Photo by Arthur Gretton, CMU Machine Learning Protestors at G20

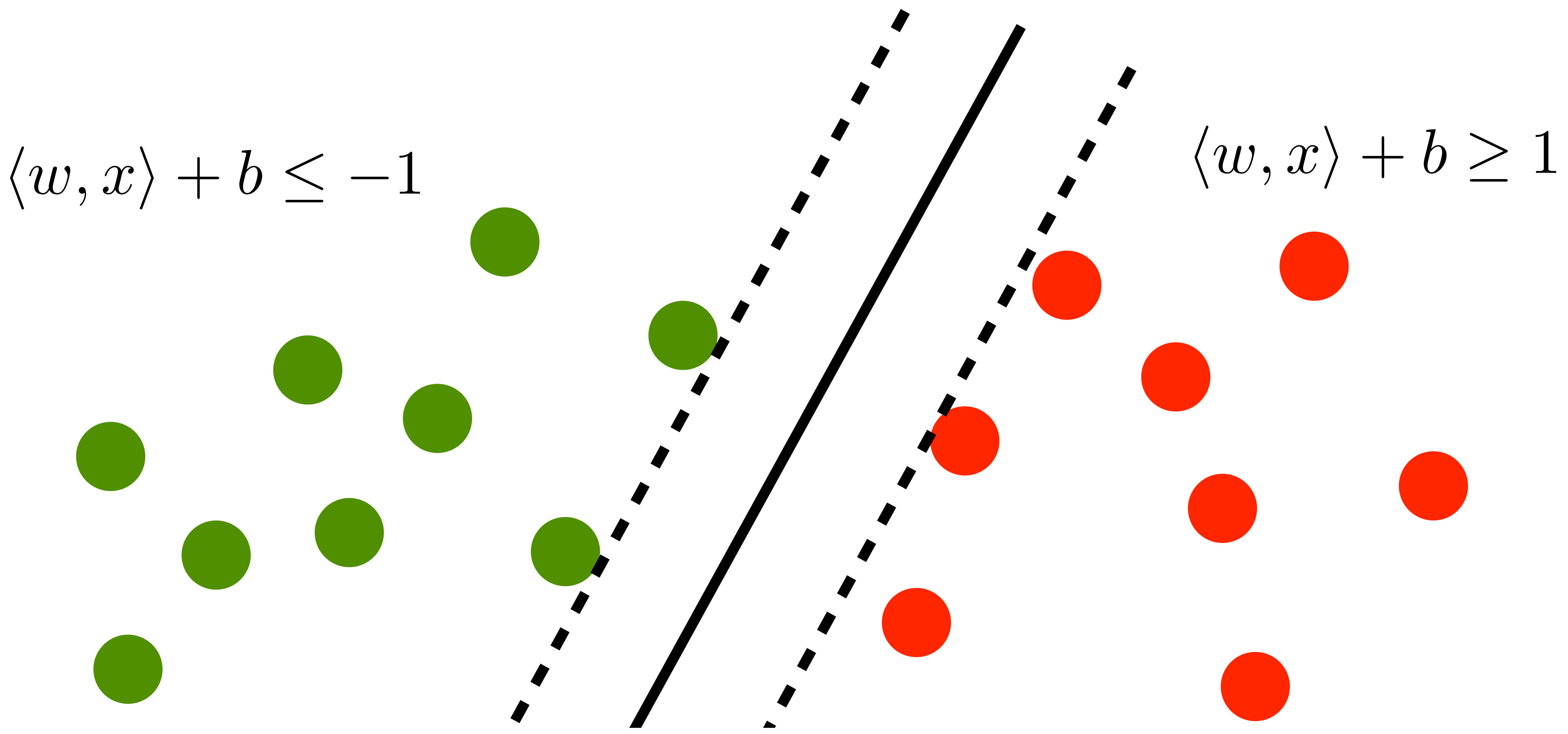
# BBM406 Fundamentals of Machine Learning

