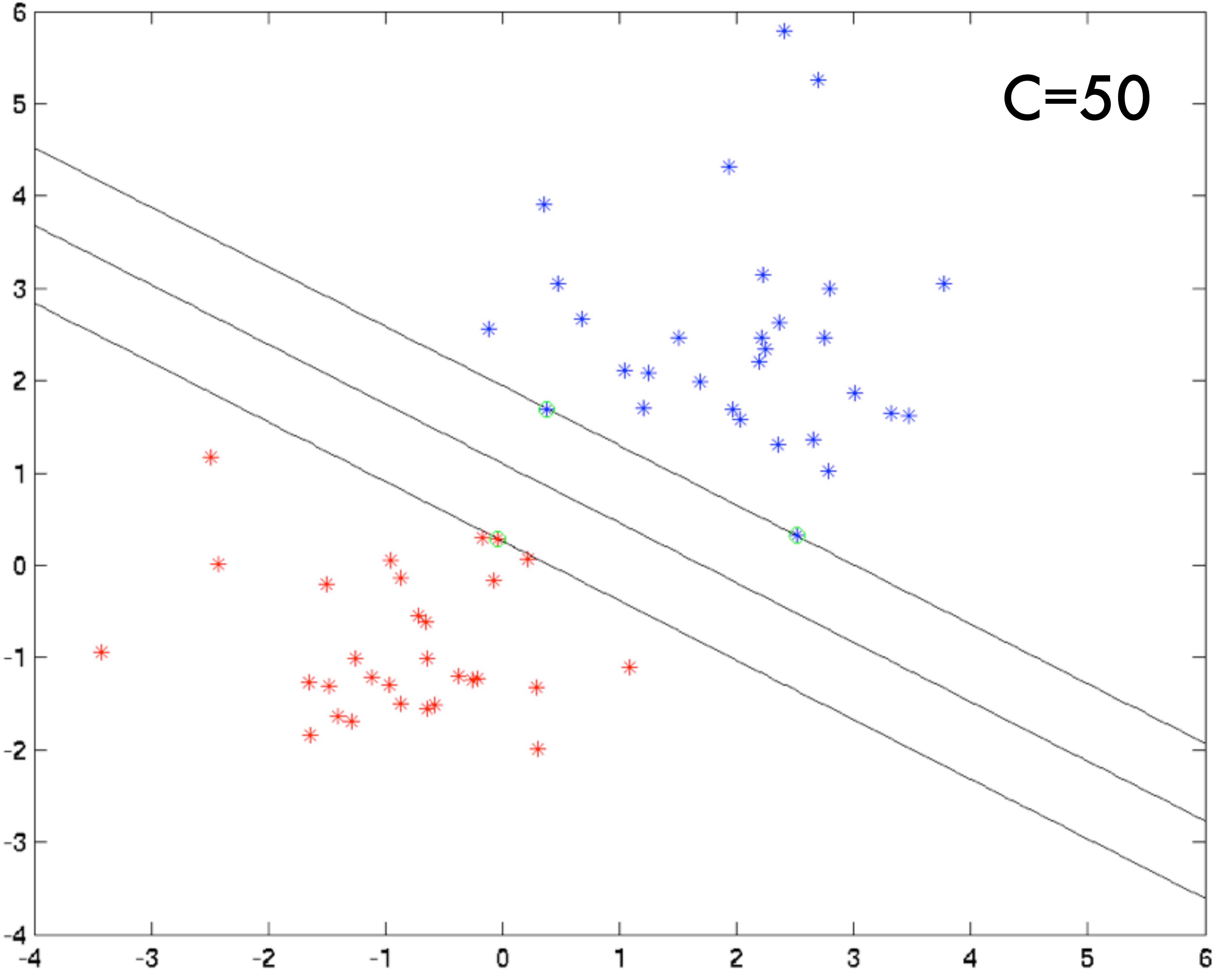
C=1
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

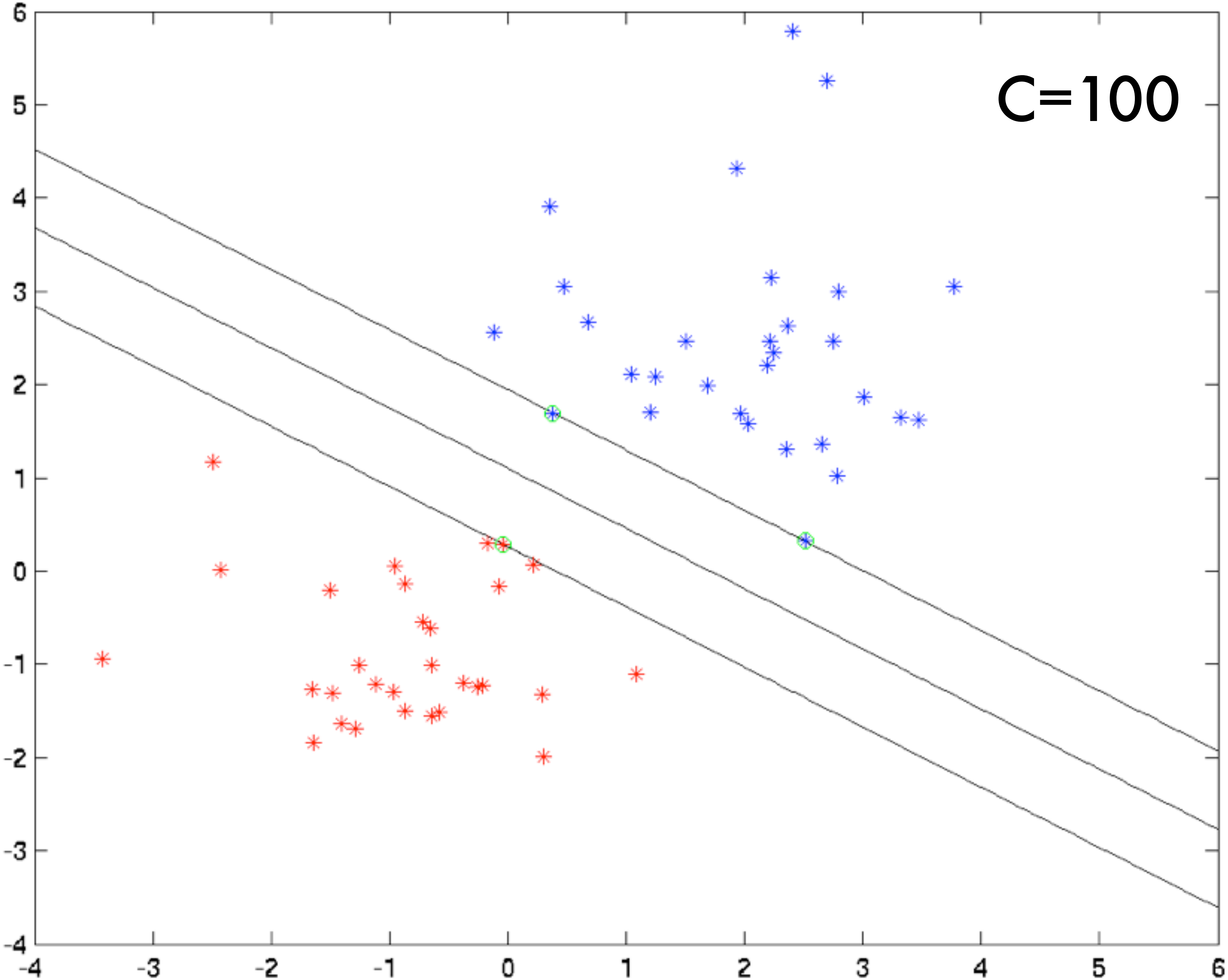

C=5
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

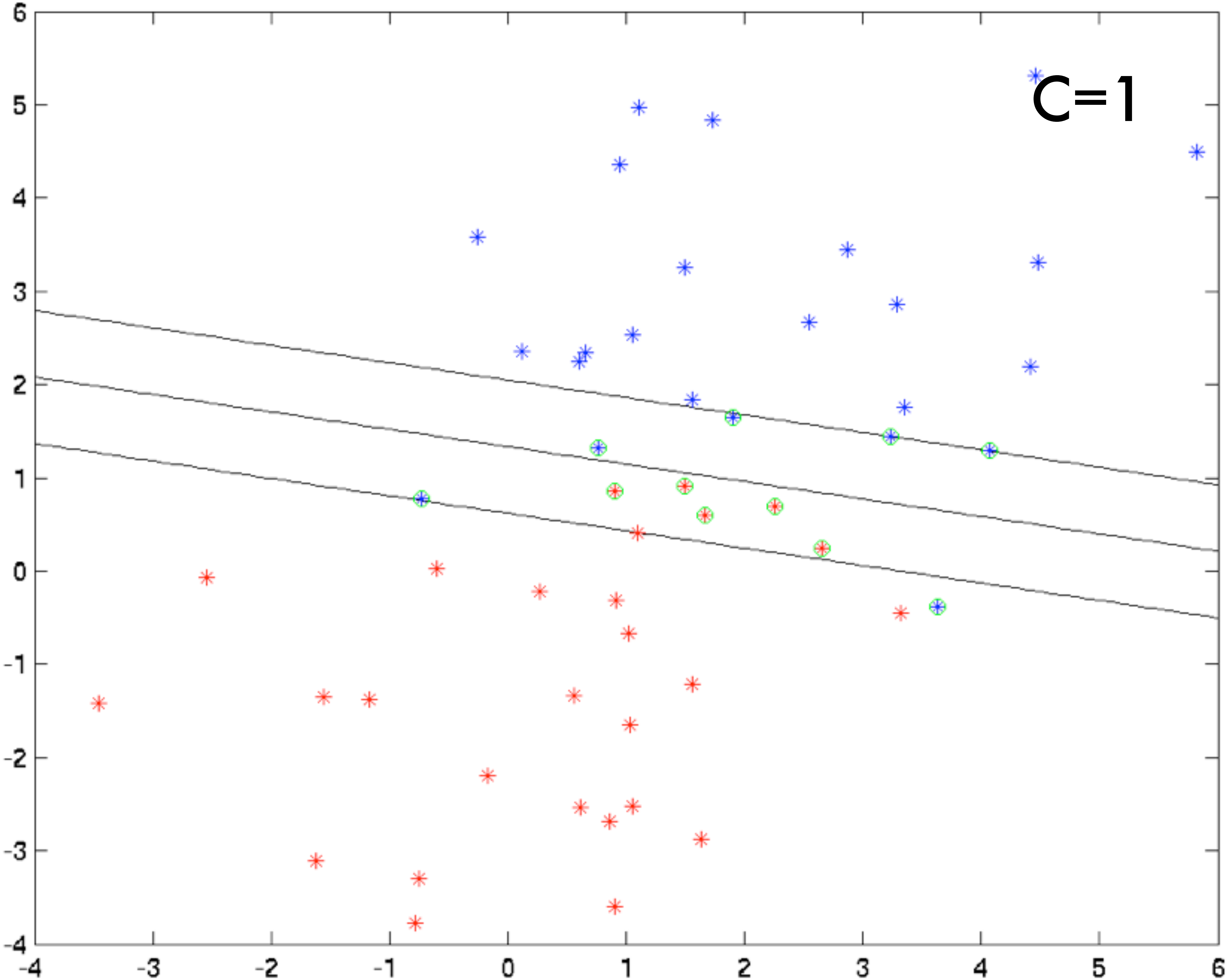

C=10
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

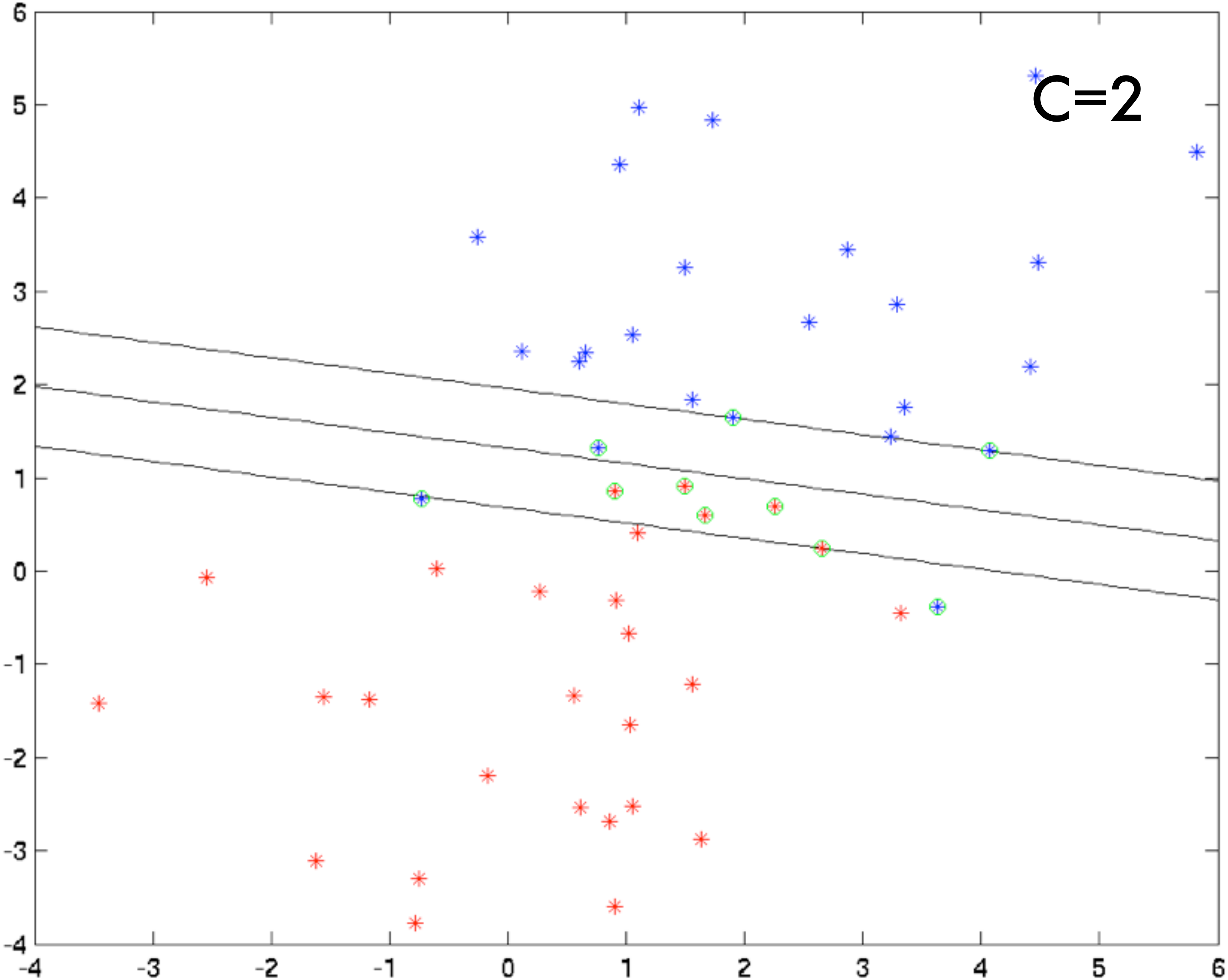

C=20
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

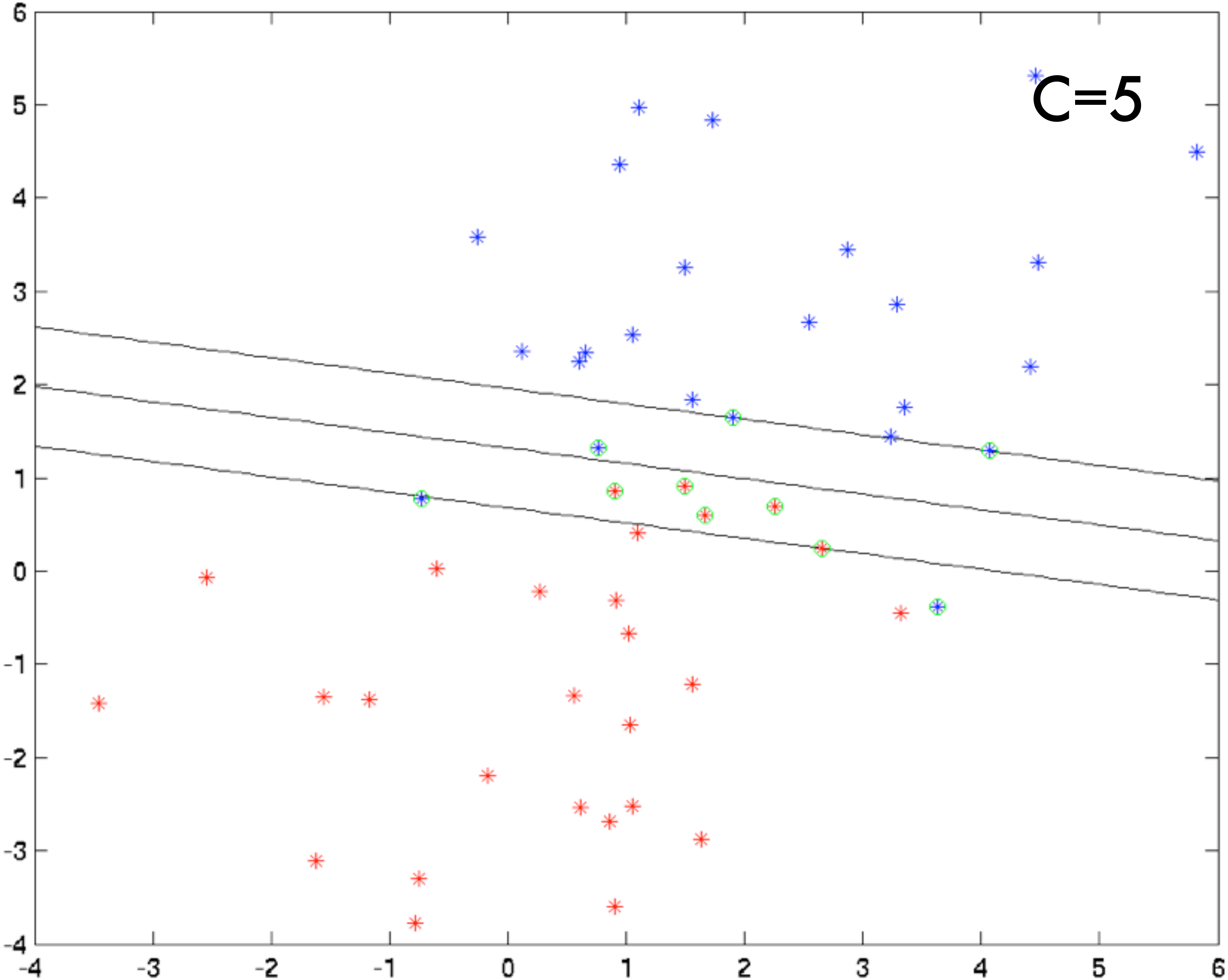


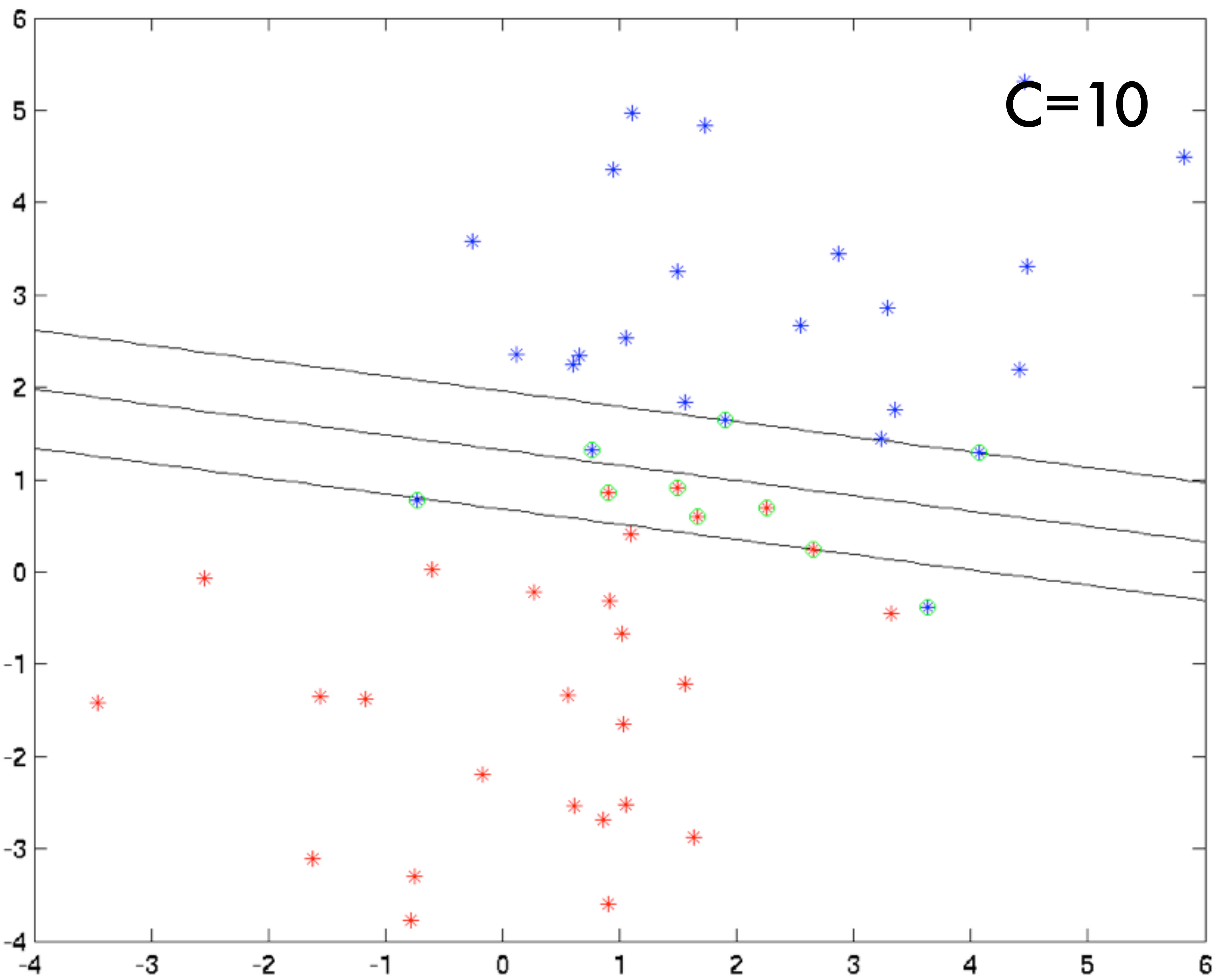
C=50
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

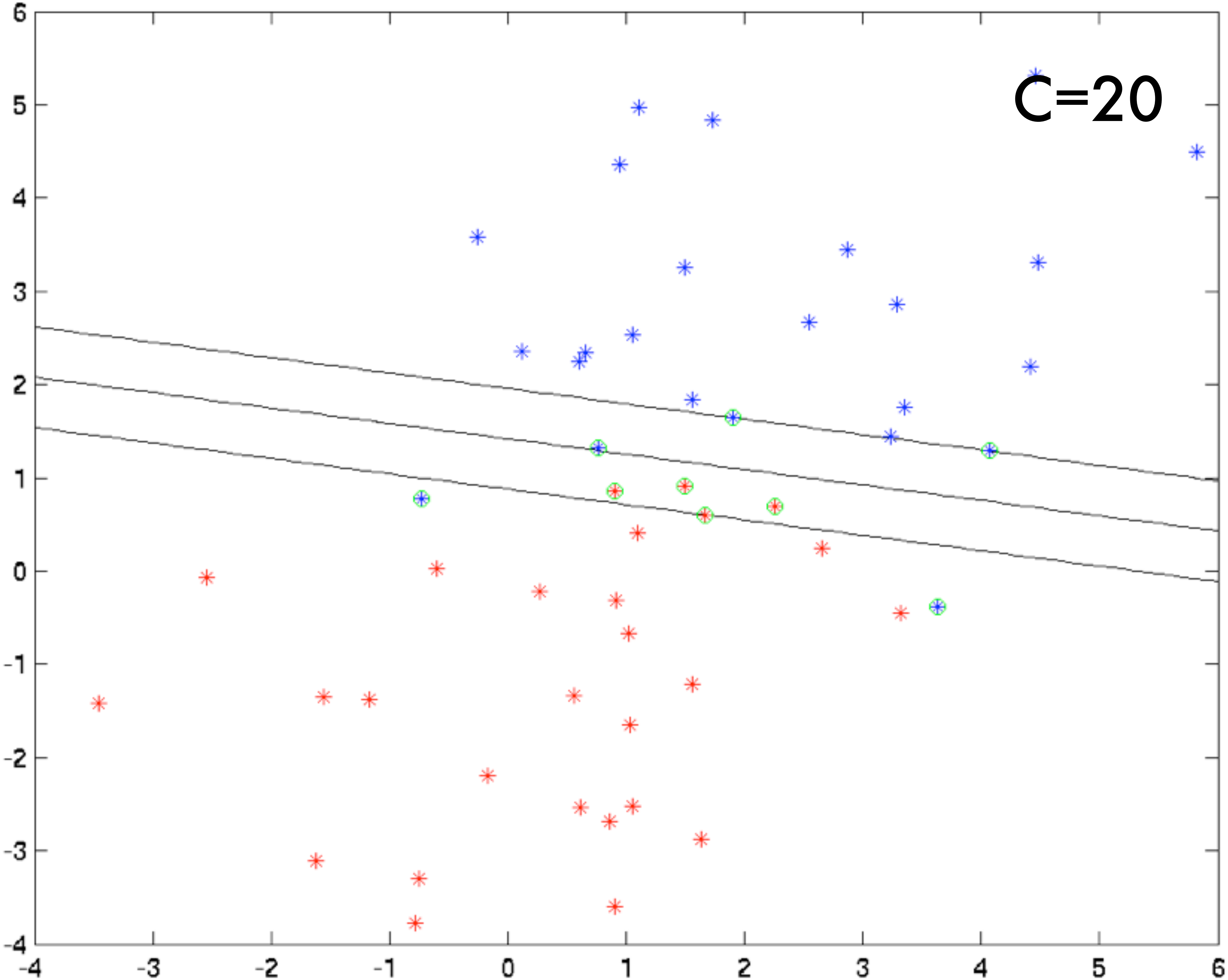

C=100
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

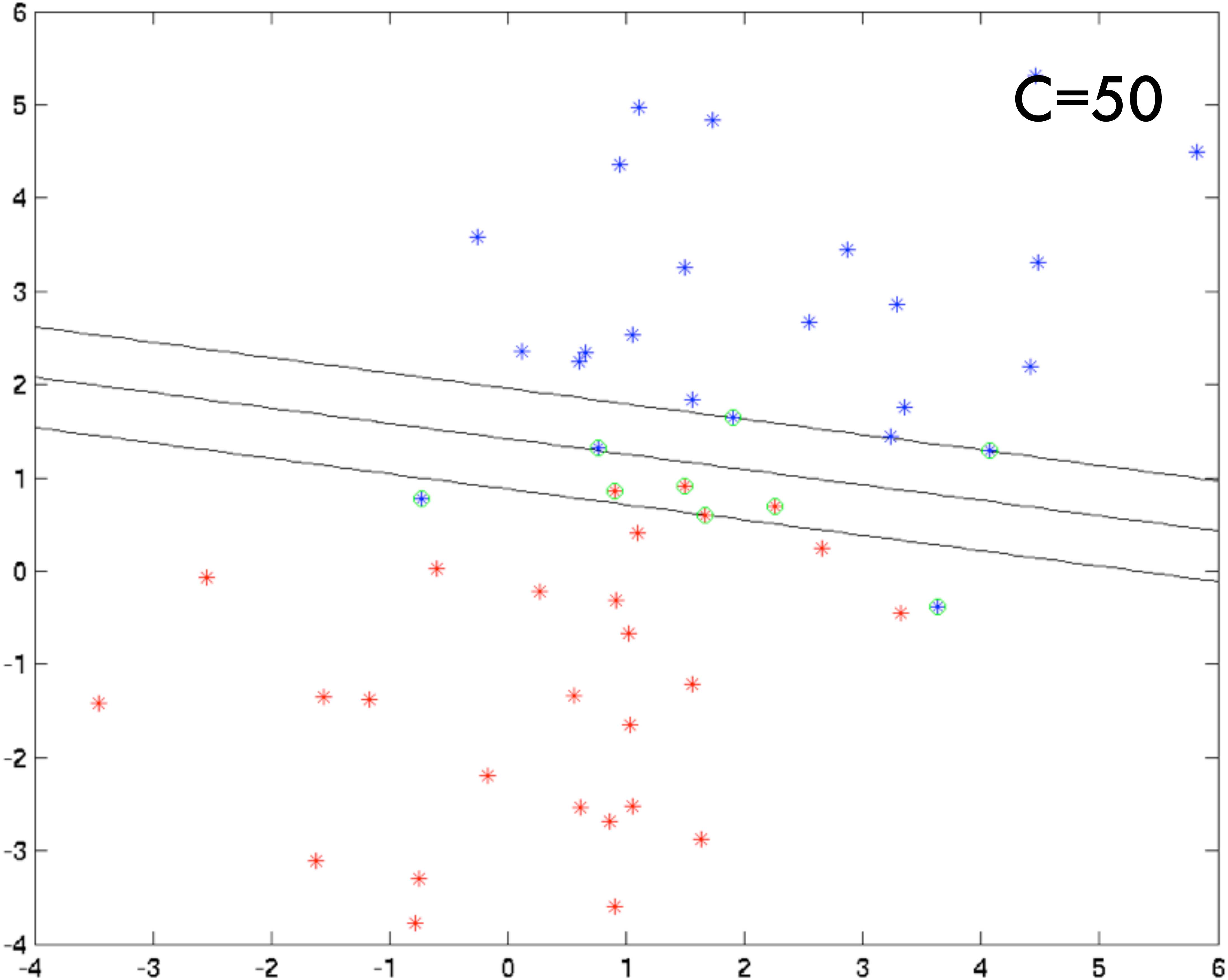

C=1
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

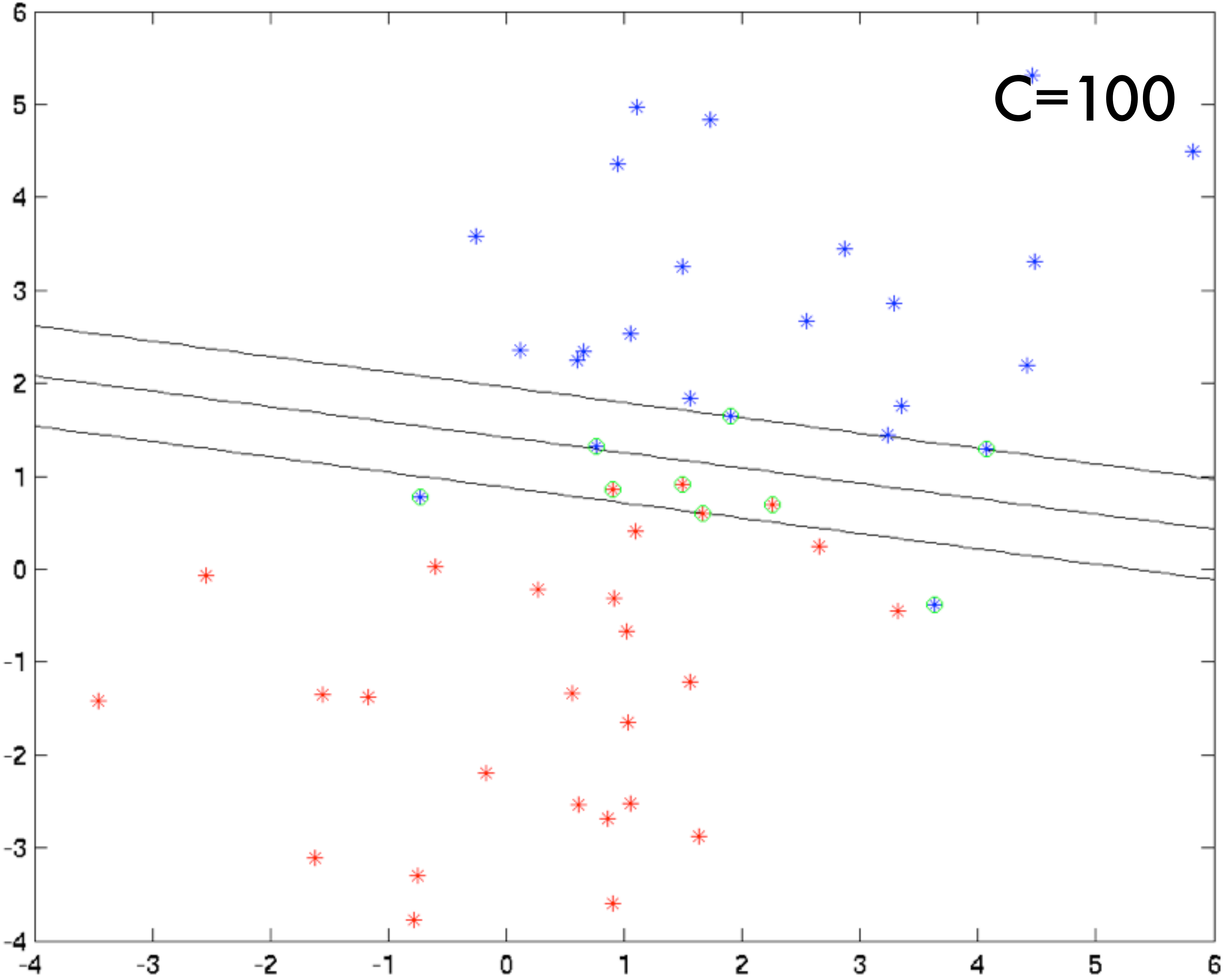

C=2
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


C=5
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


C=10
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


C=20
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


C=50
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6



C=100
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

Solving the optimization problem

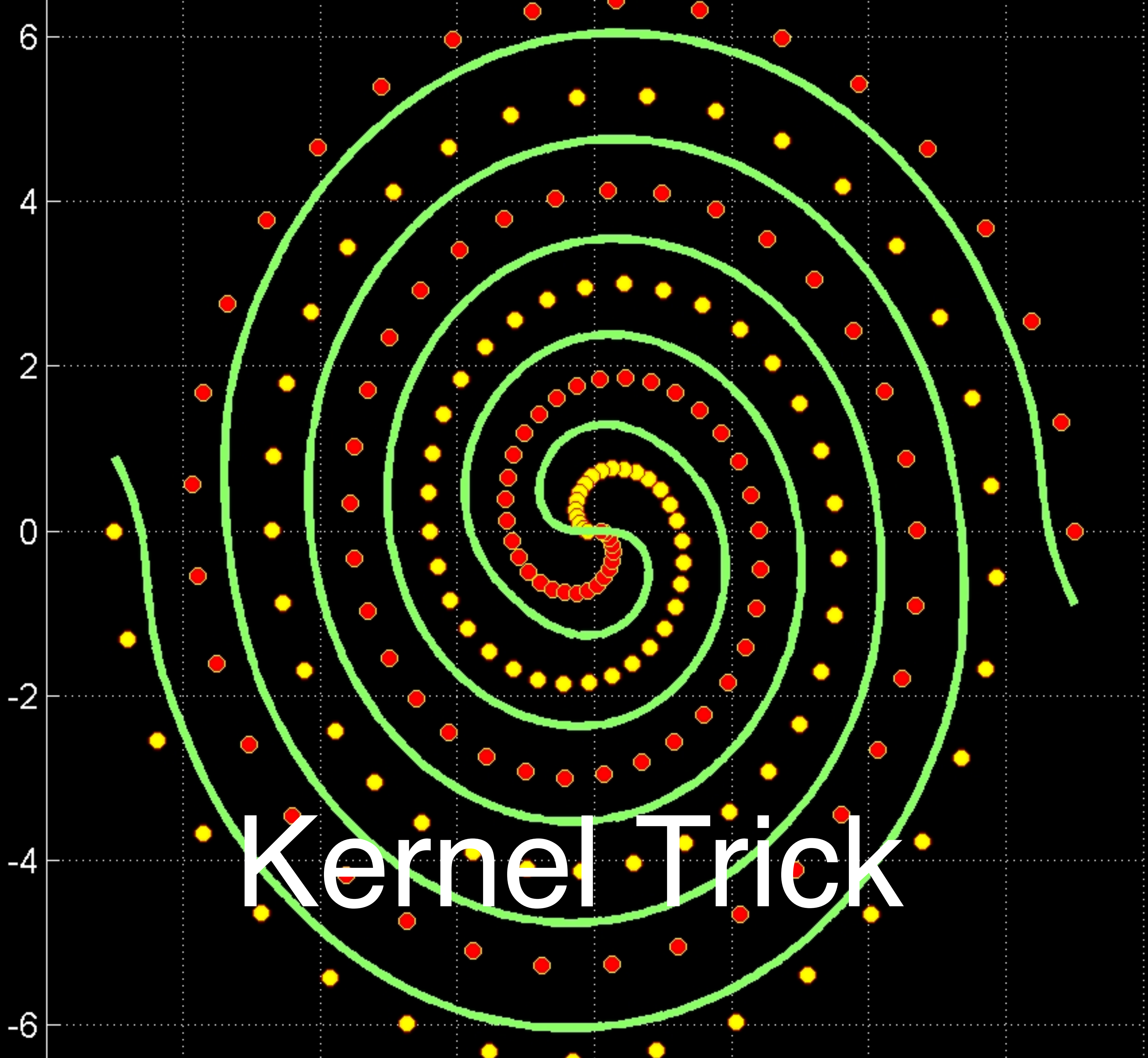
- Dual problem

$$\underset{\alpha}{\text{maximize}} - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_i \alpha_i$$

$$\text{subject to } \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, C]$$

- If problem is small enough (1000s of variables) we can use off-the-shelf solver (CVXOPT, CPLEX, OOQP, LOQO)
- For larger problem use fact that only SVs matter and solve in blocks (active set method).

Kernel Trick



The Kernel Trick

- Linear soft margin problem

$$\underset{w,b}{\text{minimize}} \ \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

$$\text{subject to } y_i\left[\langle w, x_i\rangle + b\right] \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

- Dual problem

$$\underset{\alpha}{\text{maximize}} - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle + \sum_i \alpha_i$$

$$\text{subject to } \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, C]$$

- Support vector expansion

$$f(x) = \sum_i \alpha_i y_i \langle x_i, x\rangle + b$$

The Kernel Trick

- Linear soft margin problem

$$\underset{w,b}{\text{minimize}} \ \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

$$\text{subject to } y_i\left[\langle w, \phi(x_i)\rangle + b\right] \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

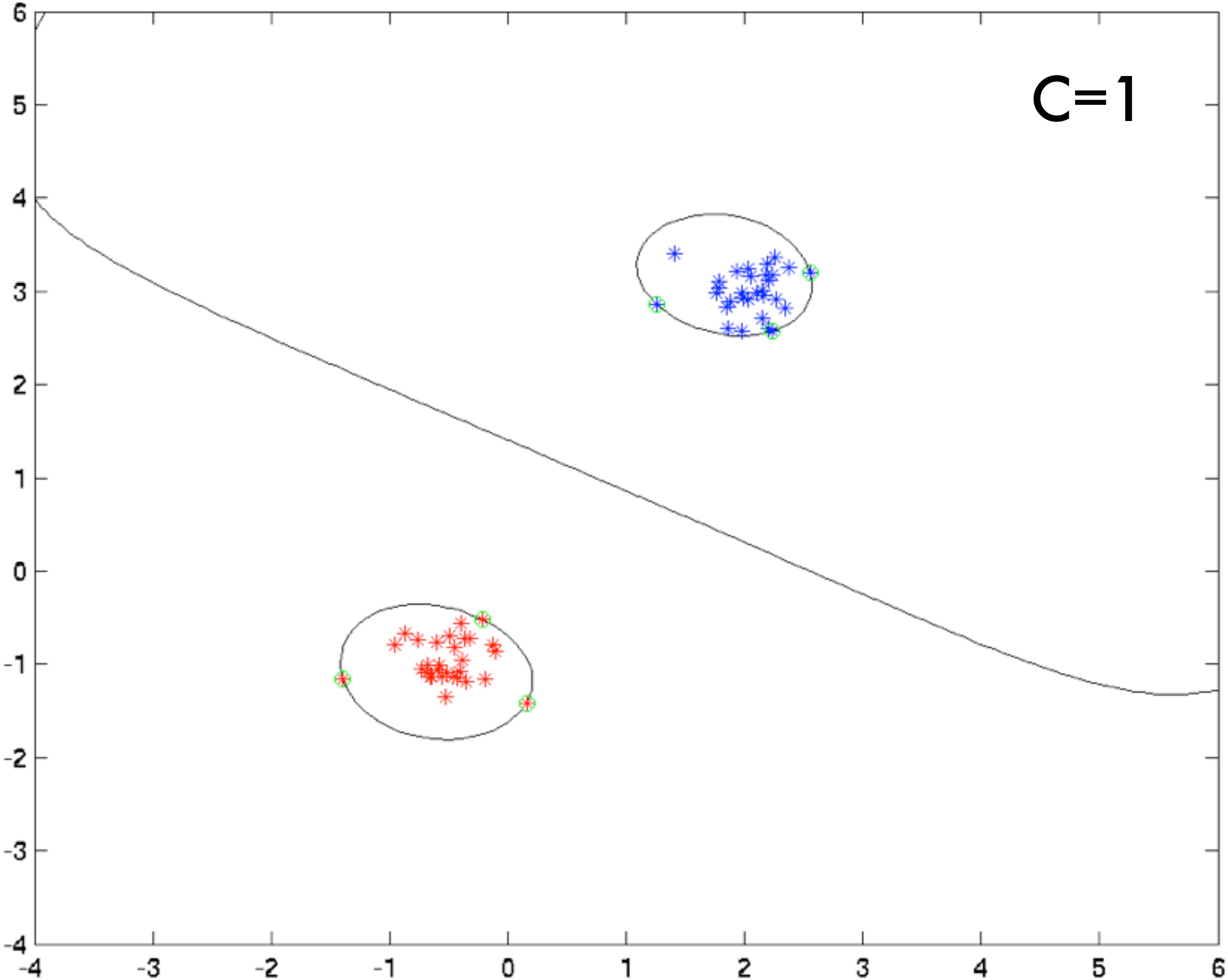
- Dual problem

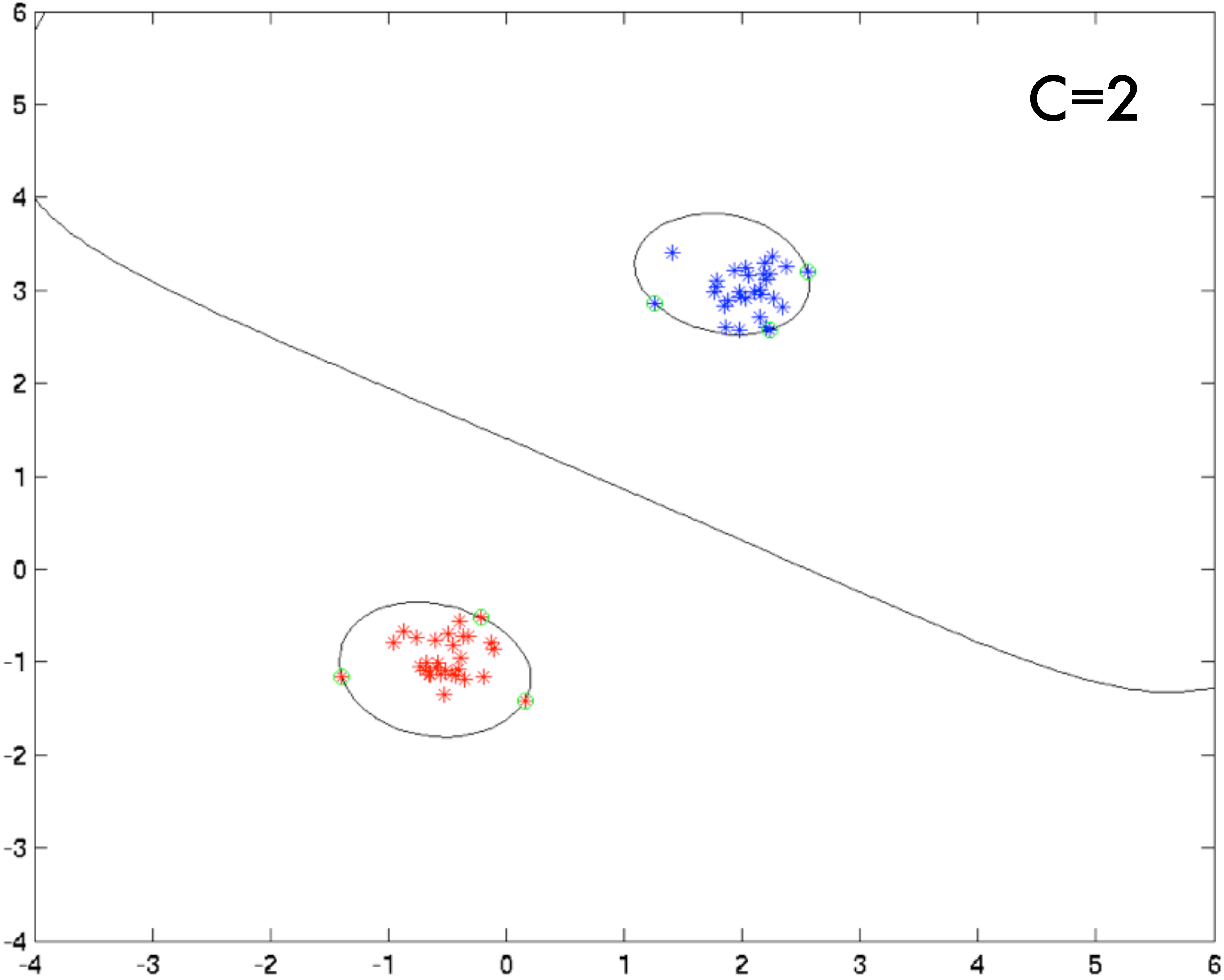
$$\underset{\alpha}{\text{maximize}} - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(x_i, x_j) + \sum_i \alpha_i$$

$$\text{subject to } \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, C]$$

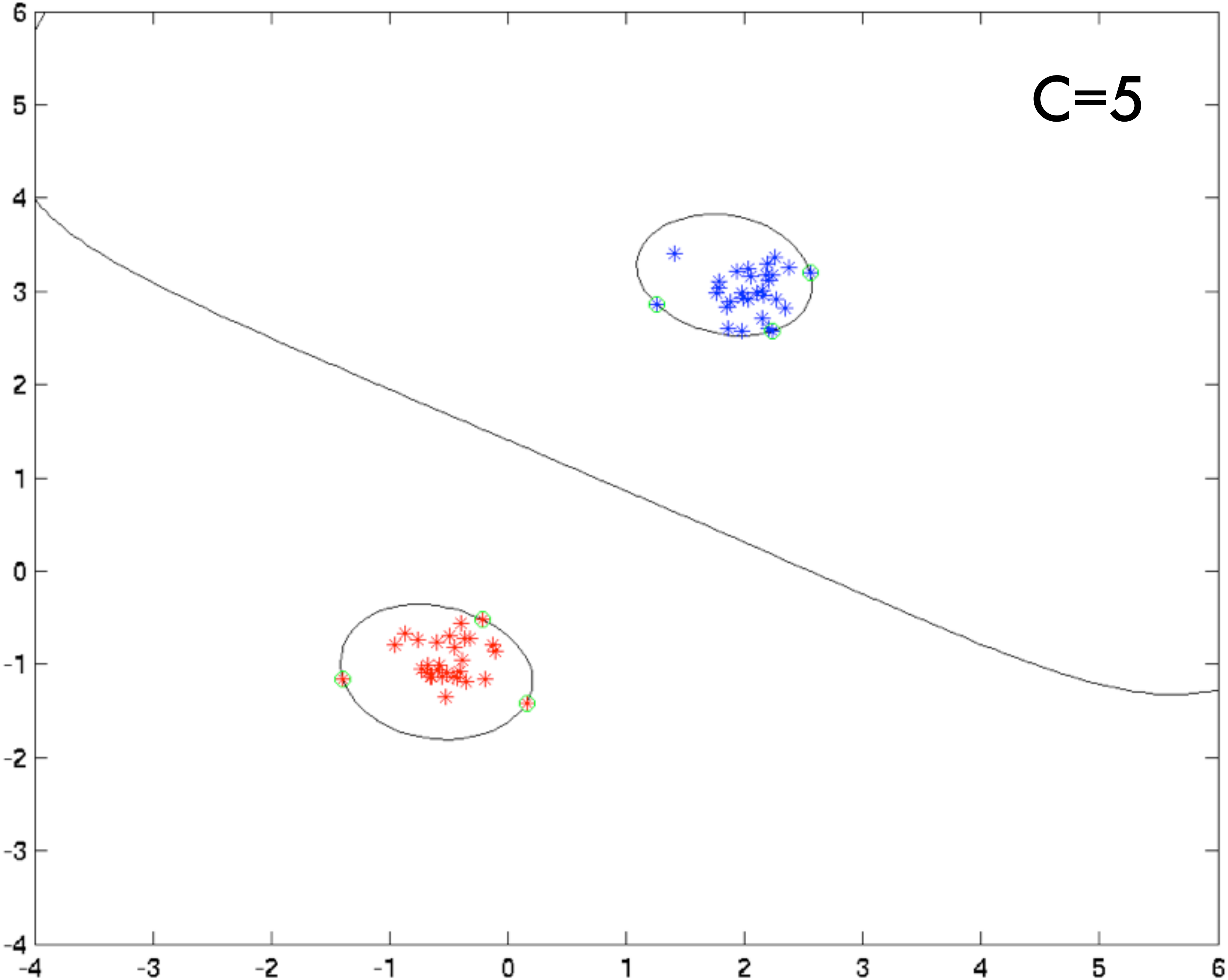
- Support vector expansion

$$f(x) = \sum_i \alpha_i y_i k(x_i, x) + b$$

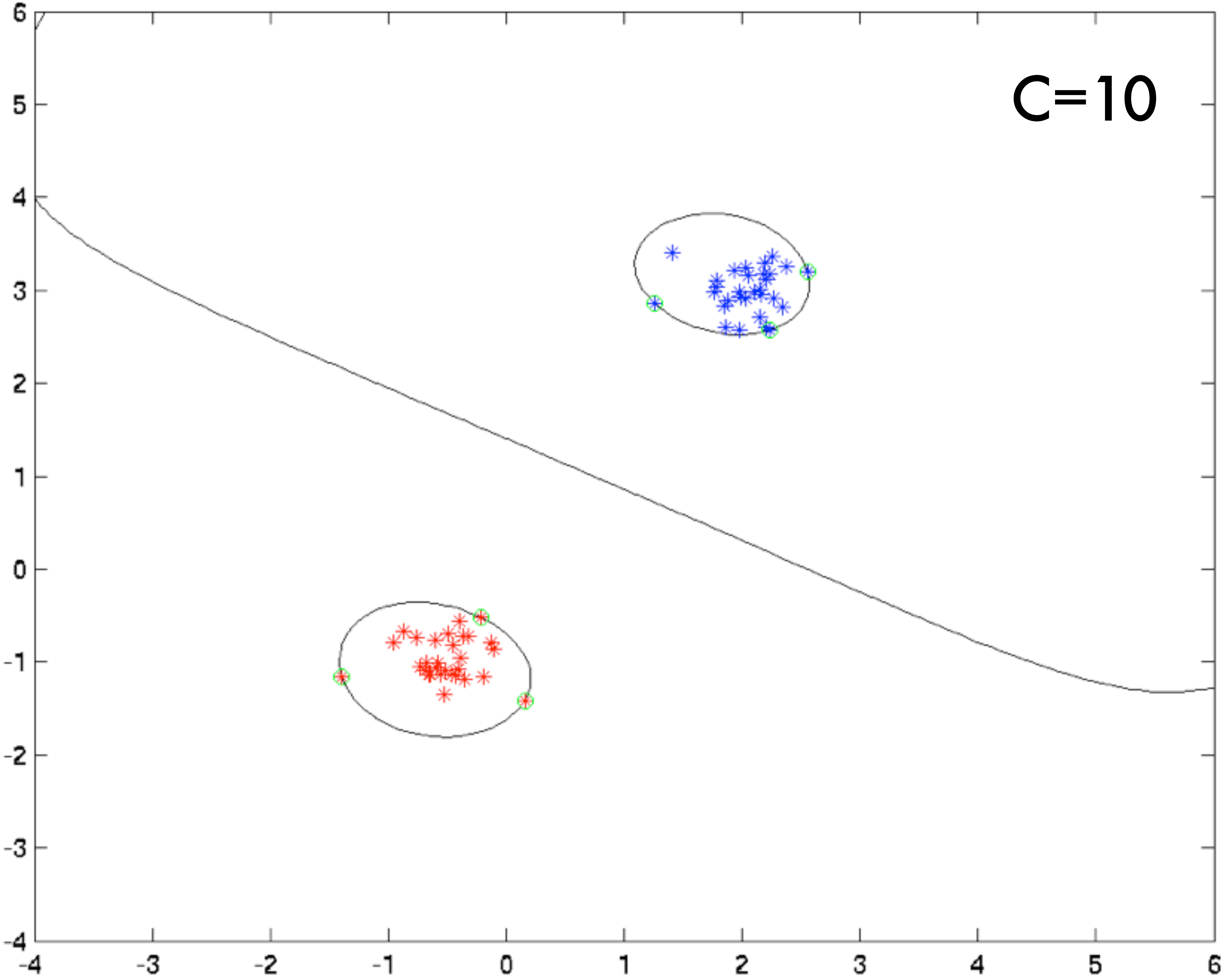

C=1
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

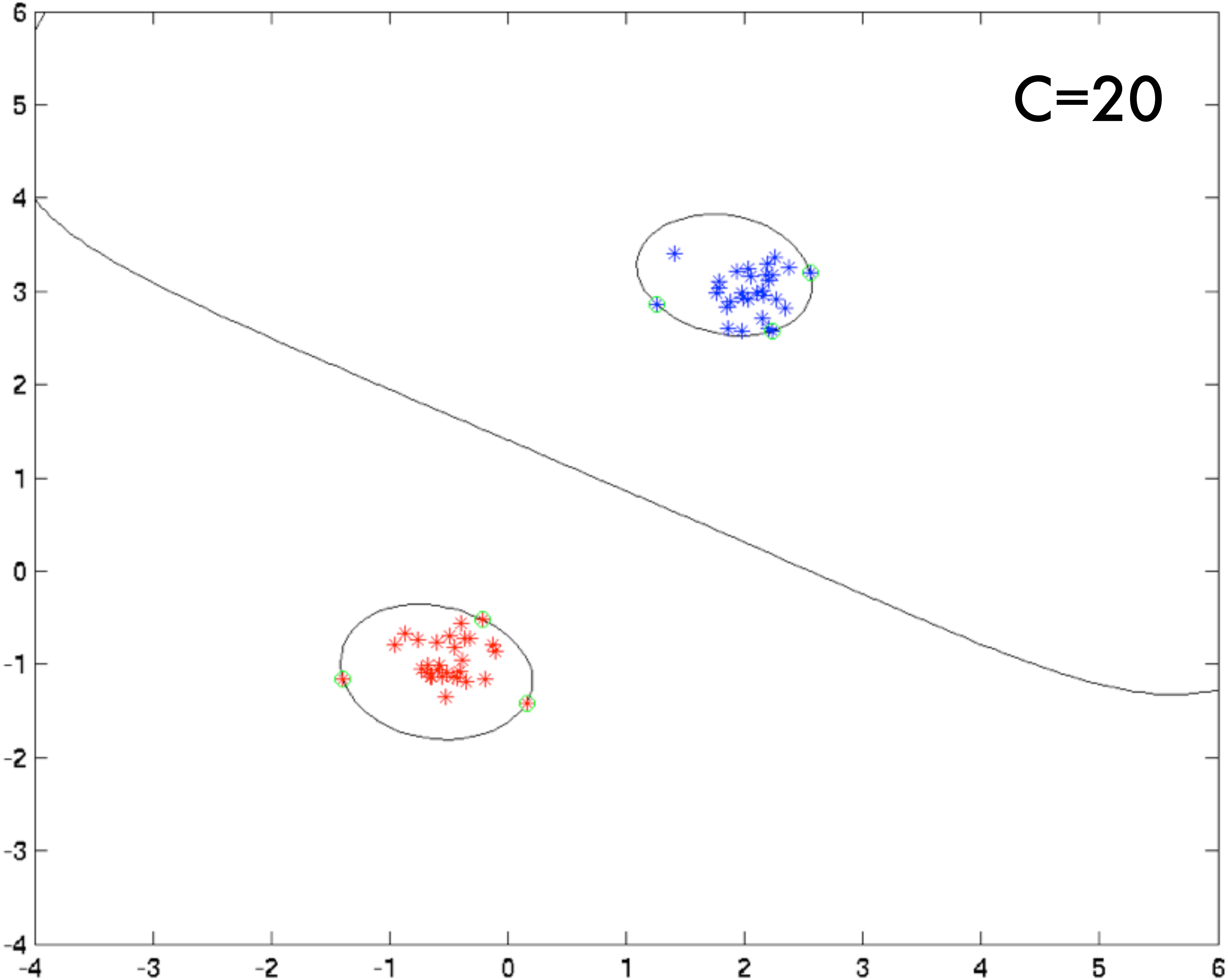


C=2
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

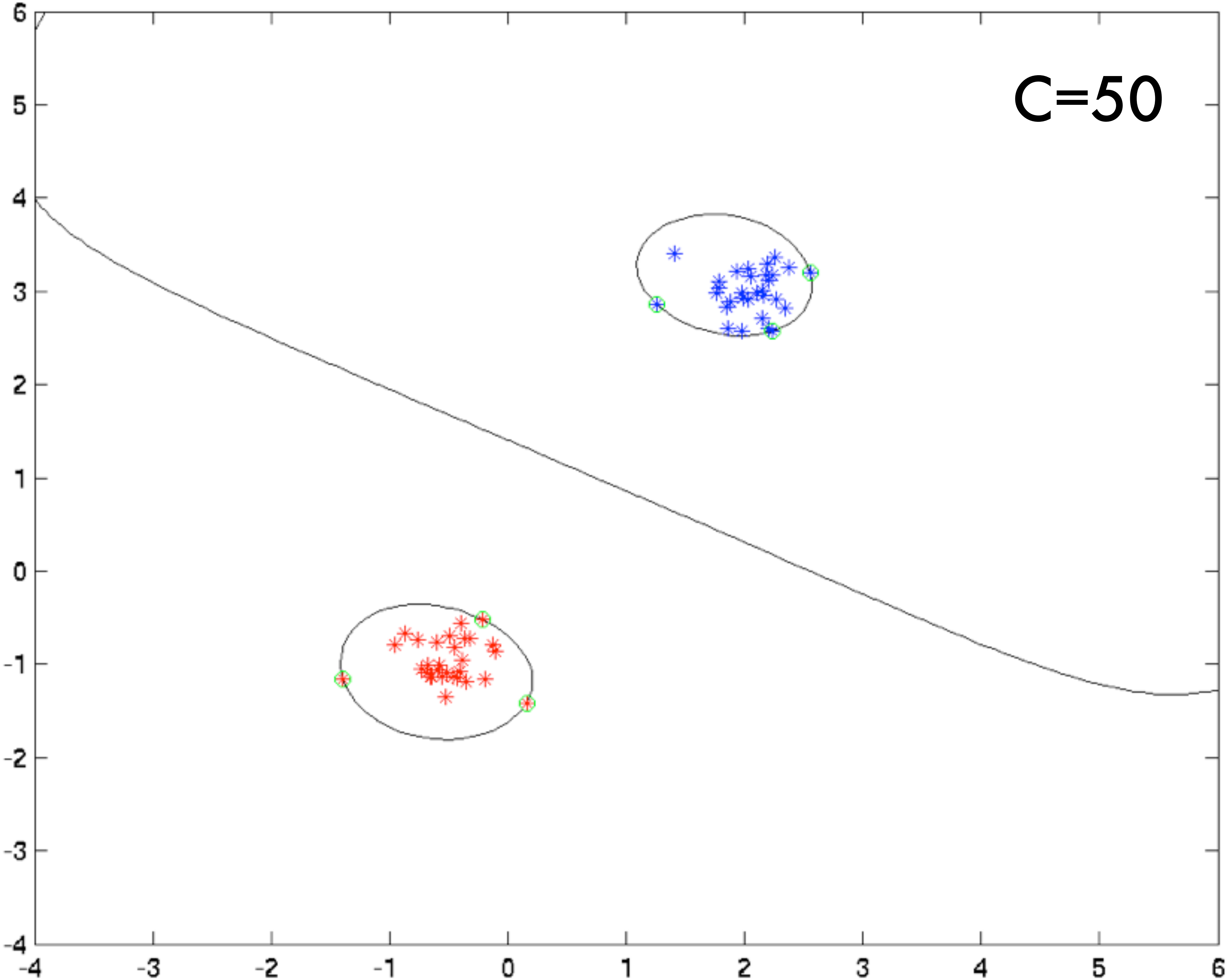


C=5
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

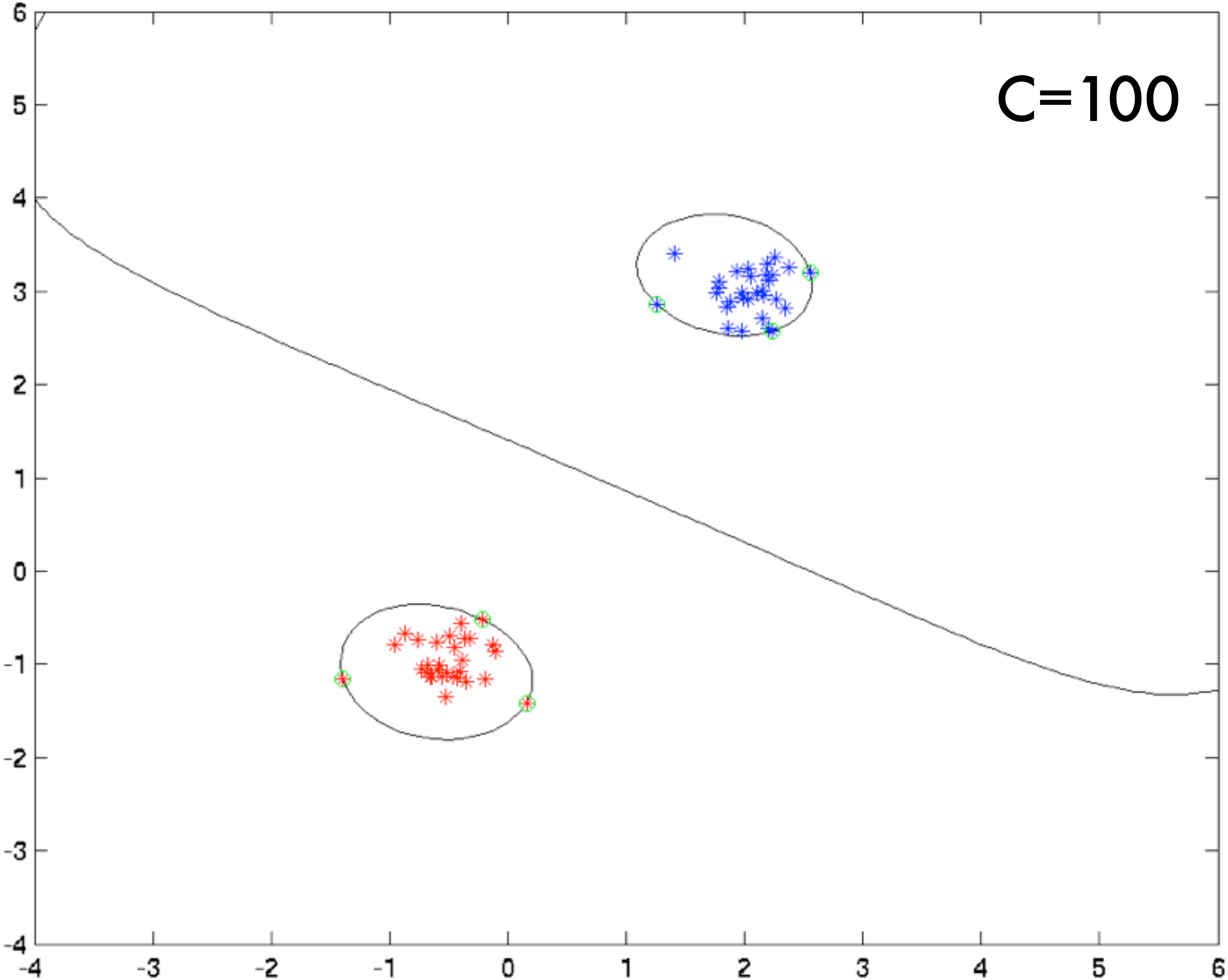

C=10
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6



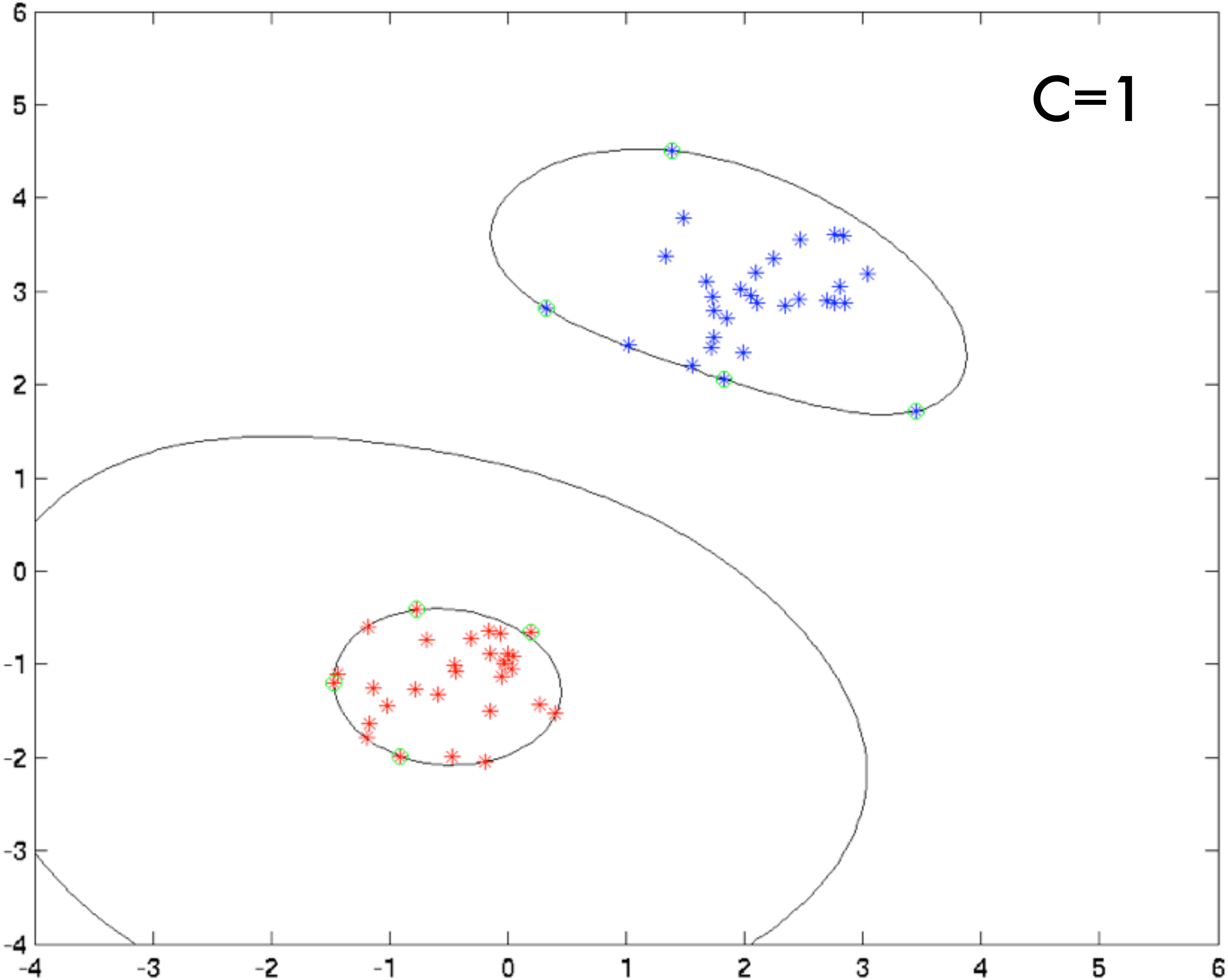
C=20
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

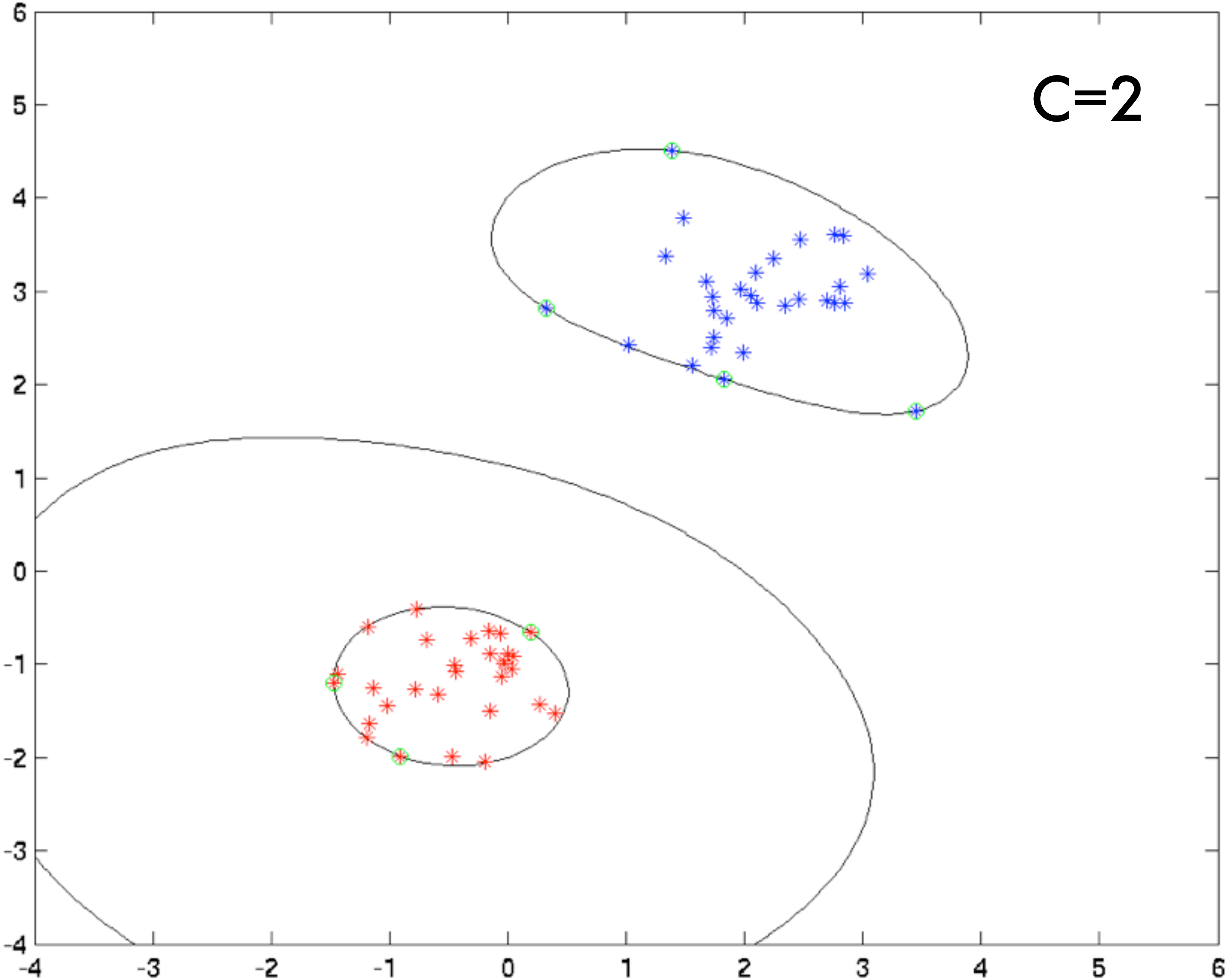


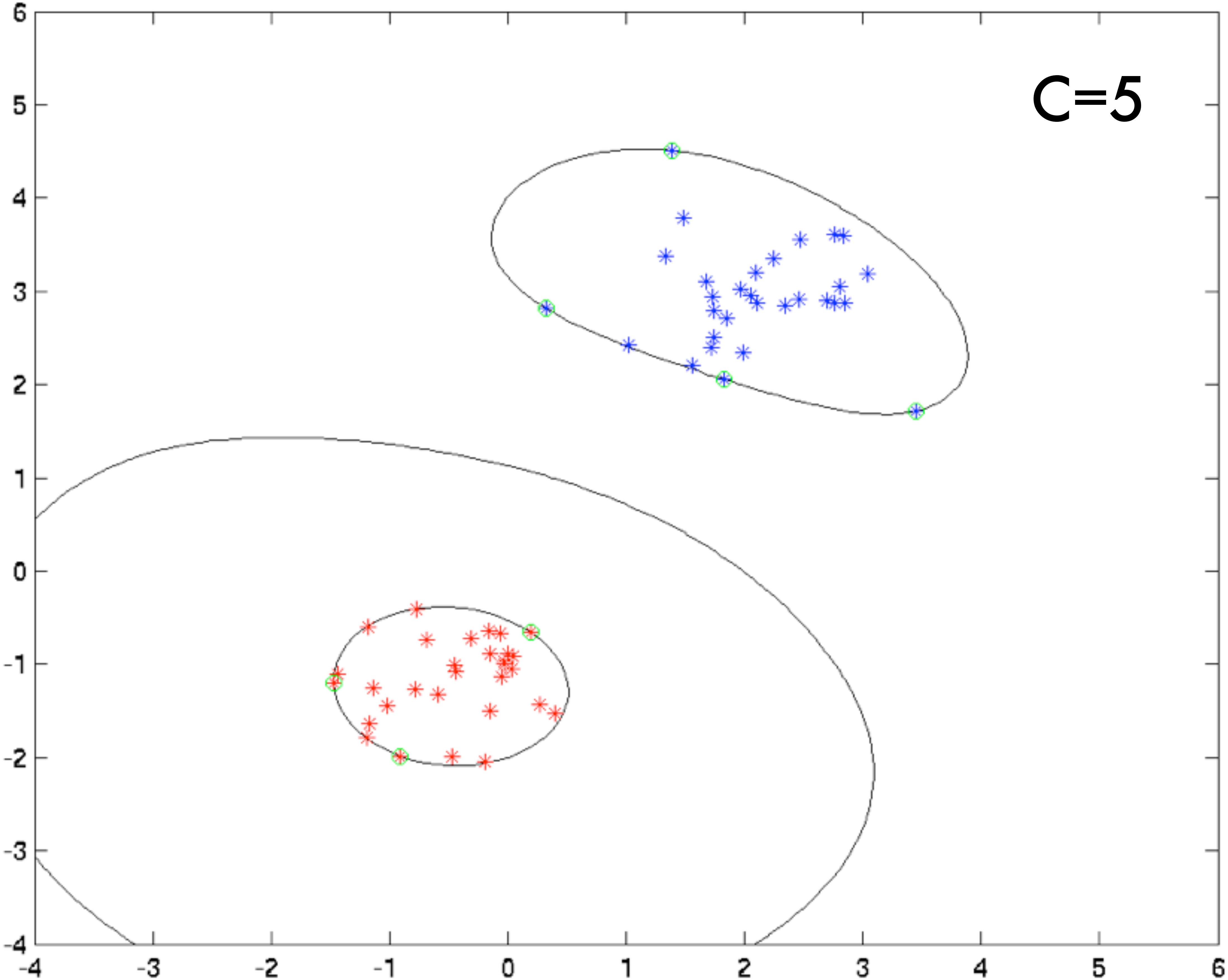
C=50
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

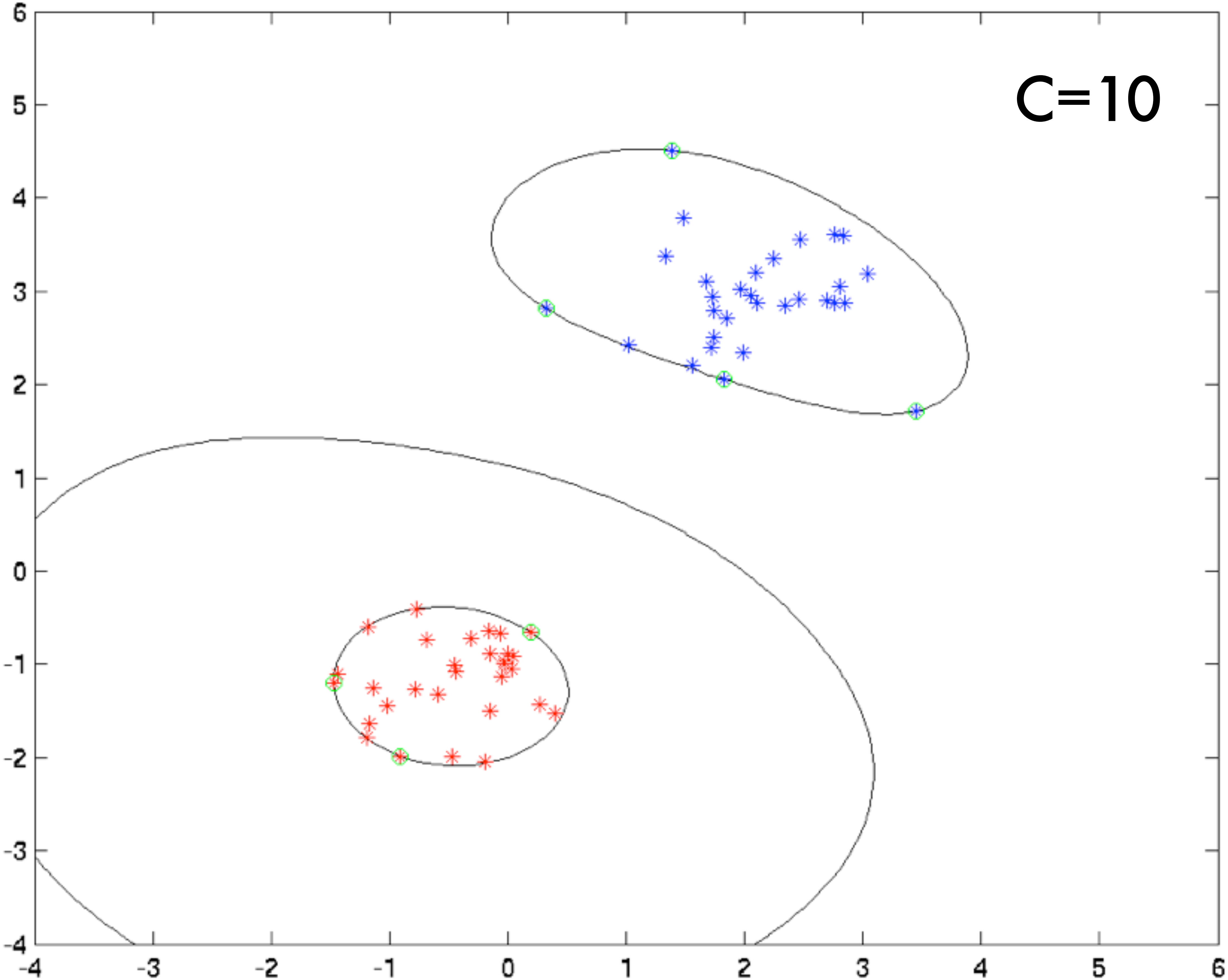


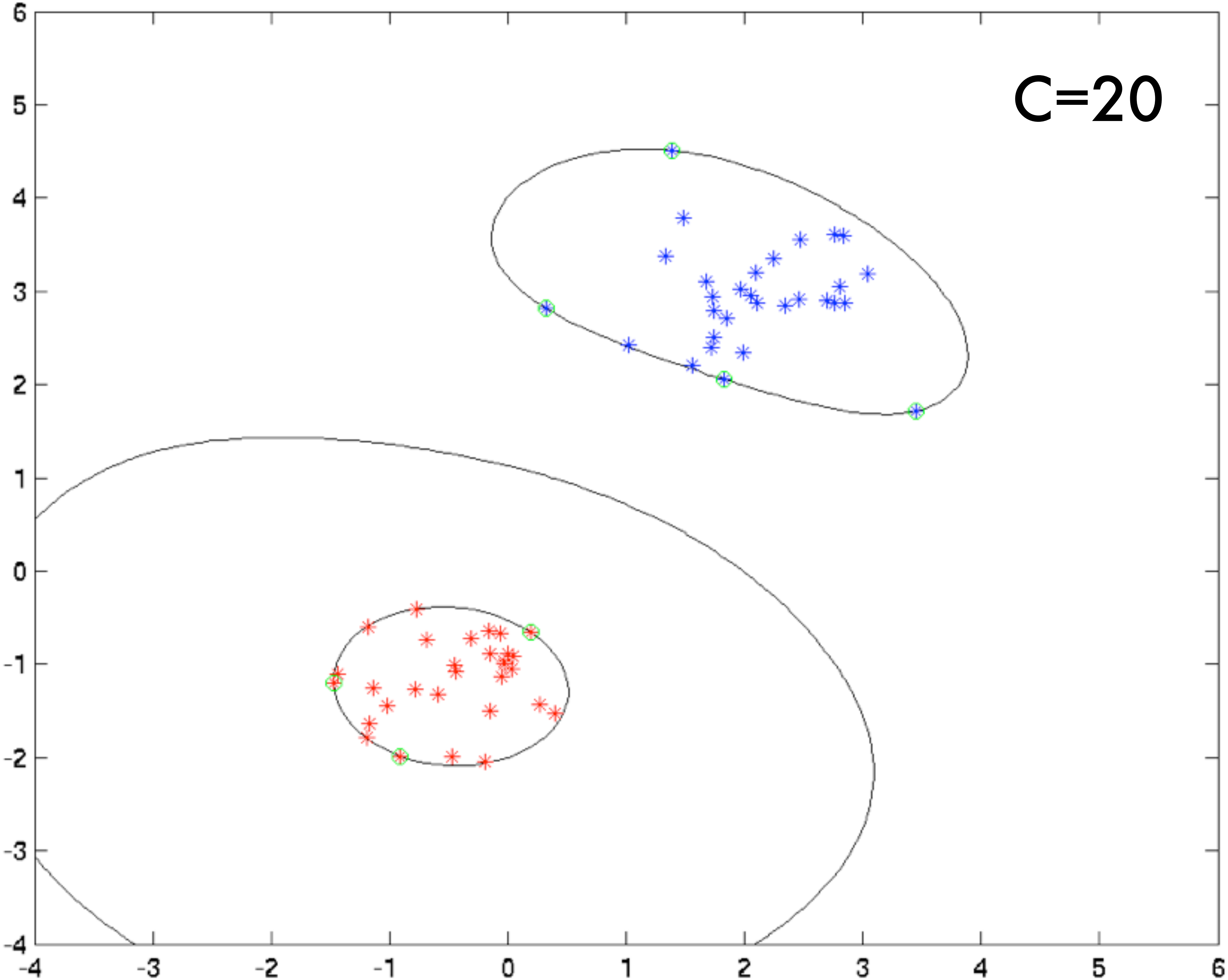
C=100
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