## Lecture 16: Soft Margin Classification Multi-class SVMs Kernels

HACETTEPE UNIVERSITY COMPUTER VISION LAB

Aykut Erdem // Hacettepe University // Fall 2019

# Last time… **Support Vector Machines**

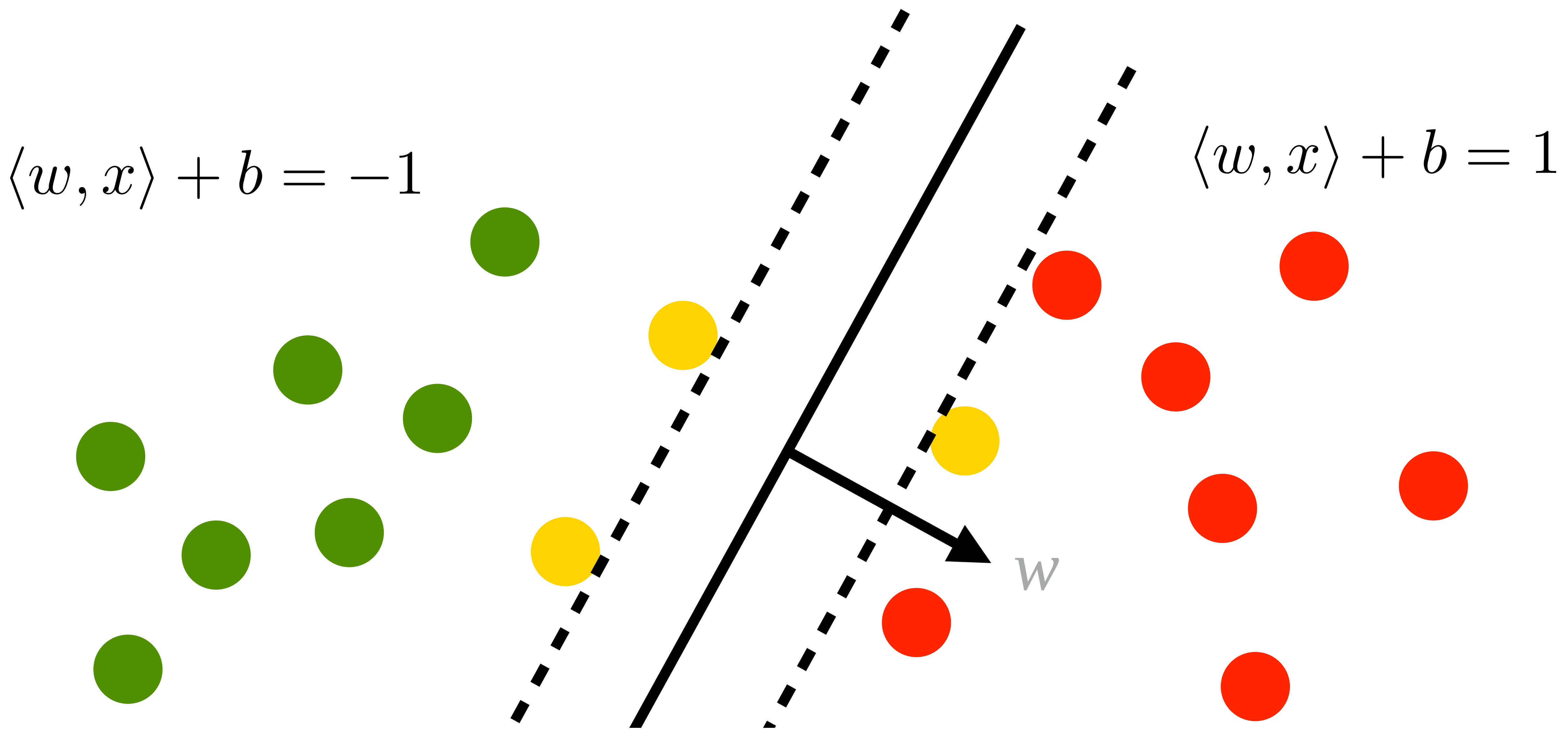

$\langle w, x\rangle + b \le -1$
$\langle w, x\rangle + b \ge 1$

linear function

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

slide by Alex Smola

# Last time... **Support Vector Machines**

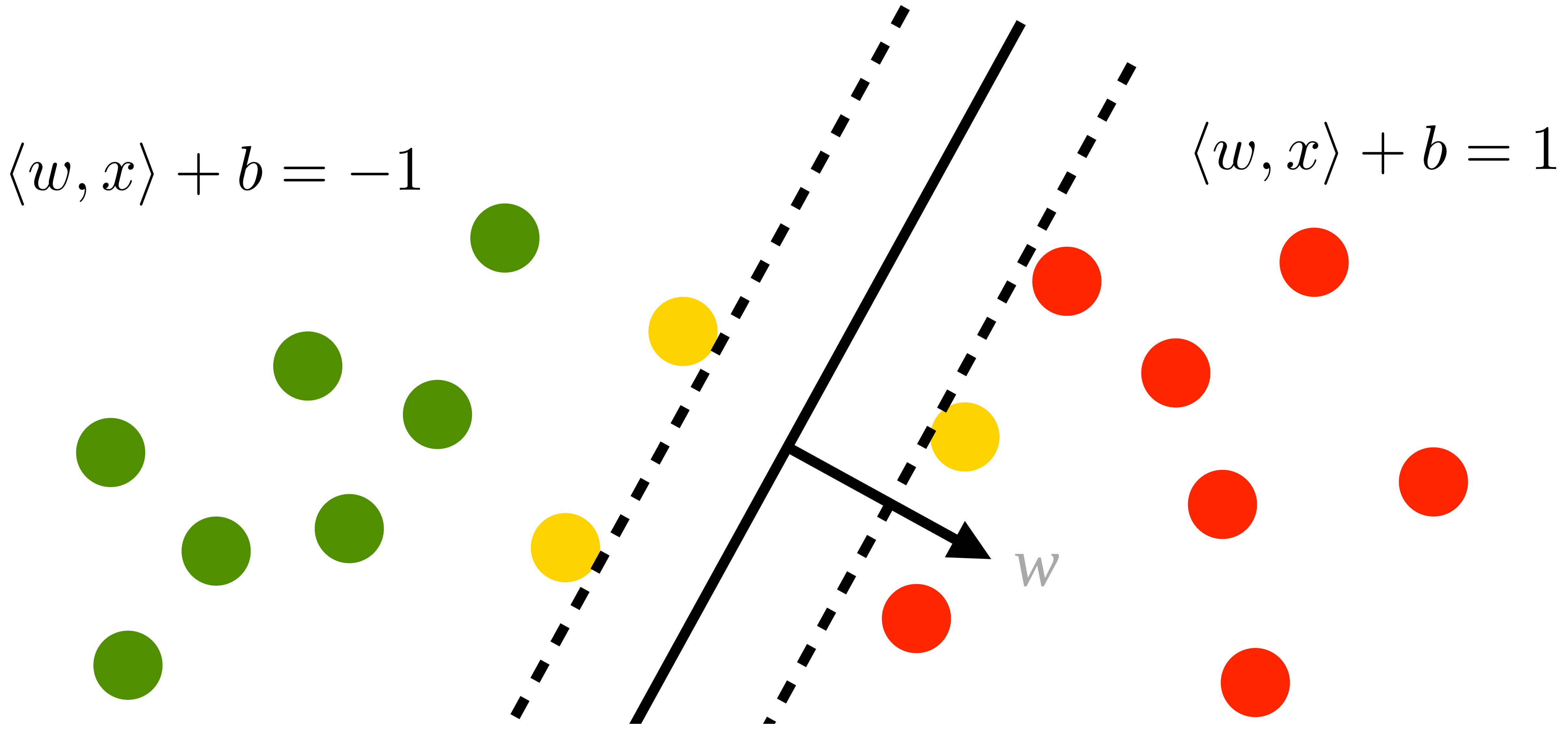


optimization problem

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

slide by Alex Smola

# Last time... **Support Vector Machines**


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

optimization problem

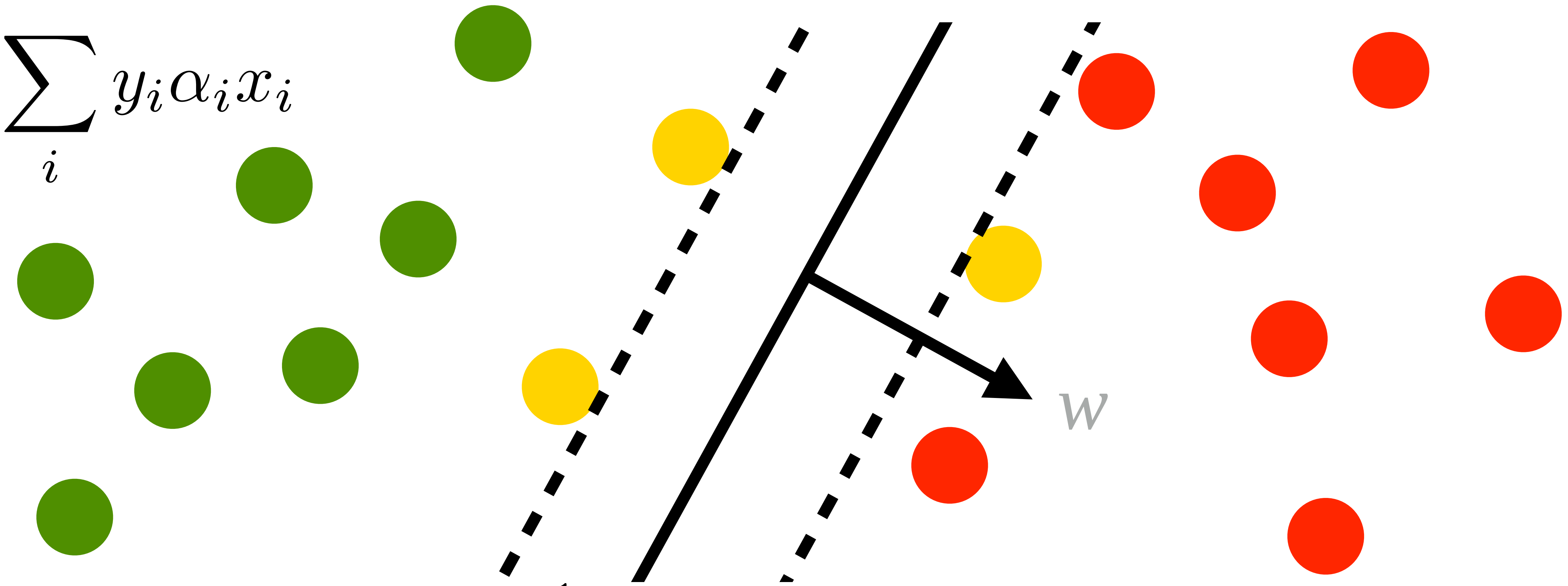
$$\underset{w,b}{\text{minimize}} \frac{1}{2} \|w\|^2 \text{ subject to } y_i \left[\langle x_i, w \rangle + b\right] \geq 1$$

slide by Alex Smola

# Last time... **Support Vector Machines**

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

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

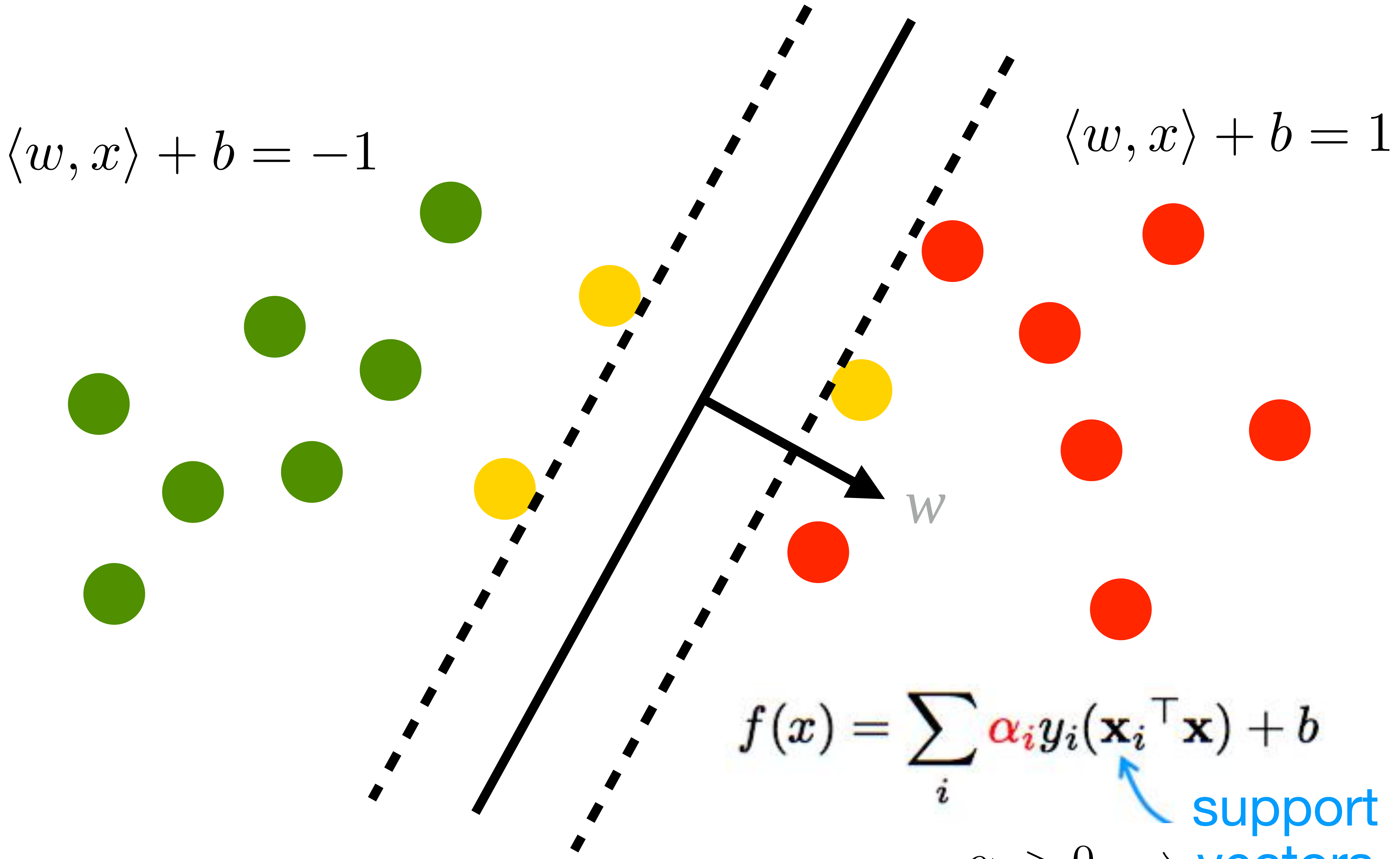


$$\underset{\alpha}{\text{maximize}} -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle + \sum_i \alpha_i$$

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

slide by Alex Smola

# Last time... **Large Margin Classifier**


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

$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

$\alpha_i > 0 \implies$ support vectors

## Today

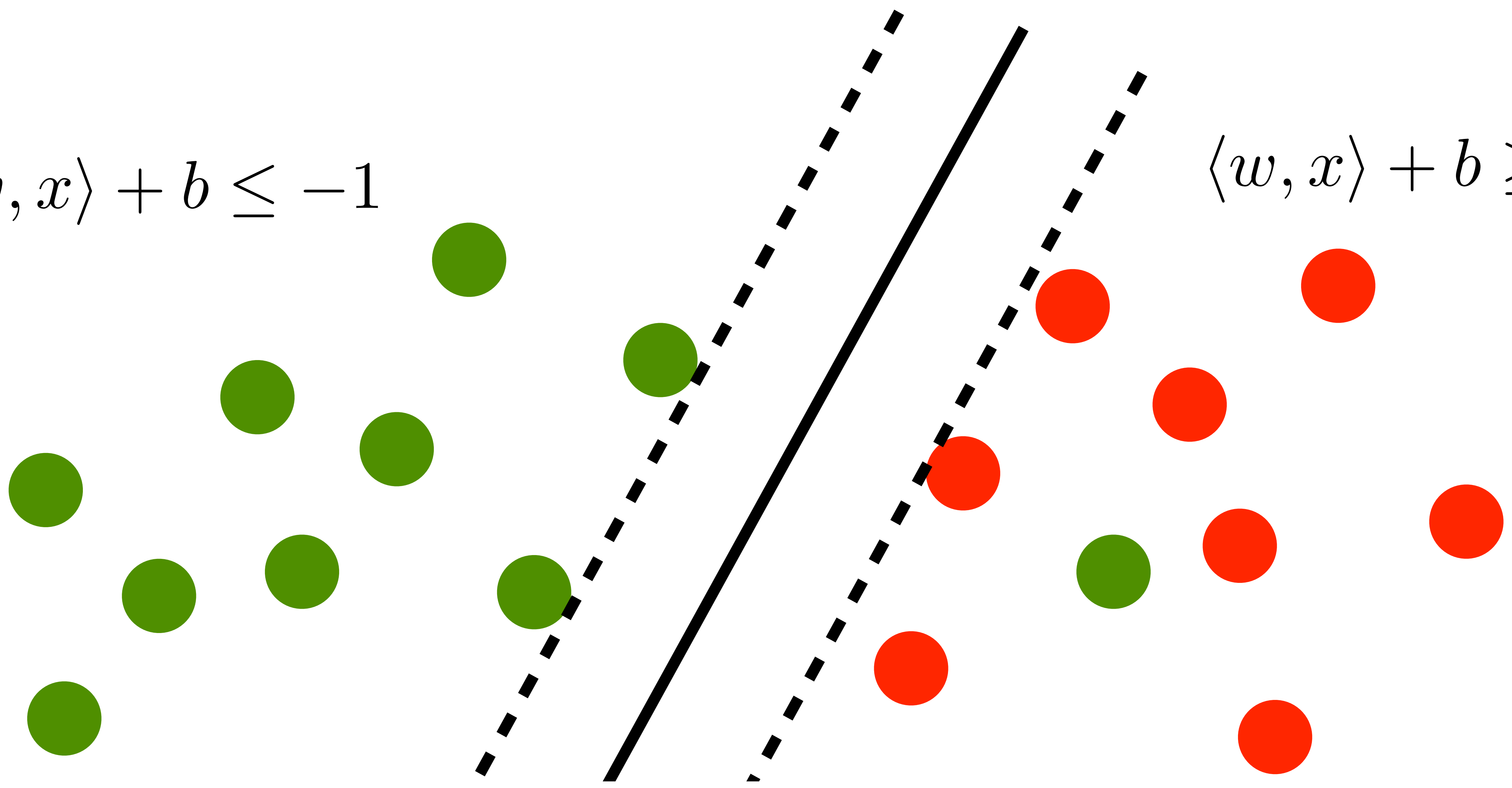
- Soft margin classification
- Multi-class classification
- Introduction to kernels