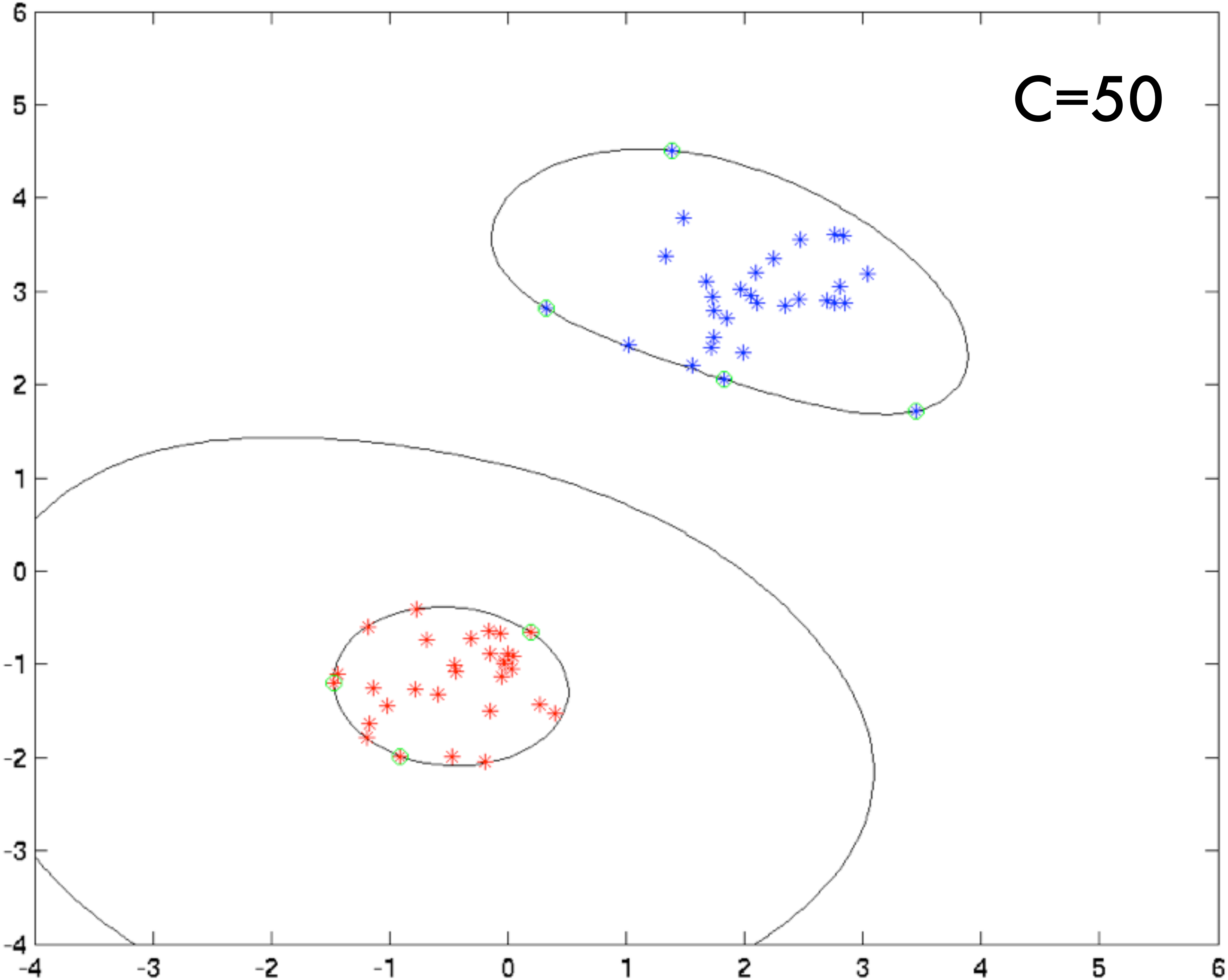

C=1
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


C=2
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

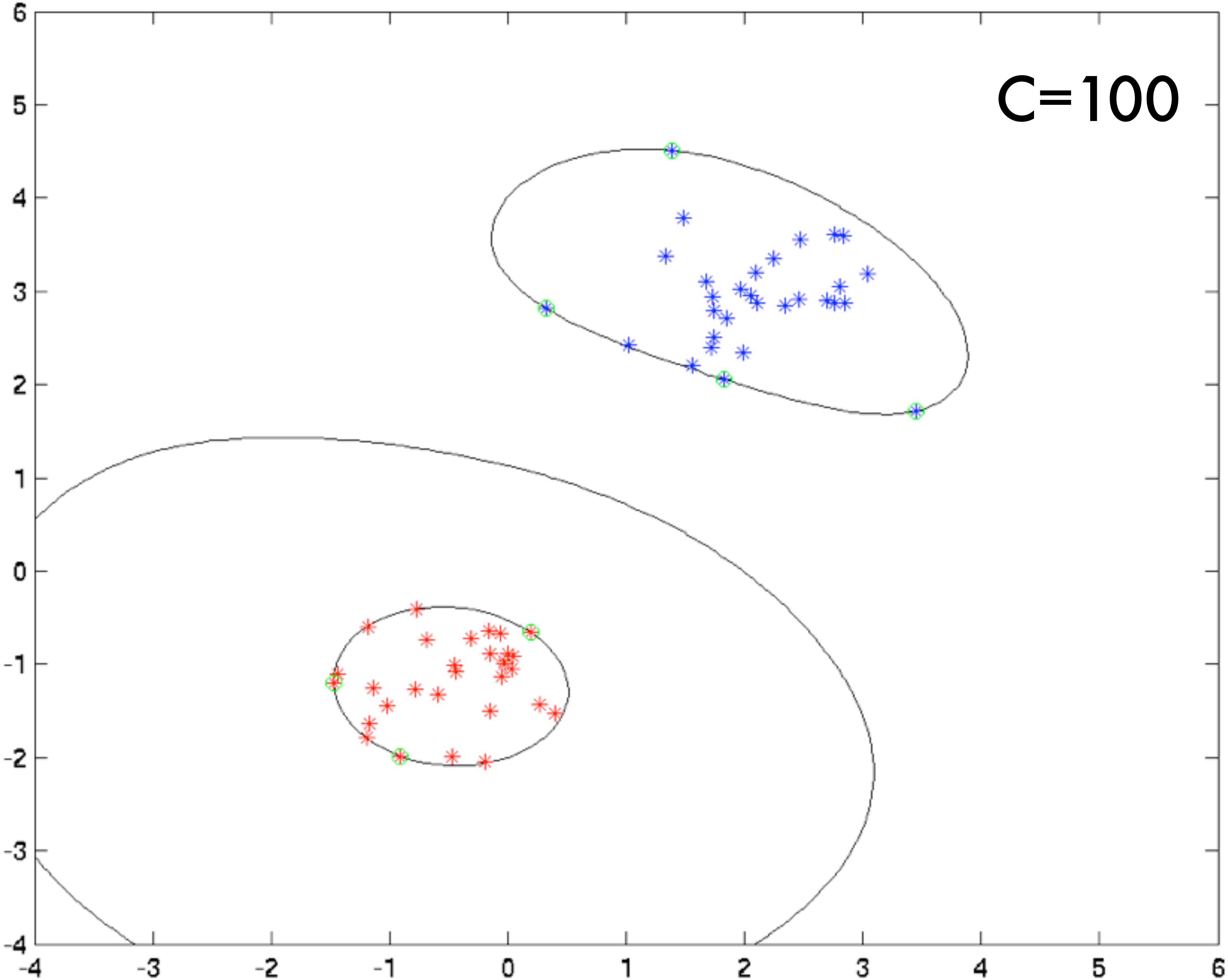

C=5
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

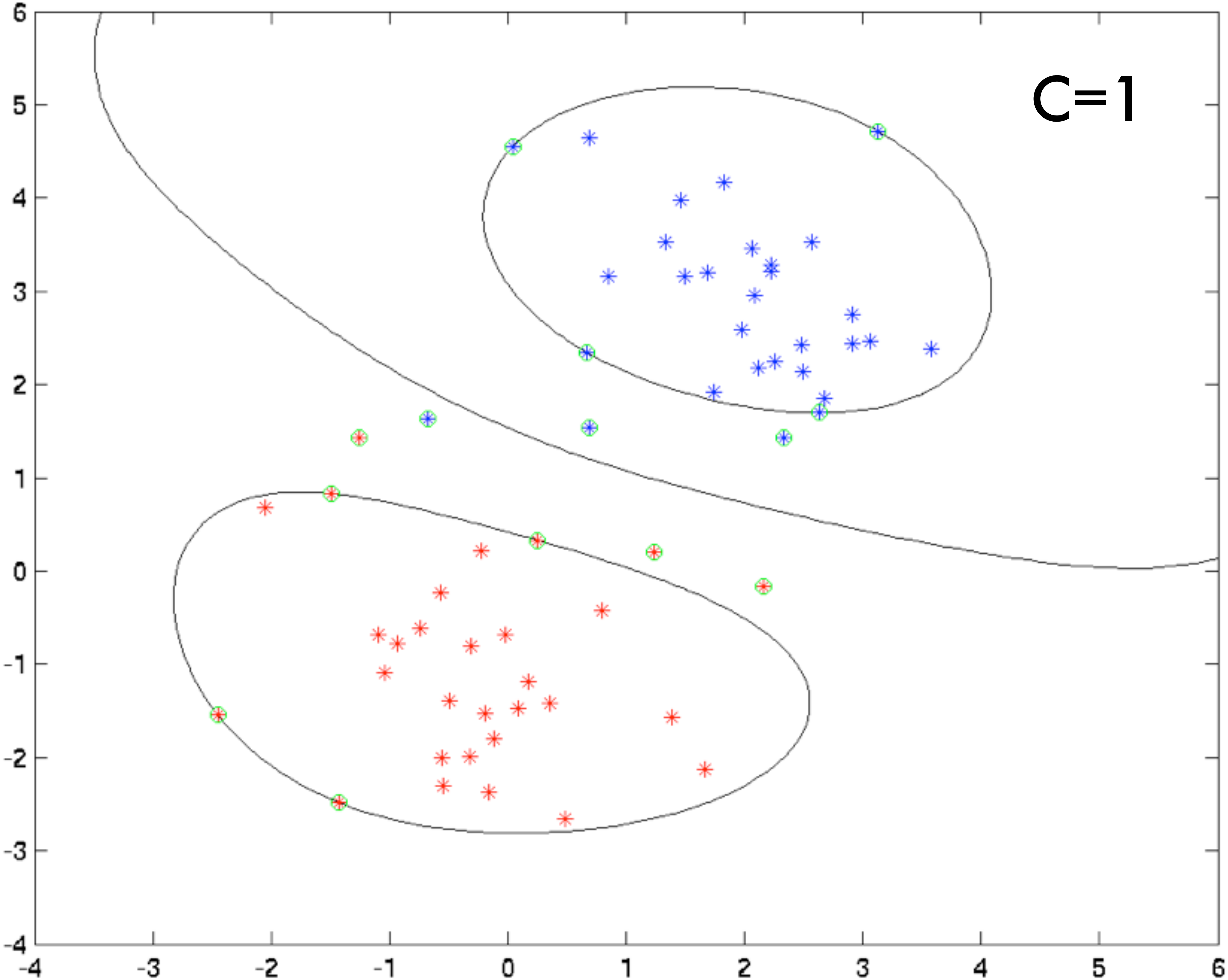

C=10
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


C=20
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

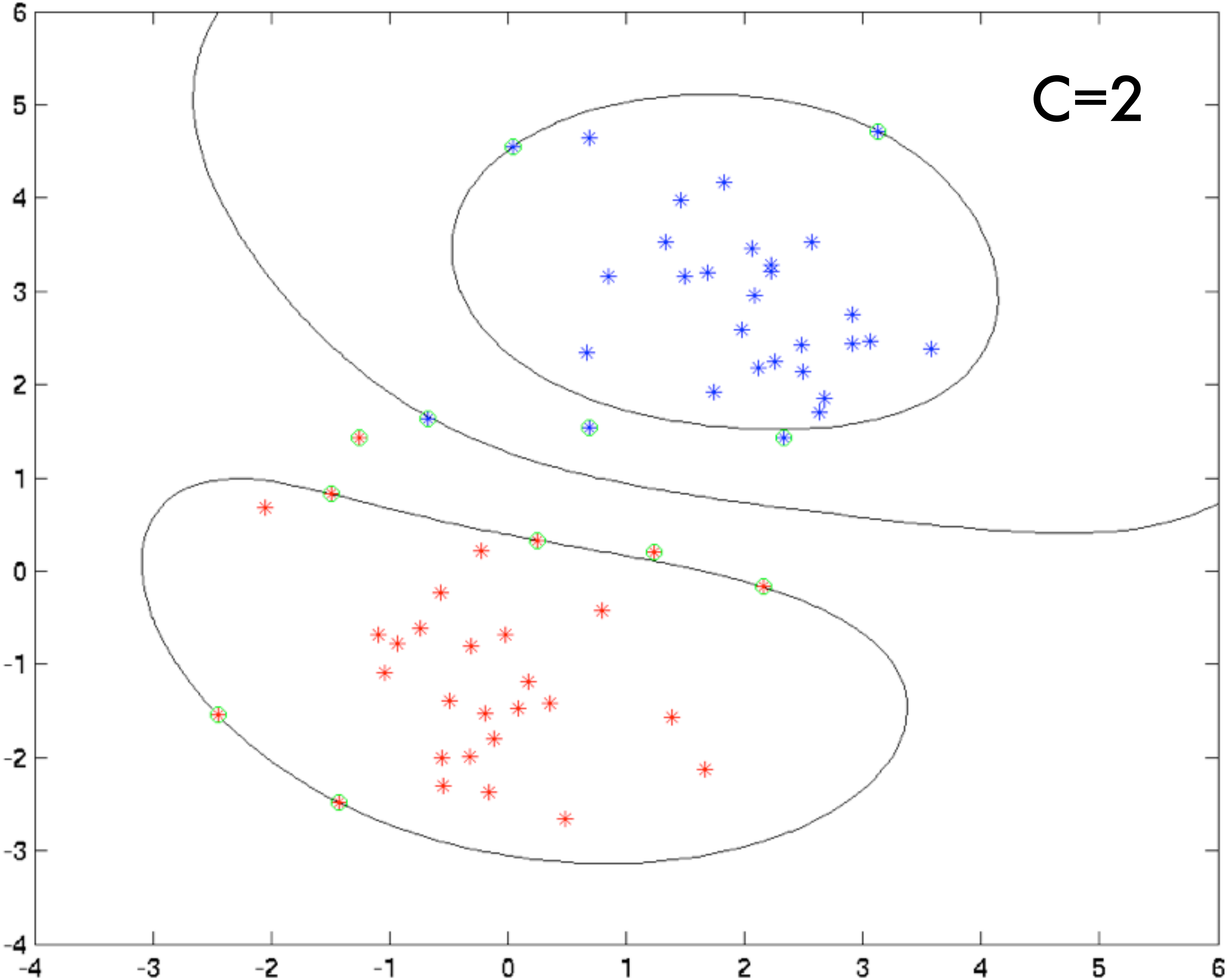


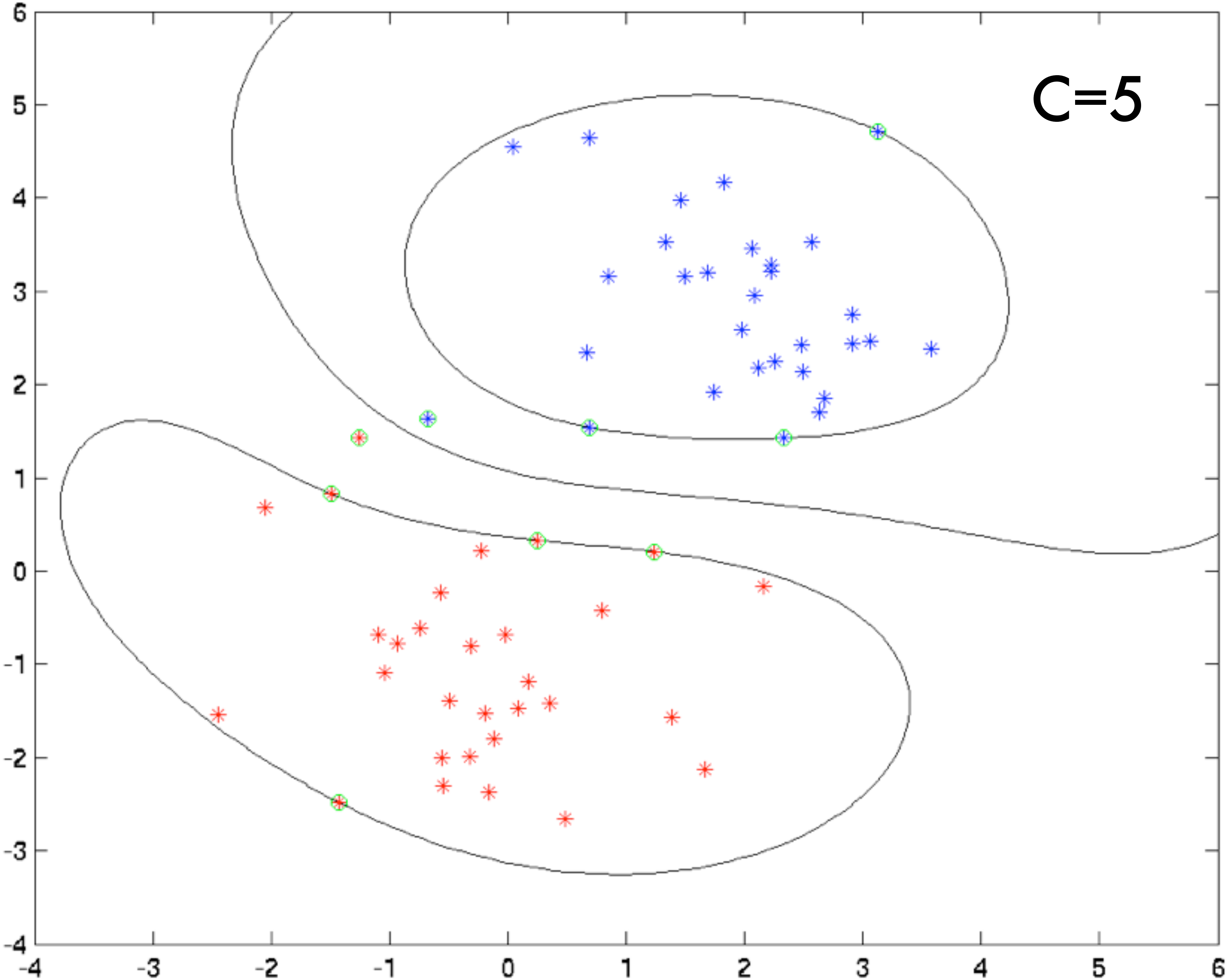
C=50
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

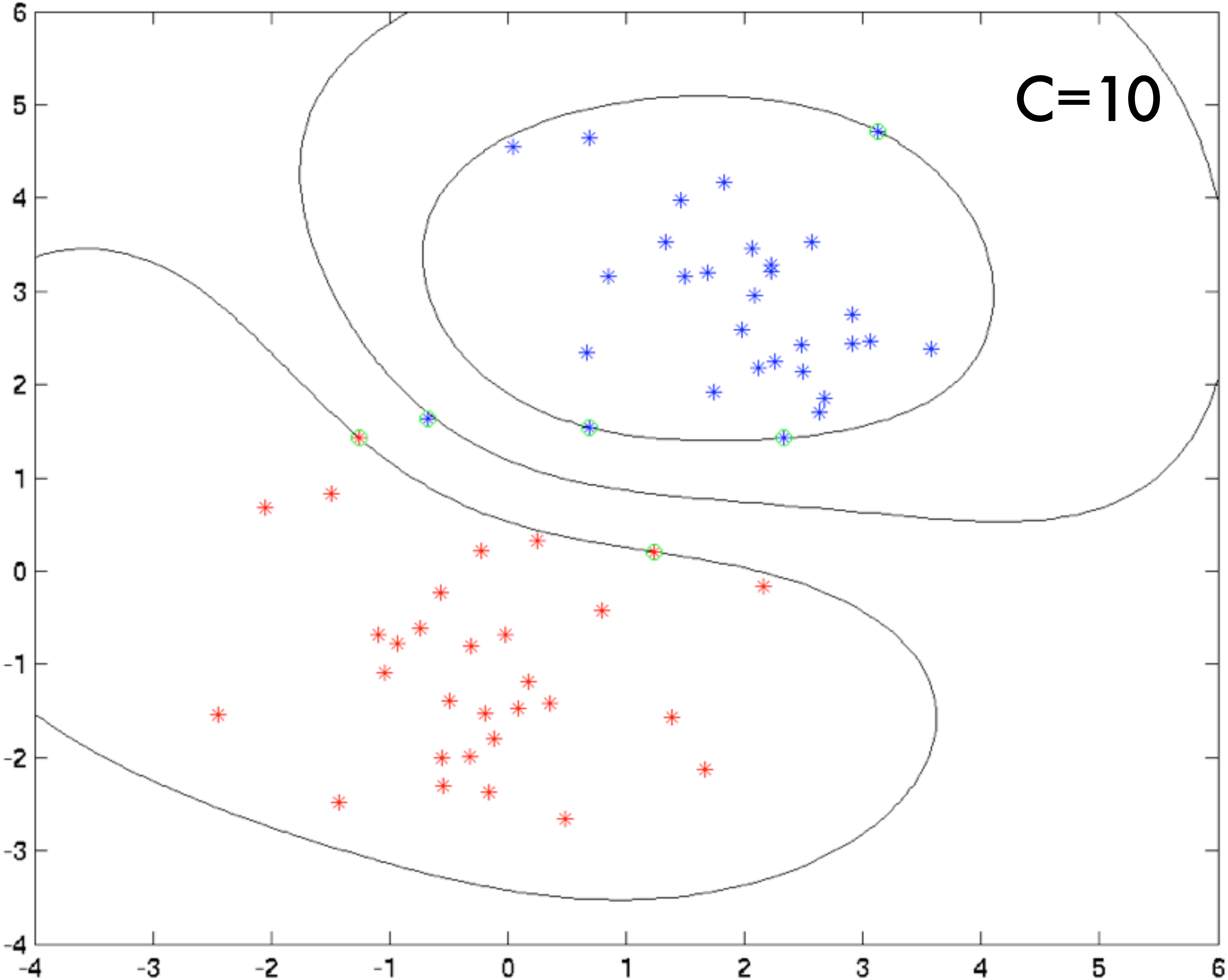

C=100
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6



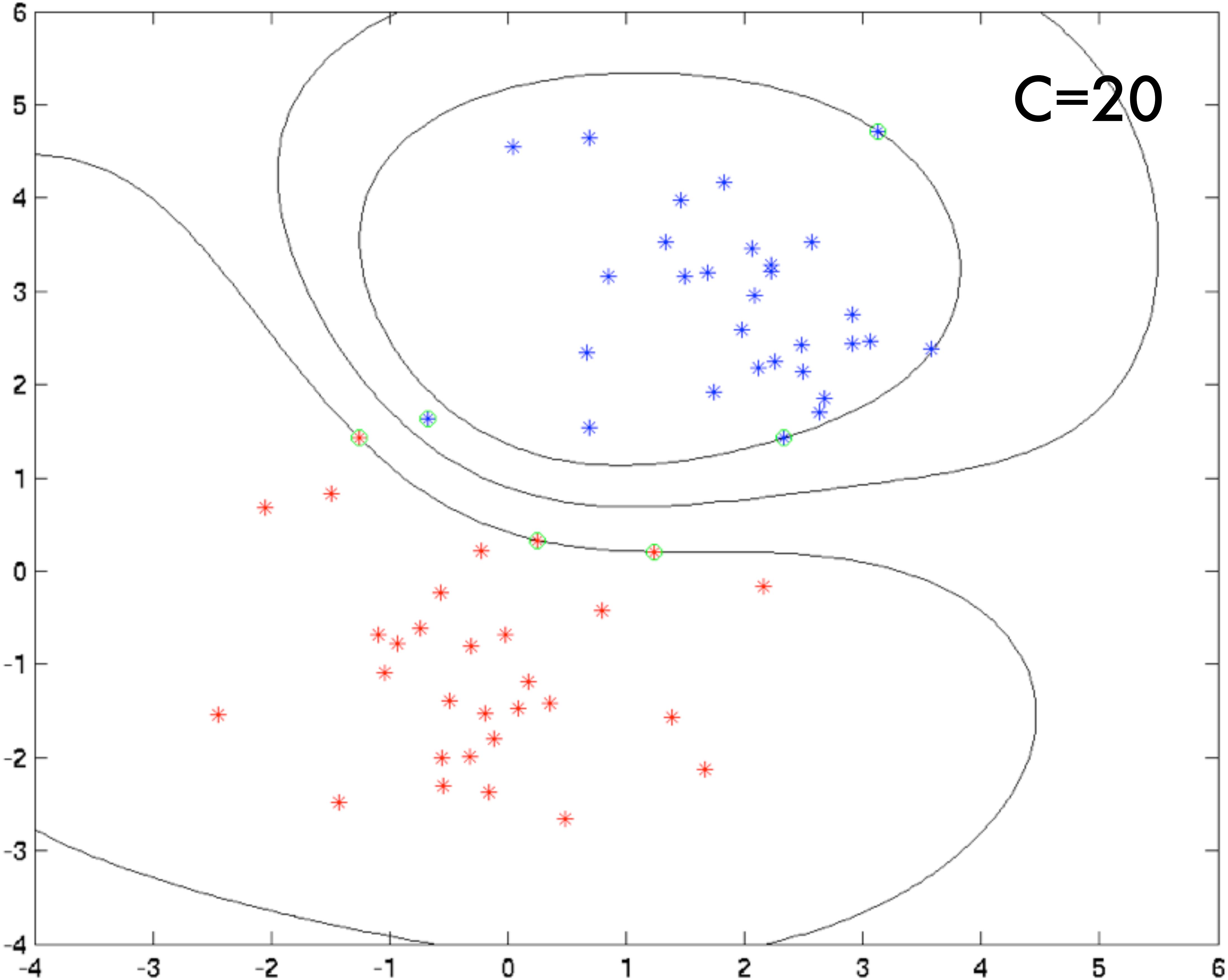
C=1
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

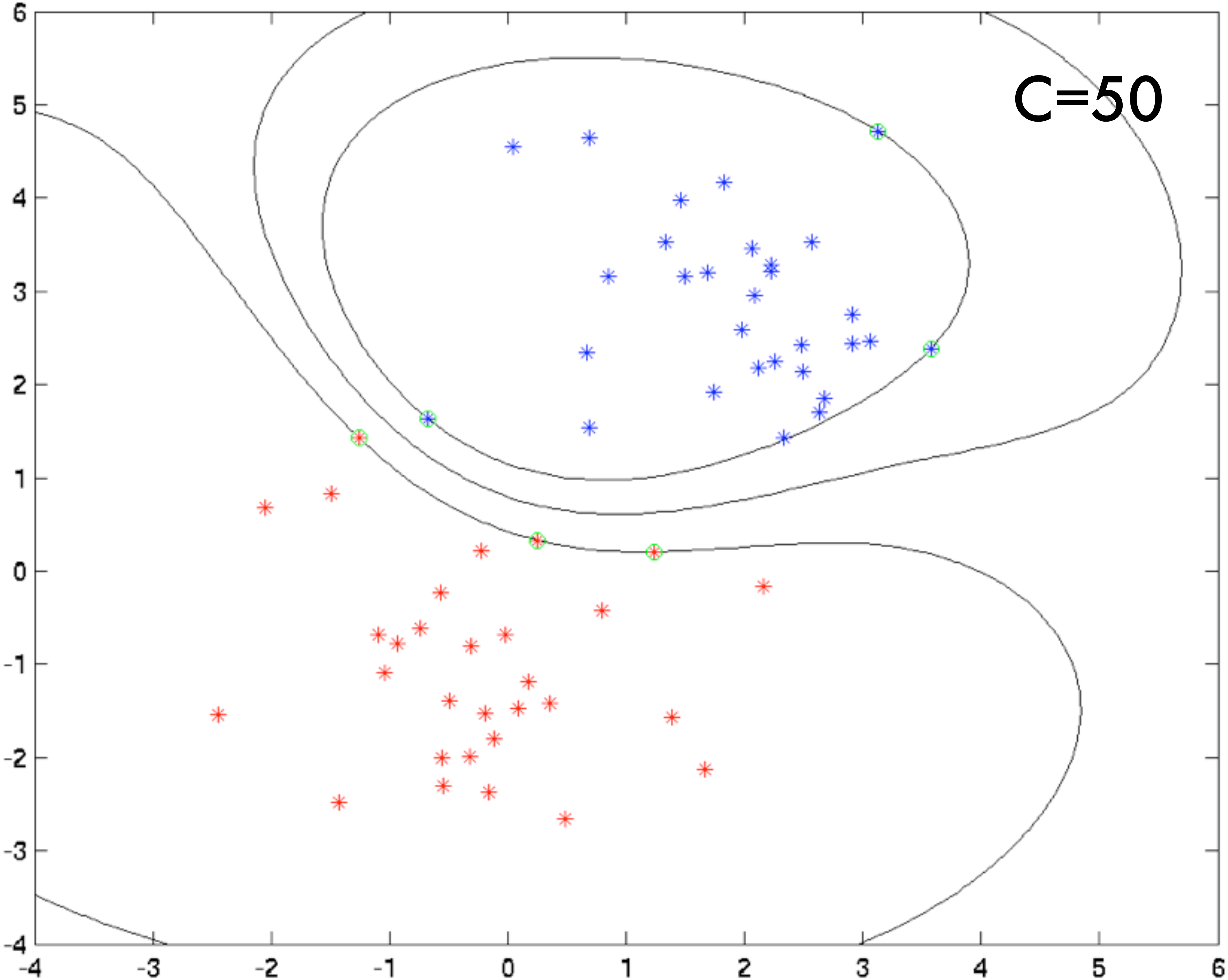

C=2
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


C=5
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

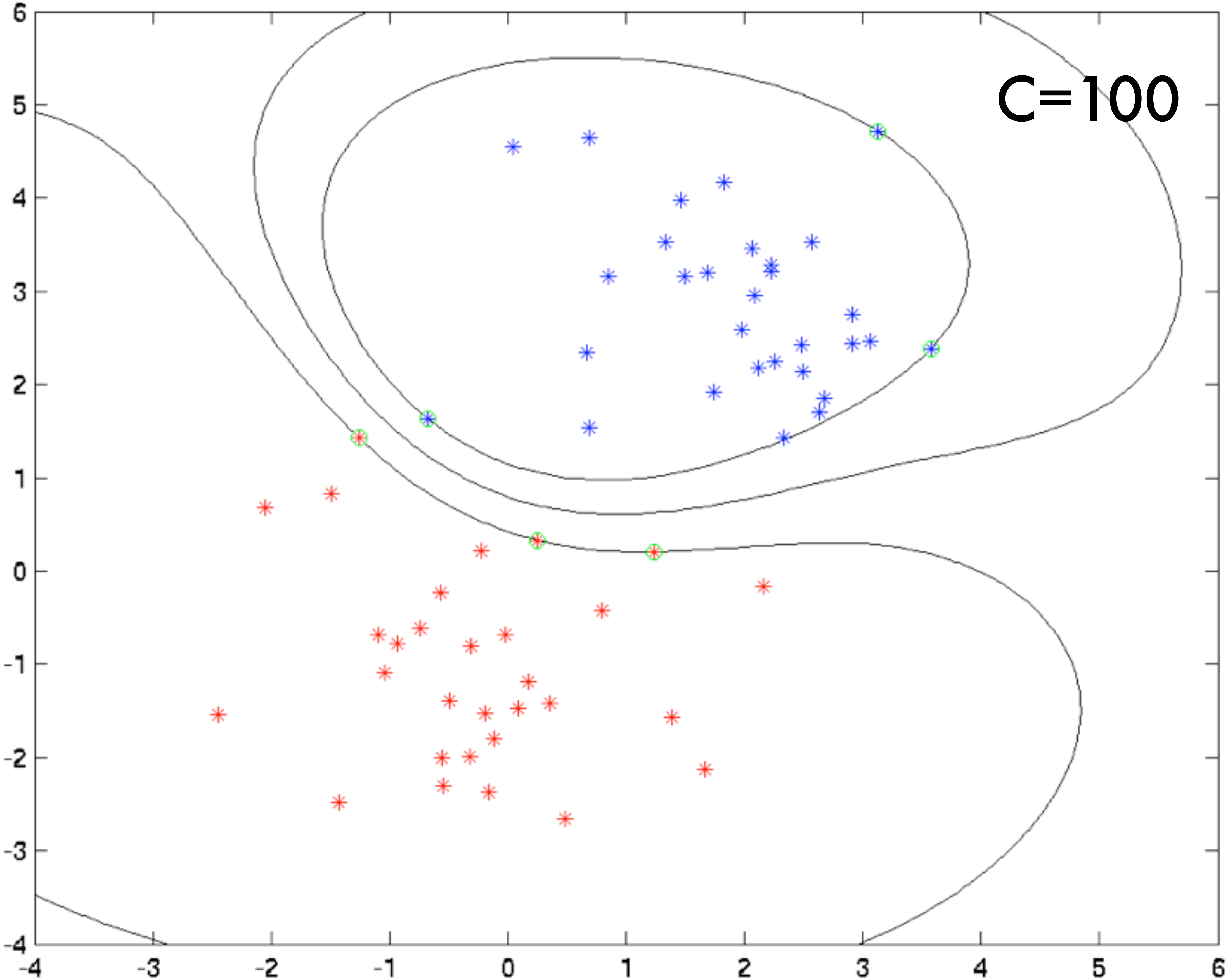


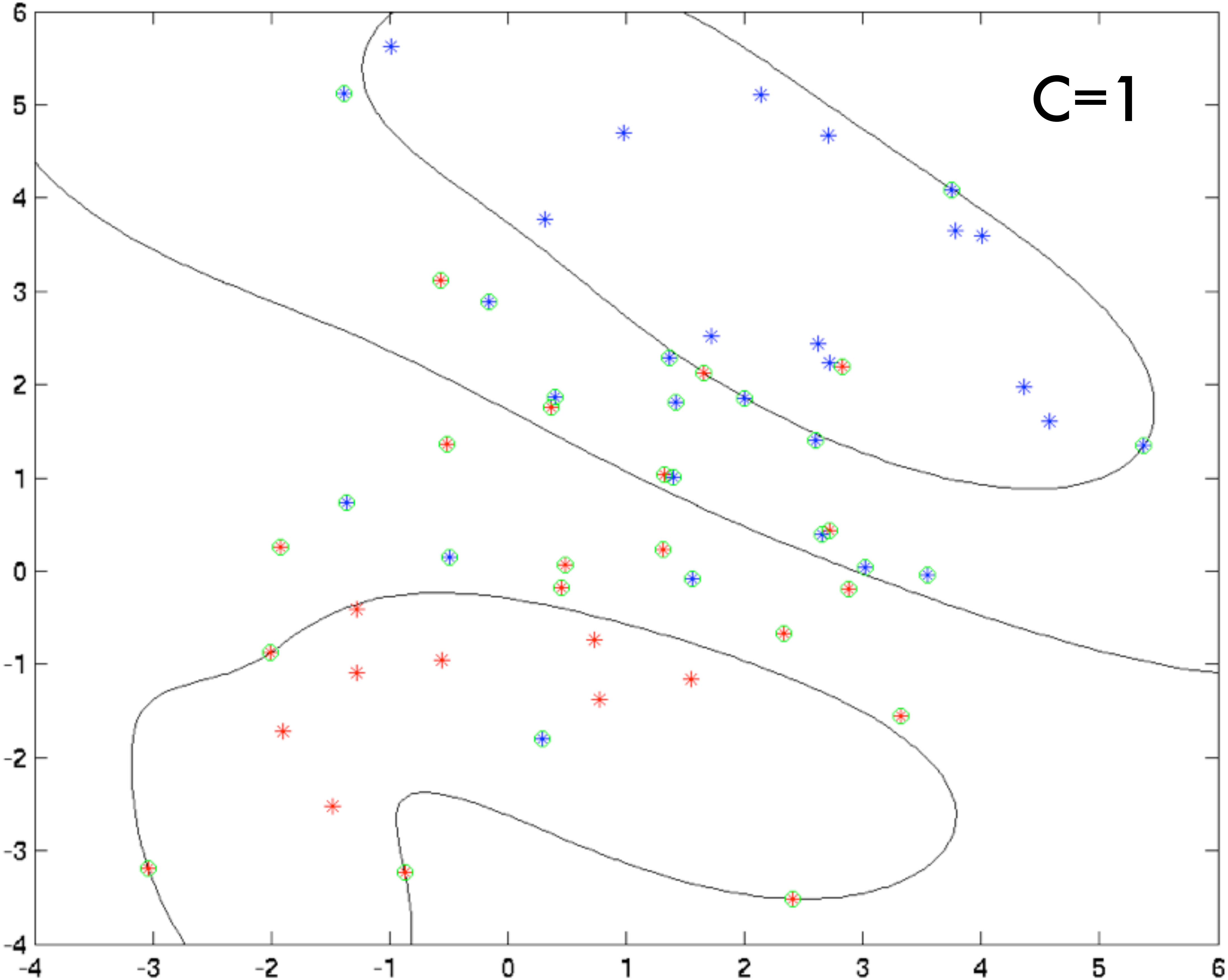
C=10
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