## Soft Margin Classification

# Large Margin Classifier

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

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



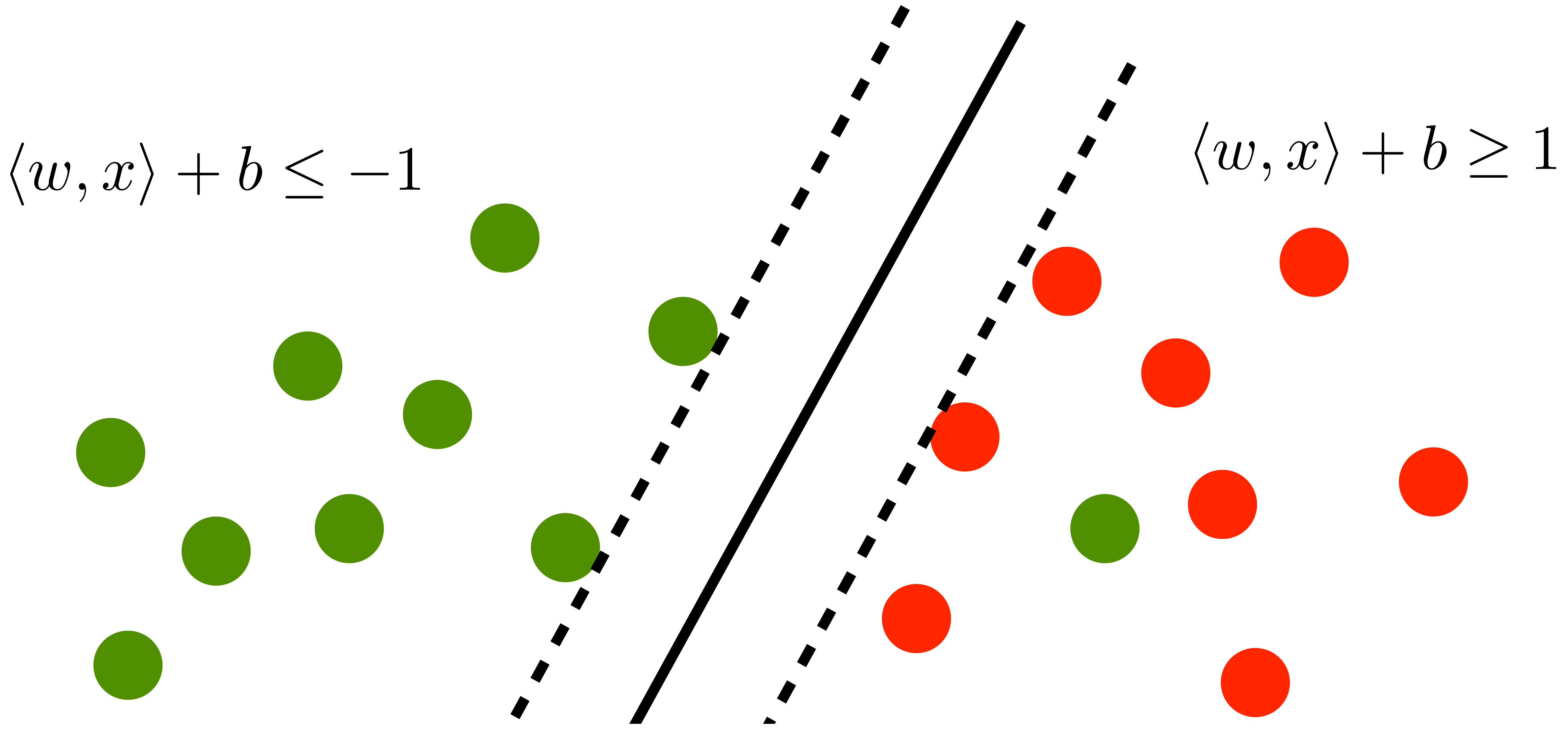
linear function

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

linear separator is impossible

slide by Alex Smola

# Large Margin Classifier


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

minimum error separator is impossible

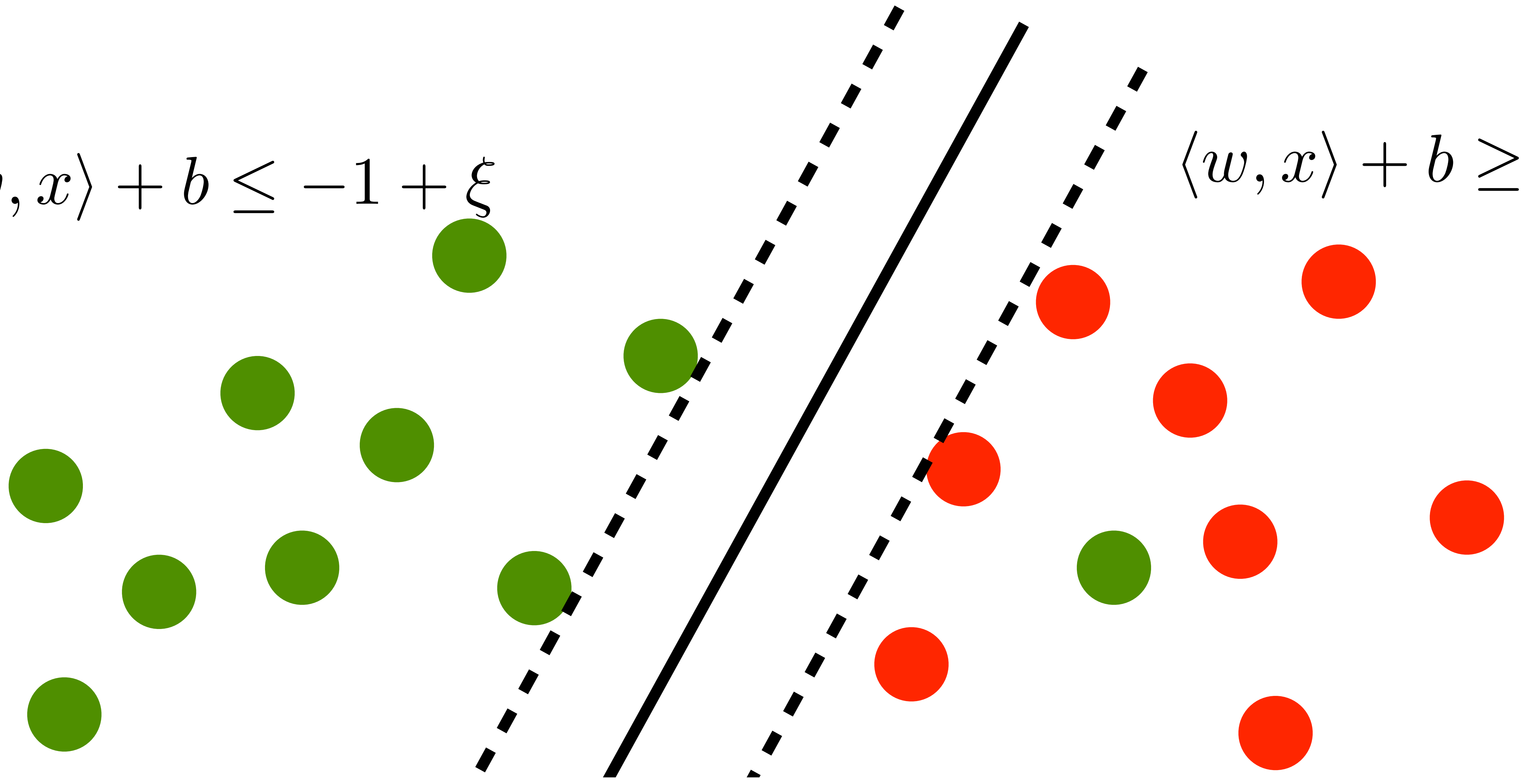
Theorem (Minsky & Papert)
Finding the minimum error separating hyperplane is NP hard

slide by Alex Smola

# Adding Slack Variables

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

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



minimize amount of slack

Convex optimization problem

slide by Alex Smola

# 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)

slide by Alex Smola

# Adding Slack Variables

- Hard margin problem

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

- With slack variables

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

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

Problem is always feasible. Proof:
$w = 0 \text{ and } b = 0 \text{ and } \xi_i = 1$ (also yields upper bound)