C=20
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

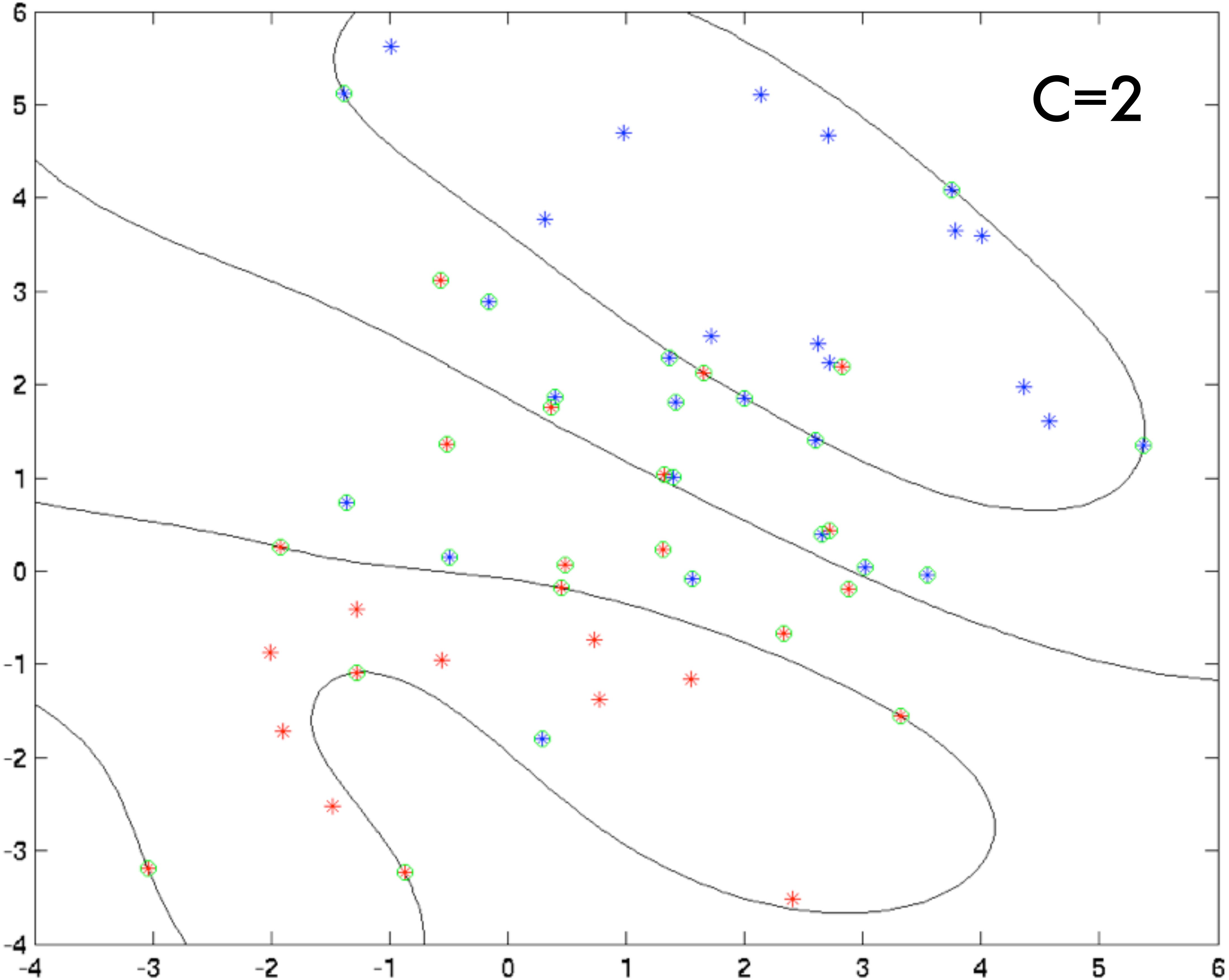


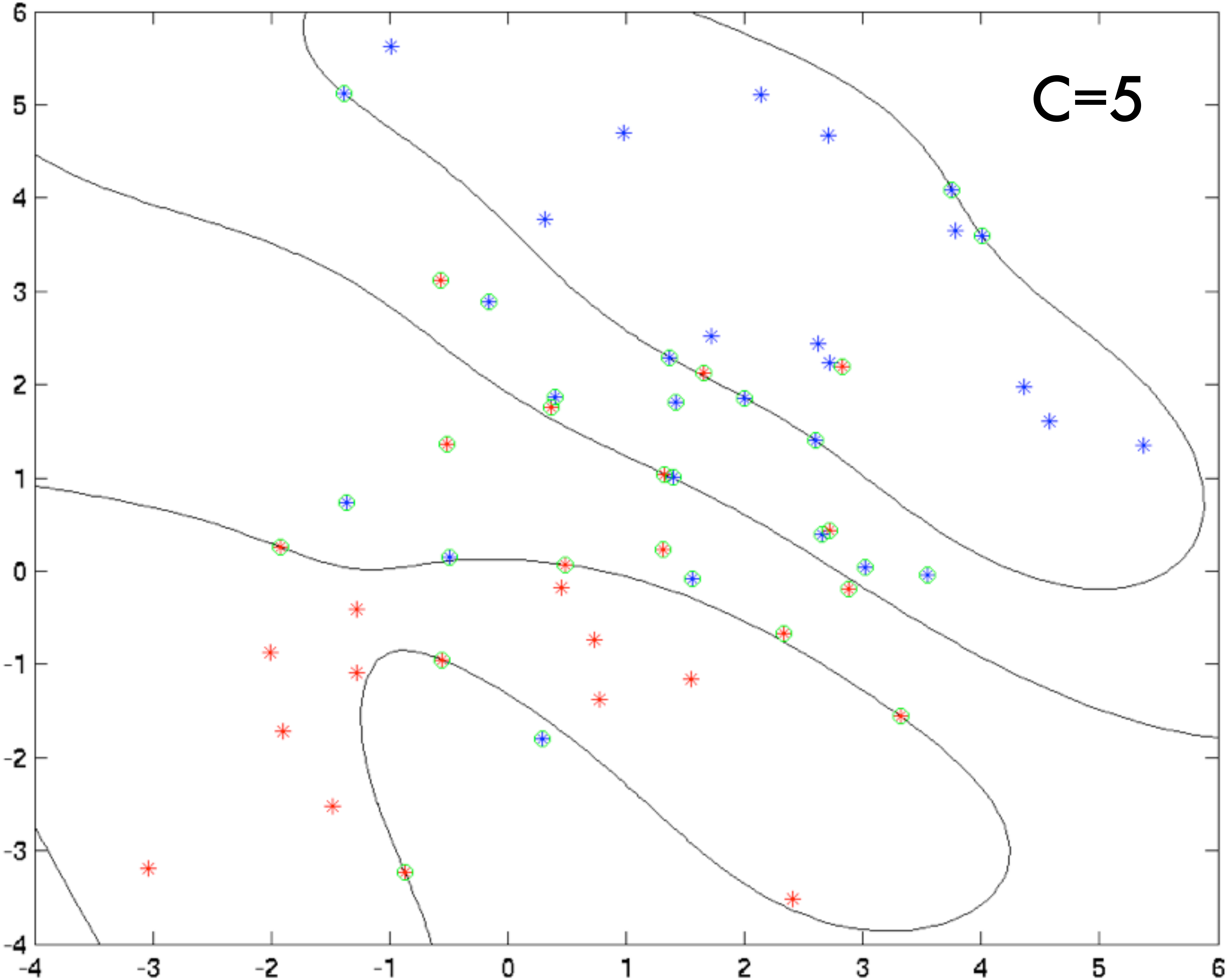
C=50


C=100
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

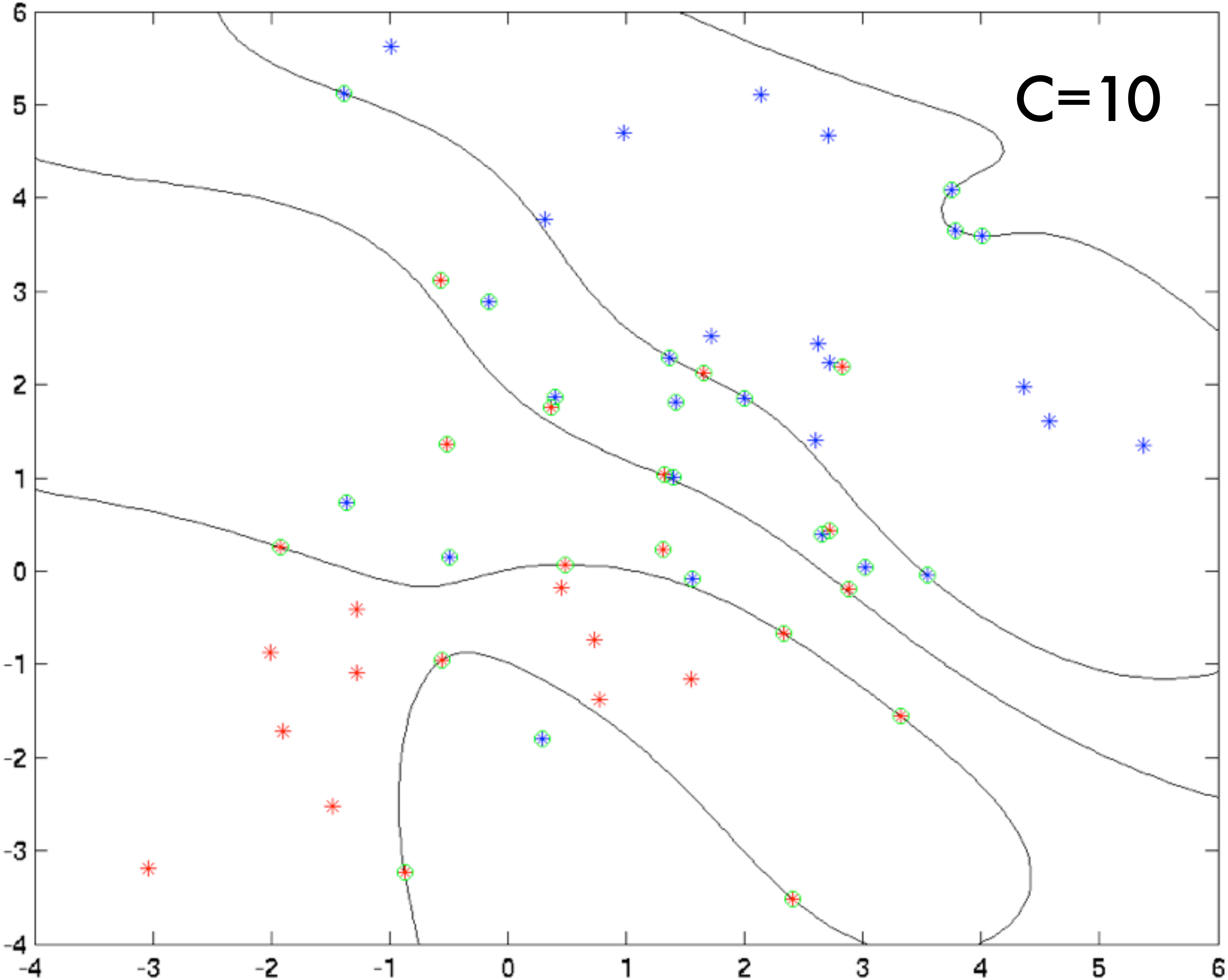


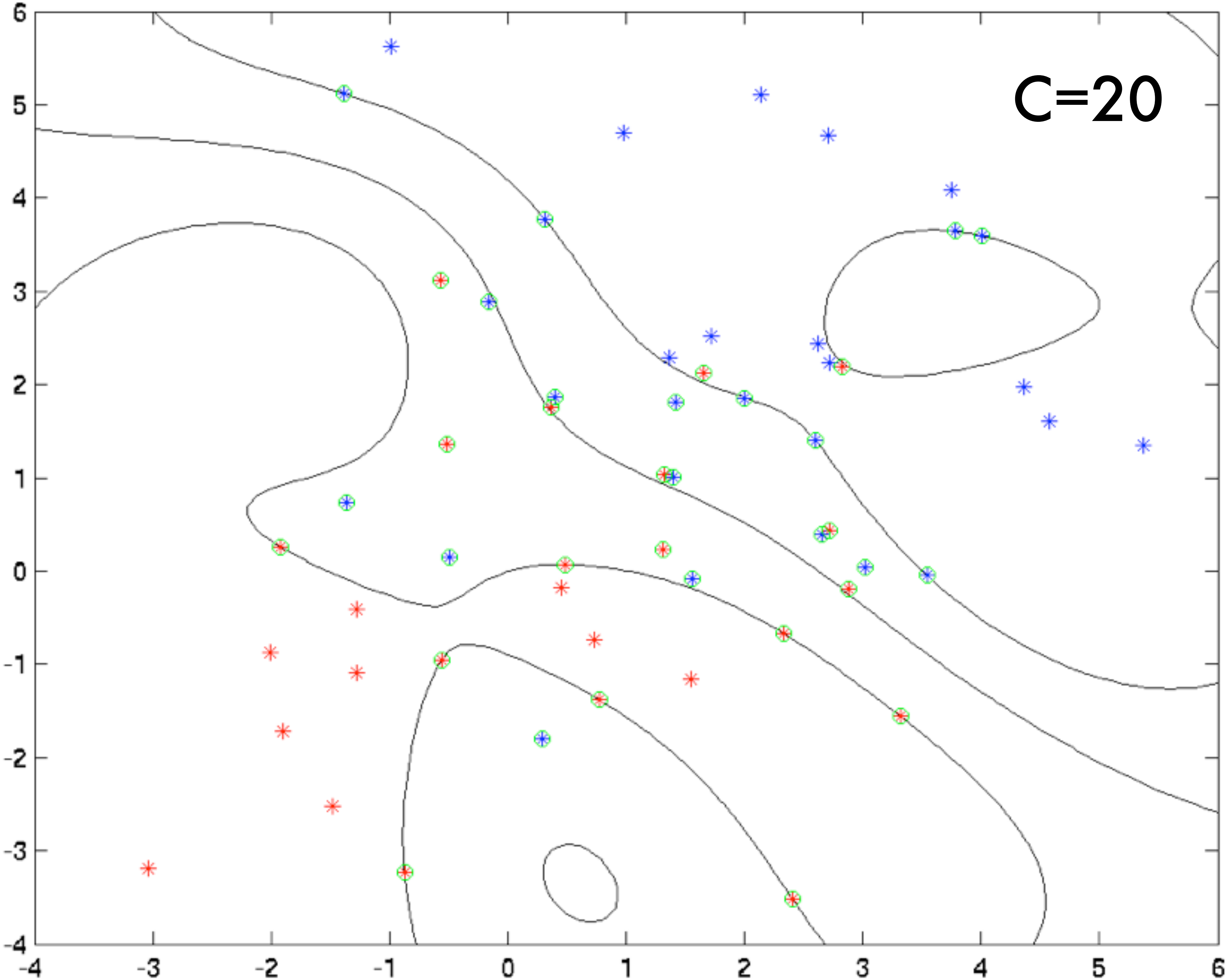
C=1
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

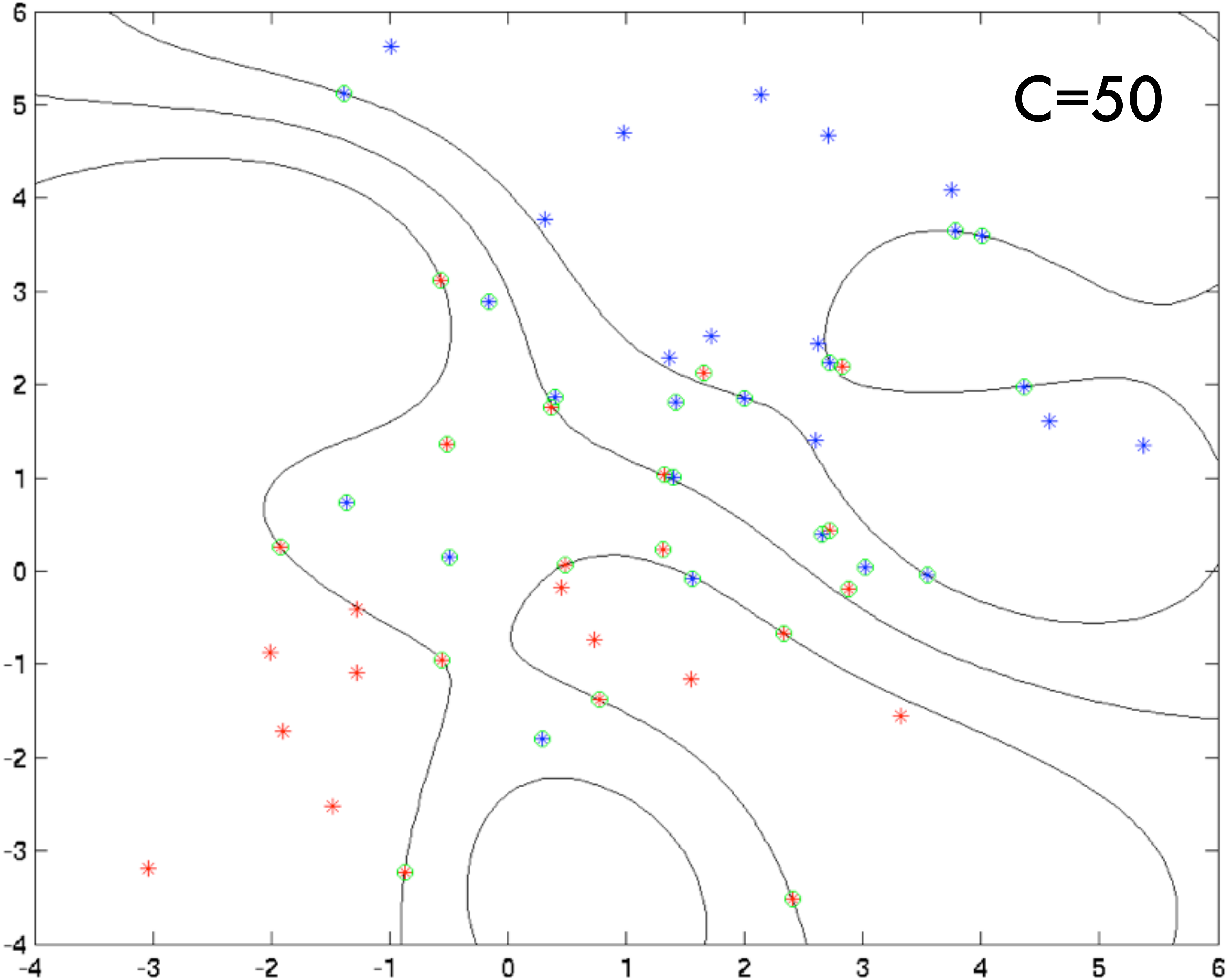

C=2
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

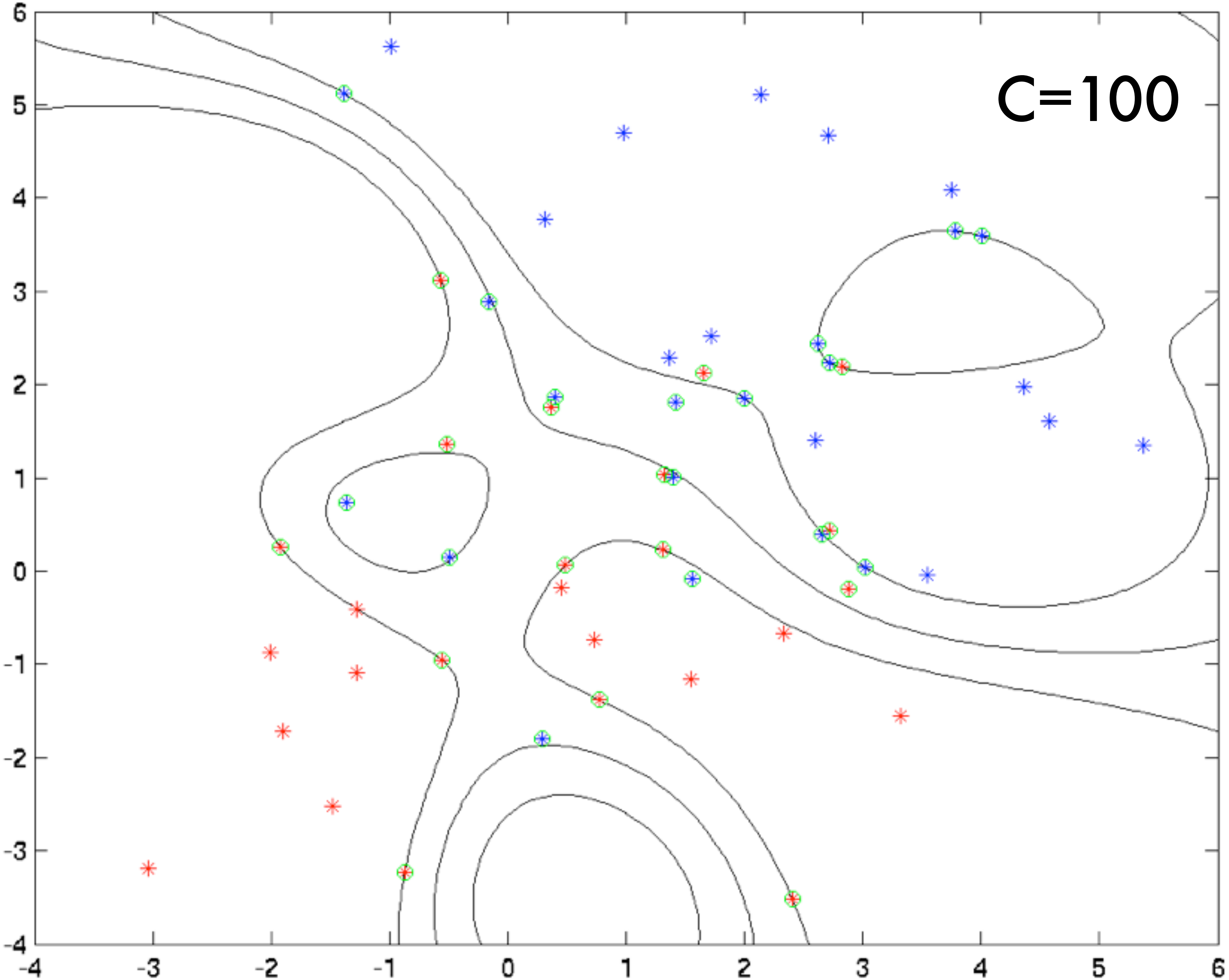


C=5

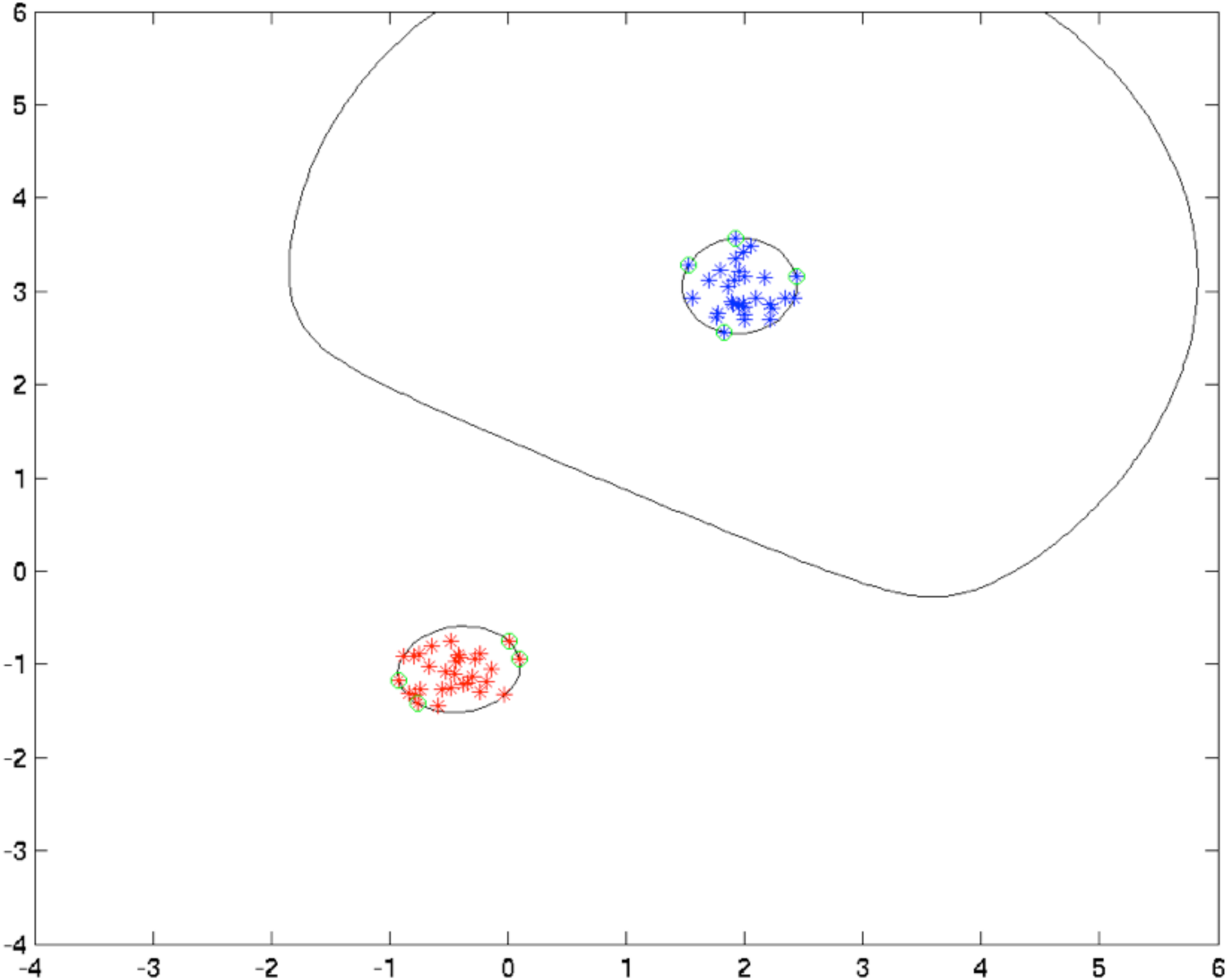

C=10
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


C=20
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

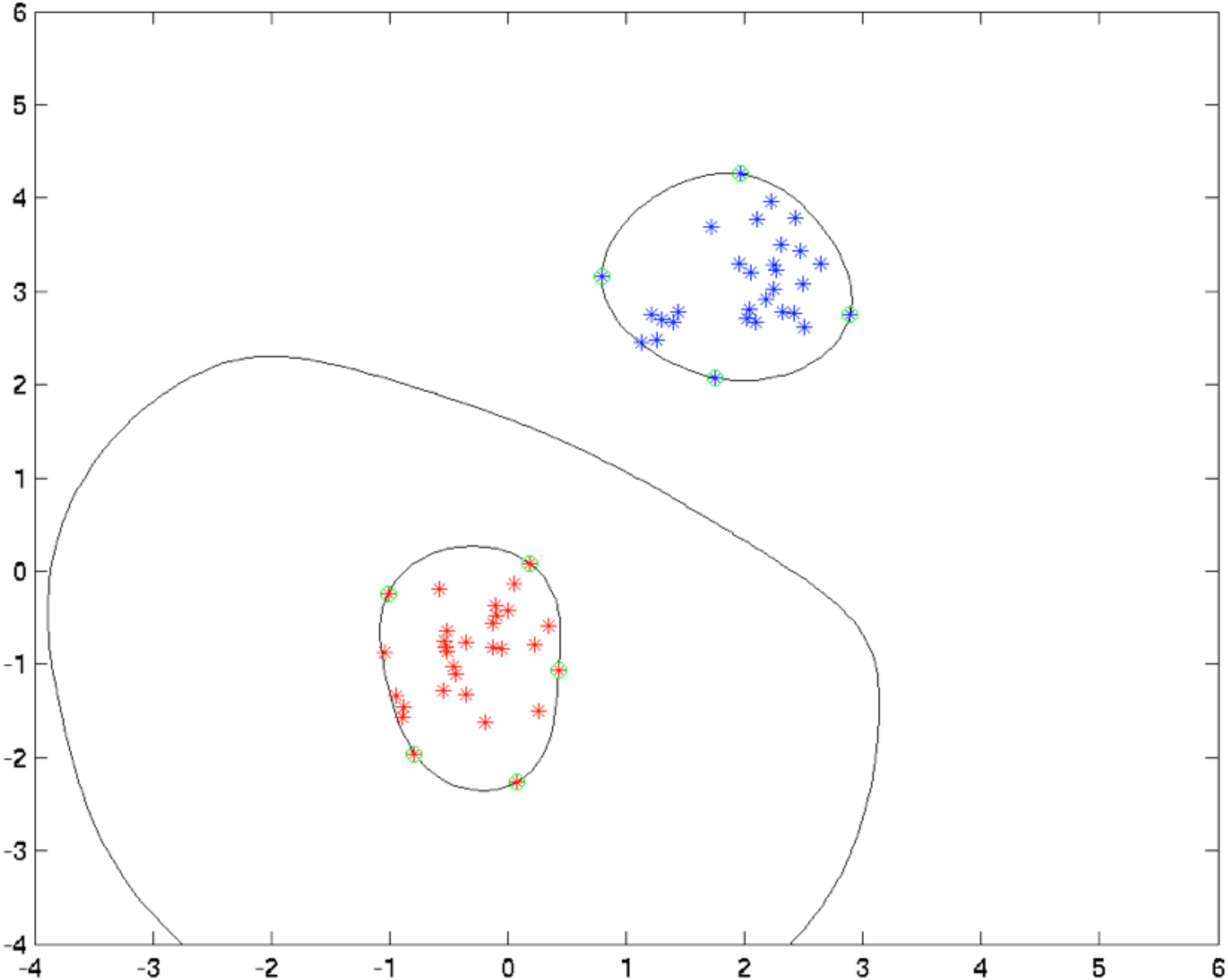

C=50
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

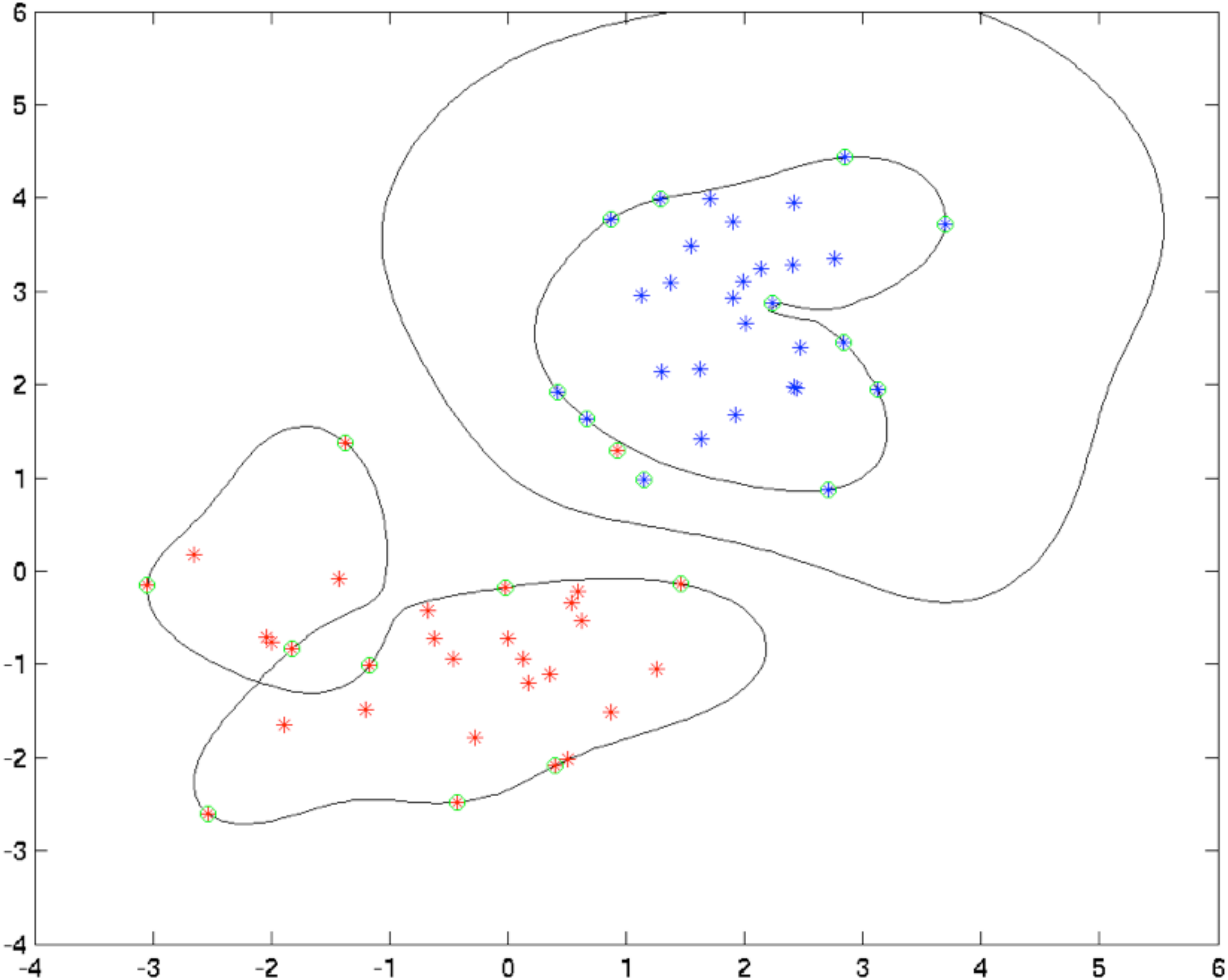

C=100
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

And now with a narrower kernel

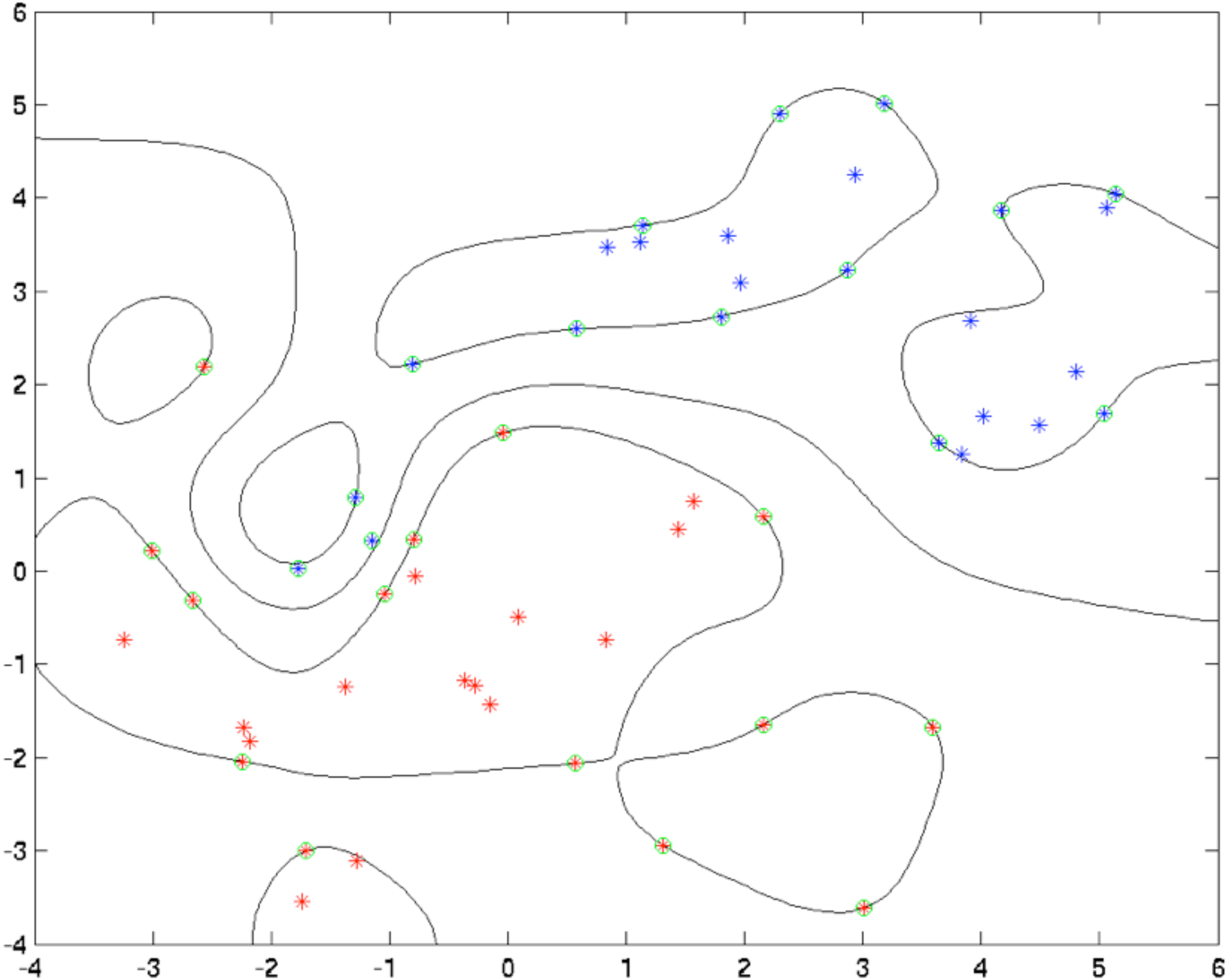


6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6


6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

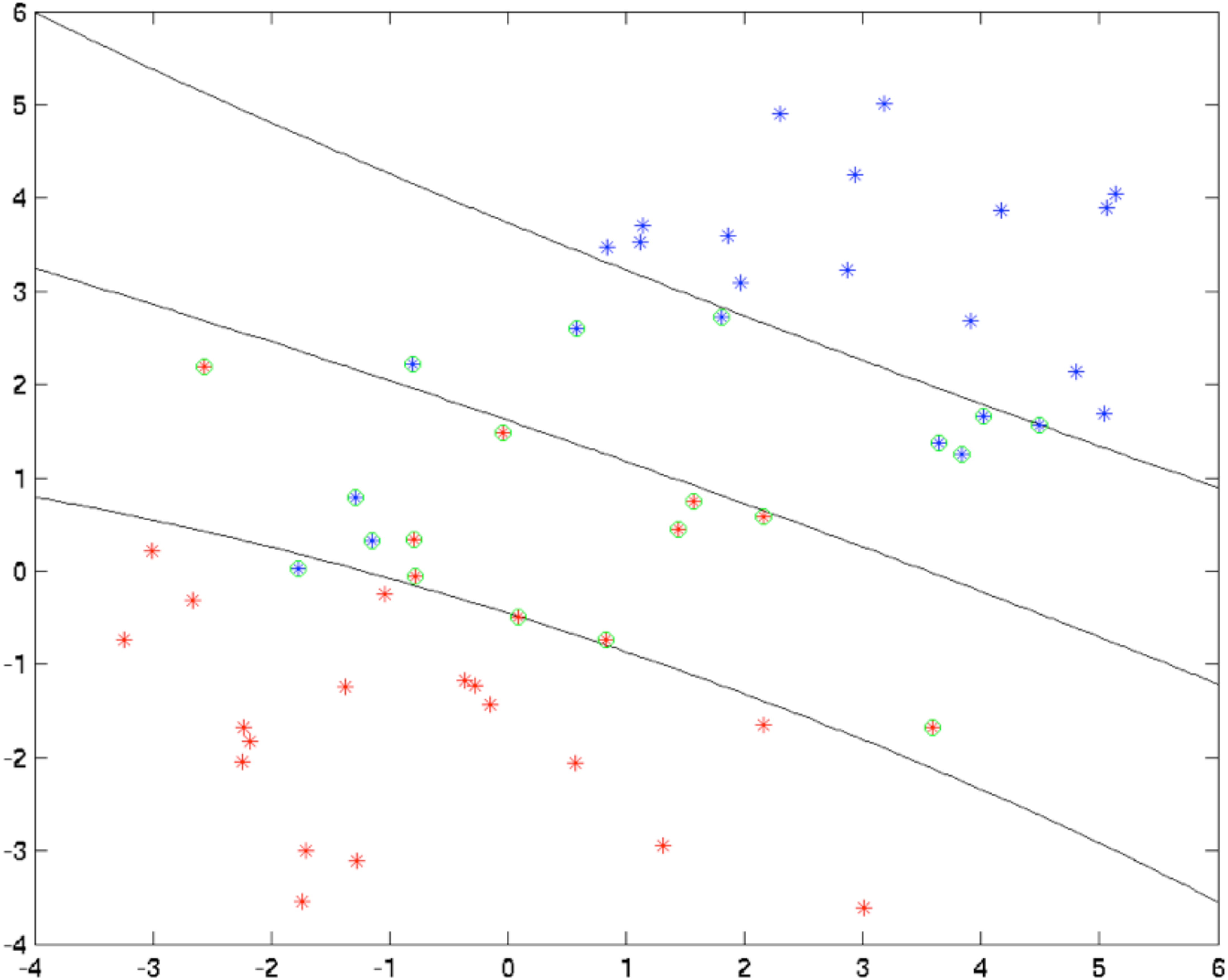


6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6



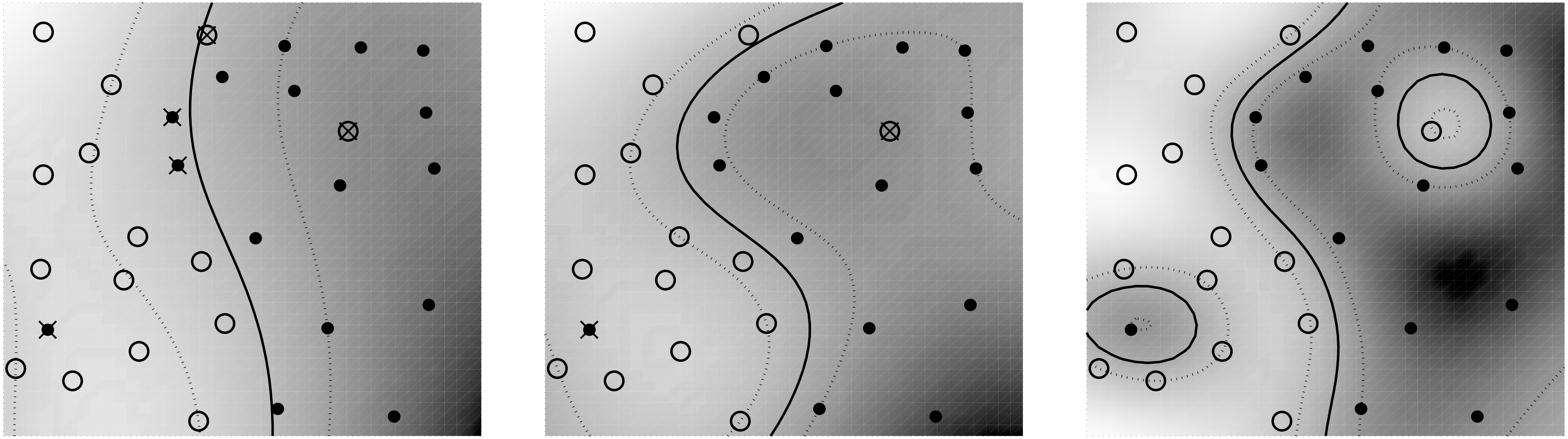
6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

And now with a very wide kernel



6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
5
6

Nonlinear separation



- Increasing C allows for more nonlinearities
- Decreases number of errors
- SV boundary need not be contiguous
- Kernel width adjusts function class

"Under hypnosis you revealed that in your last eight lives you were ... er ... a cat."