slide by Alex Smola

# 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 $w, b, \xi$ is at saddle point with $\alpha, \eta$

- Derivatives in $w, b, \xi$ need to vanish

slide by Alex Smola

# 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

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

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

bound influence

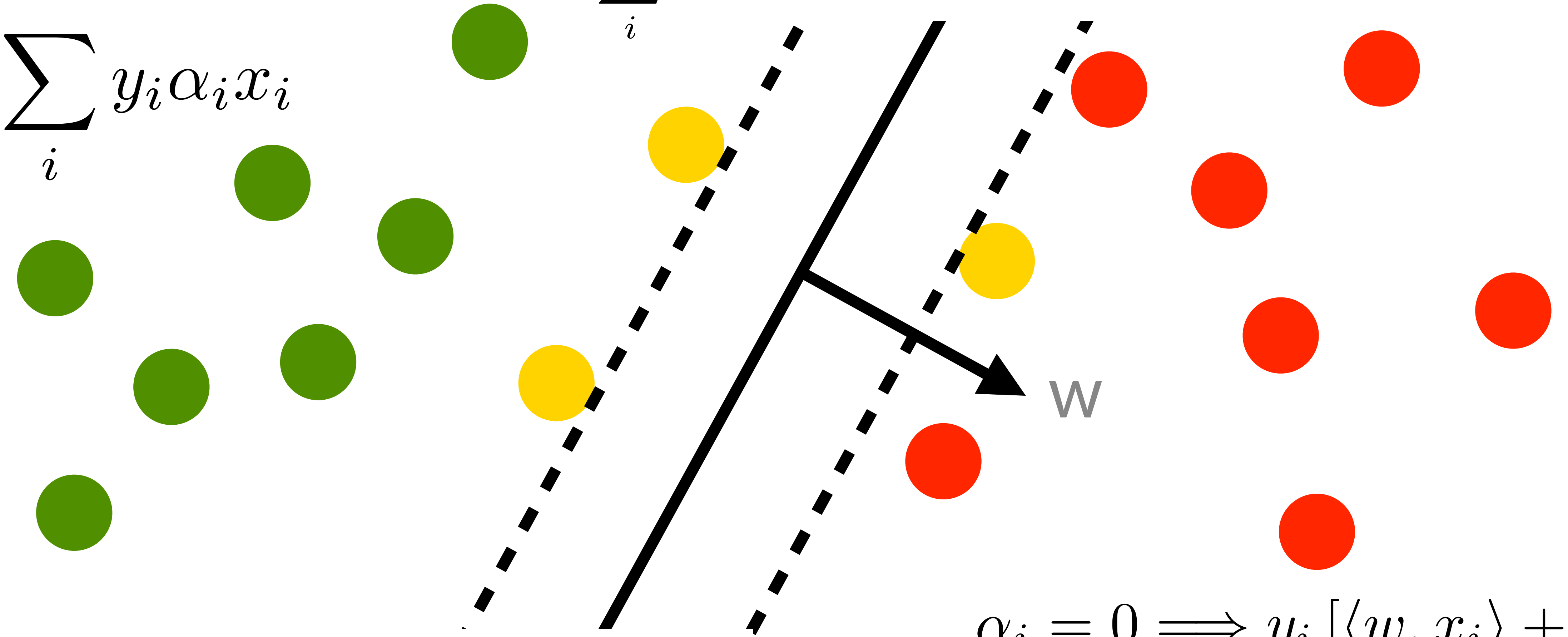
slide by Alex Smola

# Karush Kuhn Tucker Conditions

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

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

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



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

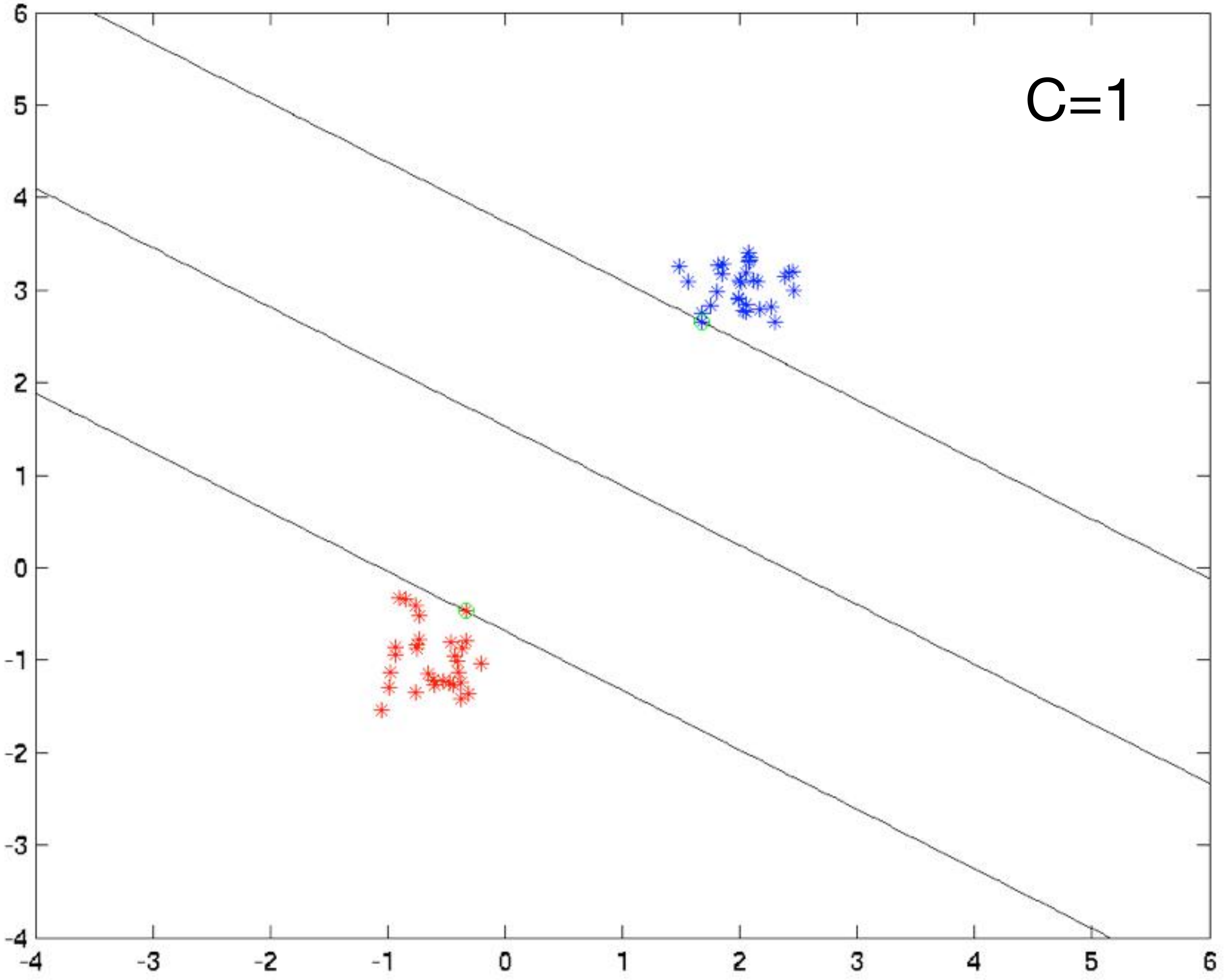
$$\alpha_i\left[y_i\left[\langle w, x_i\rangle + b\right] + \xi_i - 1\right] = 0$$

$$\eta_i\xi_i = 0$$

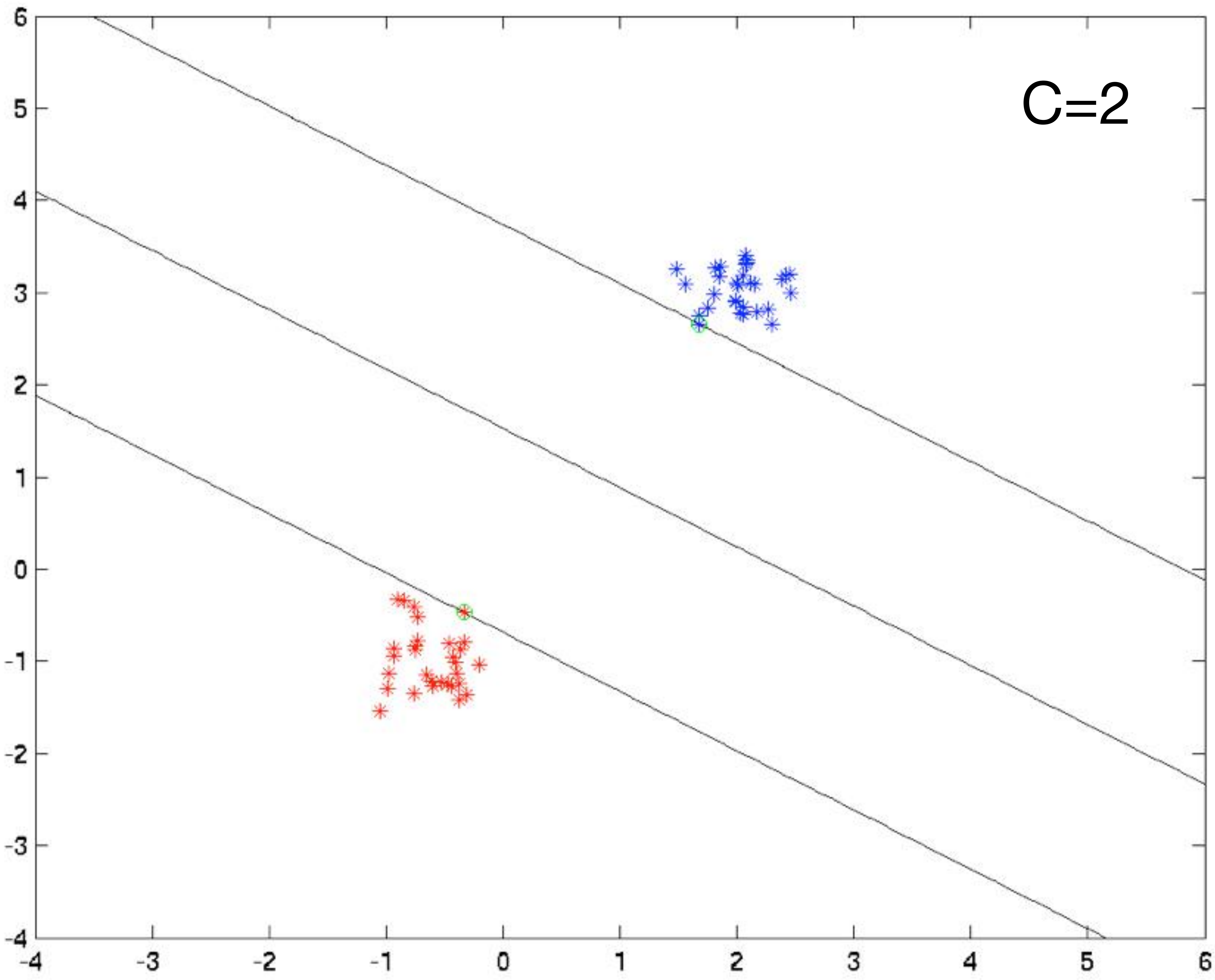
$$0 < \alpha_i < C \Longrightarrow y_i\left[\langle w, x_i\rangle + b\right] = 1$$