Regression Estimation

- Find function f minimizing regression error

$$R[f] := \mathbf{E}_{x,y \sim p(x,y)} \left[l(y, f(x)) \right]$$

- Compute empirical average

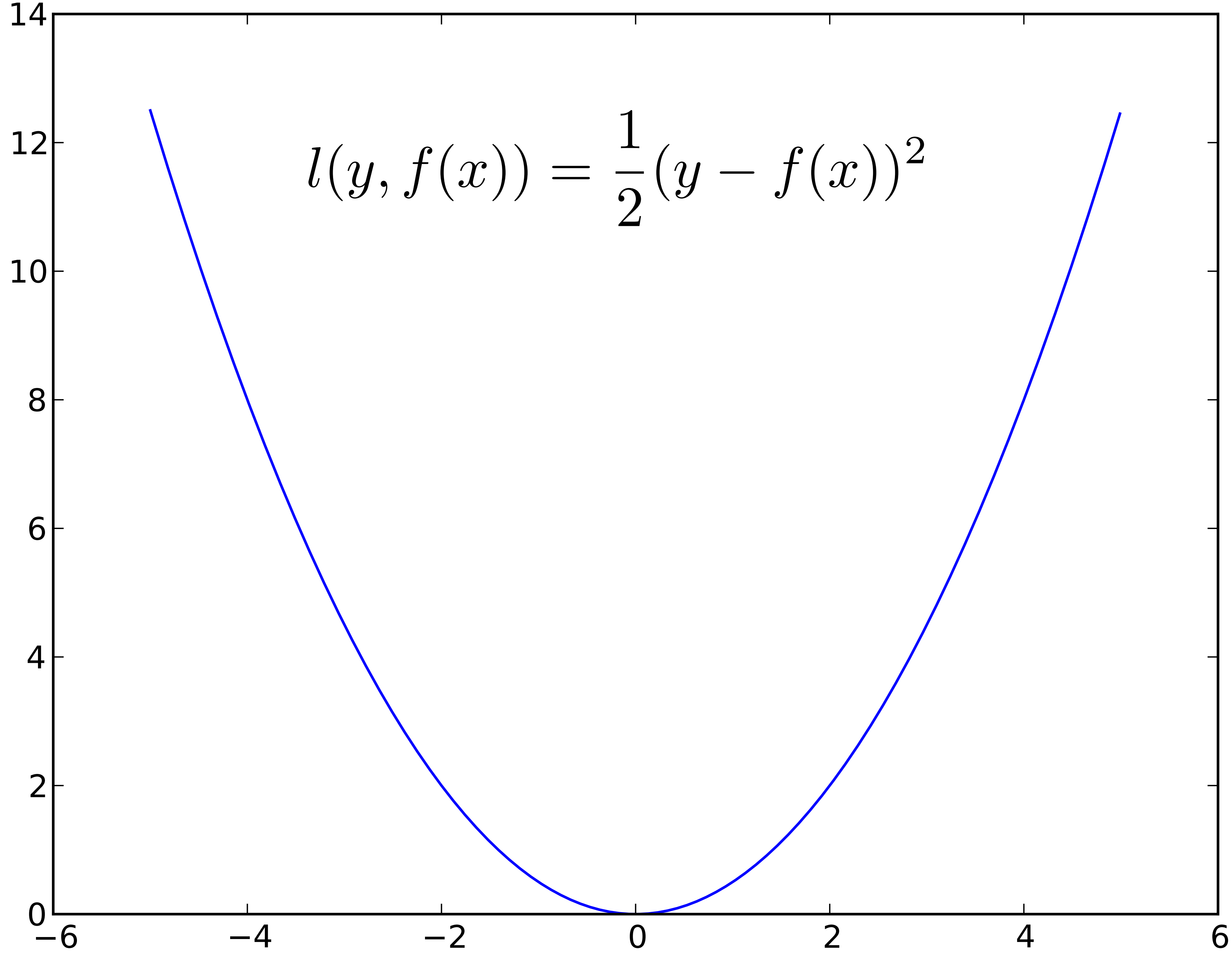
$$R_{\text{emp}}[f] := \frac{1}{m} \sum_{i=1}^{m} l(y_i, f(x_i))$$

  Overfitting as we minimize empirical error

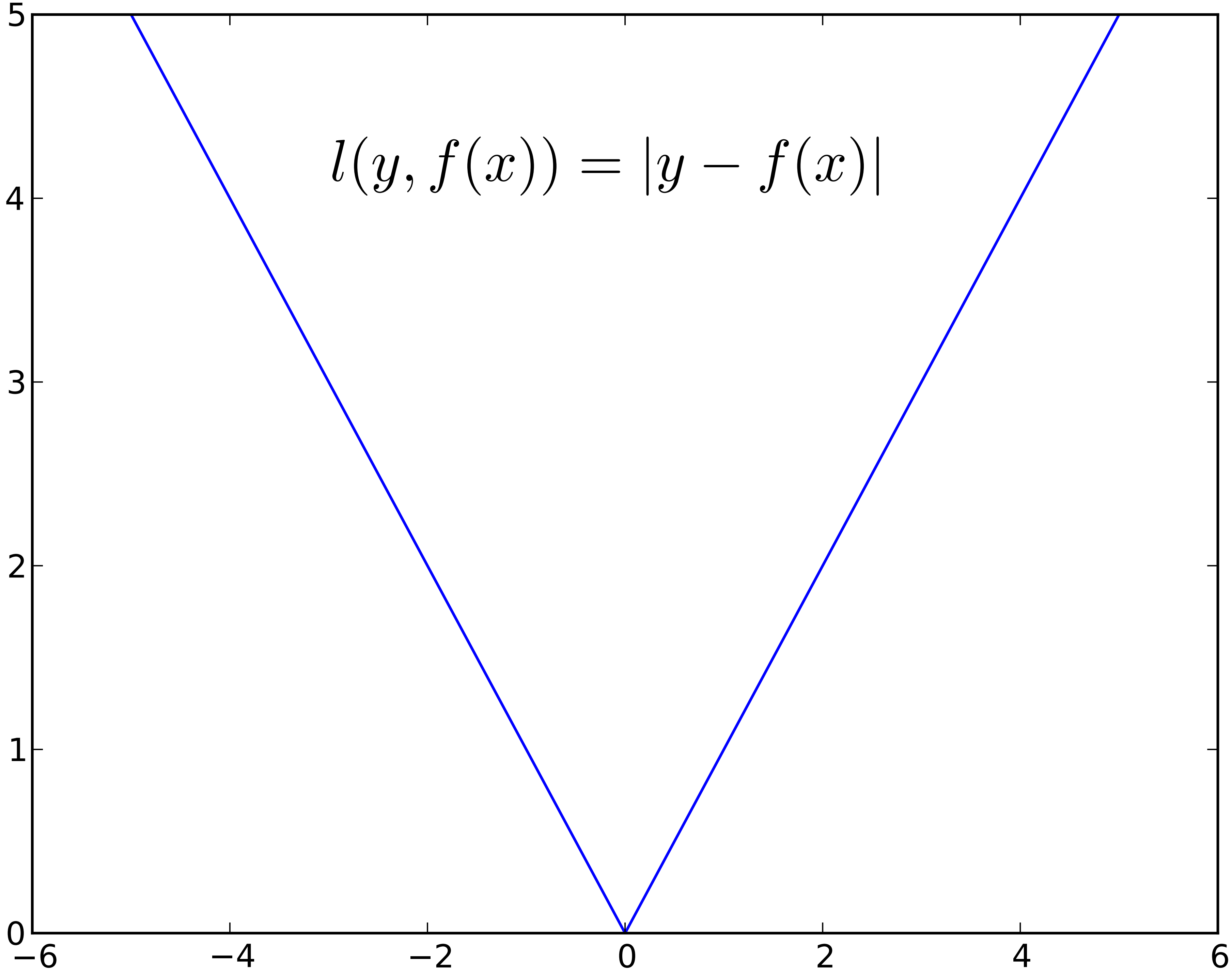
- Add regularization for capacity control

$$R_{\text{reg}}[f] := \frac{1}{m} \sum_{i=1}^{m} l(y_i, f(x_i)) + \lambda \Omega[f]$$

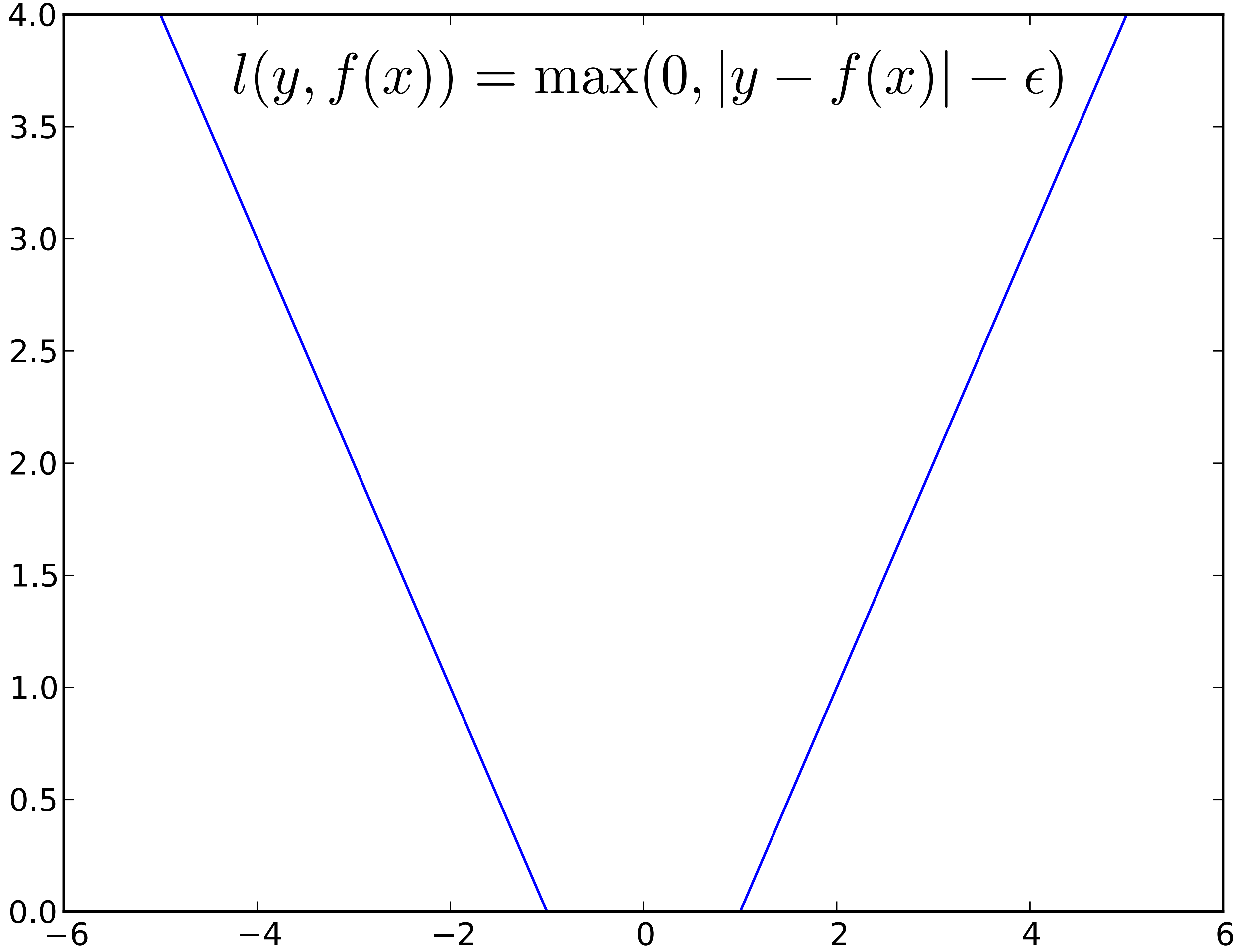
Squared loss



l1 loss



ε-insensitive Loss



Penalized least mean squares

- Optimization problem

$$\underset{w}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} (y_i - \langle x_i, w \rangle)^2 + \frac{\lambda}{2} \|w\|^2$$

- Solution

$$\begin{aligned} \partial_w [\ldots] &= \frac{1}{m} \sum_{i=1}^{m} \left[x_i x_i^\top w - x_i y_i \right] + \lambda w \\ &= \left[\frac{1}{m} X X^\top + \lambda \mathbf{1} \right] w - \frac{1}{m} X y = 0 \end{aligned}$$

$$\text{hence } w = \left[X X^\top + \lambda m \mathbf{1} \right]^{-1} X y$$

Outer product matrix in X

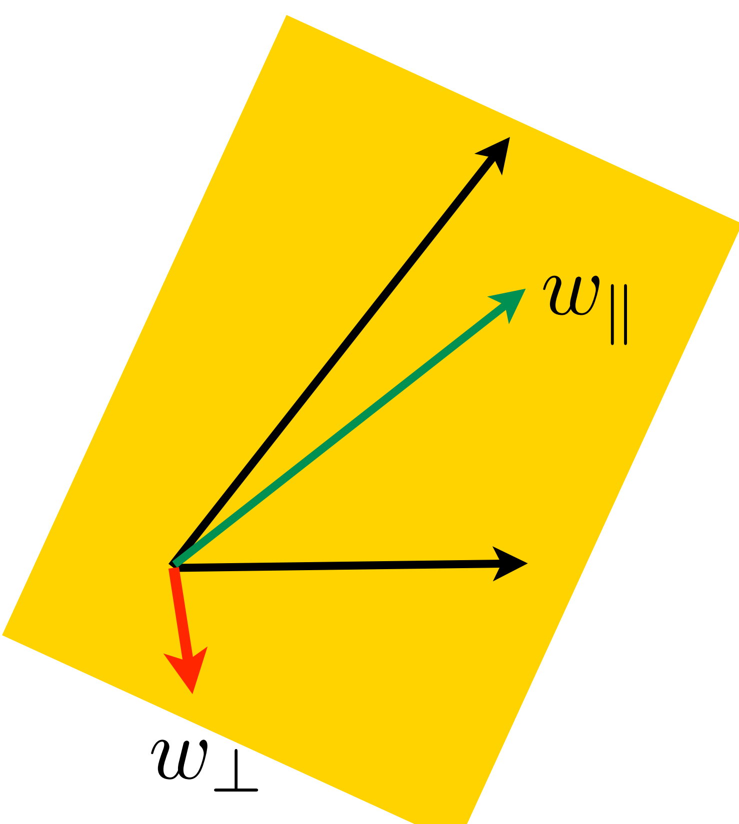
Conjugate Gradient
Sherman Morrison Woodbury

Penalized least mean squares ... now with kernels

- Optimization problem

$$\underset{w}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} (y_i - \langle \phi(x_i), w \rangle)^2 + \frac{\lambda}{2} \|w\|^2$$

- Representer Theorem (Kimeldorf & Wahba, 1971)



$$\|w\|^2 = \|w_{\|}\|^2 + \|w_{\perp}\|^2$$

empirical risk dependent

Penalized least mean squares ... now with kernels

- Optimization problem

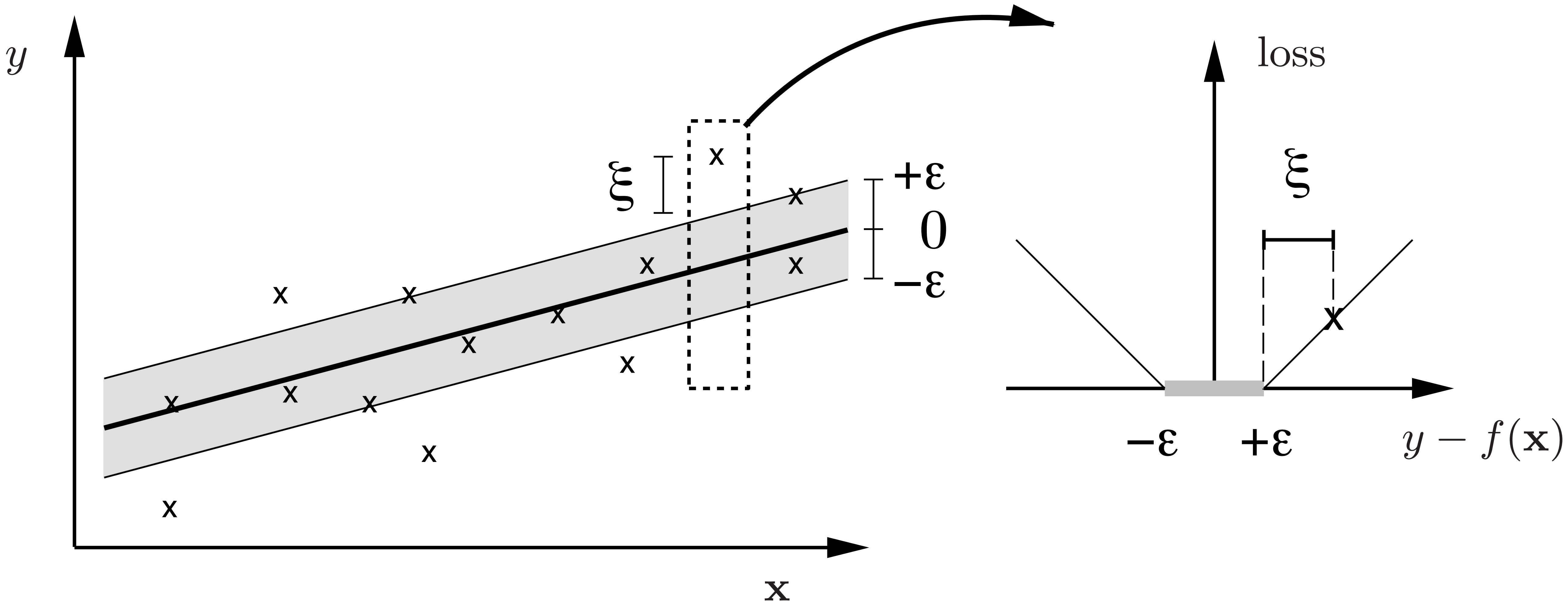
$$\underset{w}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} (y_i - \langle \phi(x_i), w \rangle)^2 + \frac{\lambda}{2} \|w\|^2$$

- Representer Theorem (Kimeldorf & Wahba, 1971)
  - Optimal solution is in span of data $w = \sum_i \alpha_i \phi(x_i)$
  - Proof - risk term only depends on data via $\phi(x_i)$
  - Regularization ensures that orthogonal part is 0
- Optimization problem in terms of w

$$\underset{\alpha}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} \left(y_i - \sum_j K_{ij} \alpha_j \right)^2 + \frac{\lambda}{2} \sum_{i,j} \alpha_i \alpha_j K_{ij}$$

solve for $\alpha = (K + m\lambda \mathbf{1})^{-1} y$ as linear system

SVM Regression



don't care about deviations within the tube

SVM Regression (ϵ-insensitive loss)

- Optimization Problem (as constrained QP)

$$\underset{w,b}{\text{minimize}} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}[\xi_i + \xi_i^*]$$

$$\text{subject to} \ \langle w, x_i\rangle + b \le y_i + \epsilon + \xi_i \ \text{ and } \xi_i \ge 0$$

$$\langle w, x_i\rangle + b \ge y_i - \epsilon - \xi_i^* \ \text{ and } \xi_i^* \ge 0$$

- Lagrange Function

$$L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}[\xi_i + \xi_i^*] - \sum_{i=1}^{m}[\eta_i\xi_i + \eta_i^*\xi_i^*] +$$

$$\sum_{i=1}^{m}\alpha_i\left[\langle w, x_i\rangle + b - y_i - \epsilon - \xi_i\right] + \sum_{i=1}^{m}\alpha_i^*\left[y_i - \epsilon - \xi_i^* - \langle w, x_i\rangle - b\right]$$

SVM Regression (є-insensitive loss)

- First order conditions

$$\partial_w L = 0 = w + \sum_i [\alpha_i - \alpha_i^*] x_i$$

$$\partial_b L = 0 = \sum_i [\alpha_i - \alpha_i^*]$$

$$\partial_{\xi_i} L = 0 = C - \eta_i - \alpha_i$$

$$\partial_{\xi_i^*} L = 0 = C - \eta_i^* - \alpha_i^*$$

- Dual problem

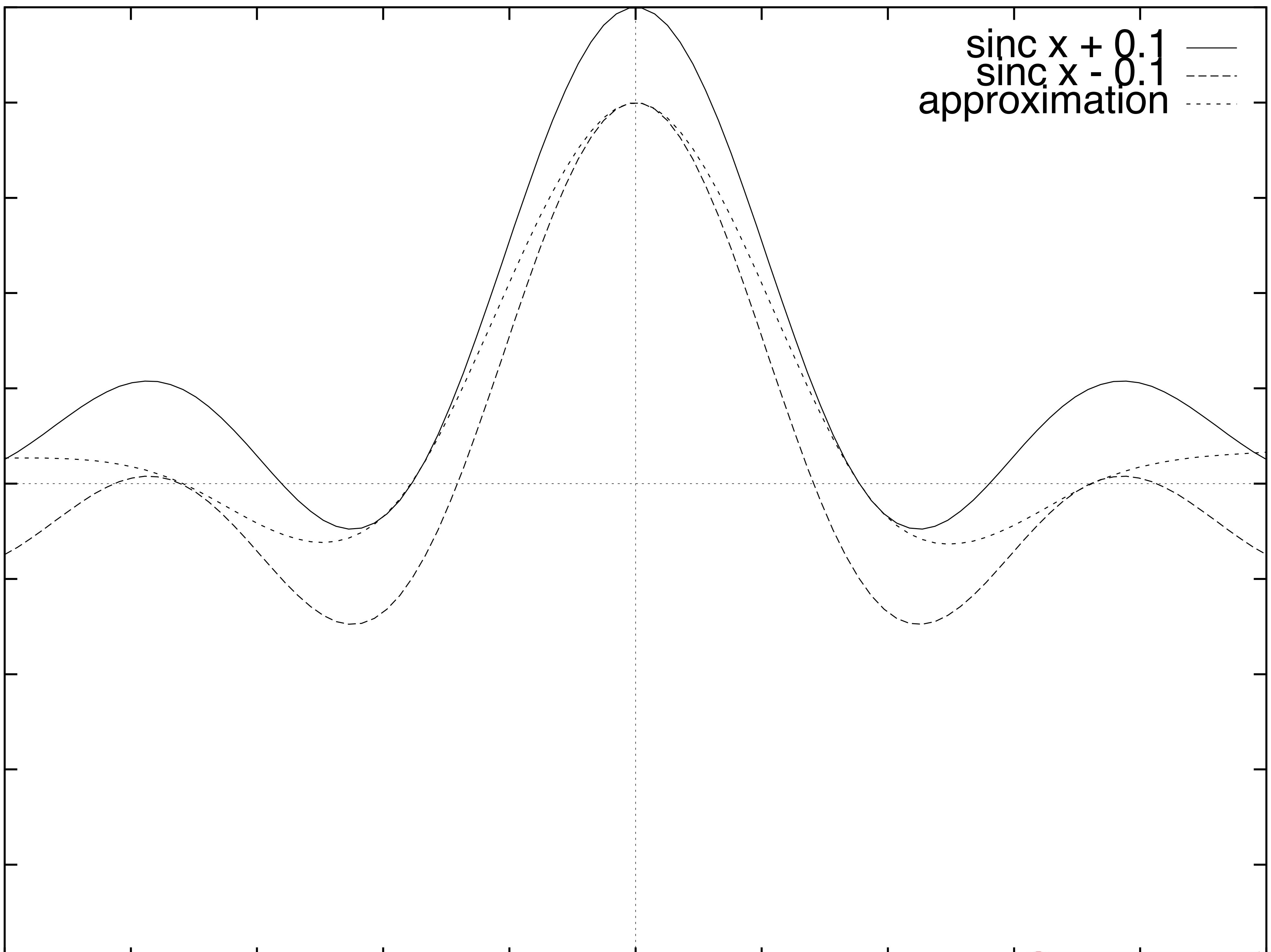
$$\underset{\alpha,\alpha^*}{\text{minimize}} \ \frac{1}{2}(\alpha - \alpha^*)^\top K(\alpha - \alpha^*) + \epsilon 1^\top(\alpha + \alpha^*) + y^\top(\alpha - \alpha^*)$$

$$\text{subject to } 1^\top(\alpha - \alpha^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C]$$

Properties

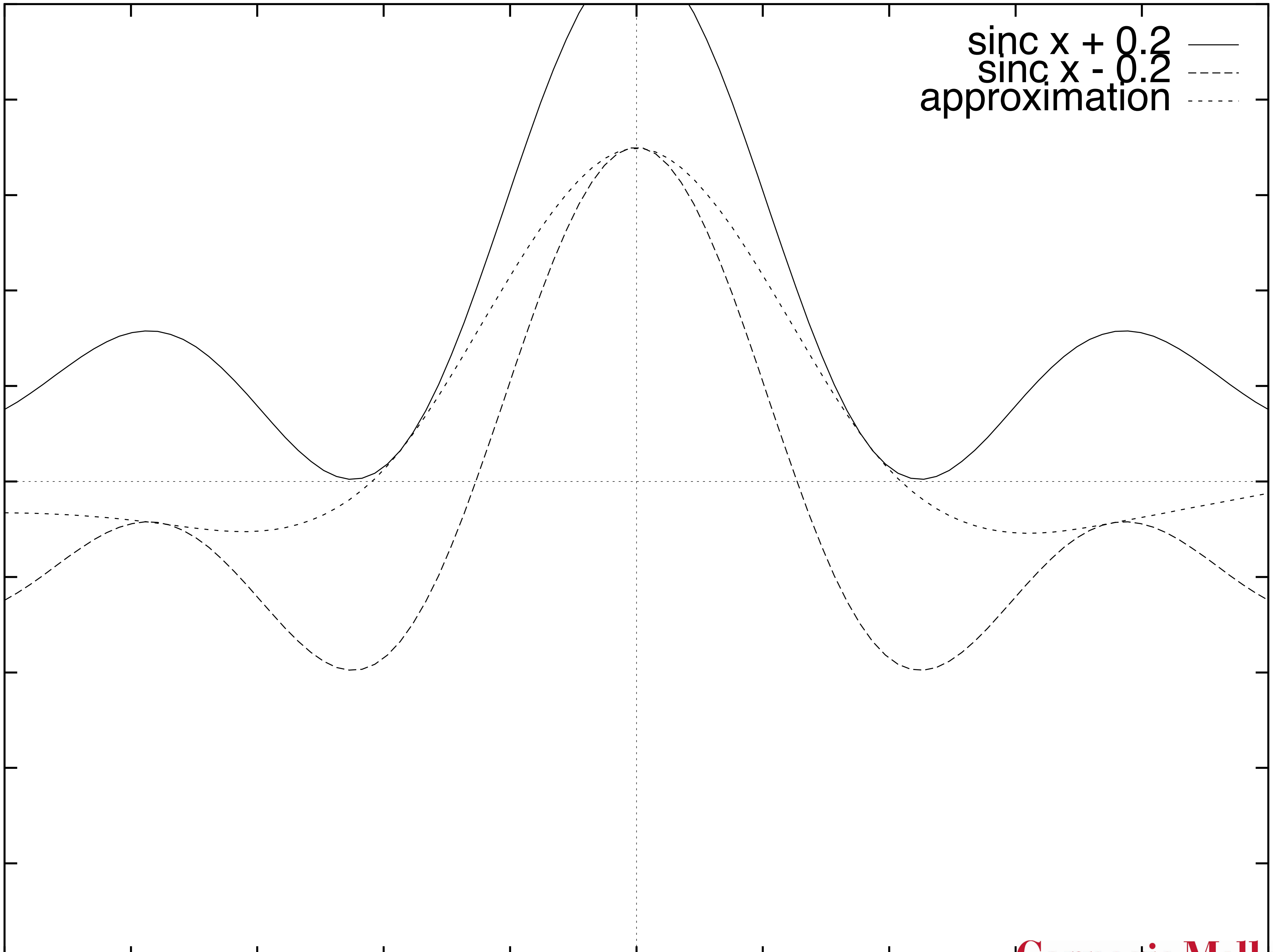
- Ignores 'typical' instances with small error
- Only upper or lower bound active at any time
- QP in 2n variables as cheap as SVM problem
- Robustness with respect to outliers
  - l_1 loss yields same problem without epsilon
  - Huber's robust loss yields similar problem but with added quadratic penalty on coefficients

Regression example

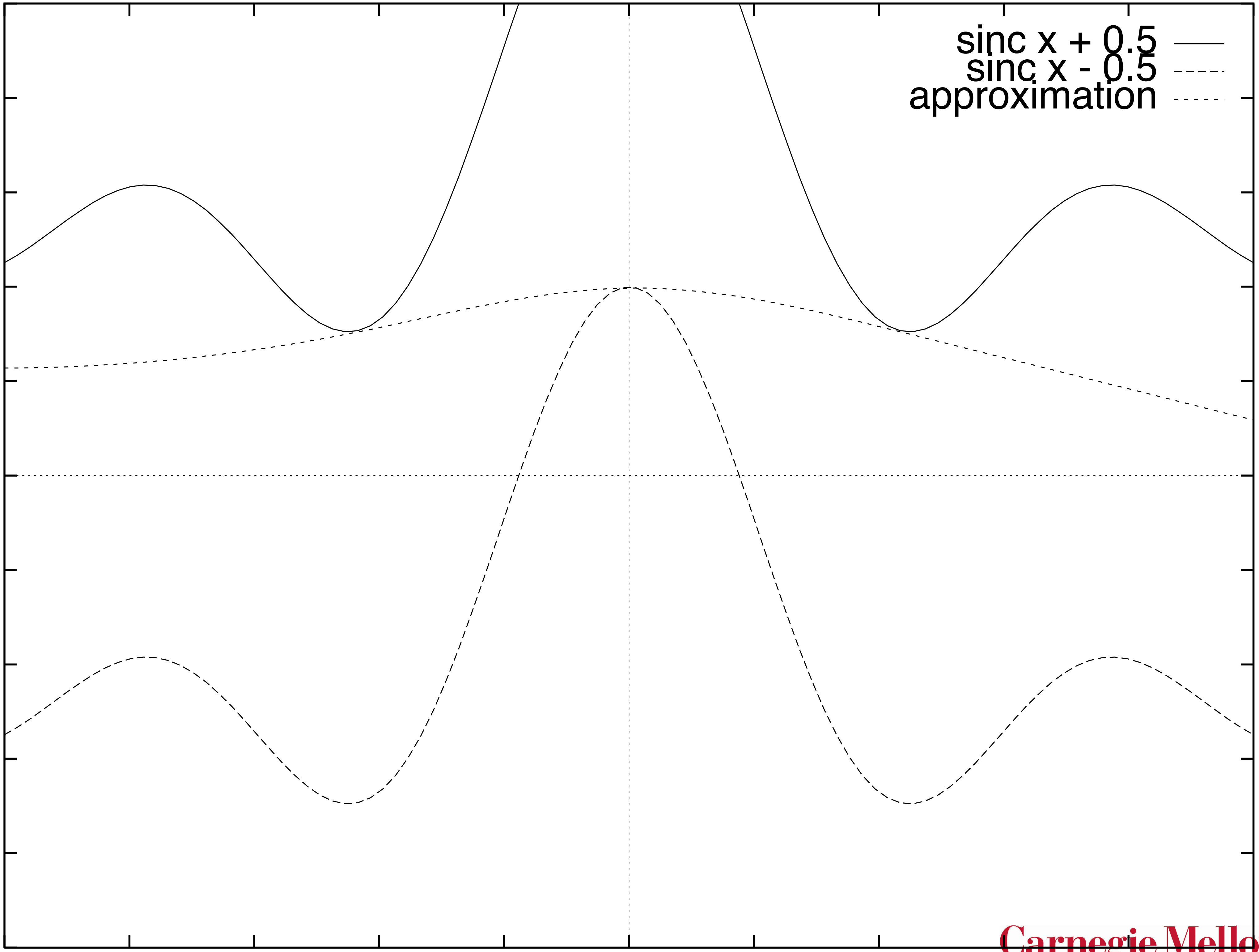


Carnegie Mellon University

Regression example

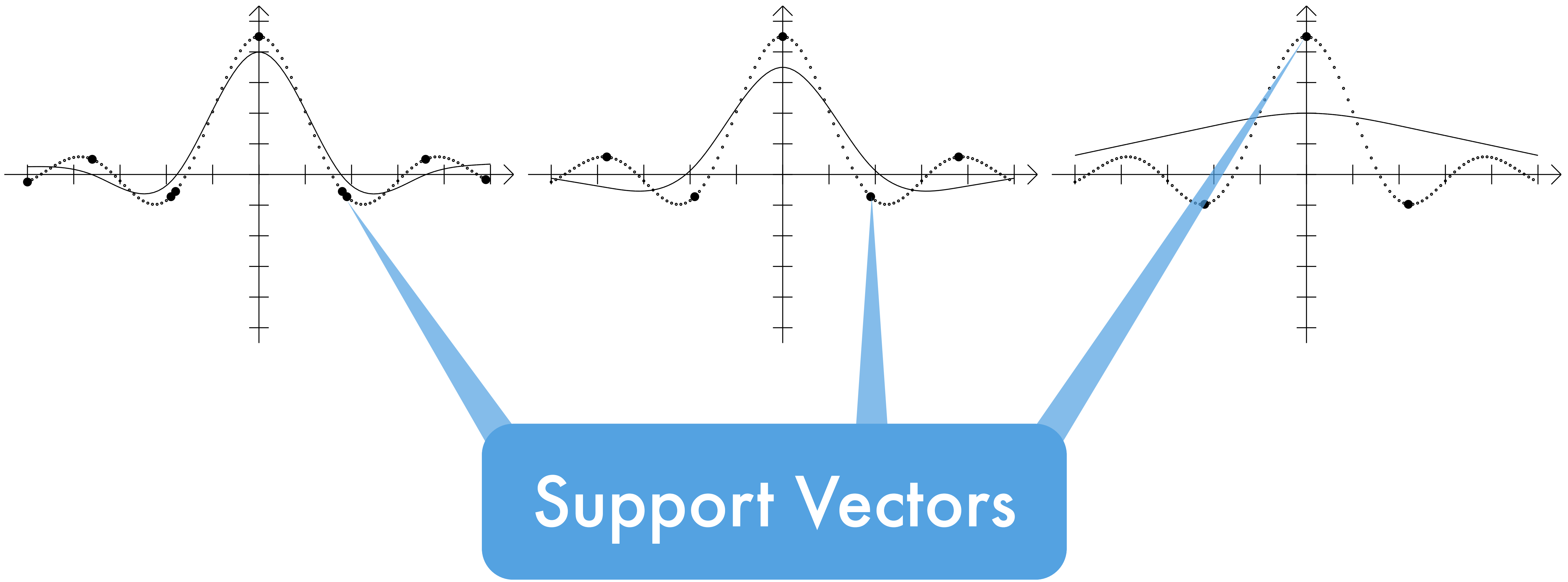


Regression example



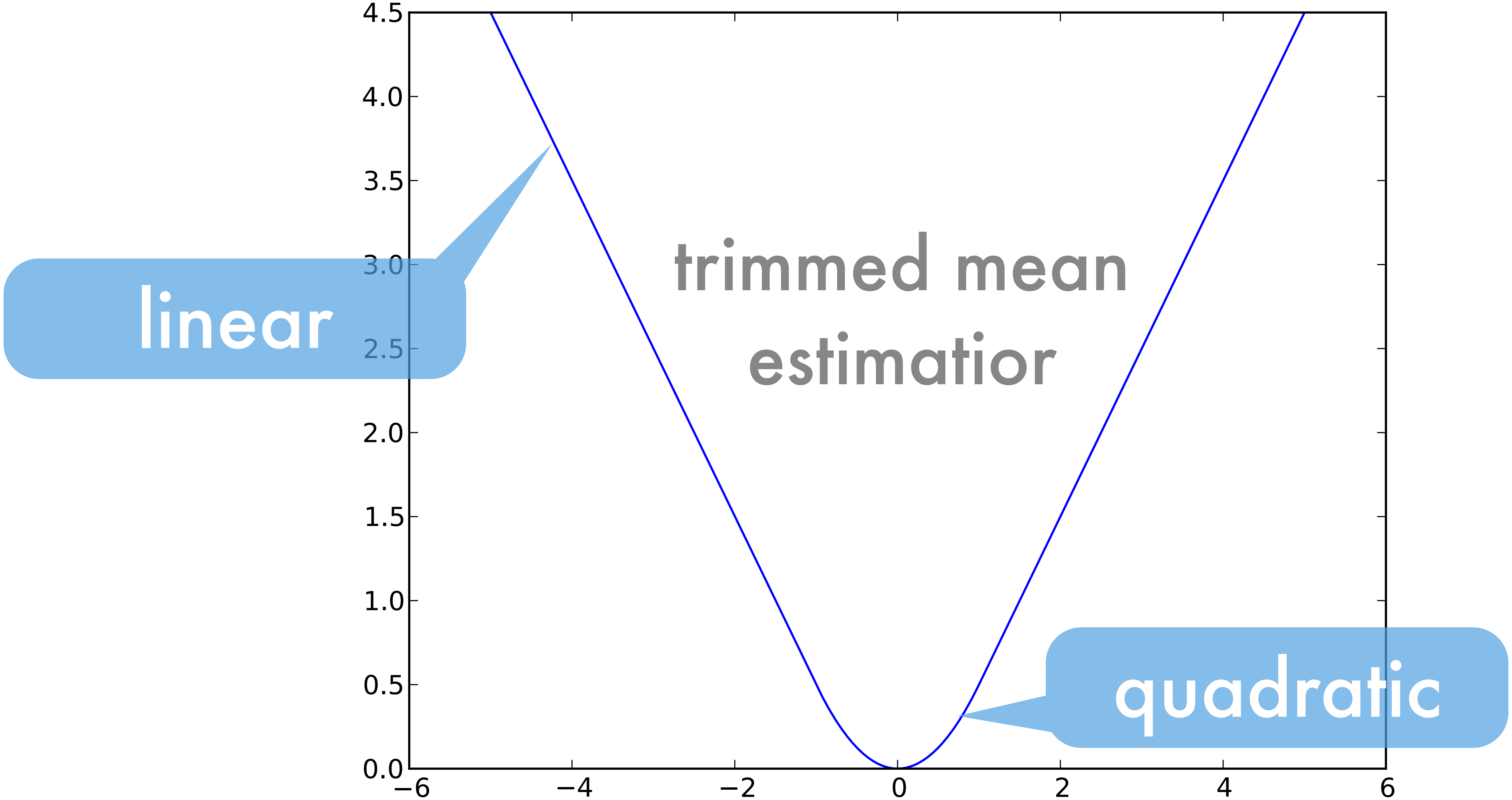
Carnegie Mellon University

Regression example

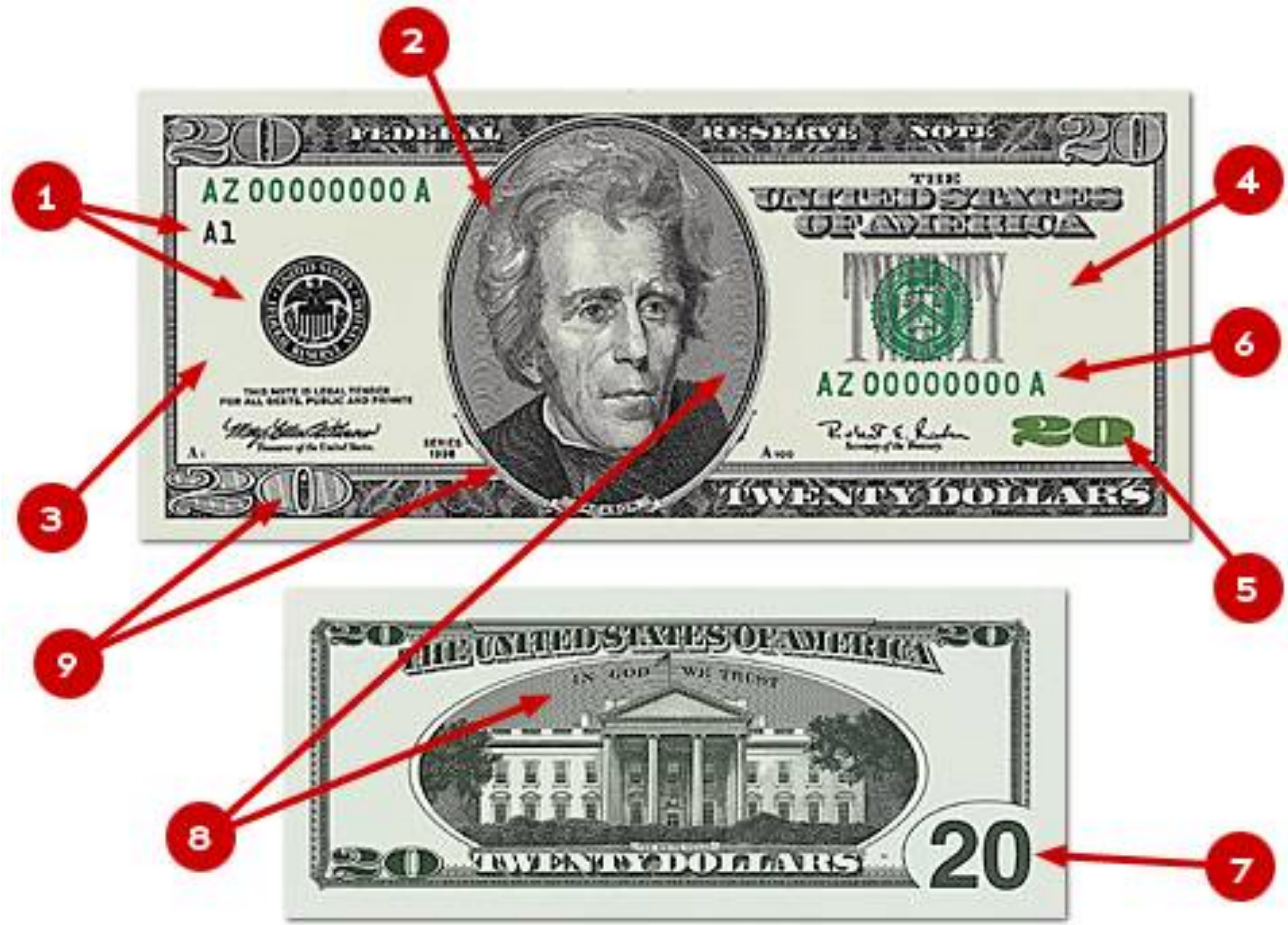


Huber's robust loss

$$l(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{if } |y - f(x)| < 1 \\ |y - f(x)| - \frac{1}{2} & \text{otherwise} \end{cases}$$



Novelty Detection



Basic Idea

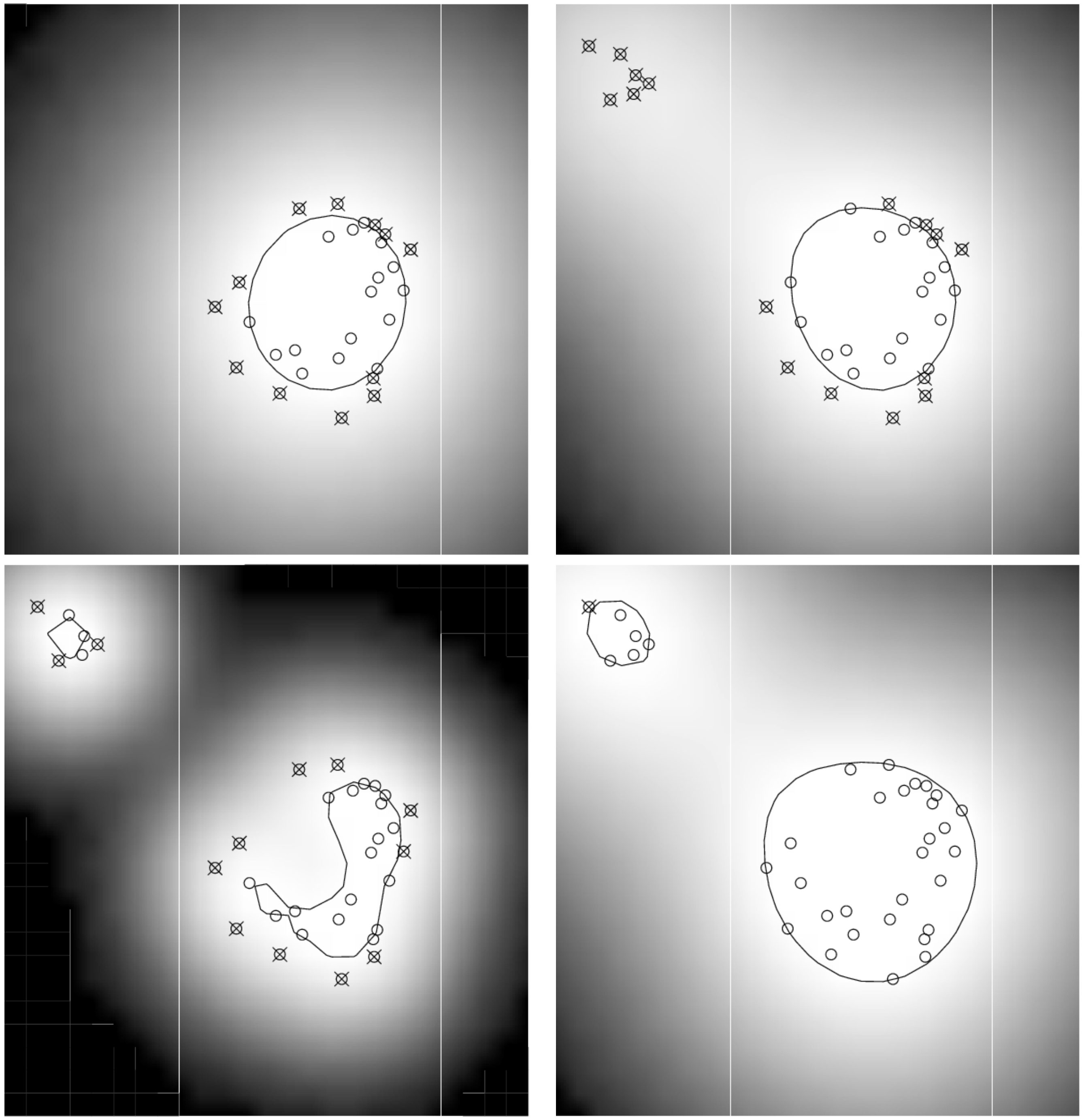
Data

Observations (x_i) generated from some $\mathrm{P}(x)$, e.g.,

- network usage patterns
- handwritten digits
- alarm sensors
- factory status

Task

Find unusual events, clean database, distinguish typical examples.