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

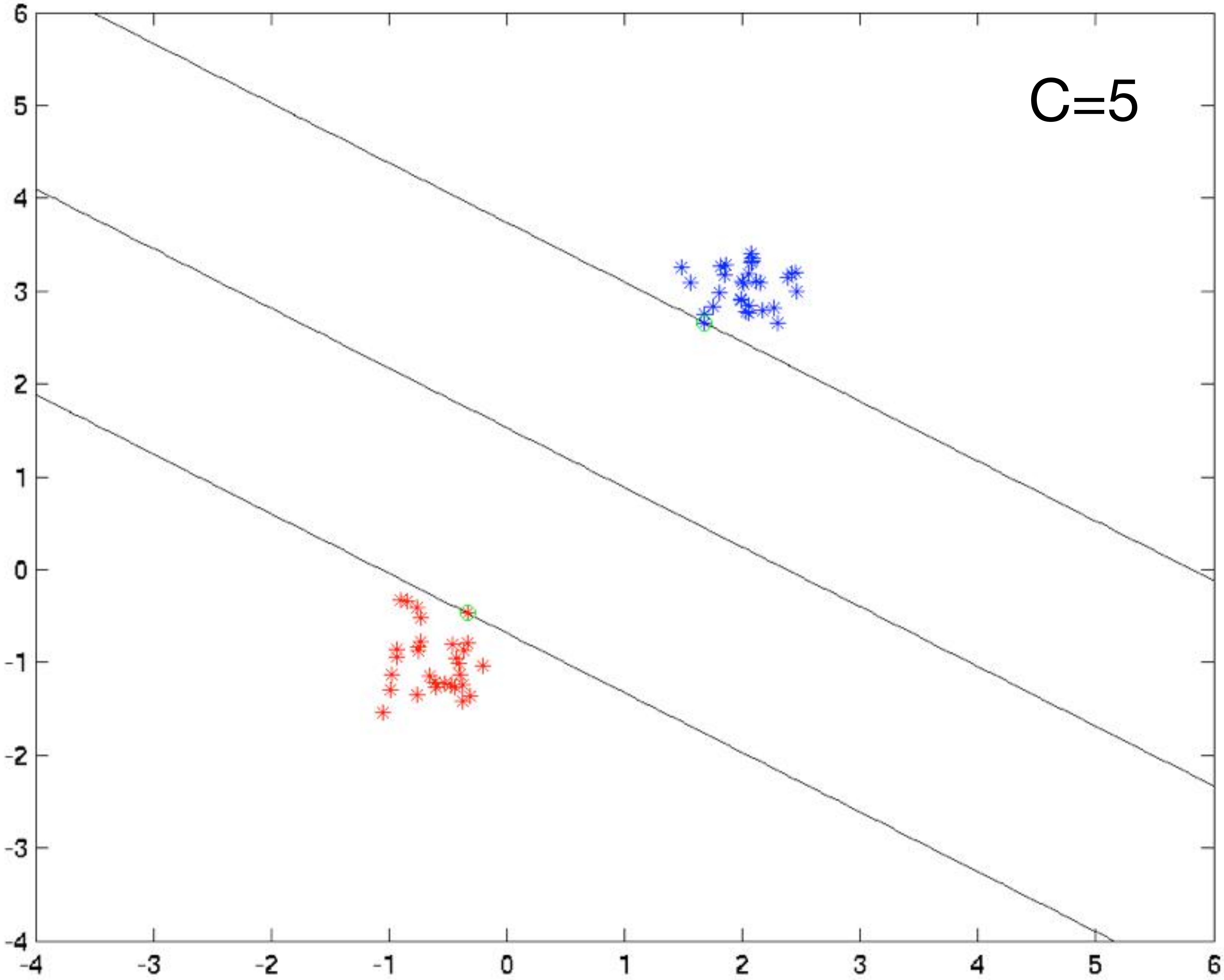
slide by Alex Smola



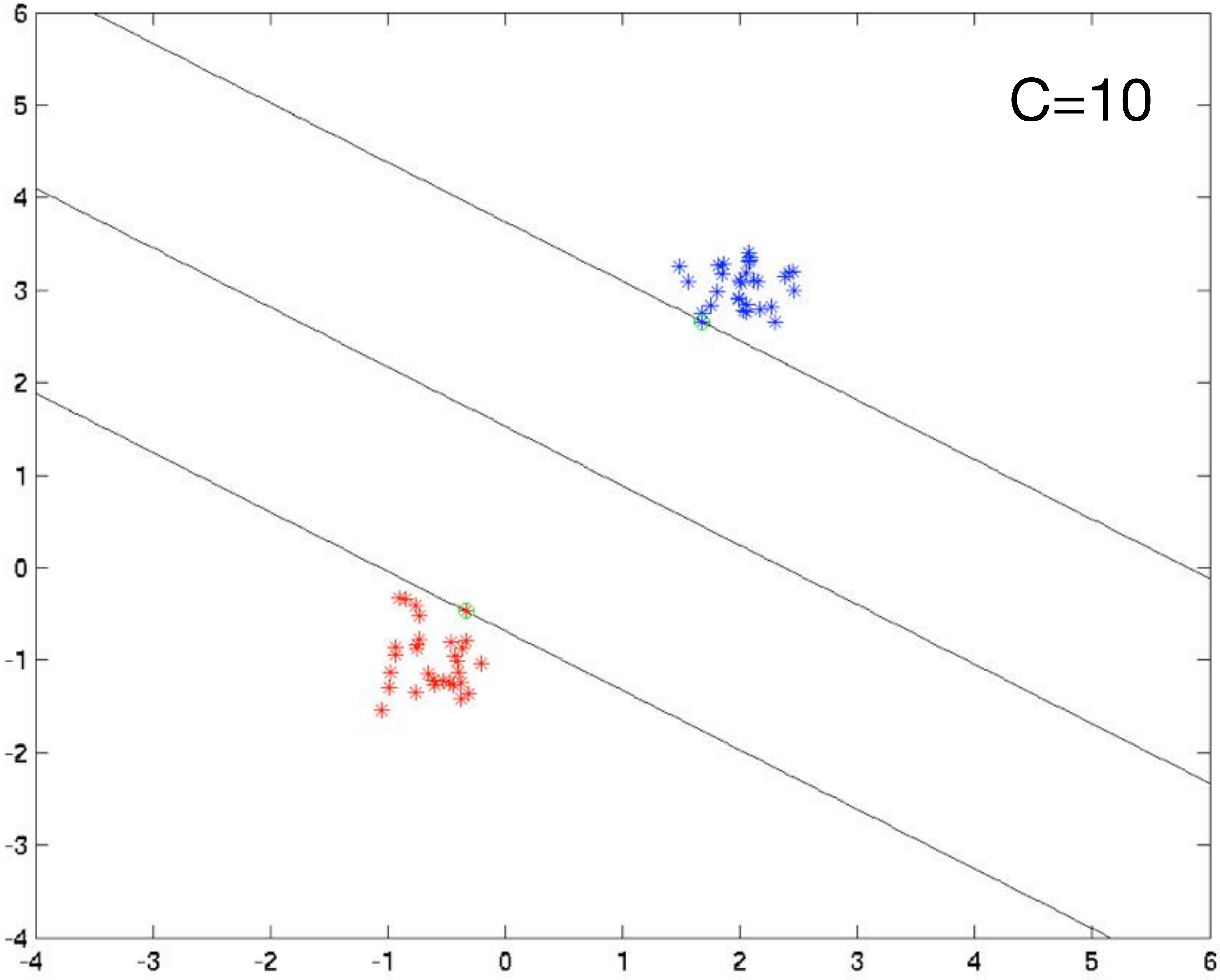
slide by Alex Smola



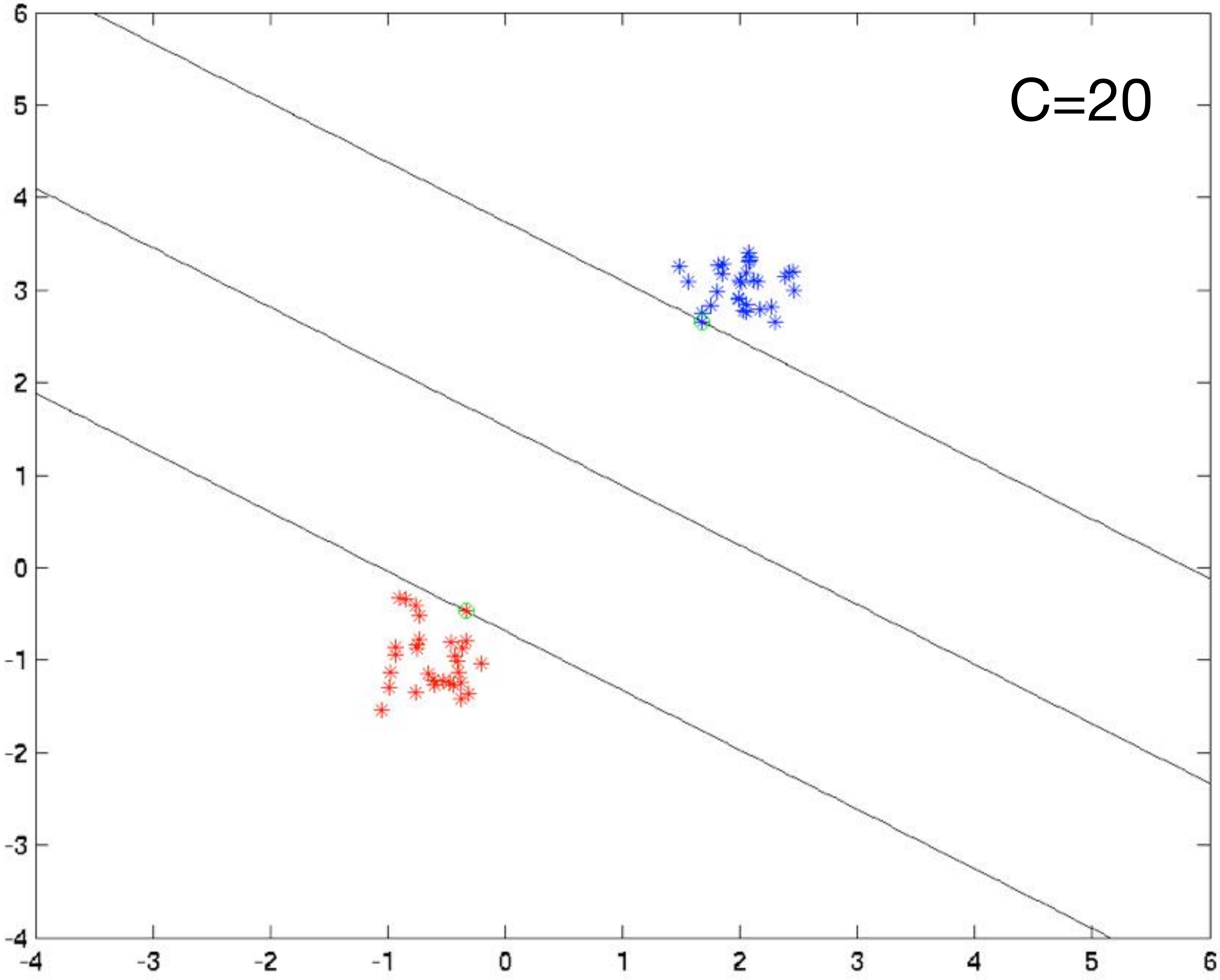