Carnegie Mellon University

Applications

Network Intrusion Detection

Detect whether someone is trying to hack the network, downloading tons of MP3s, or doing anything else *unusual* on the network.

Jet Engine Failure Detection

You can't destroy jet engines just to see *how* they fail.

Database Cleaning

We want to find out whether someone stored bogus information in a database (typos, etc.), mislabelled digits, ugly digits, bad photographs in an electronic album.

Fraud Detection

Credit Cards, Telephone Bills, Medical Records

Self calibrating alarm devices

Car alarms (adjusts itself to where the car is parked), home alarm (furniture, temperature, windows, etc.)

Novelty Detection via

Key Idea

- Novel data is one that we don't see frequently.
- It must lie in low density regions.

Step 1: Estimate density

- Observations $x_1, \ldots, x_m$
- Density estimate via Parzen windows

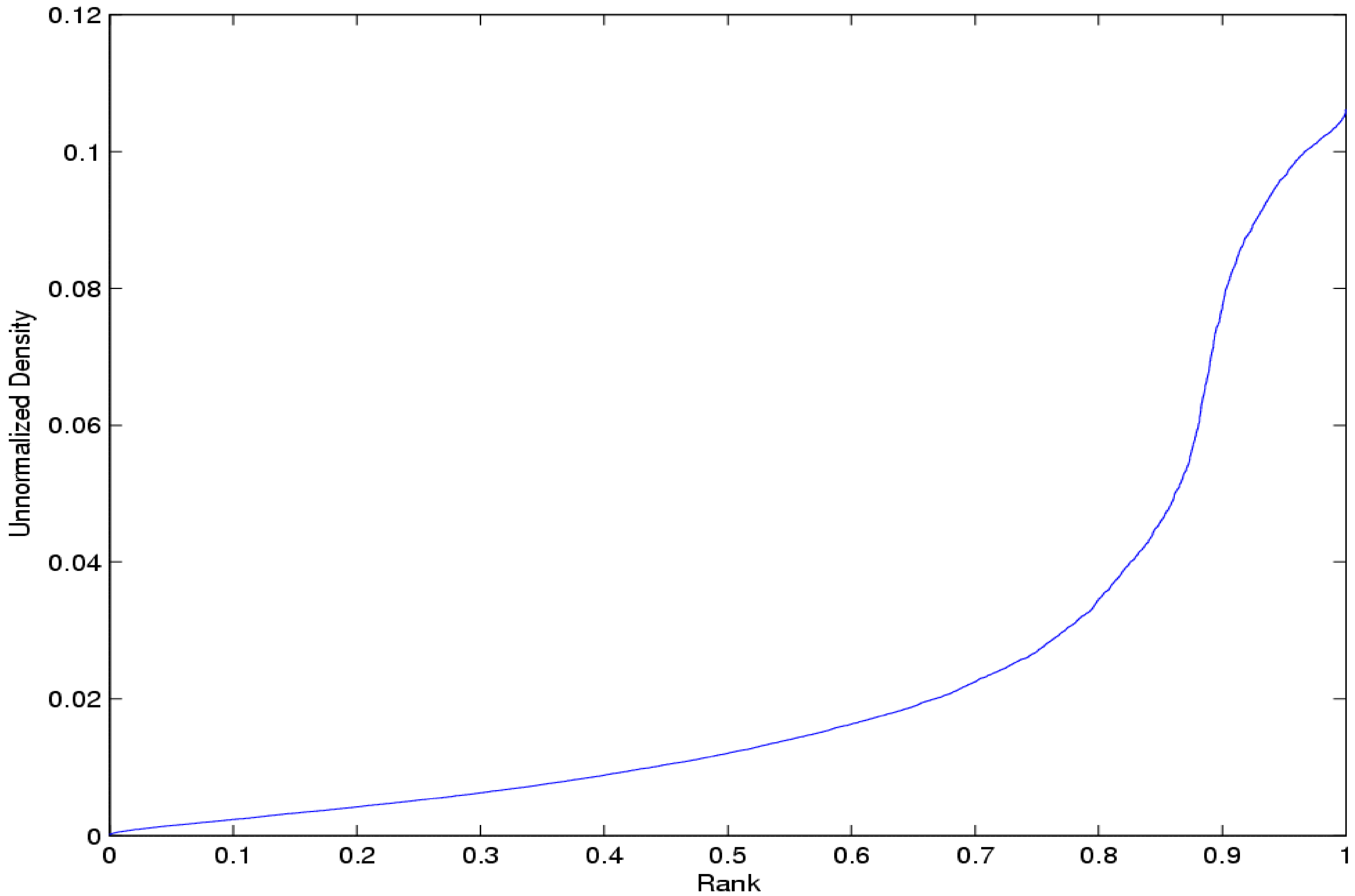
Step 2: Thresholding the density

- Sort data according to density and use it for rejection
- Practical implementation: compute

$$p(x_i) = \frac{1}{m} \sum_j k(x_i, x_j) \text{ for all } i$$

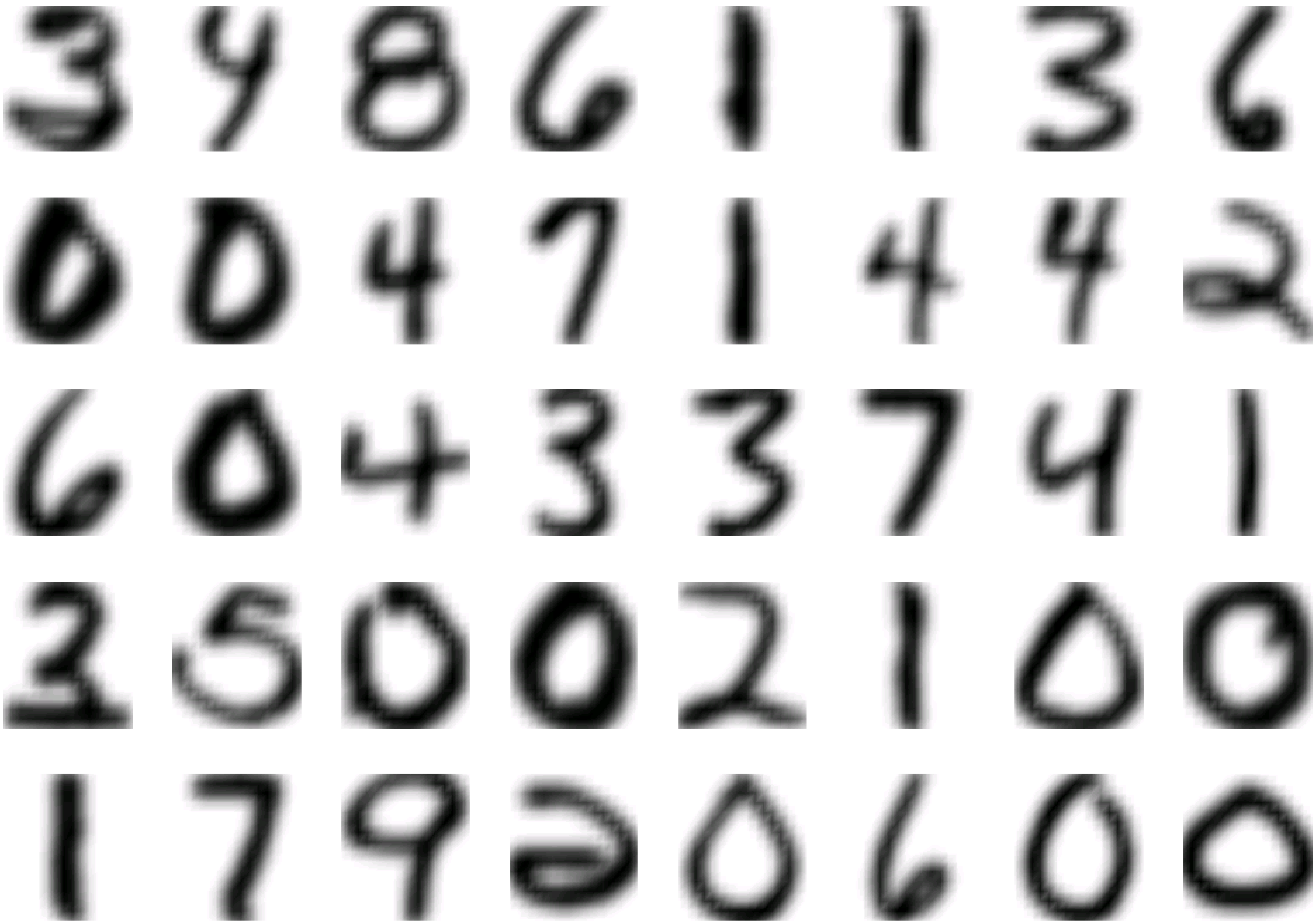
  and sort according to magnitude.
- Pick smallest $p(x_i)$ as novel points.

Order Statistics of Densities



iversity

Typical Data



Outliers

A better way

Problems

- We do not care about estimating the density properly in regions of high density (waste of capacity).
- We only care about the relative density for thresholding purposes.
- We want to eliminate a certain fraction of observations and tune our estimator specifically for this fraction.

Solution

- Areas of low density can be approximated as the **level set** of an auxiliary function. No need to estimate $p(x)$ directly — use proxy of $p(x)$.
- Specifically: find $f(x)$ such that x is novel if $f(x) \leq c$ where c is some constant, i.e. $f(x)$ describes the amount of novelty.

Problems with density estimation

- Exponential Family for density estimation

$$p(x|\theta) = \exp\left(\langle\phi(x), \theta\rangle - g(\theta)\right)$$

- MAP estimation

$$\underset{\theta}{\operatorname{minimize}} \sum_i g(\theta) - \langle\phi(x_i), \theta\rangle + \frac{1}{2\sigma^2}\left\|\theta\right\|^2$$

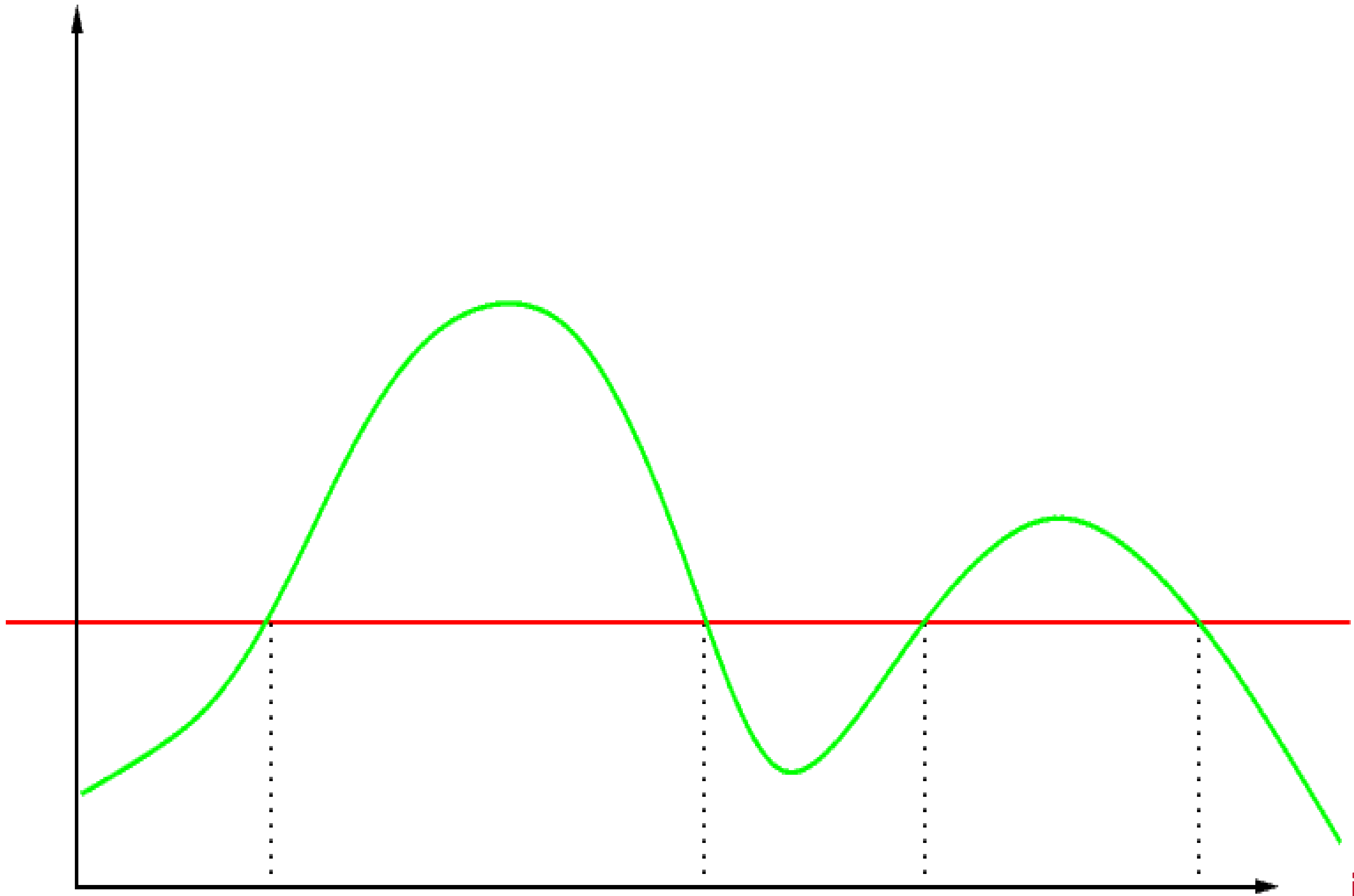
Advantages

- Convex optimization problem
- Concentration of measure

Problems

- Normalization $g(\theta)$ may be painful to compute
- For density estimation we need no normalized $p(x|\theta)$
- No need to perform particularly well in high density regions

Thresholding



Optimization Problem

Optimization Problem

MAP
$$\sum_{i=1}^{m} -\log p(x_i|\theta) + \frac{1}{2\sigma^2}\|\theta\|^2$$

Novelty
$$\sum_{i=1}^{m} \max\left(-\log \frac{p(x_i|\theta)}{\exp(\rho - g(\theta))}, 0\right) + \frac{1}{2}\|\theta\|^2$$

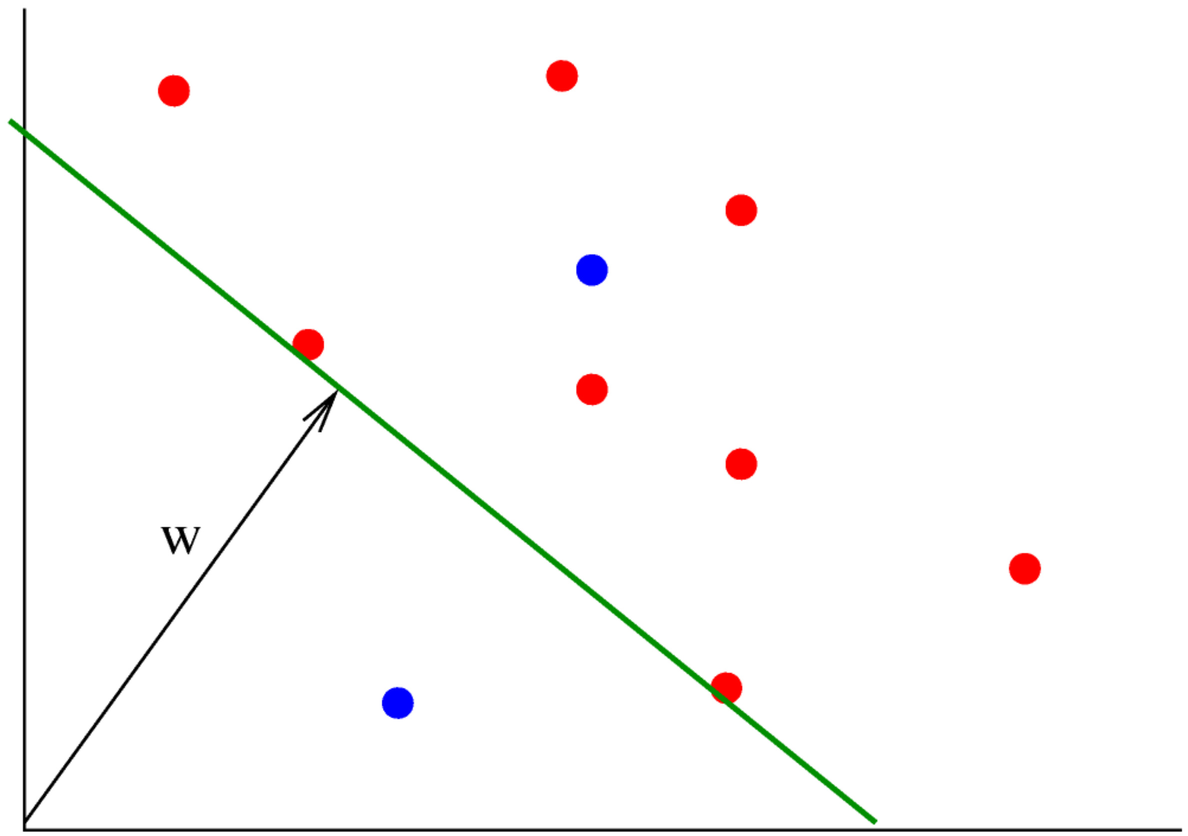
$$\sum_{i=1}^{m} \max(\rho - \langle\phi(x_i), \theta\rangle, 0) + \frac{1}{2}\|\theta\|^2$$

Advantages

- No normalization $g(\theta)$ needed
- No need to perform particularly well in high density regions (estimator focuses on low-density regions)
- Quadratic program

Maximum Distance

Idea Find hyperplane, given by $f(x) = \langle w, x \rangle + b = 0$ that has **maximum distance from origin** yet is still closer to the origin than the observations.



Hard Margin

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & \langle w, x_i \rangle \geq 1 \end{aligned}$$

Soft Margin

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & \langle w, x_i \rangle \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

Optimization Problem

Primal Problem

minimize $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$

subject to $\langle w, x_i\rangle - 1 + \xi_i \geq 0$ and $\xi_i \geq 0$

Lagrange Function L

- Subtract constraints, multiplied by Lagrange multipliers (α_i and η_i), from Primal Objective Function.
- Lagrange function L has **saddlepoint** at optimum.

$$L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i\left(\langle w, x_i\rangle - 1 + \xi_i\right) - \sum_{i=1}^{m}\eta_i\xi_i$$

subject to $\alpha_i, \eta_i \geq 0$.

Dual Problem

Optimality Conditions

$$\partial_w L = w - \sum_{i=1}^{m} \alpha_i x_i = 0 \implies w = \sum_{i=1}^{m} \alpha_i x_i$$
$$\partial_{\xi_i} L = C - \alpha_i - \eta_i = 0 \implies \alpha_i \in [0, C]$$

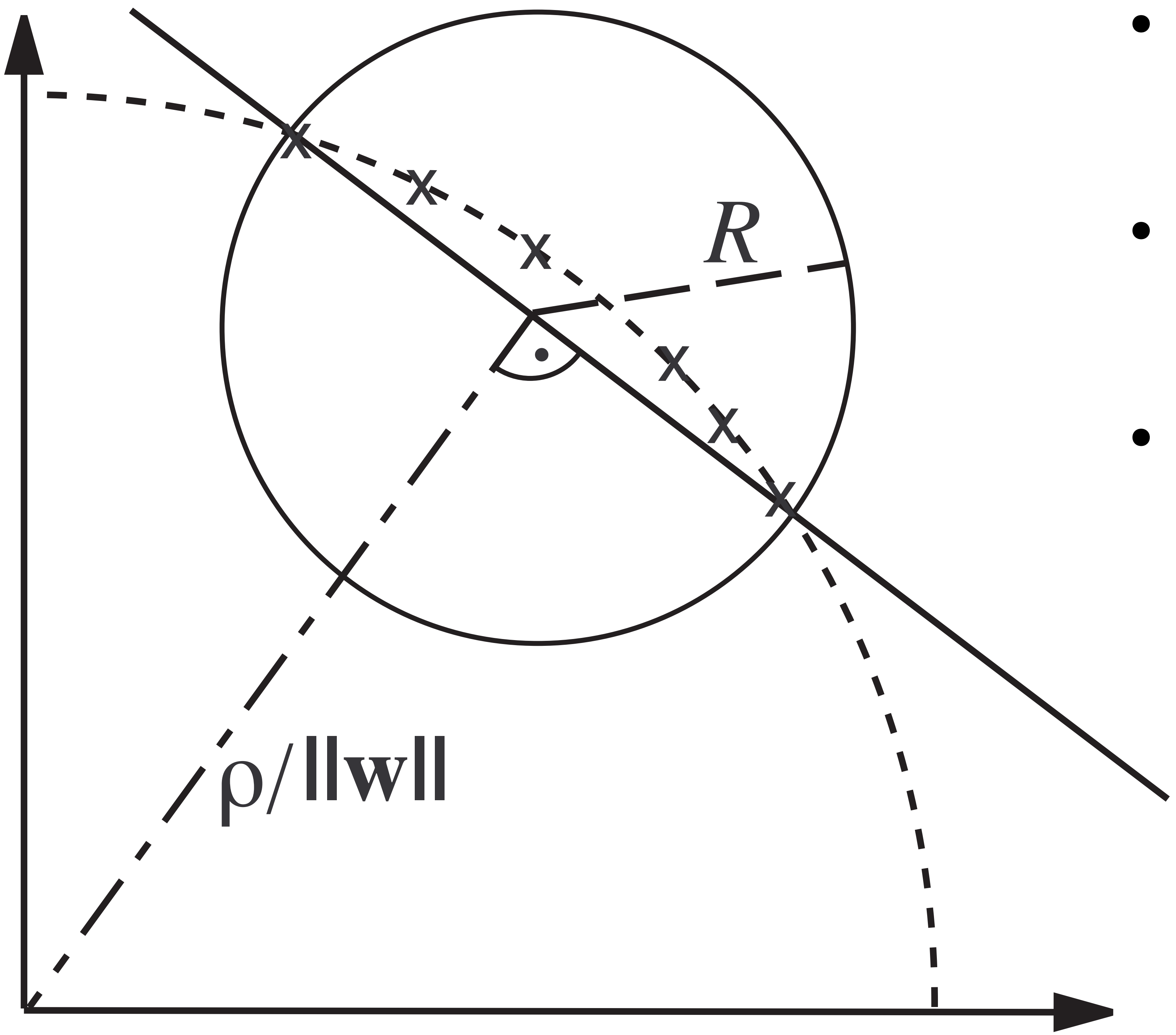
Now **substitute** the optimality conditions **back into** L.

Dual Problem

$$\text{minimize} \quad \frac{1}{2} \sum_{i=1}^{m} \alpha_i \alpha_j \langle x_i, x_j \rangle - \sum_{i=1}^{m} \alpha_i$$
$$\text{subject to} \quad \alpha_i \in [0, C]$$

All this is only possible due to the convexity of the primal problem.

Minimum enclosing ball



- Observations on surface of ball
- Find minimum enclosing ball
- Equivalent to single class SVM

Adaptive thresholds

Problem

- Depending on C, the number of novel points will vary.
- We would like to **specify the fraction** ν beforehand.

Solution

Use hyperplane separating data from the origin

$$H := \{x | \langle w, x \rangle = \rho\}$$

where the threshold ρ is **adaptive**.

Intuition

- Let the hyperplane shift by shifting ρ
- Adjust it such that the 'right' number of observations is considered novel.
- Do this automatically

Optimization Problem

Primal Problem

$$\text{minimize } \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \xi_i - m\nu\rho$$

$$\text{where } \langle w, x_i \rangle - \rho + \xi_i \geq 0$$

$$\xi_i \geq 0$$

Dual Problem

$$\text{minimize } \frac{1}{2}\sum_{i=1}^{m} \alpha_i \alpha_j \langle x_i, x_j \rangle$$

$$\text{where } \alpha_i \in [0, 1] \text{ and } \sum_{i=1}^{m} \alpha_i = \nu m.$$

The v-property theorem

- Optimization problem

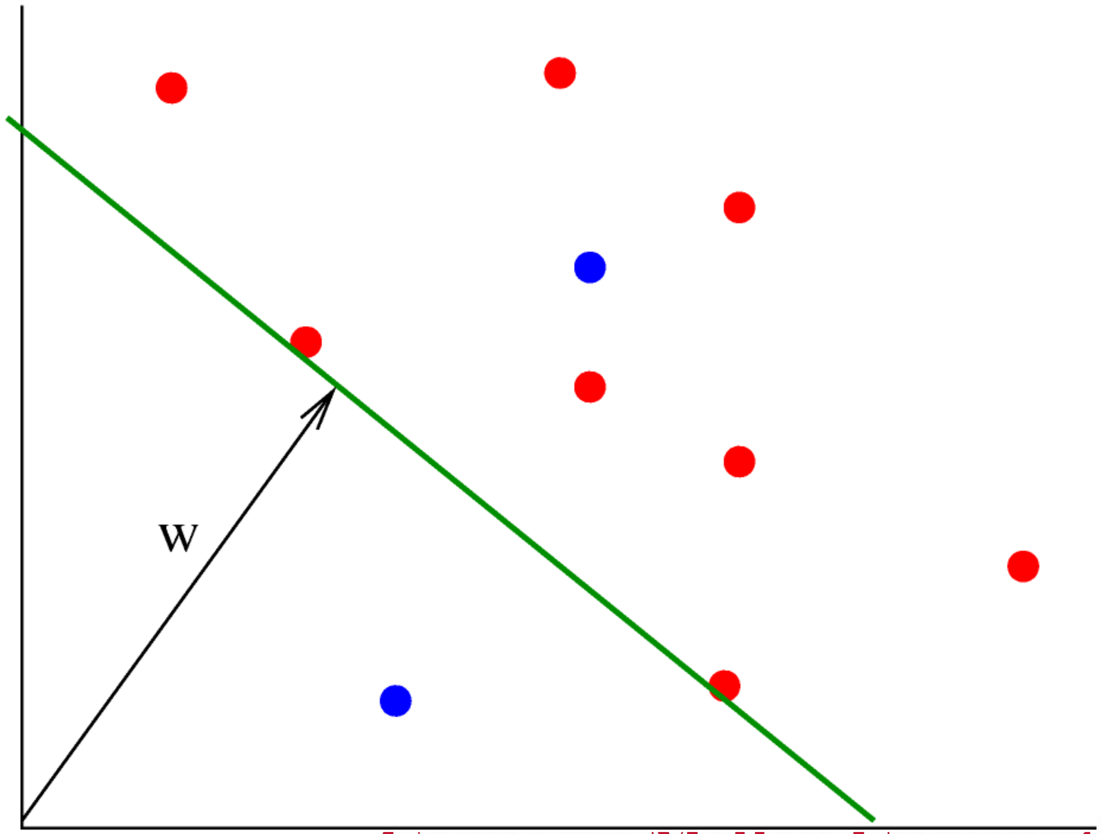
$$\underset{w}{\text{minimize}} \;\; \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \xi_i - m\nu\rho$$

$$\text{subject to } \langle w, x_i \rangle \geq \rho - \xi_i \text{ and } \xi_i \geq 0$$

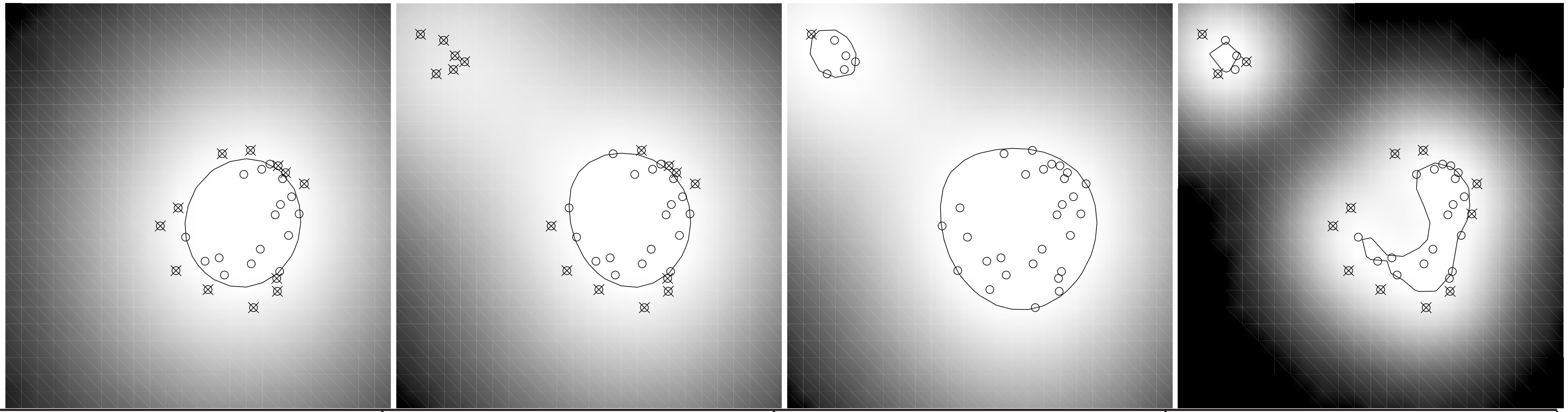
- Solution satisfies
  - At most a fraction of v points are novel
  - At most a fraction of (1-v) points aren't novel
  - Fraction of points on boundary vanishes for large m (for non-pathological kernels)

Proof

- Move boundary at optimality
  - For smaller threshold m- points on wrong side of margin contribute $\delta(m_- - \nu m) \leq 0$
  - For larger threshold m+ points not on 'good' side of margin yield
  $$\delta(m_+ - \nu m) \geq 0$$
  - Combining inequalities
  $$\frac{m_-}{m} \leq \nu \leq \frac{m_+}{m}$$
- Margin set of measure 0



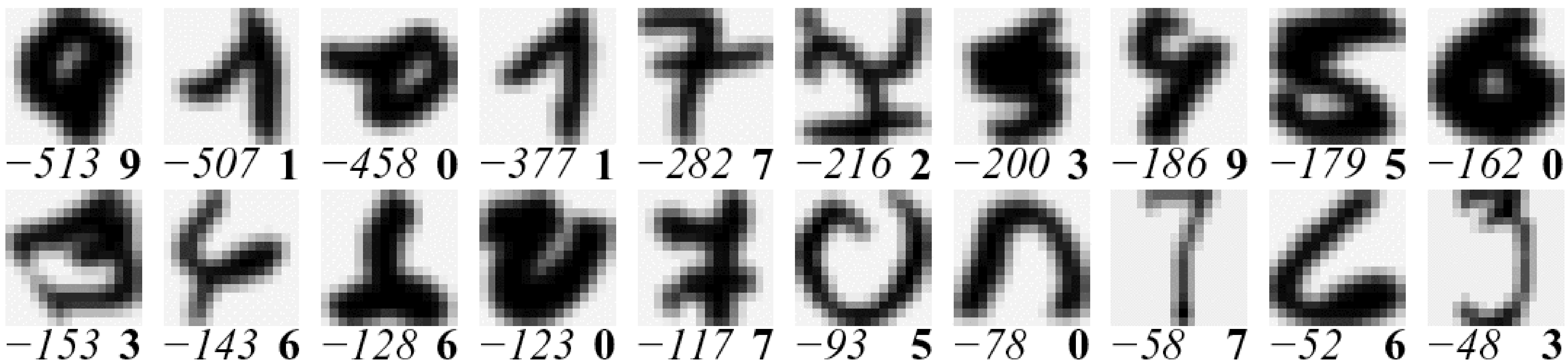
Toy example



ν, width c	0.5, 0.5	0.5, 0.5	0.1, 0.5	0.5, 0.1
frac. SVs/OLs	0.54, 0.43	0.59, 0.47	0.24, 0.03	0.65, 0.38
margin $\rho/\|\mathbf{w}\|$	0.84	0.70	0.62	0.48

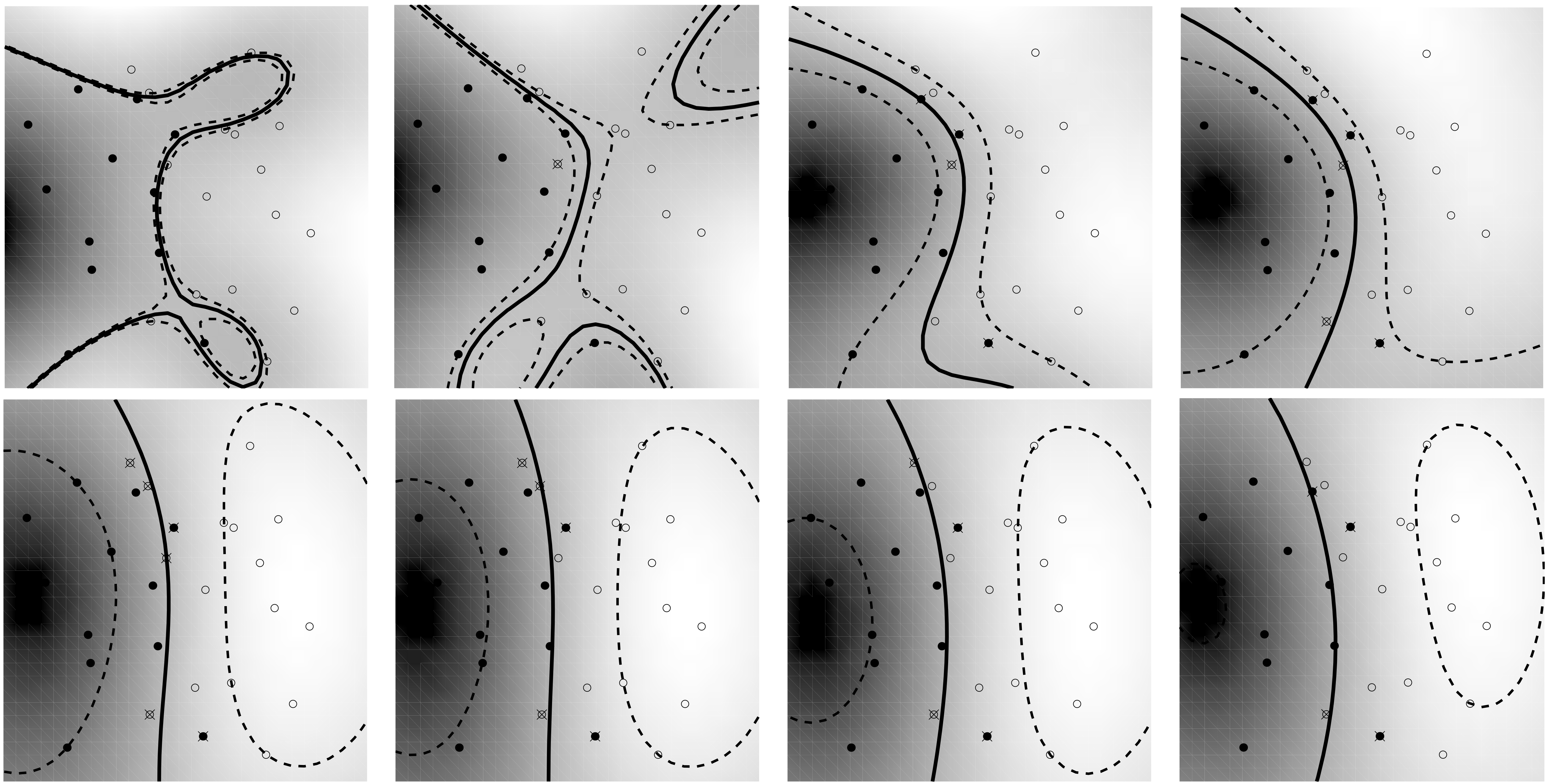
threshold and smoothness requirements

Novelty detection for OCR



- Better estimates since we only optimize in low density regions.
- Specifically tuned for small number of outliers.
- Only estimates of a level-set.
- For $\nu = 1$ we get the Parzen-windows estimator back.