slide by Alex Smola



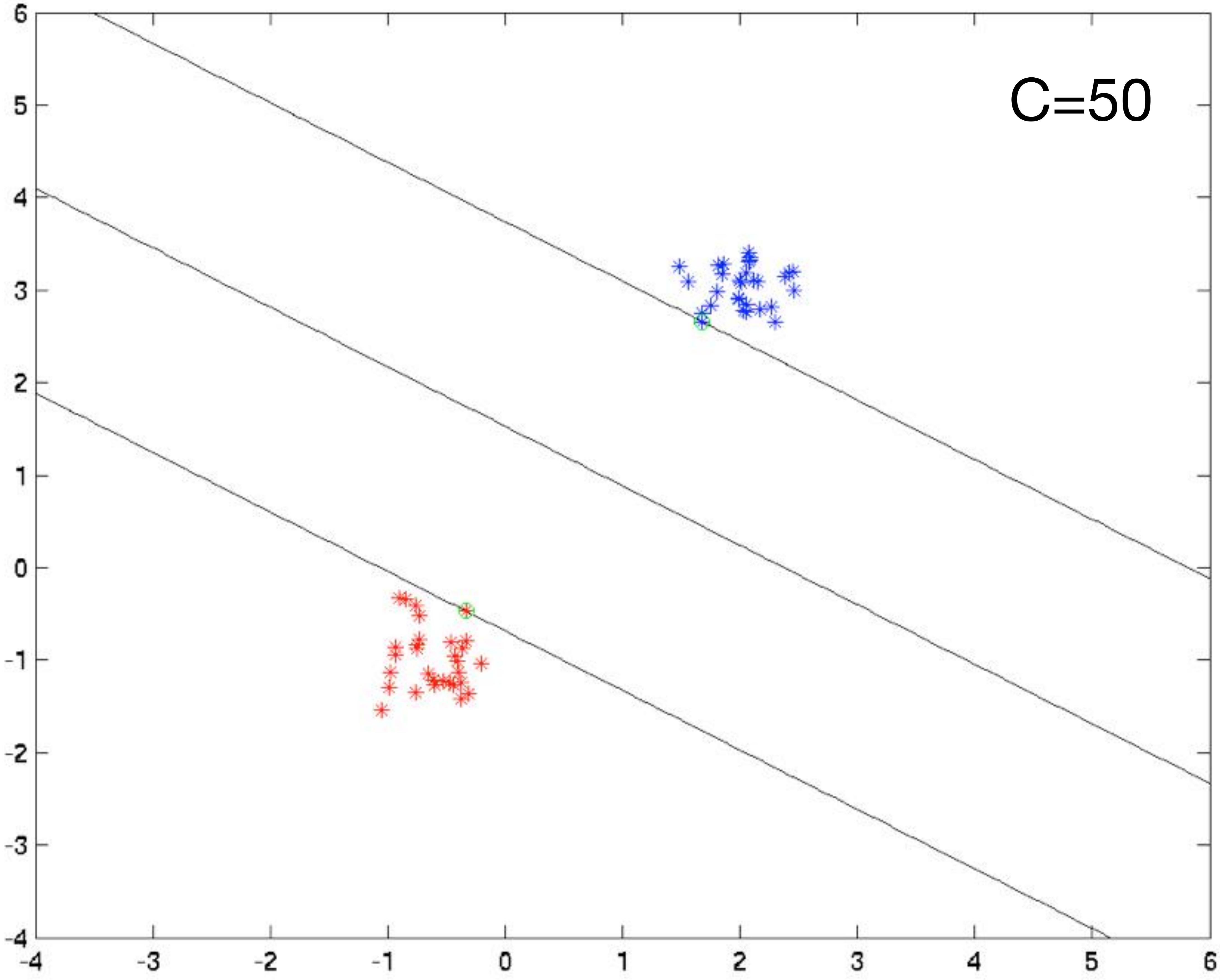
slide by Alex Smola



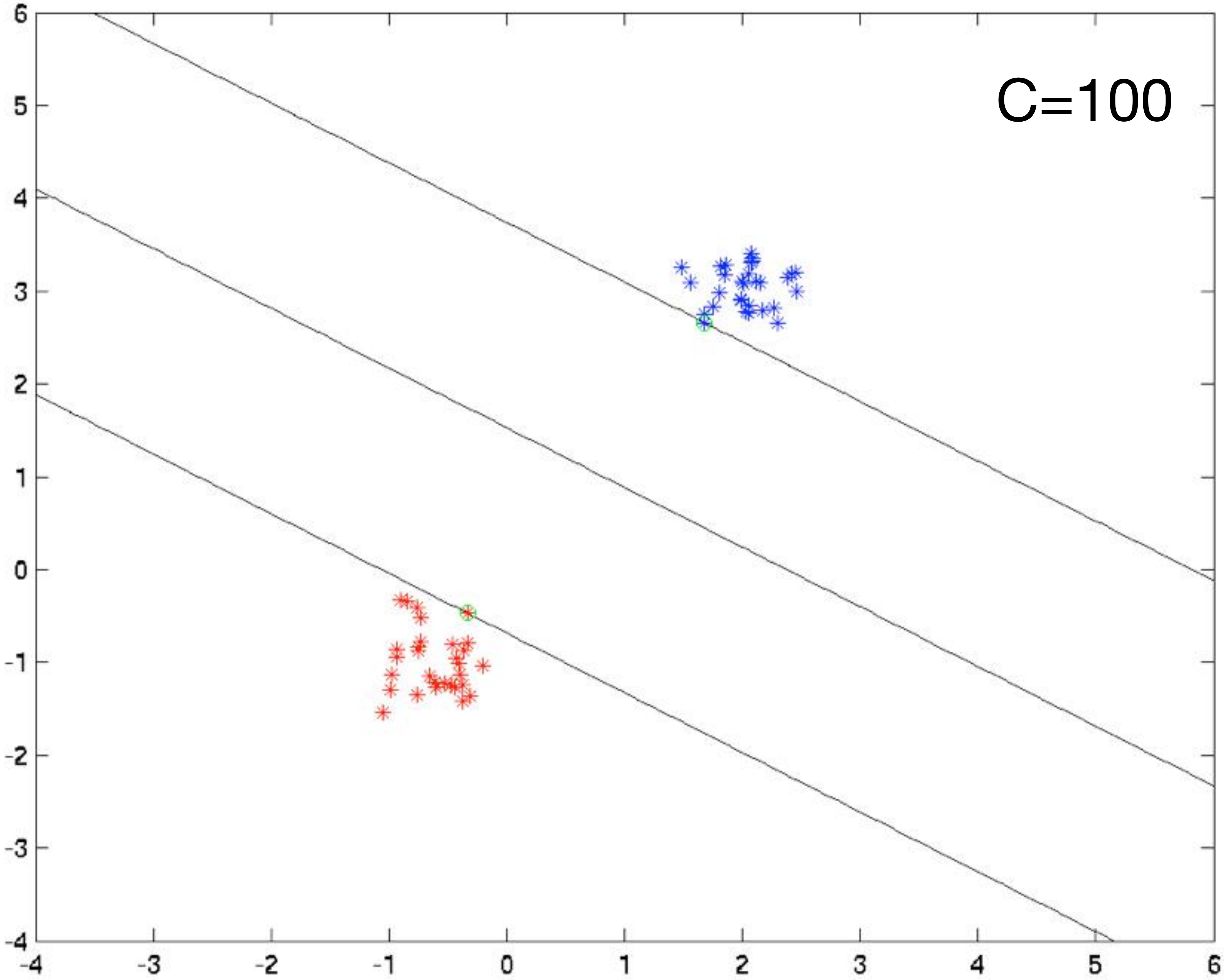
slide by Alex Smola



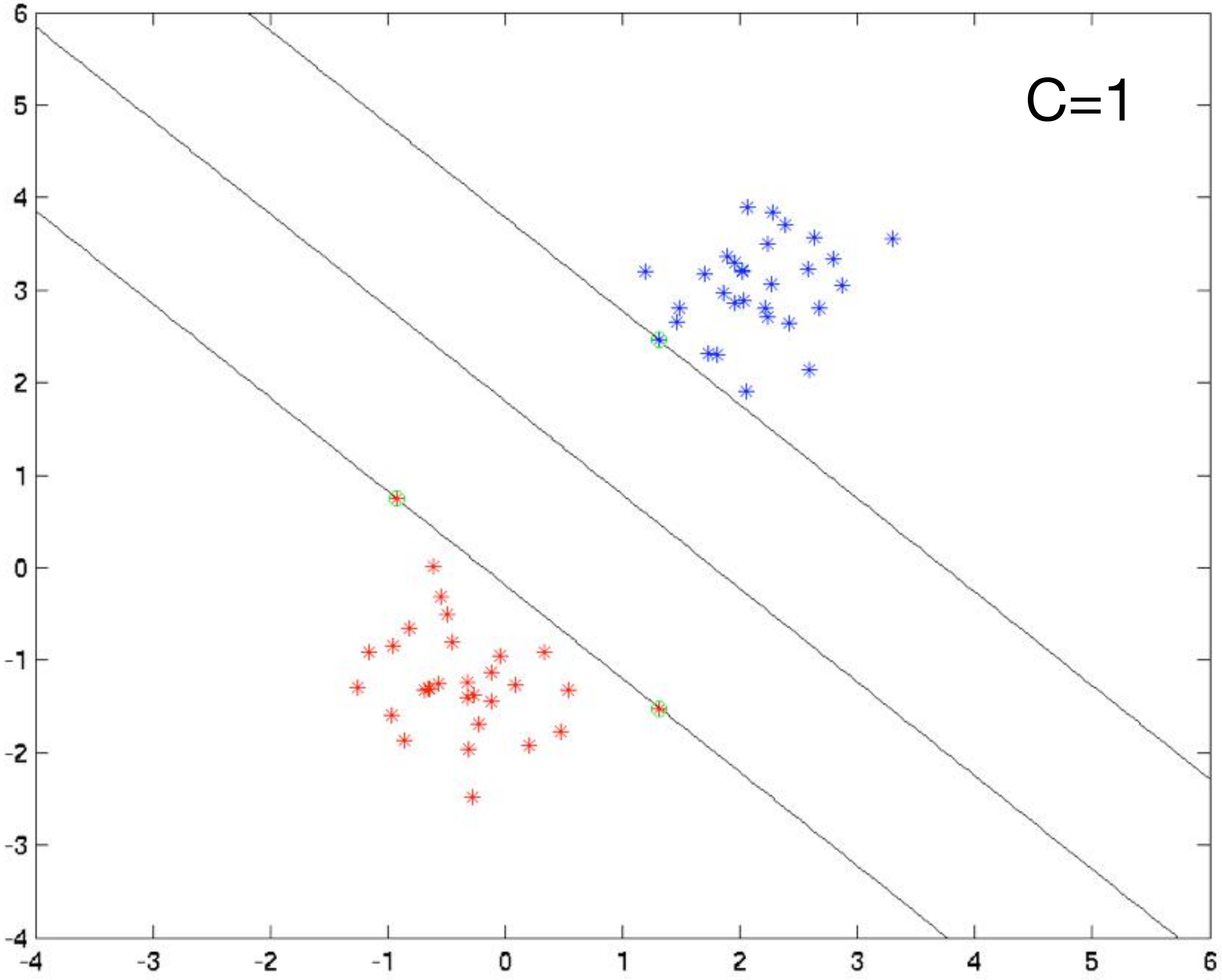
slide by Alex Smola



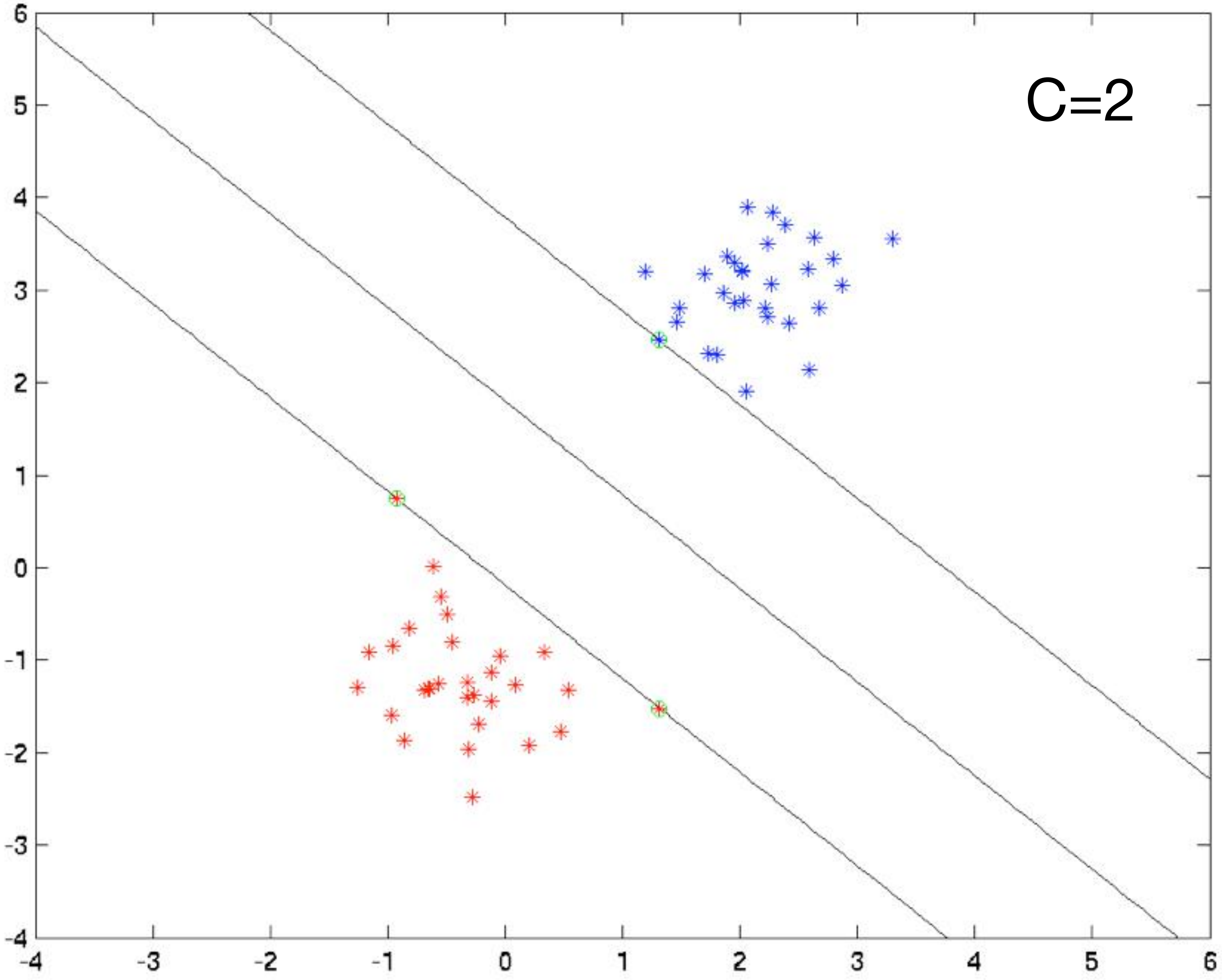
slide by Alex Smola



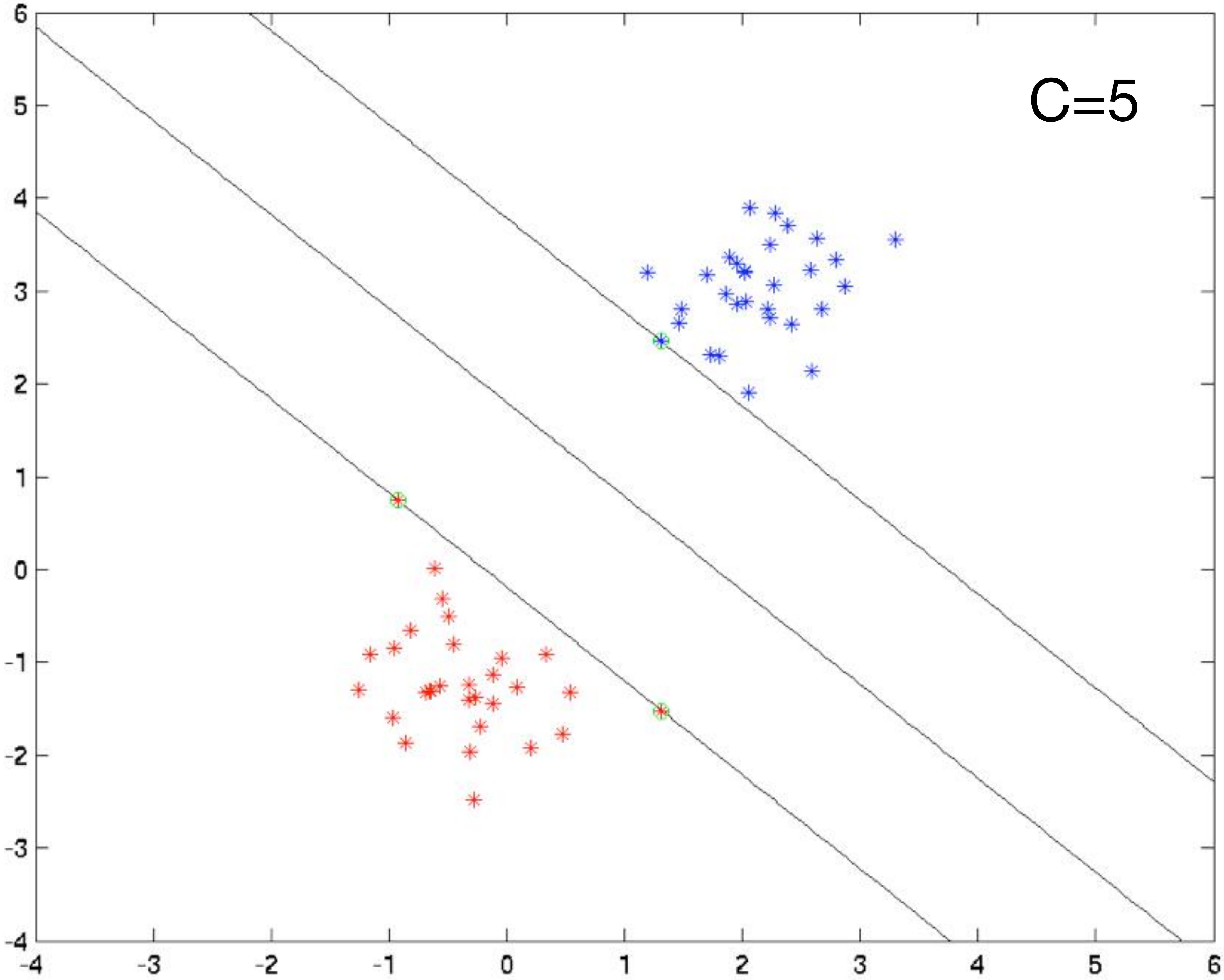
slide by Alex Smola



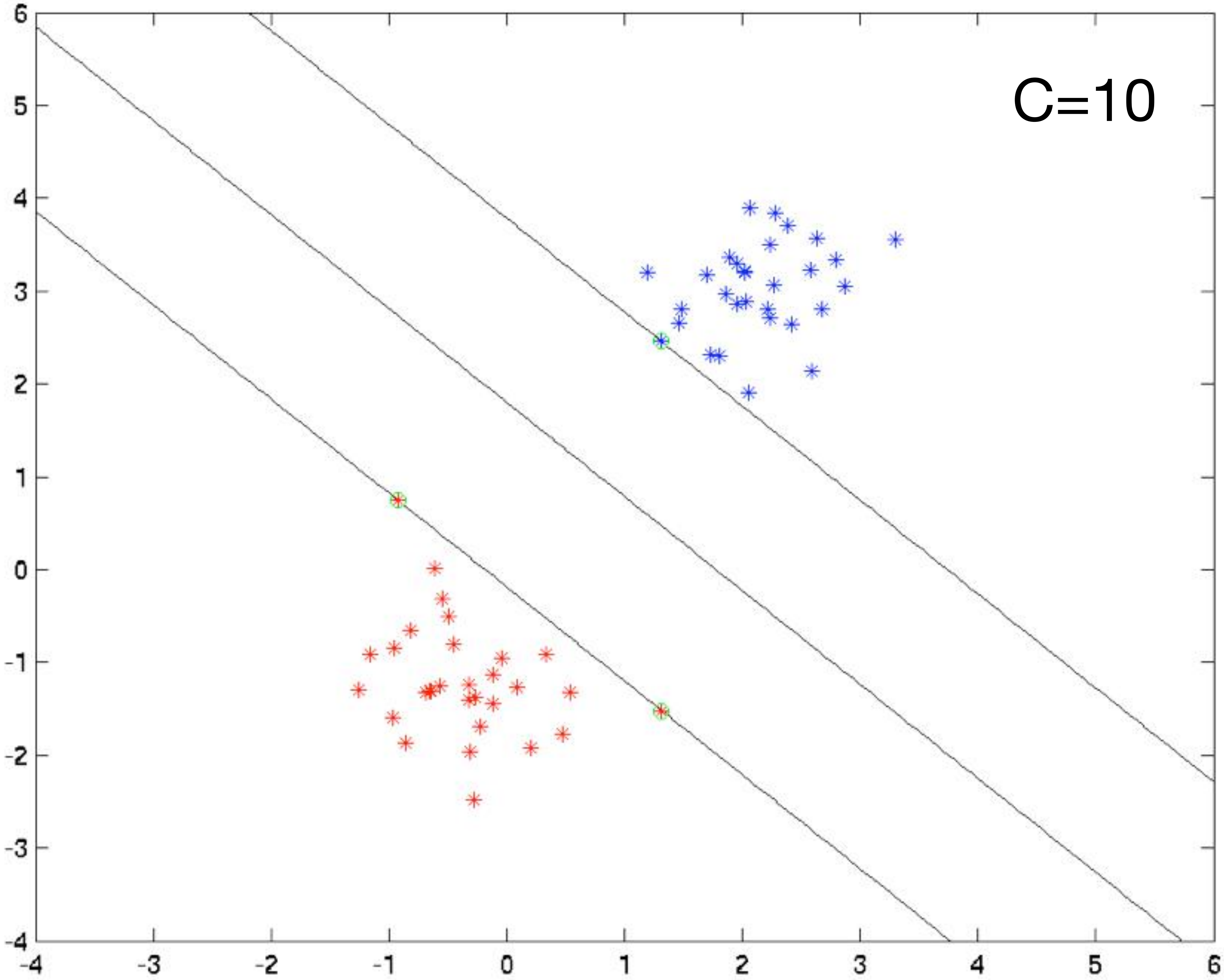
slide by Alex Smola



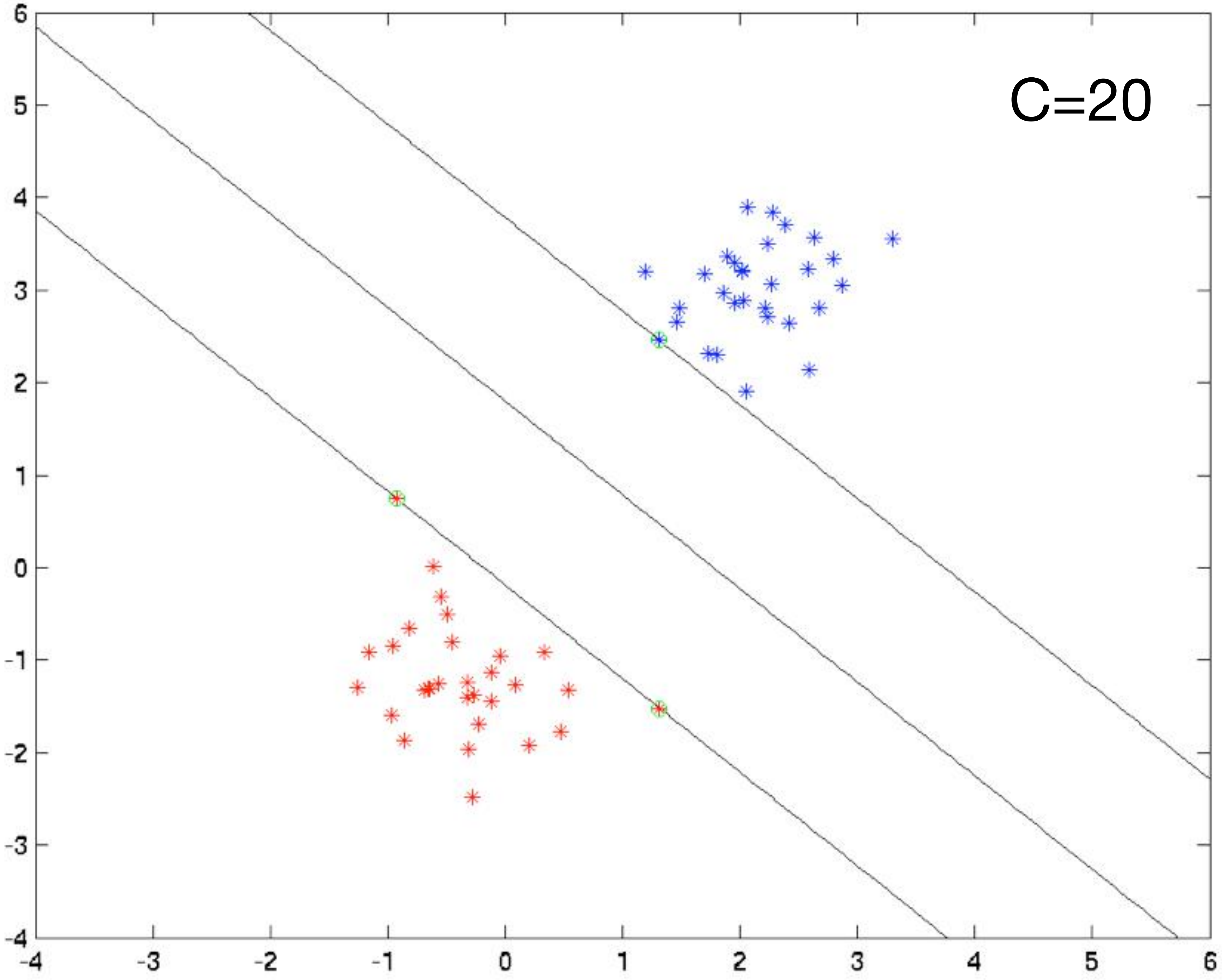