Classification with the ν-



changing kernel width and threshold

4.4 Optimization

4 (Generalized) Linear Methods

Alexander Smola

Introduction to Machine Learning 10-701

http://alex.smola.org/teaching/10-701-15

MACHINE LEARNING DEPARTMENT

Carnegie Mellon University

Efficient Convex Optimization

Constrained Quadratic Program

- Optimization Problem

$$\underset{\alpha}{\text{minimize}}\ \frac{1}{2}\alpha^\top Q\alpha + l^\top\alpha \text{ subject to } C\alpha + b \le 0$$

  - Support Vector classification
  - Support Vector regression
  - Novelty detection
- Solving it
  - Off the shelf solvers for small problems
  - Solve sequence of subproblems
  - Optimization in primal space (the w space)

Subproblems

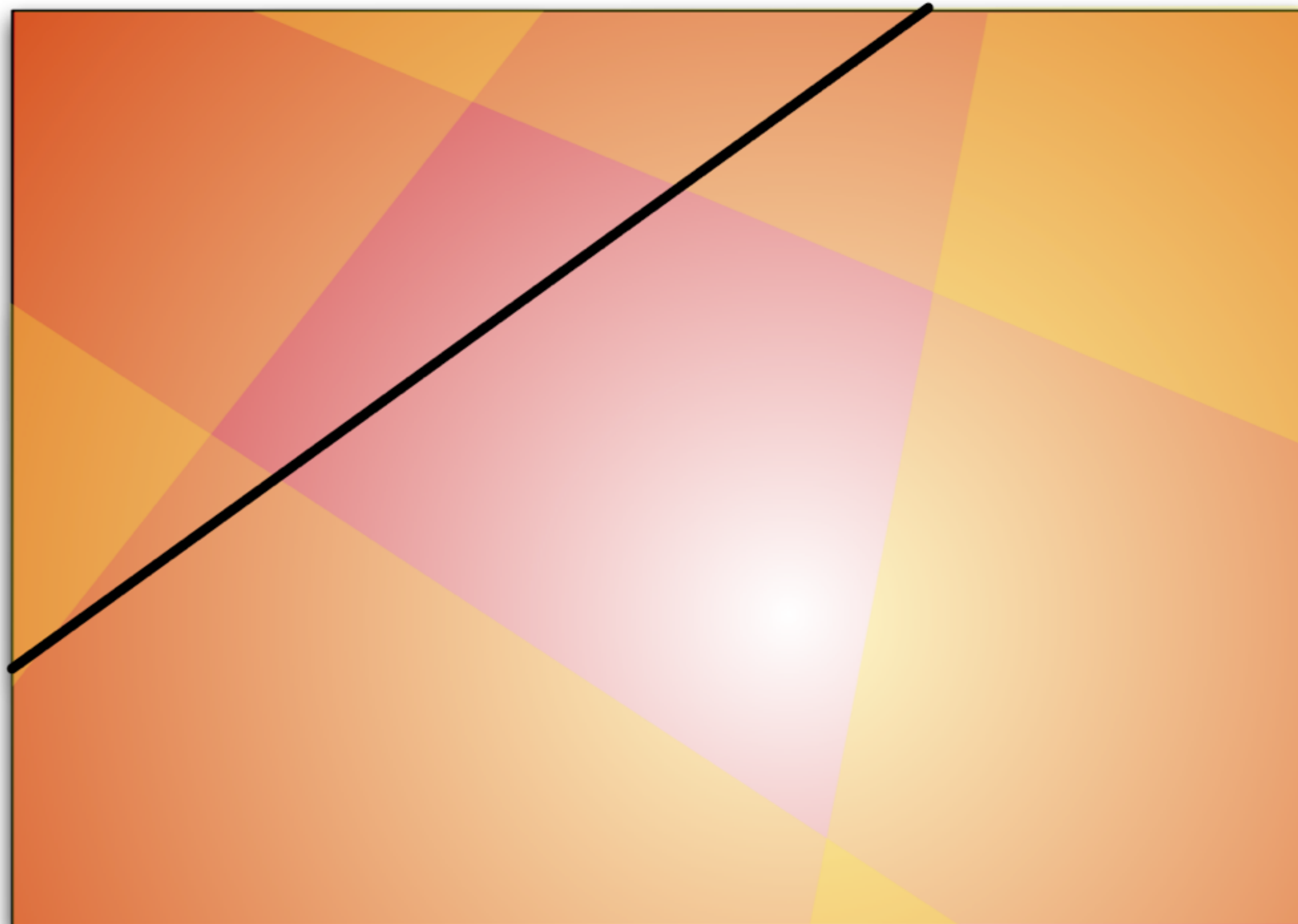
- Original optimization problem

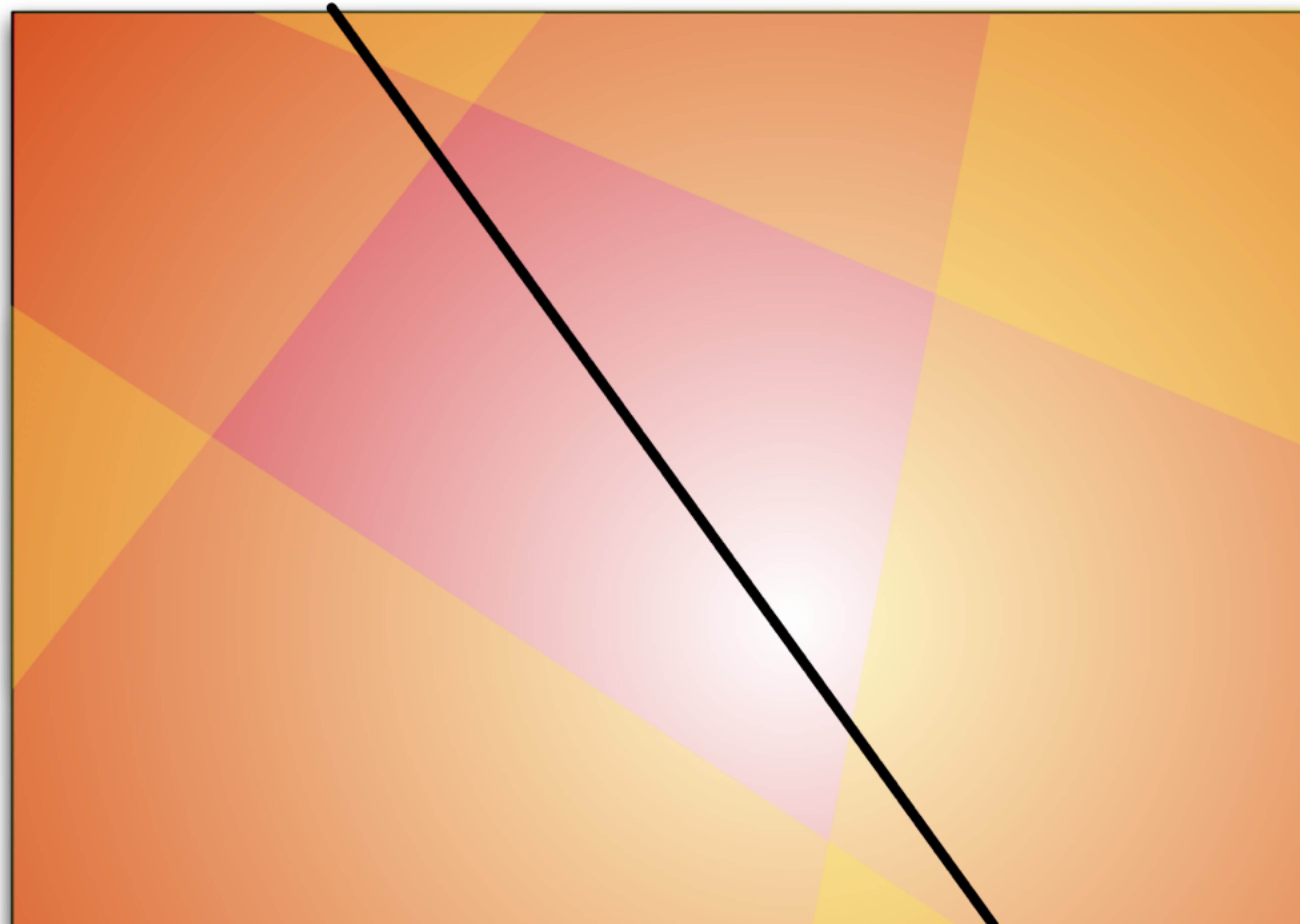
$$\underset{\alpha}{\text{minimize}} \frac{1}{2}\alpha^\top Q\alpha + l^\top \alpha \text{ subject to } C\alpha + b \leq 0$$

- Key Idea - solve subproblems one at a time and decompose into active and fixed set $\alpha = (\alpha_a, \alpha_f)$

$$\underset{\alpha}{\text{minimize}} \;\; \frac{1}{2}\alpha_a^\top Q_{aa}\alpha_a + \left[l_a + Q_{af}\alpha_f\right]^\top \alpha_a$$
$$\text{subject to } C_a\alpha_a + \left[b + C_f\alpha_f\right] \leq 0$$

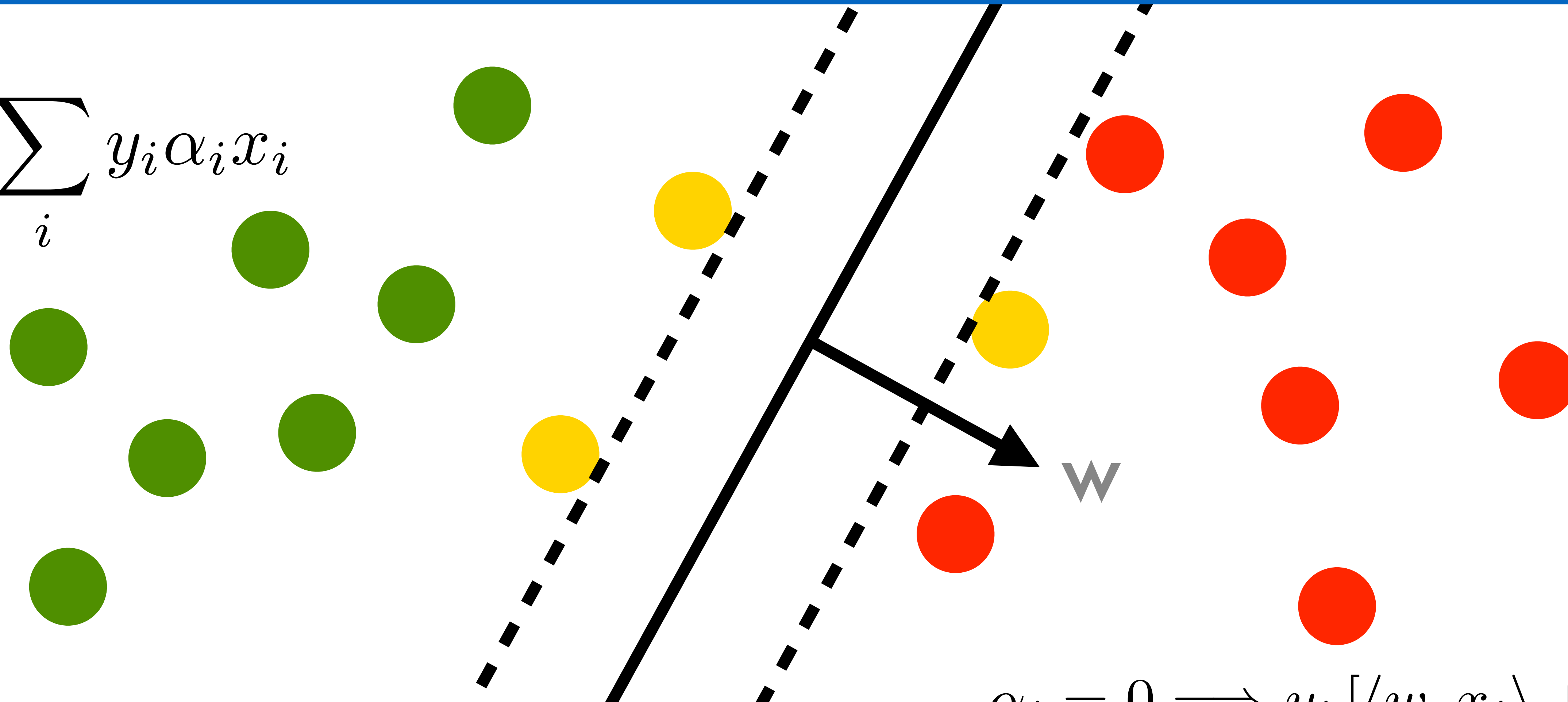
  - Subproblem is again a convex problem
  - Updating subproblems is cheap





Picking observations

$$w = \sum_i y_i \alpha_i x_i$$



$$\alpha_i \left[y_i \left[\langle w, x_i \rangle + b \right] + \xi_i - 1 \right] = 0$$

$$\eta_i \xi_i = 0$$

$$\alpha_i = 0 \Longrightarrow y_i \left[\langle w, x_i \rangle + b \right] \geq 1$$

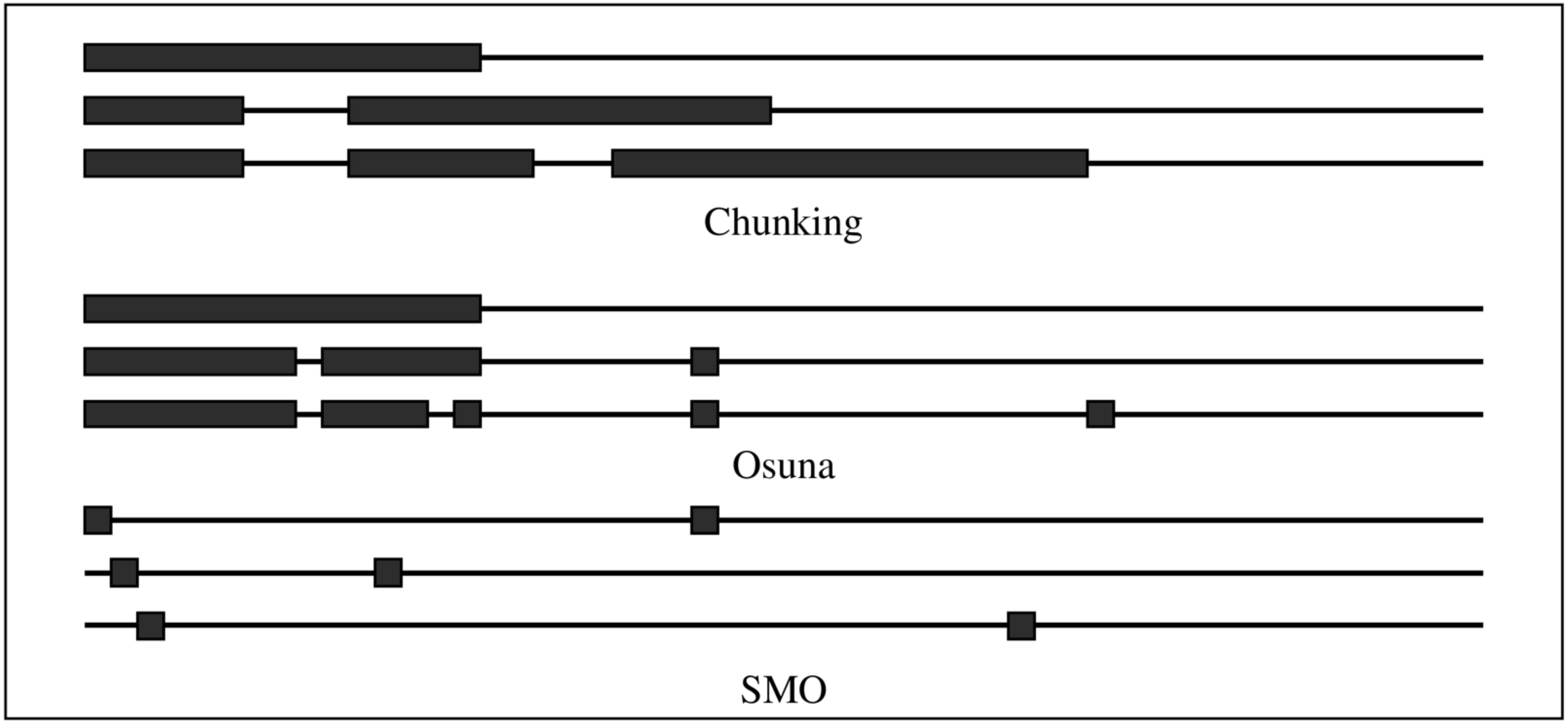
$$0 < \alpha_i < C \Longrightarrow y_i \left[\langle w, x_i \rangle + b \right] = 1$$

$$\alpha_i = C \Longrightarrow y_i \left[\langle w, x_i \rangle + b \right] \leq 1$$

- Most violated margin condition
- Points on the boundary
- Points with nonzero Lagrange multiplier that are correct

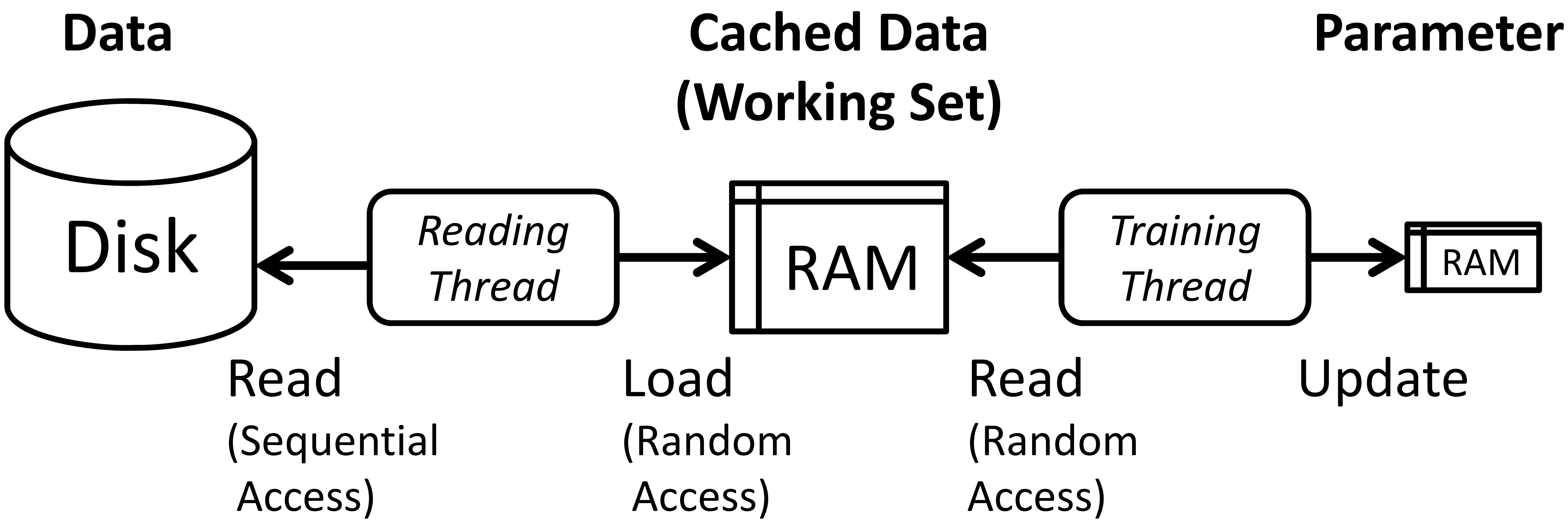
Selecting variables

- Incrementally increase (chunking)
- Select promising subset of actives (SVMLight)
- Select pairs of variables (SMO)



Being smart about hardware

- Data flow from disk to CPU

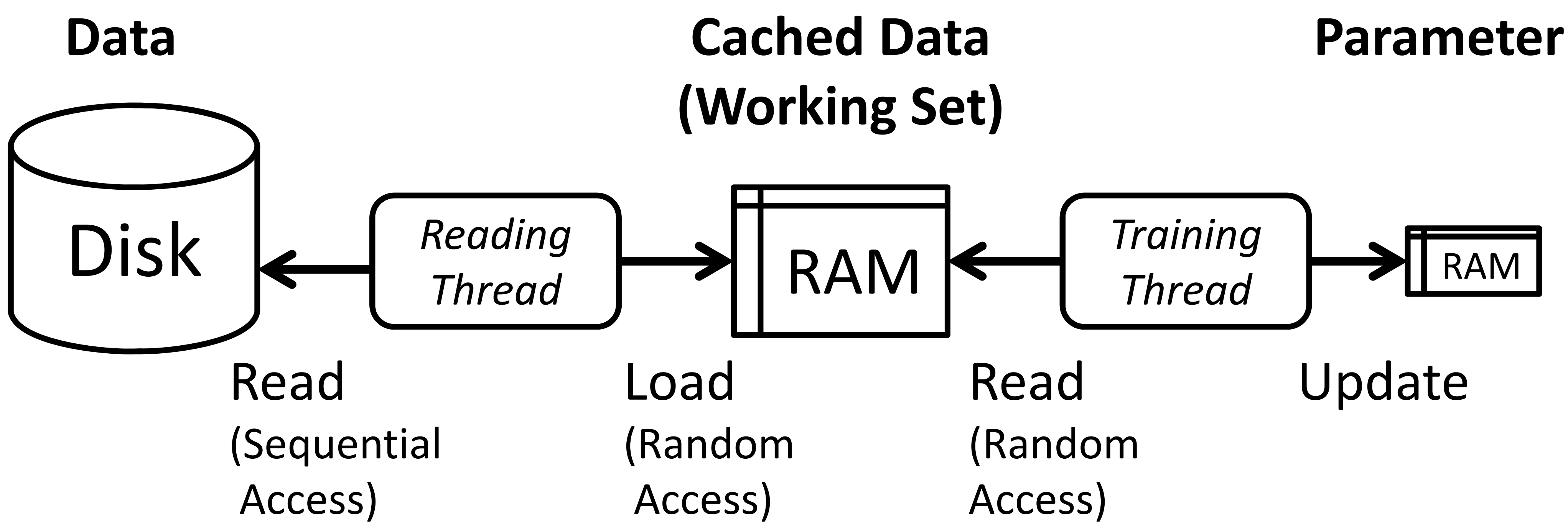


- IO speeds

System	Capacity	Bandwidth	IOPs
Disk	3TB	150MB/s	10^2
SSD	256GB	500MB/s	$5 \cdot 10^4$
RAM	16GB	30GB/s	10^8
Cache	16MB	100GB/s	10^9

Being smart about hardware

- Data flow from disk to CPU

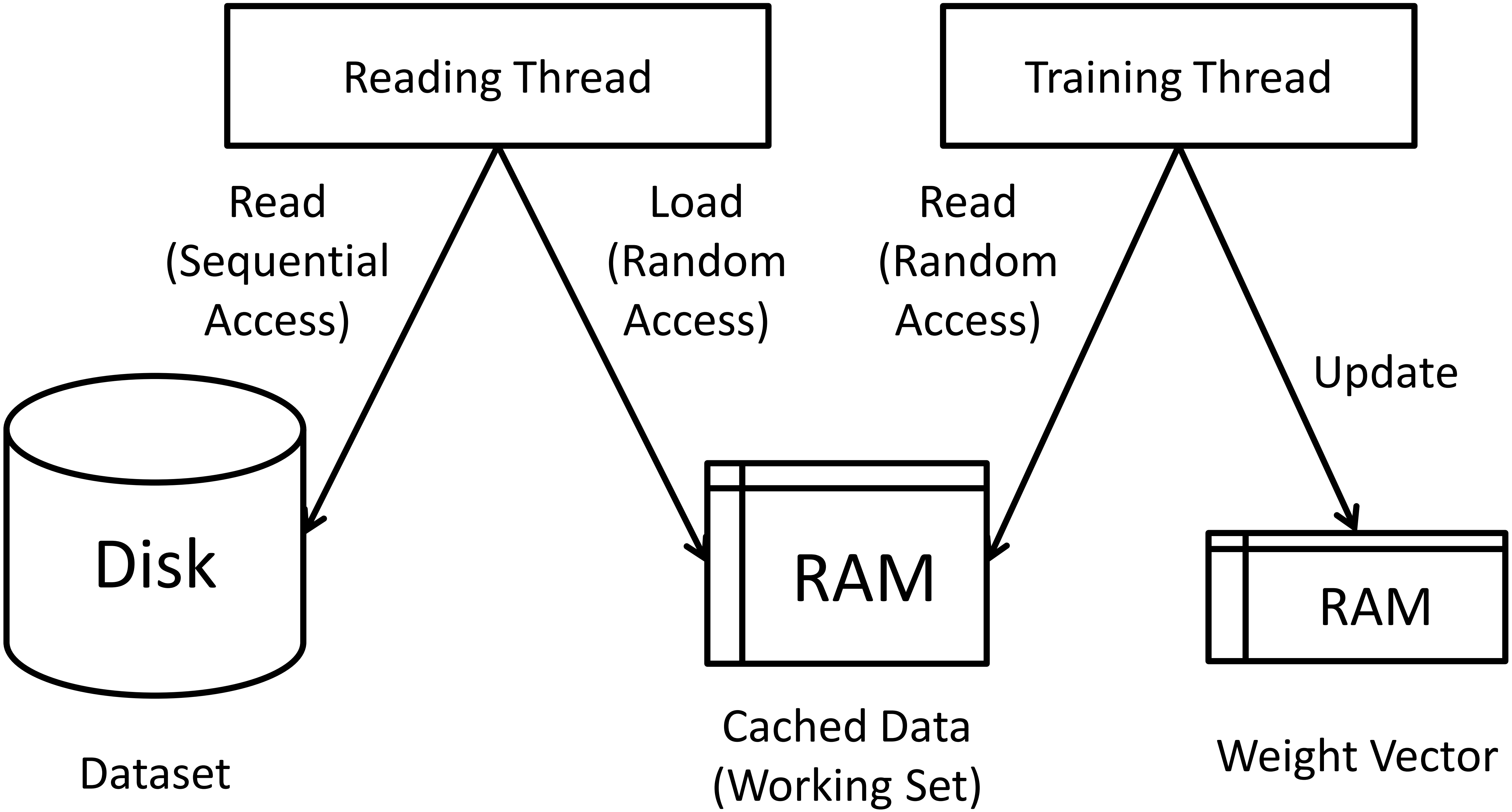


- IO speeds

System	Capacity	Bandwidth	IOPs
Disk	3TB	150MB/s	10^2
SSD	256GB	500MB/s	$5 \cdot 10^4$
RAM	16GB	30GB/s	10^8
Cache	16MB	100GB/s	10^9

reuse data

Dataflow



Reading Thread
Training Thread
Read
(Sequential
Access)
Load
(Random
Access)
Read
(Random
Access)
Update
Disk
RAM
RAM
Dataset
Cached Data
(Working Set)
Weight Vector

Algorithm - 2 loops

Reader
while not converged **do**
  read example (x, y) from disk
  if buffer full **then** evict random (x', y') from memory
  insert new (x, y) into ring buffer in memory
end while

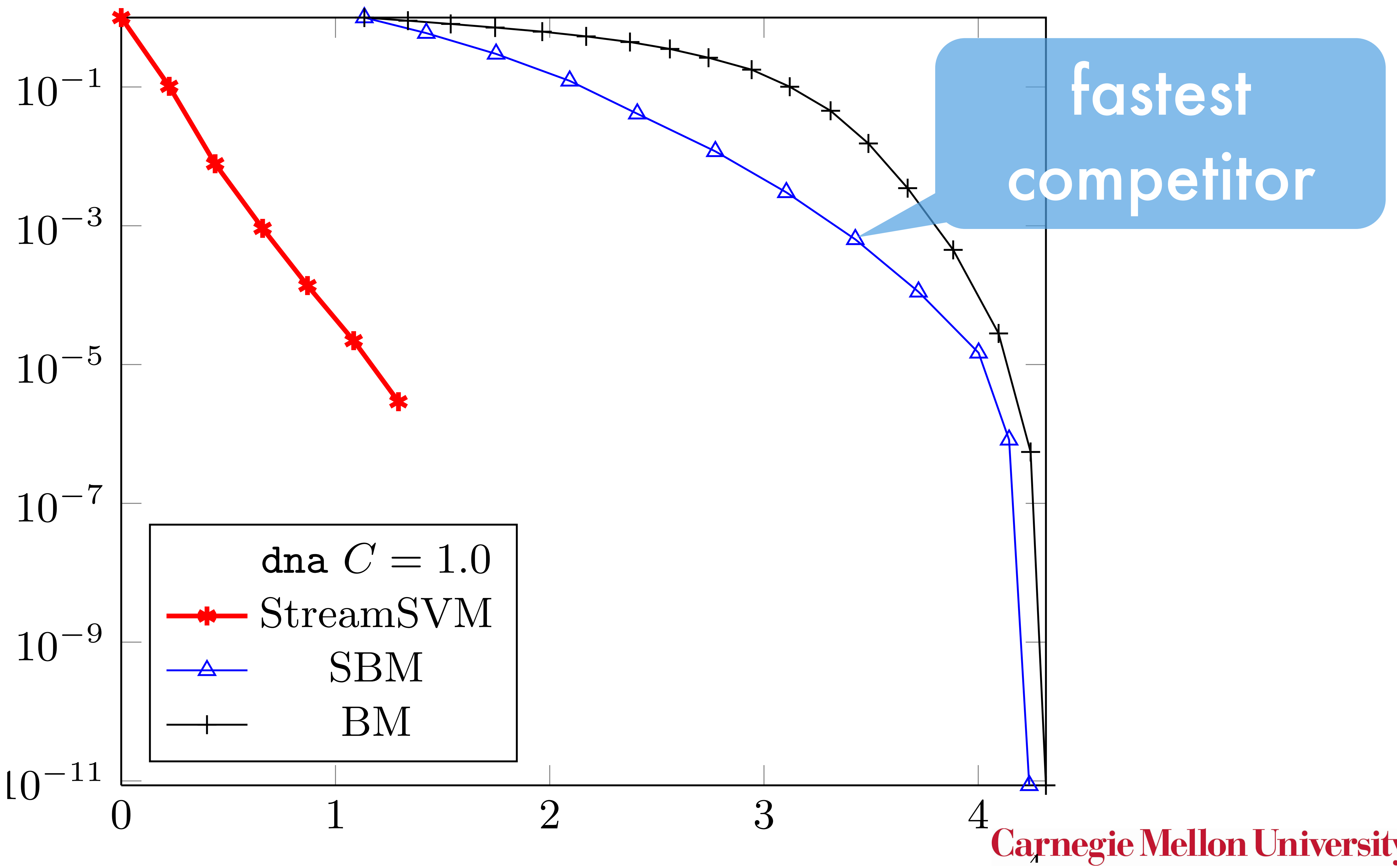
at disk speed

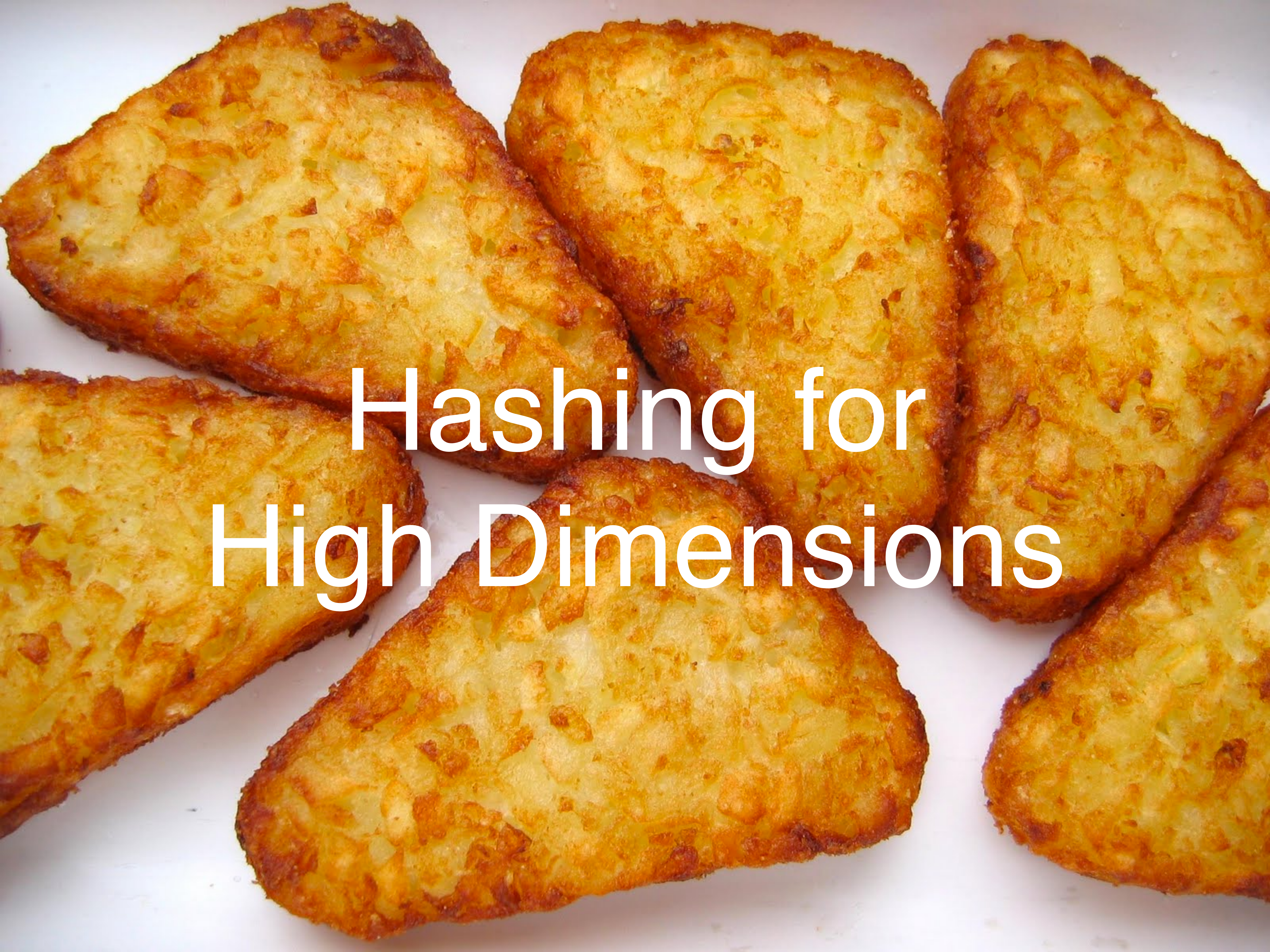
Trainer
while not converged **do**
  randomly pick example (x, y) from memory
  update dual parameter α
  update weight vector w
  if deemed to be uninformative **then** evict (x, y) from memory
end while

at RAM speed

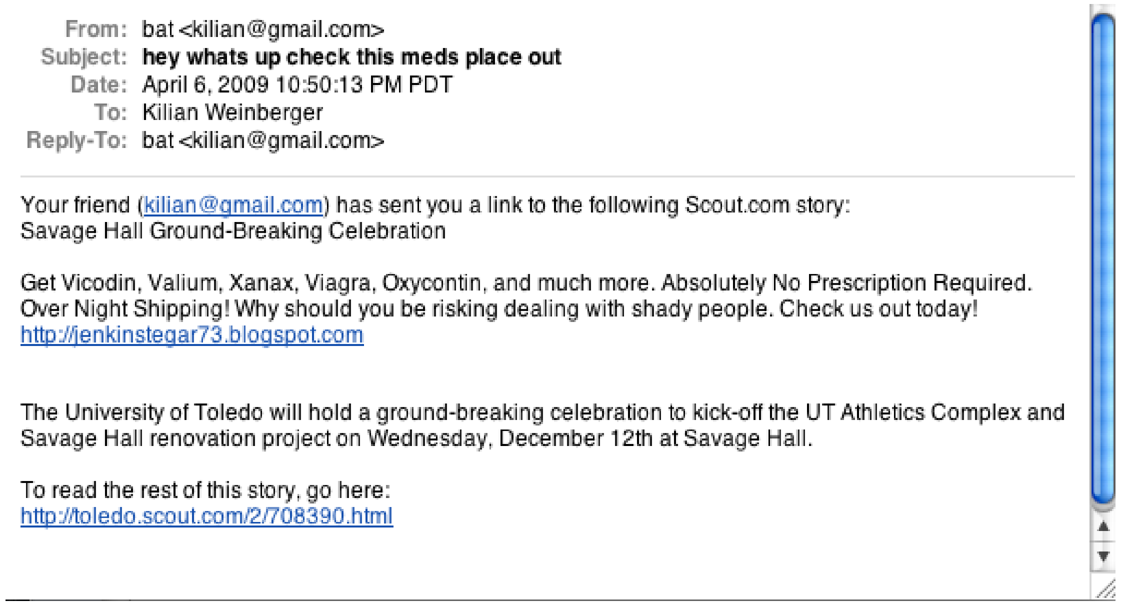
margin criterion

Runtime Example




Hashing for
High Dimensions

Spam Classification



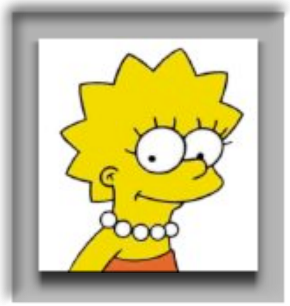
From: bat <kilian@gmail.com>
Subject: **hey whats up check this meds place out**
Date: April 6, 2009 10:50:13 PM PDT
To: Kilian Weinberger
Reply-To: bat <kilian@gmail.com>

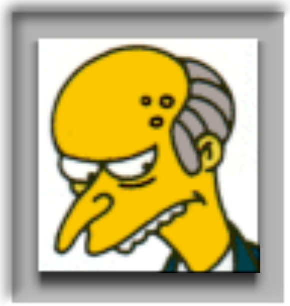
Your friend (kilian@gmail.com) has sent you a link to the following Scout.com story:
Savage Hall Ground-Breaking Celebration

Get Vicodin, Valium, Xanax, Viagra, Oxycontin, and much more. Absolutely No Prescription Required. Over Night Shipping! Why should you be risking dealing with shady people. Check us out today!
http://jenkinstegar73.blogspot.com

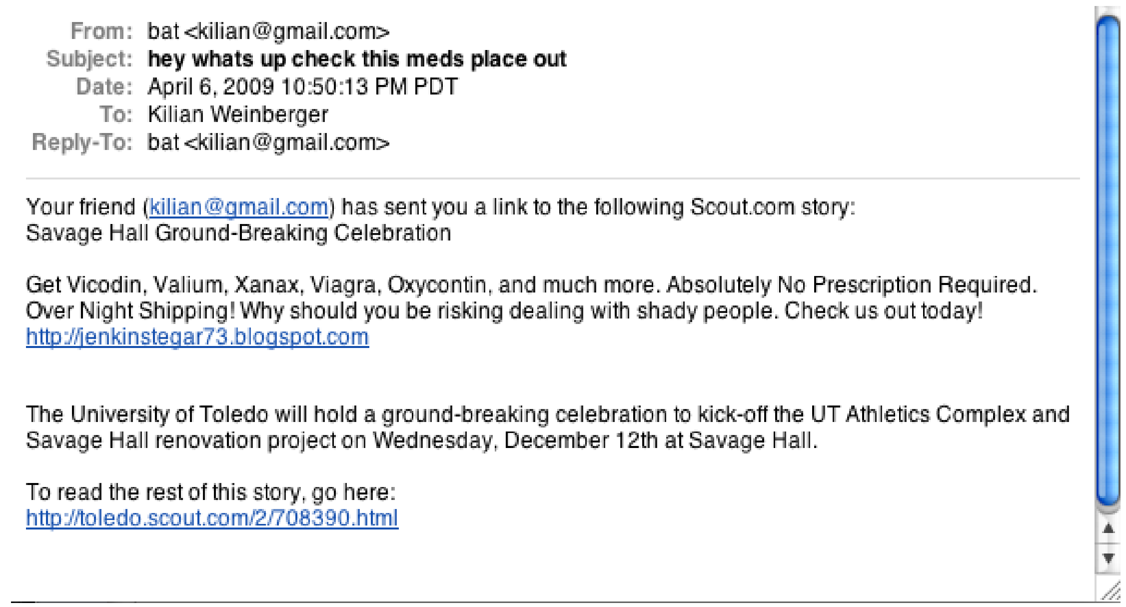
The University of Toledo will hold a ground-breaking celebration to kick-off the UT Athletics Complex and Savage Hall renovation project on Wednesday, December 12th at Savage Hall.

To read the rest of this story, go here:
http://toledo.scout.com/2/708390.html




Spam Classification



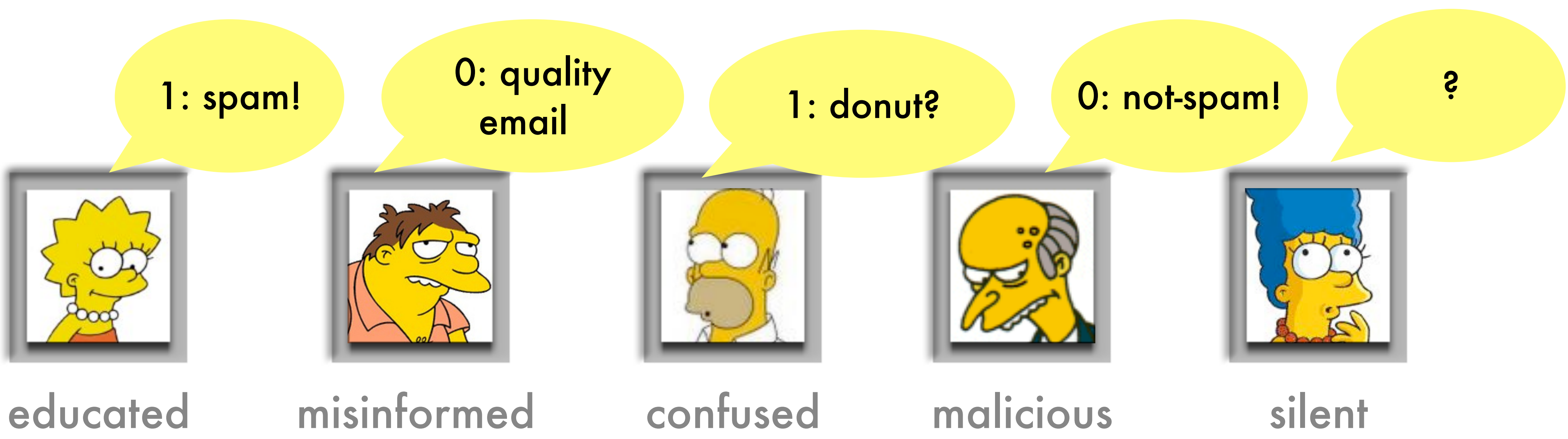
From: bat <kilian@gmail.com>
Subject: **hey whats up check this meds place out**
Date: April 6, 2009 10:50:13 PM PDT
To: Kilian Weinberger
Reply-To: bat <kilian@gmail.com>

Your friend (kilian@gmail.com) has sent you a link to the following Scout.com story:
Savage Hall Ground-Breaking Celebration

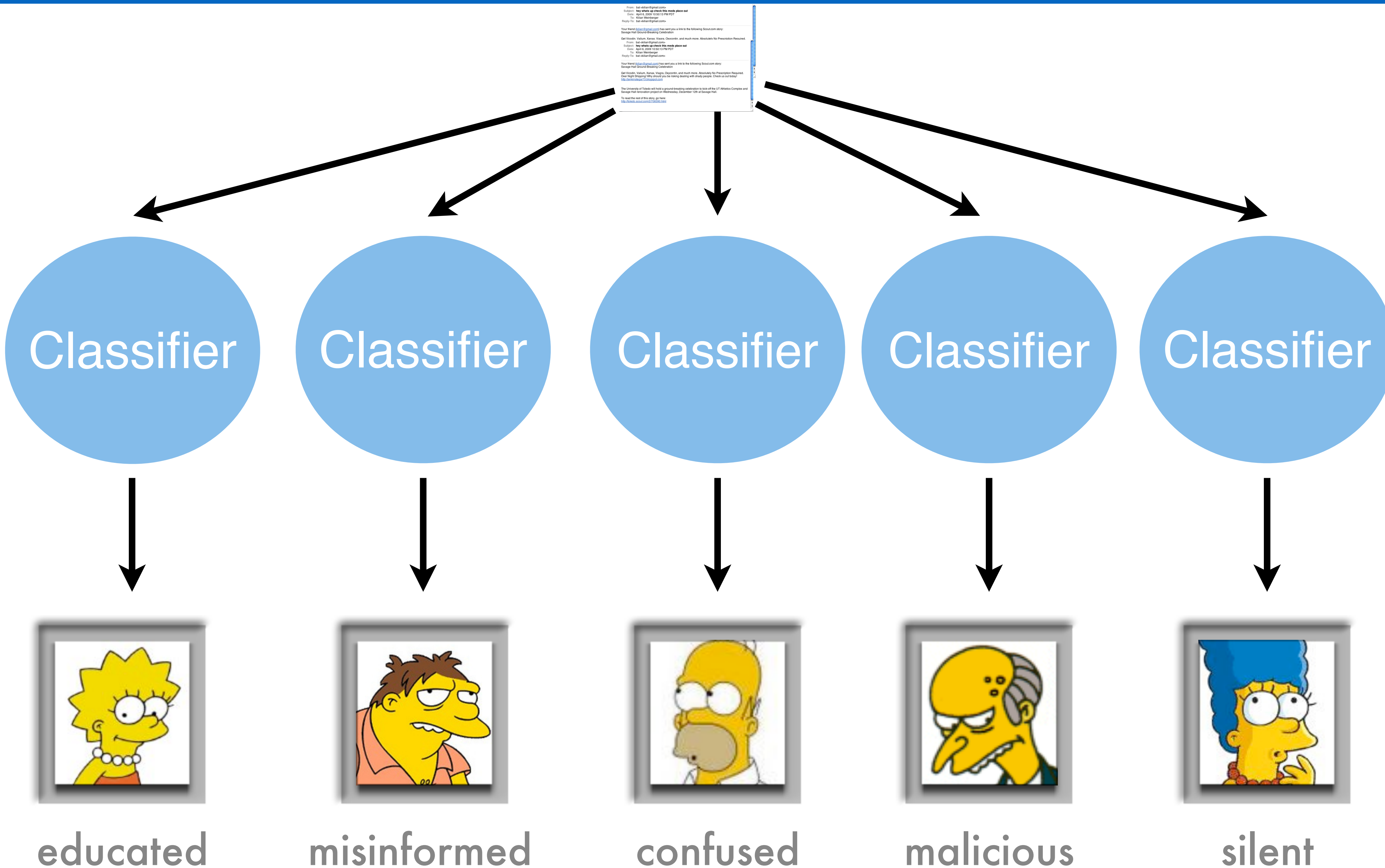
Get Vicodin, Valium, Xanax, Viagra, Oxycontin, and much more. Absolutely No Prescription Required. Over Night Shipping! Why should you be risking dealing with shady people. Check us out today!
http://jenkinstegar73.blogspot.com

The University of Toledo will hold a ground-breaking celebration to kick-off the UT Athletics Complex and Savage Hall renovation project on Wednesday, December 12th at Savage Hall.

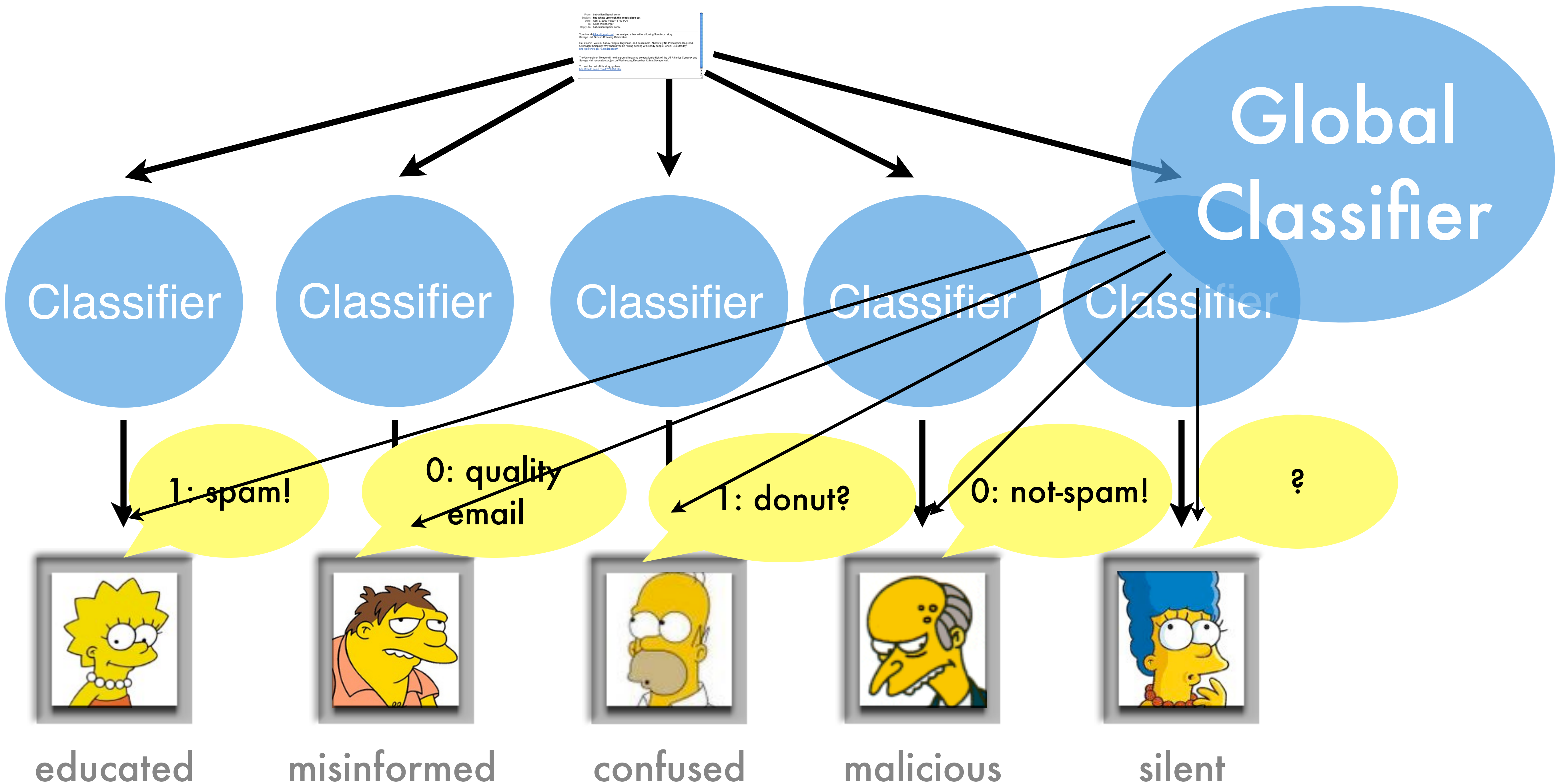
To read the rest of this story, go here:
http://toledo.scout.com/2/708390.html



Spam Classification



Spam Classification



Collaborative Classification

- **Primal representation**

$$f(x,u) = \langle \phi(x), w \rangle + \langle \phi(x), w_u \rangle = \langle \phi(x) \otimes (1 \oplus e_u), w \rangle$$

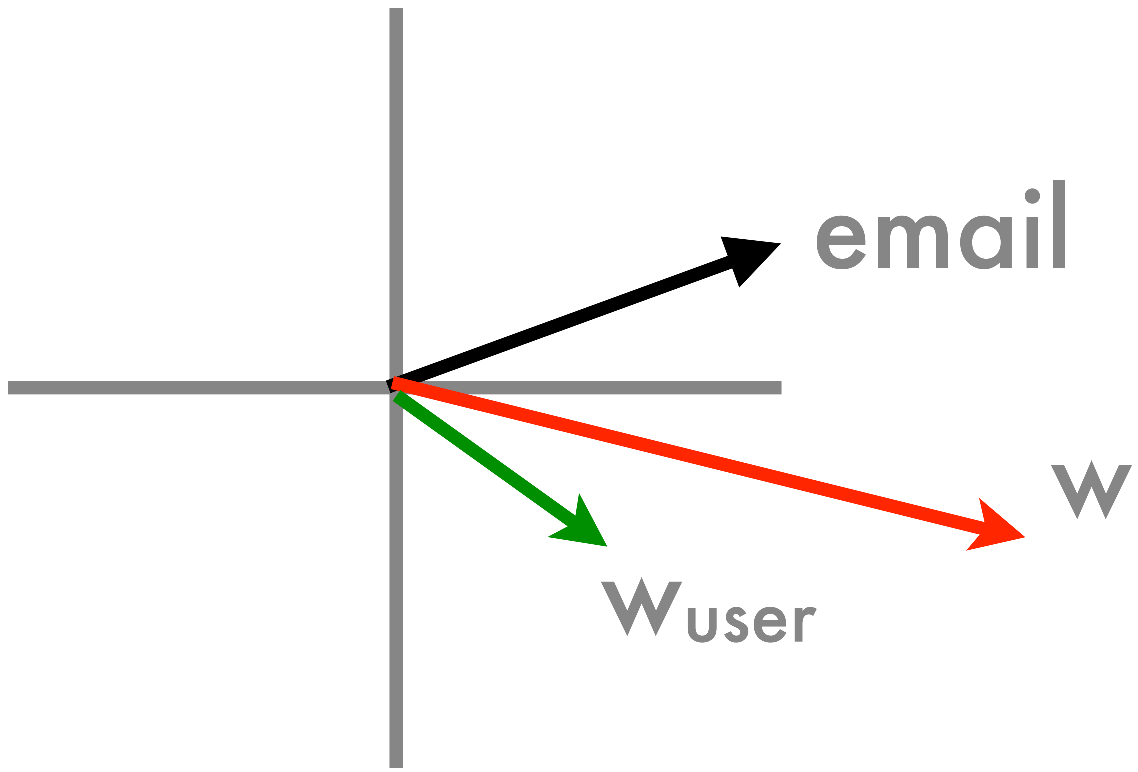
  Kernel representation

$$k((x,u),(x',u')) = k(x,x')[1 + \delta_{u,u'}]$$

  Multitask kernel (e.g. Pontil & Michelli, Daume). Usually does not scale well ...

- **Problem -** dimensionality is 10^{13}. That is 40TB of space

Collaborative Classification



- **Primal representation**

$$f(x, u) = \langle \phi(x), w \rangle + \langle \phi(x), w_u \rangle = \langle \phi(x) \otimes (1 \oplus e_u), w \rangle$$

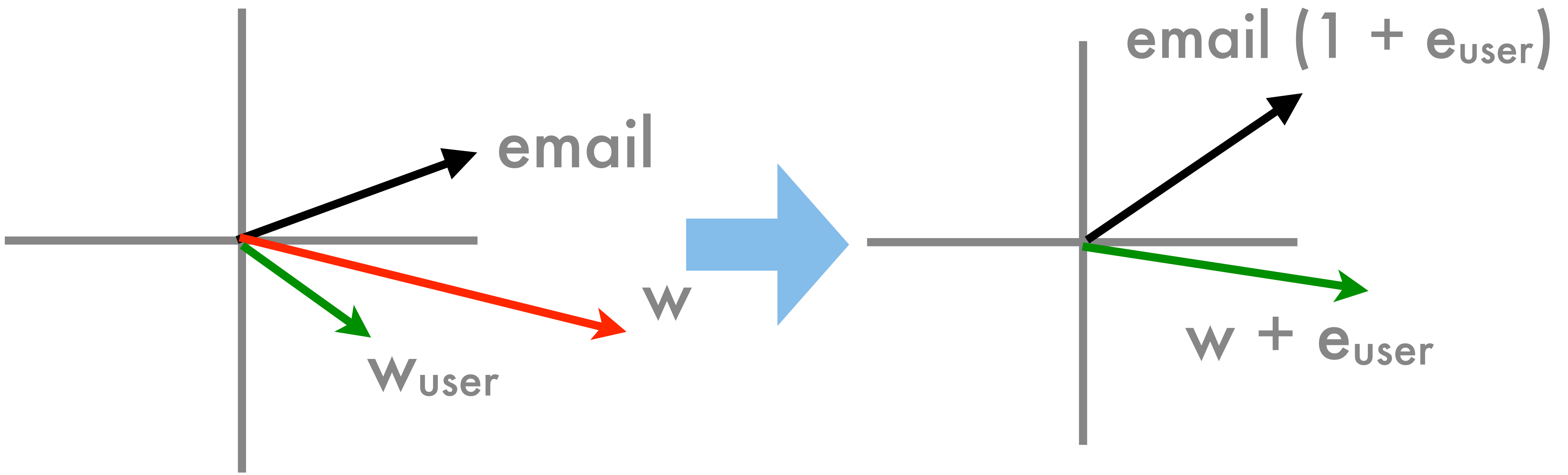
Kernel representation

$$k((x, u), (x', u')) = k(x, x')[1 + \delta_{u,u'}]$$

Multitask kernel (e.g. Pontil & Michelli, Daume). Usually does not scale well ...

- **Problem -** dimensionality is 10^{13}. That is 40TB of space

Collaborative Classification



- **Primal representation**

$$f(x,u) = \langle \phi(x), w \rangle + \langle \phi(x), w_u \rangle = \langle \phi(x) \otimes (1 \oplus e_u), w \rangle$$

Kernel representation

$$k((x,u),(x',u')) = k(x,x')[1 + \delta_{u,u'}]$$

Multitask kernel (e.g. Pontil & Michelli, Daume). Usually does not scale well ...

- **Problem -** dimensionality is 10^{13}. That is 40TB of space

Hash Kernels

Hash Kernels

instance:

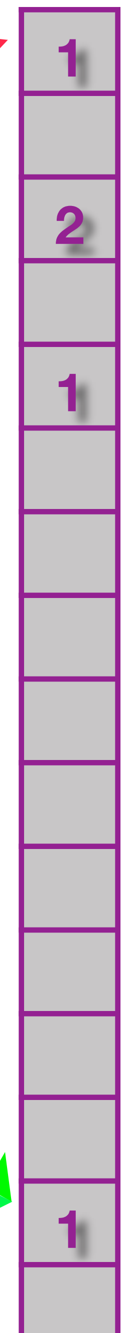
dictionary:

Hey,

please mention subtly during your talk that people should use Yahoo mail more often.
Thanks,

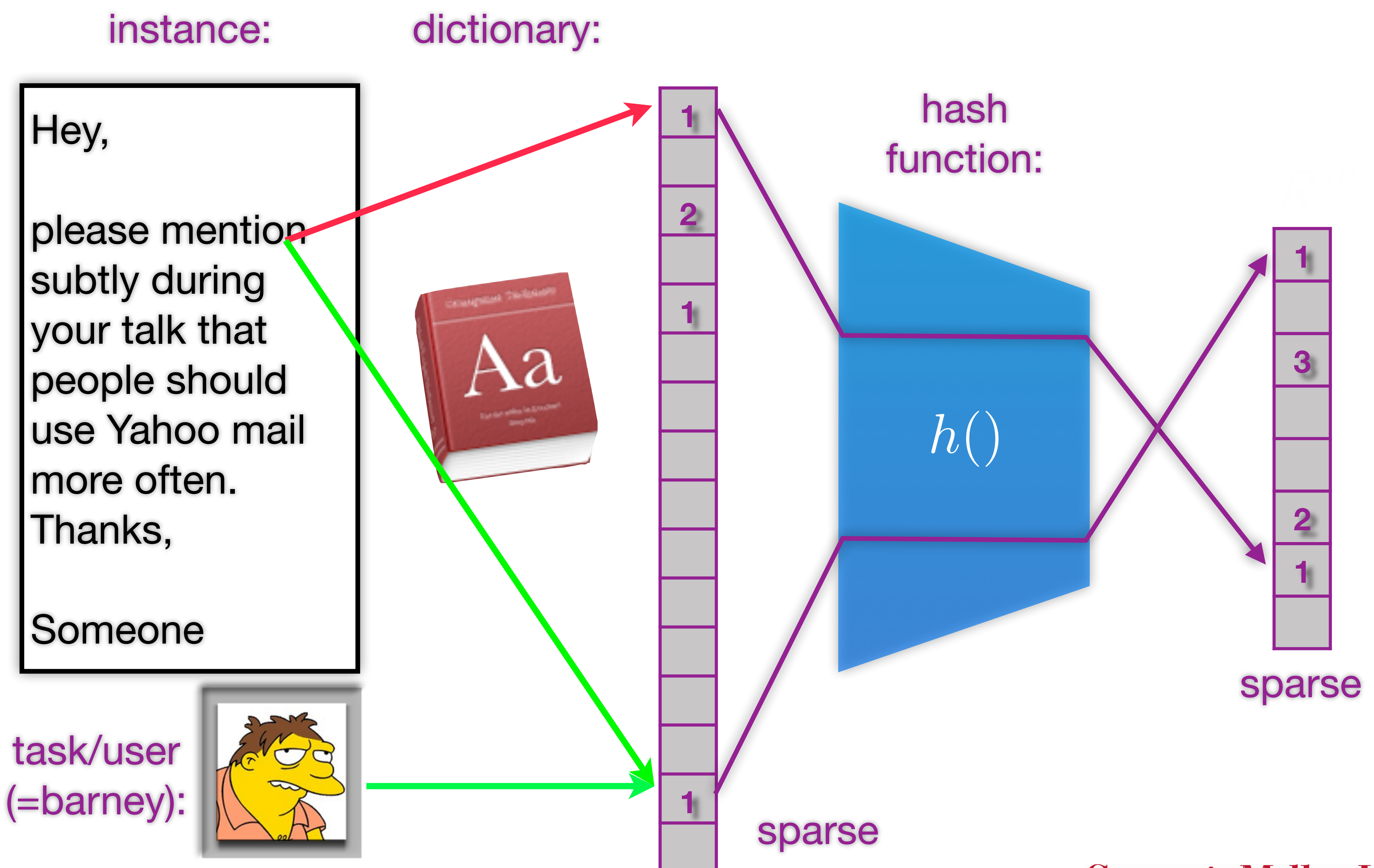
Someone

task/user (=barney):



sparse

Hash Kernels



Carnegie Mellon University

Hash Kernels

instance:

Hey,

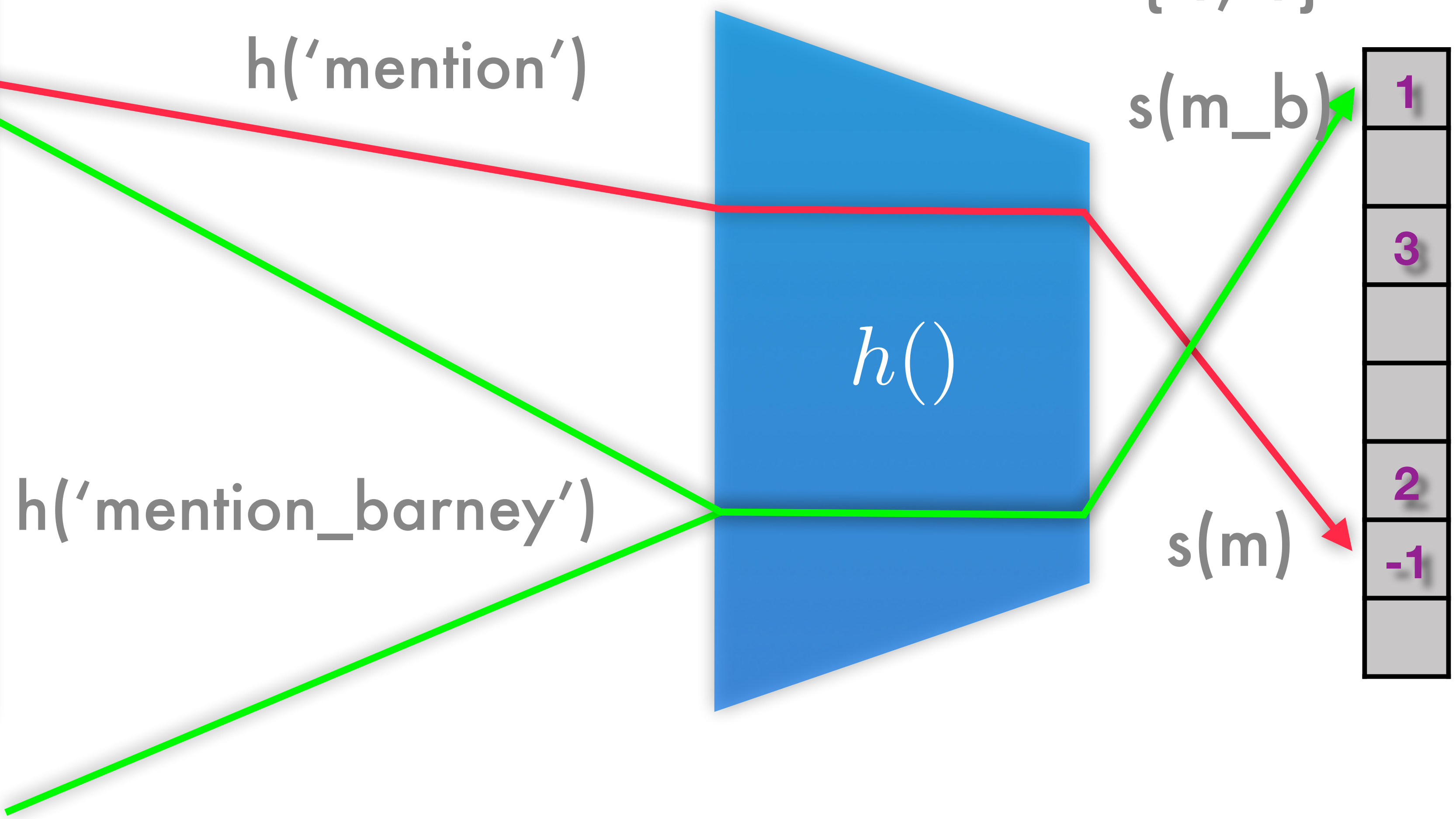
please mention subtly during your talk that people should use Yahoo mail more often.
Thanks,

Someone

task/user (=barney):

$$\sum_i \bar{w}[h(i)]\sigma(i)x_i$$

{-1, 1}



Similar to count hash
(Charikar, Chen, Farrach-Colton, 2003)

Advantages of hashing

- **No dictionary**!
  - Content drift is no problem
  - All memory used for classification
  - Finite memory guarantee (with online learning)
- **No Memory** needed for projection. (vs LSH)
- **Implicit** mapping into high dimensional space!
- It is **sparsity preserving**! (vs LSH)

Inner product preserving

- Unhashed inner product

$$\langle w, x \rangle = \sum_i w_i x_i$$

- Hashed inner product

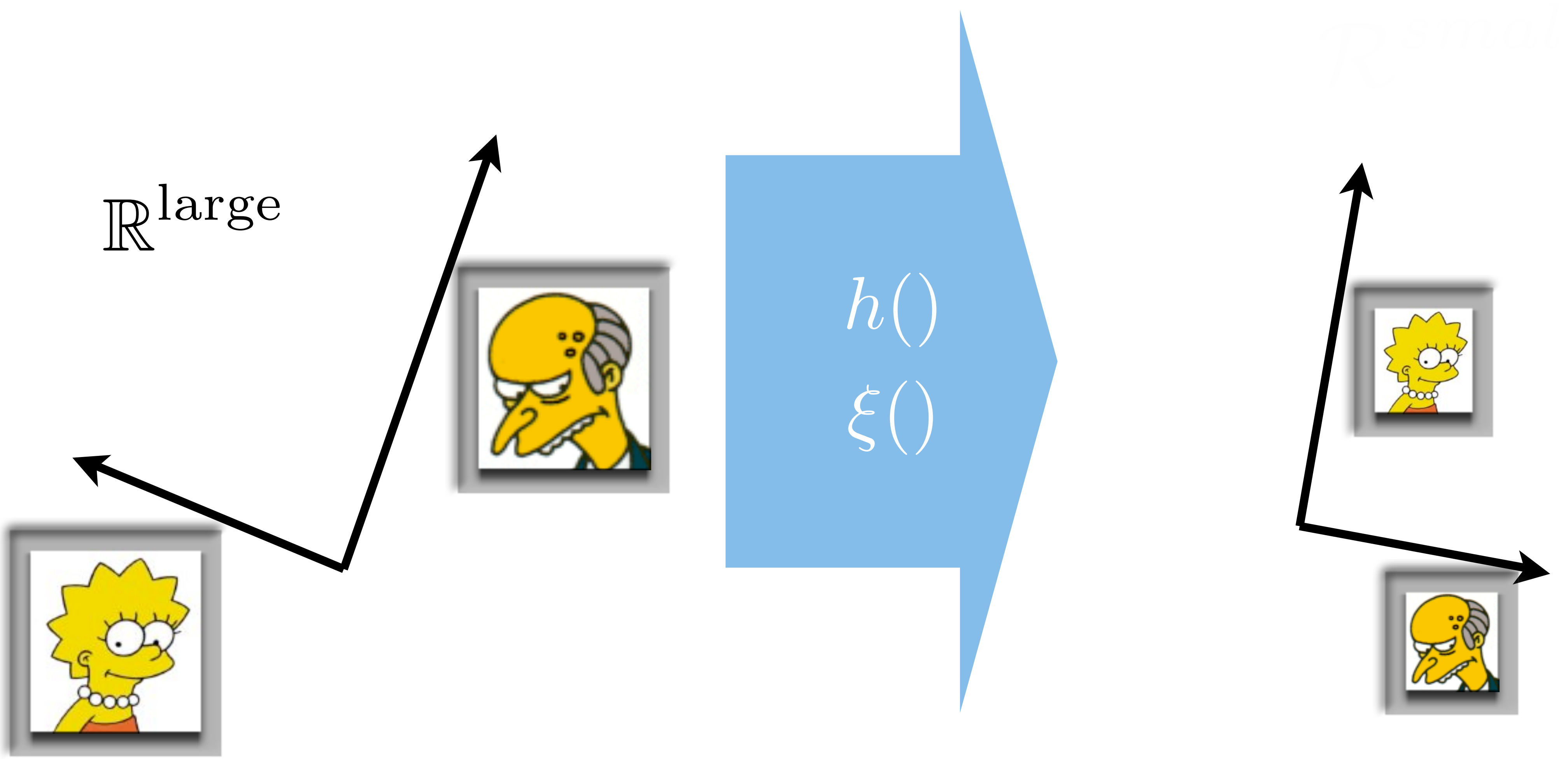
$$\langle \bar{w}, \bar{x} \rangle = \sum_j \left[\sum_{i:h(i)=j} w_i \sigma(i) \right] \left[\sum_{i:h(i)=j} x_i \sigma(i) \right]$$

- Taking expectations

$$\mathbf{E}_\sigma [\sigma(i)\sigma(i')] = \delta_{ii'}$$

hence inner product is preserved in expectation

Approximate Orthogonality



We can do multi-task learning!

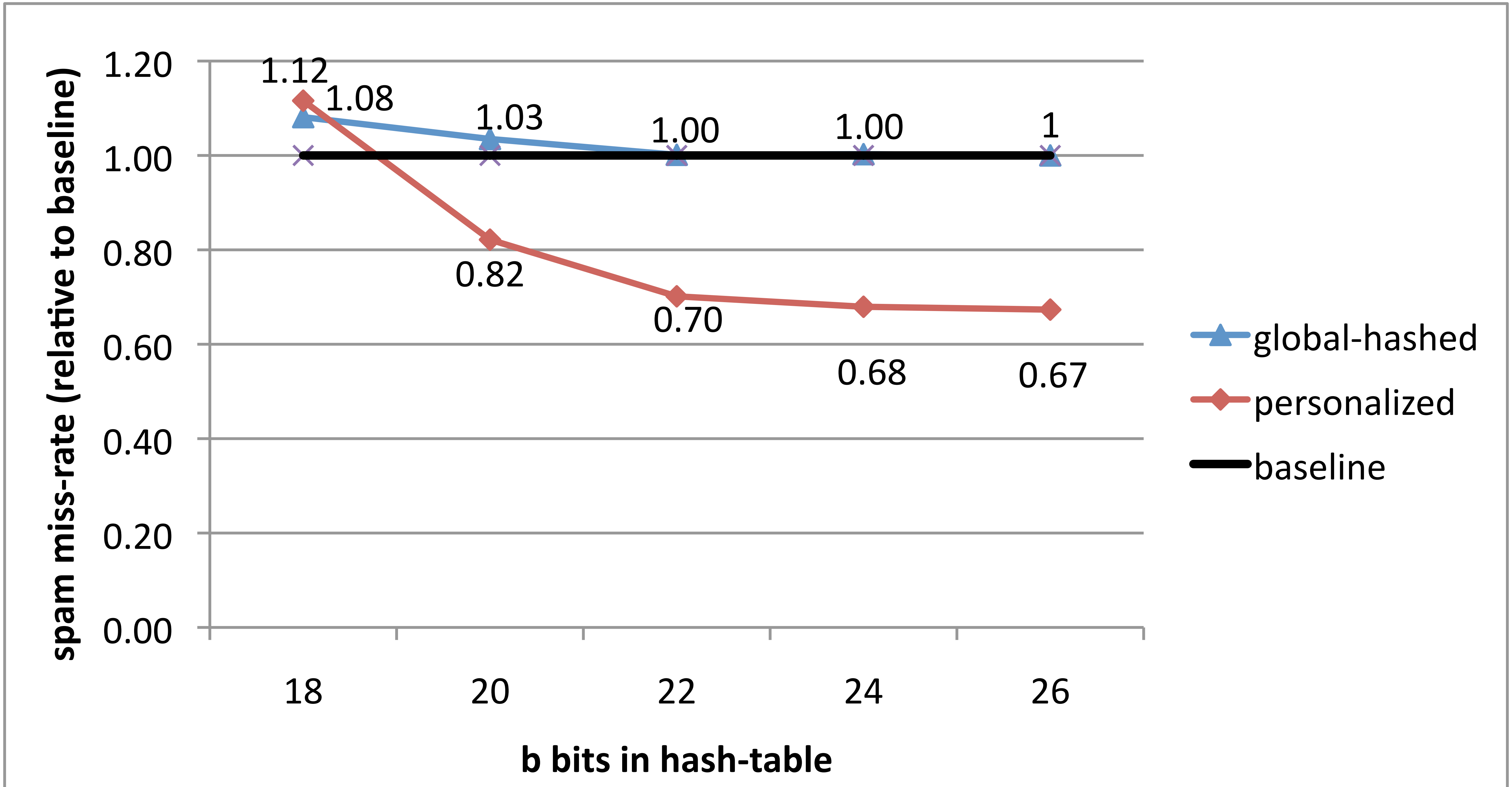
Guarantees

- For a random hash function the inner product vanishes with high probability via
$$\Pr\{|\langle w_v, h_u(x)\rangle| > \epsilon\} \leq 2e^{-C\epsilon^2 m}$$
- We can use this for multitask learning

Direct sum in Hilbert Space → Sum in Hash Space

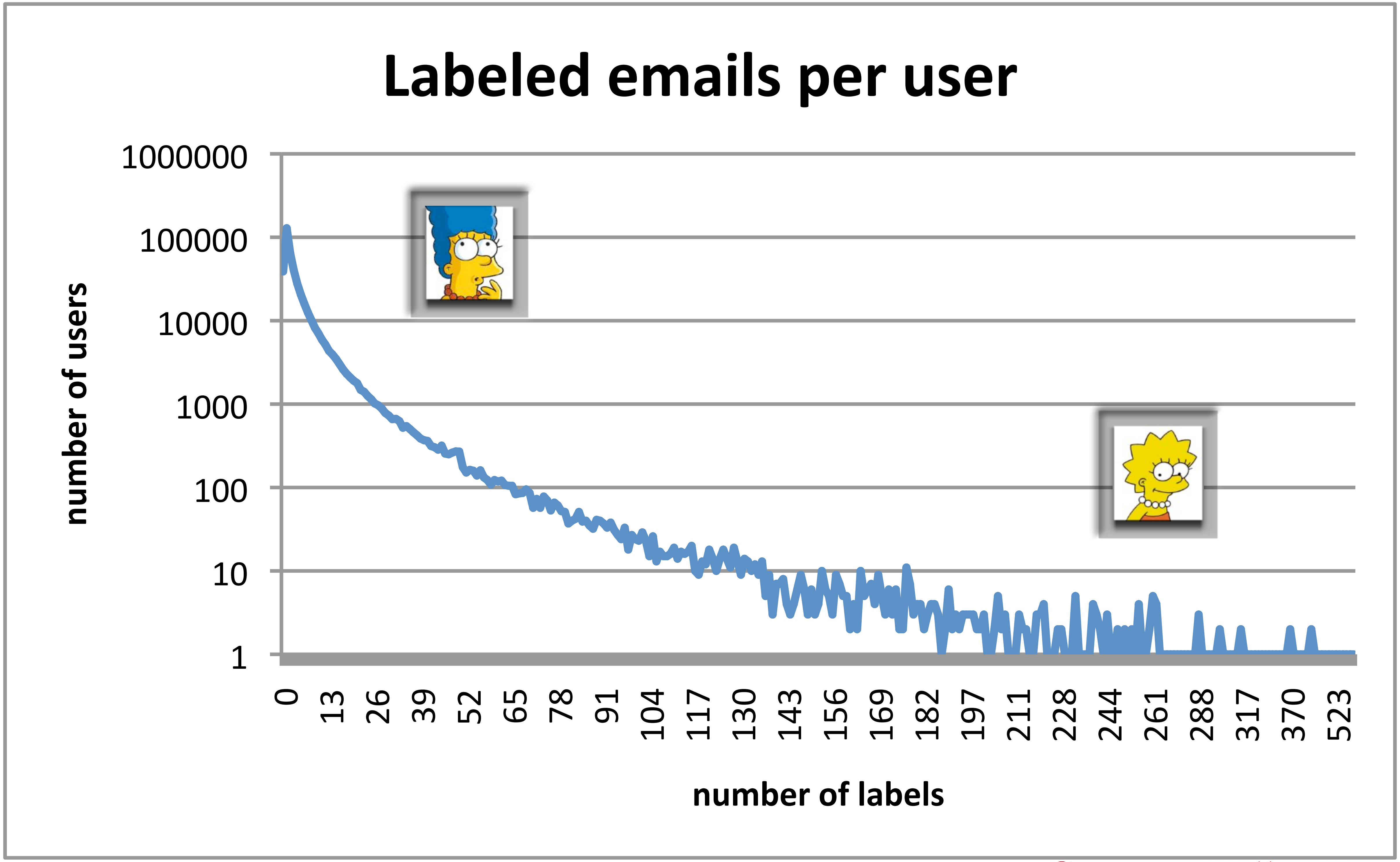
- The hashed inner product is unbiased
  Proof: take expectation over random signs
- The variance is O(1/n). Proof by brute force expansion
- Restricted isometry property (Kumar, Sarlos, Dasgupta ’10)

Spam classification results



N=20M, U=400K

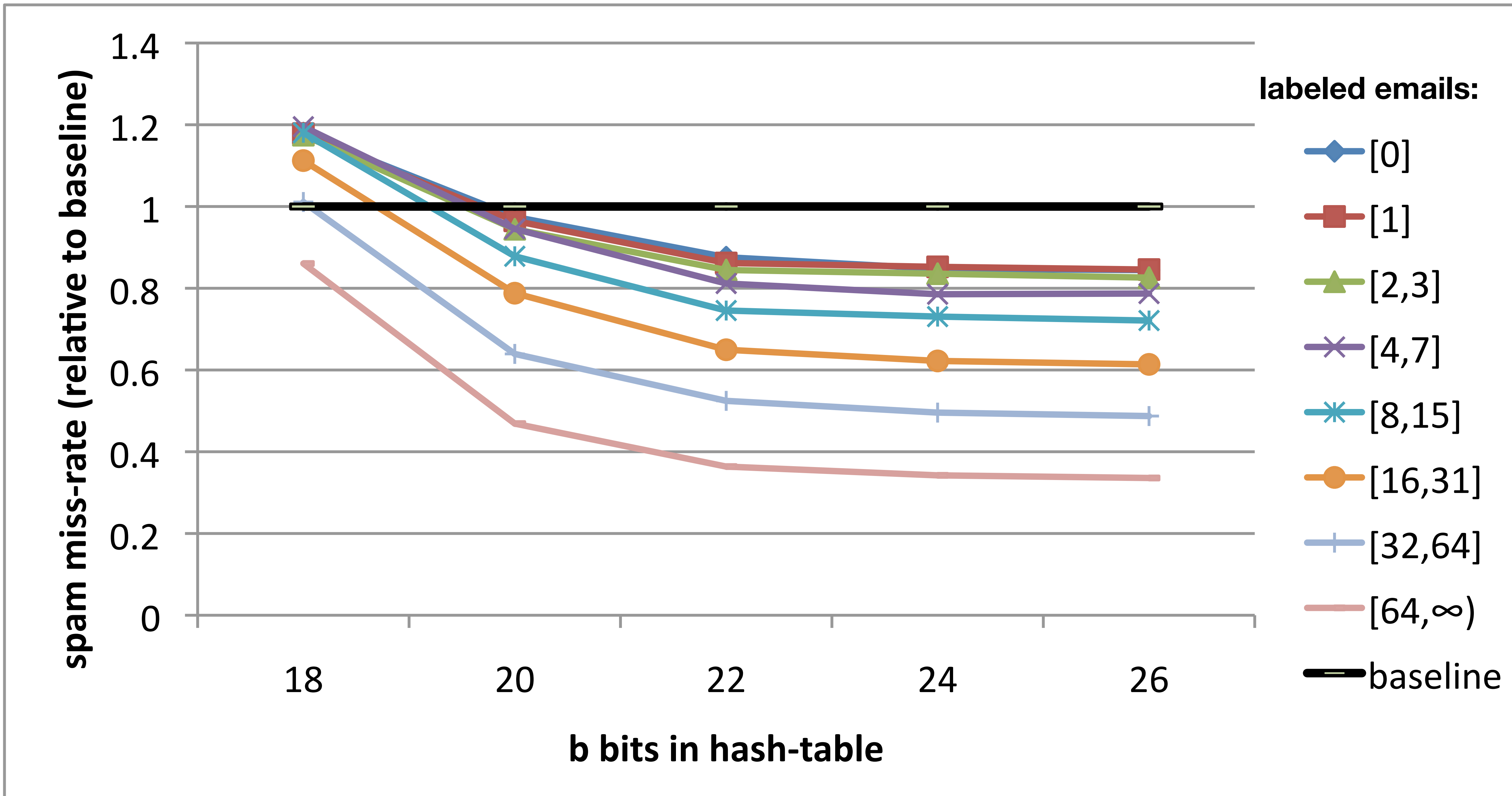
Lazy users ...



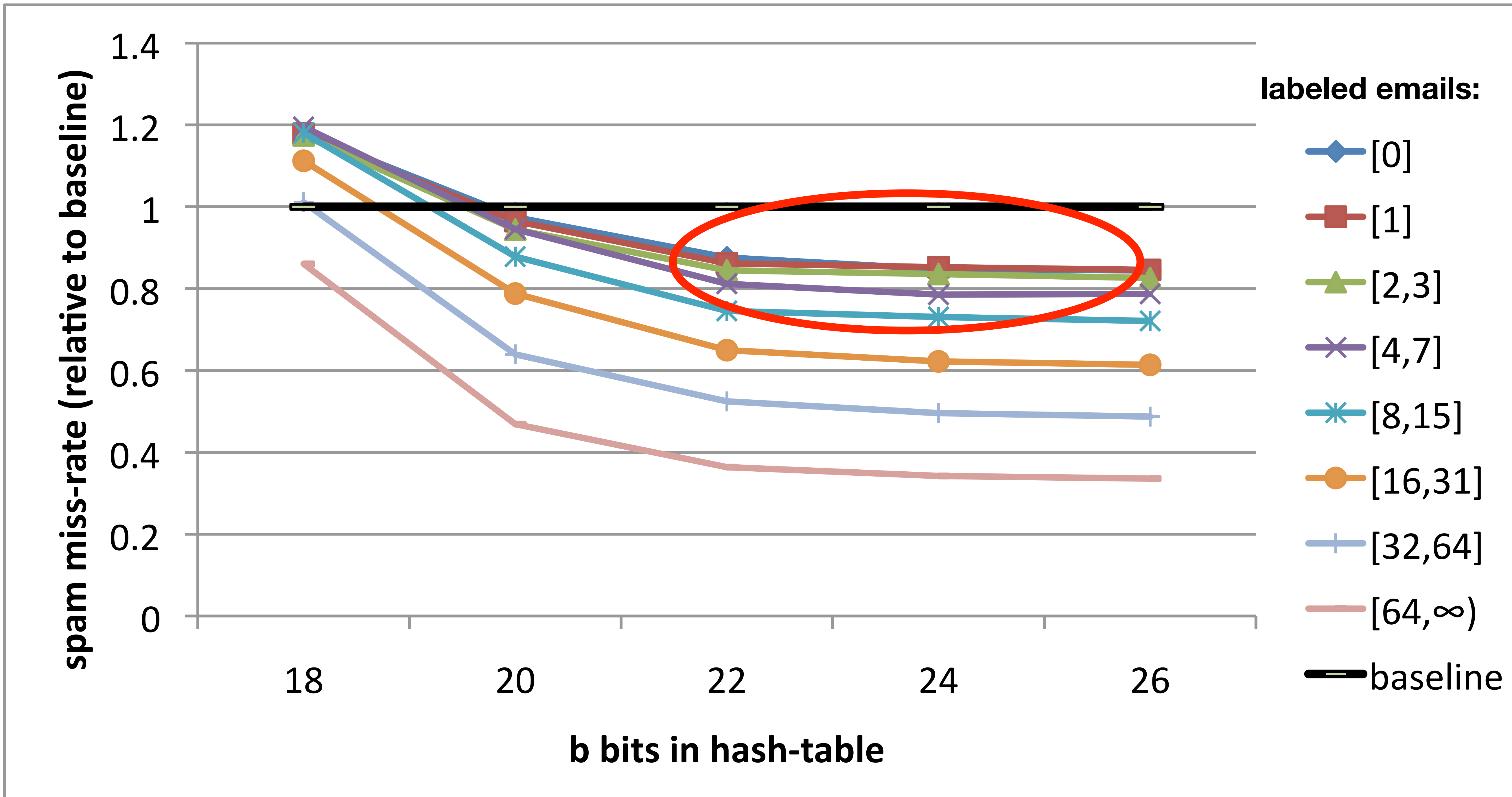
Carnegie Mellon University

Results by user group

Results by user group



Results by user group



Approximate String Matches

- General idea

$$k(x, x') = \sum_{w \in x} \sum_{w' \in x'} \kappa(w, w') \text{ for } |w - w'| \leq \delta$$

- `Pittsburgh`
- `P1ttsburgh`
- `Pitsburgh`
- `Pittsburg`
- `Pittsbrugh`

catch all

Approximate String Matches

- General idea

$$k(x, x') = \sum_{w \in x} \sum_{w' \in x'} \kappa(w, w') \text{ for } |w - w'| \leq \delta$$

- Simplification
  - Weigh by mismatch amount |w-w'|
  - Map into fragments: `dog -> (*og, d*g, do*)`
  - Hash fragments and weigh them based on mismatch amount
  - Exponential in amount of mismatch
    But not in alphabet size

Approximate String Matches

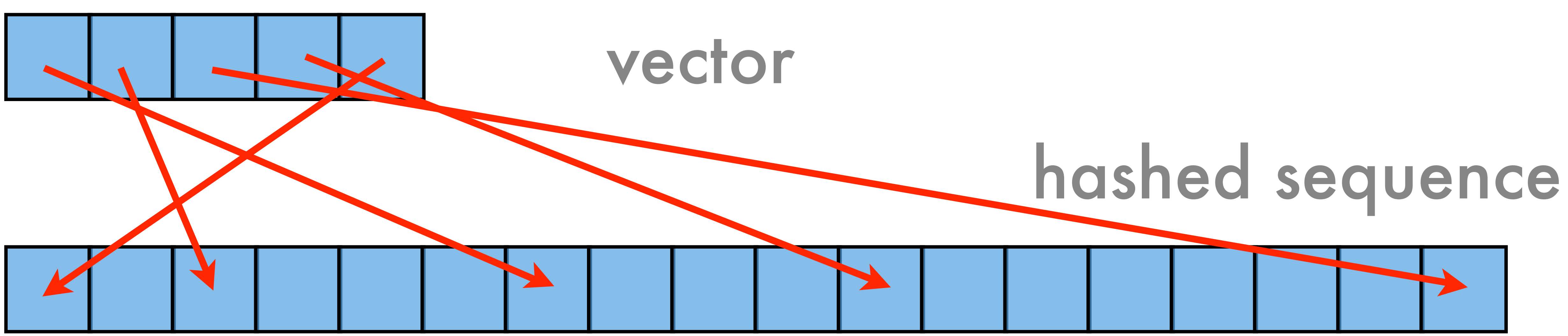
- General idea

$$k(x, x') = \sum_{w \in x} \sum_{w' \in x'} \kappa(w, w') \text{ for } |w - w'| \leq \delta$$

- `Pittsburgh`
- `P*ttsburgh`
- `Pi*tsburgh`
- `Pit*sburgh`
- `Pitt*burgh`
- ...

Memory access patterns

- Cache size is a few MBs
  Very fast random memory access
- RAM (DDR3 or better) is GBs
  - Fast sequential memory access (burst read)
  - CPU caches memory read from RAM
  - Random memory access is very slow
  - CPU caches memory read from RAM



Speeding up access

for j=1 to n access h(i,j)

- Key idea - bound the range of h(i,j)
- Linear offset
  bad collisions in i
  $$h(i,j) = h(i) + j$$
- Sum of hash functions
  bad collisions in j
  $$h(i,j) = h(i) + h'(j)$$
- Optimal Golomb Ruler (Langford)
  NP hard in general
  $$h(i,j) = h(i) + \mathrm{OGR}(j)$$
- Feistel Network / Cryptography
  $$h(i,j) = h(i) + \mathrm{crypt}(j|i)$$