slide by Alex Smola



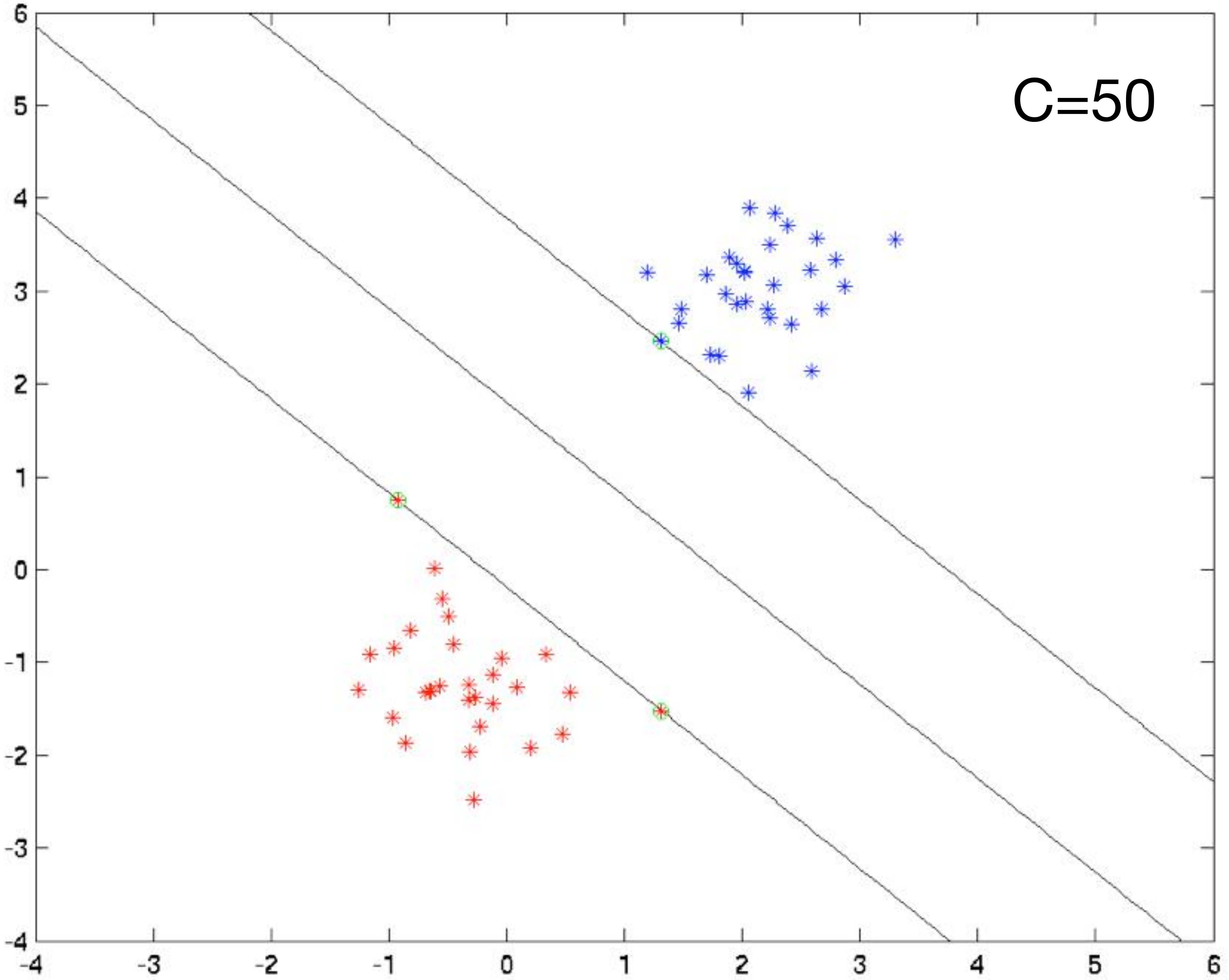
slide by Alex Smola



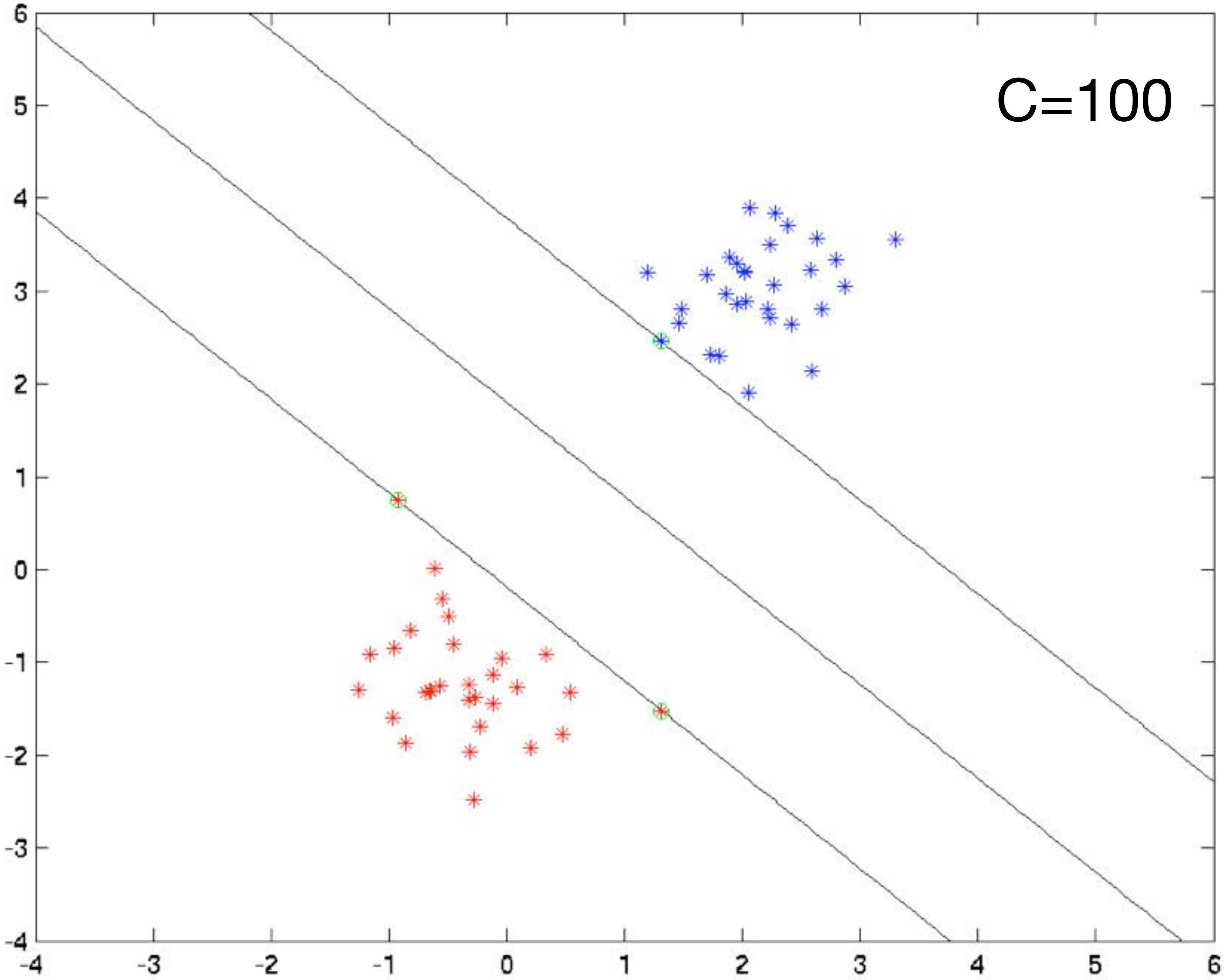
slide by Alex Smola



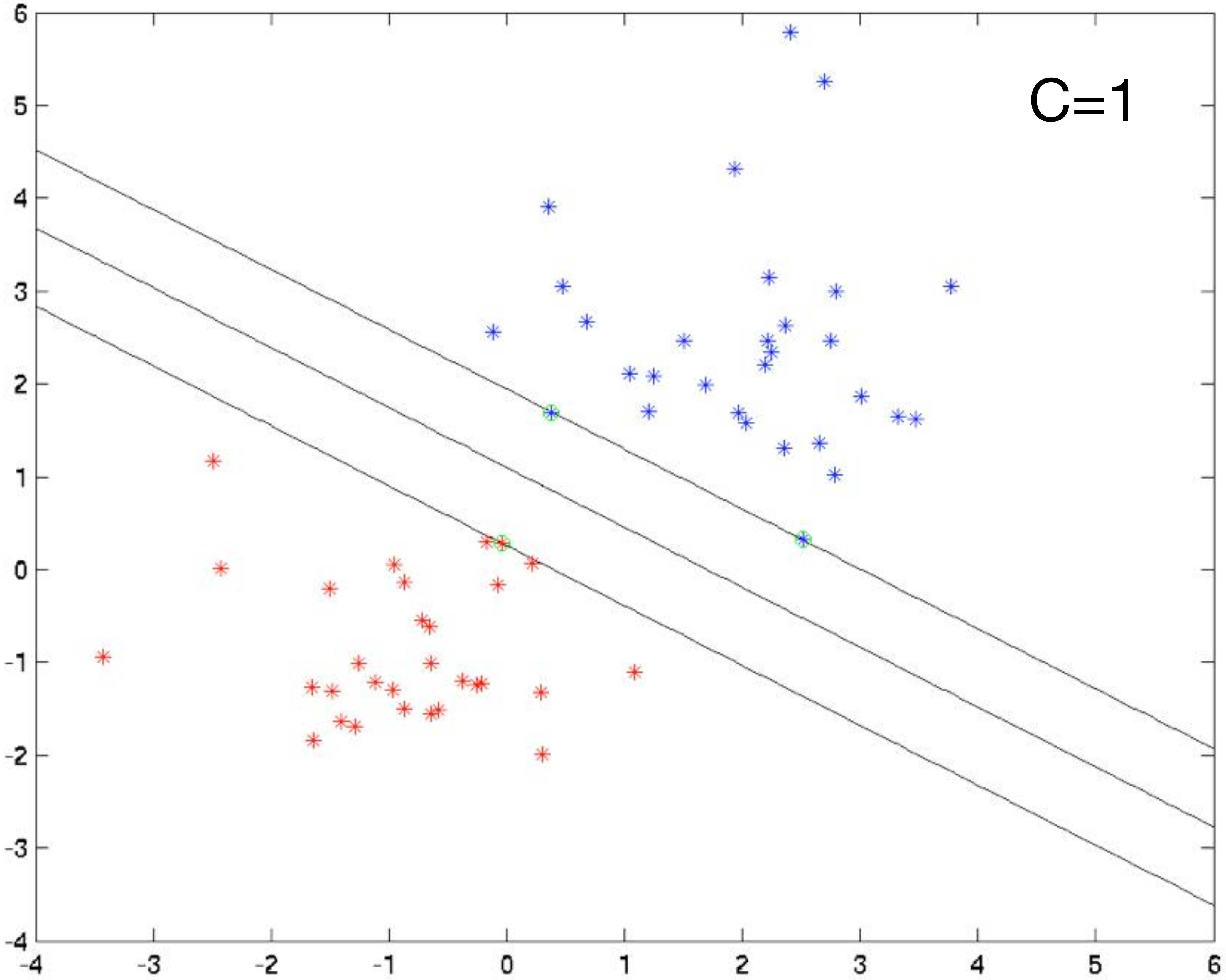
slide by Alex Smola



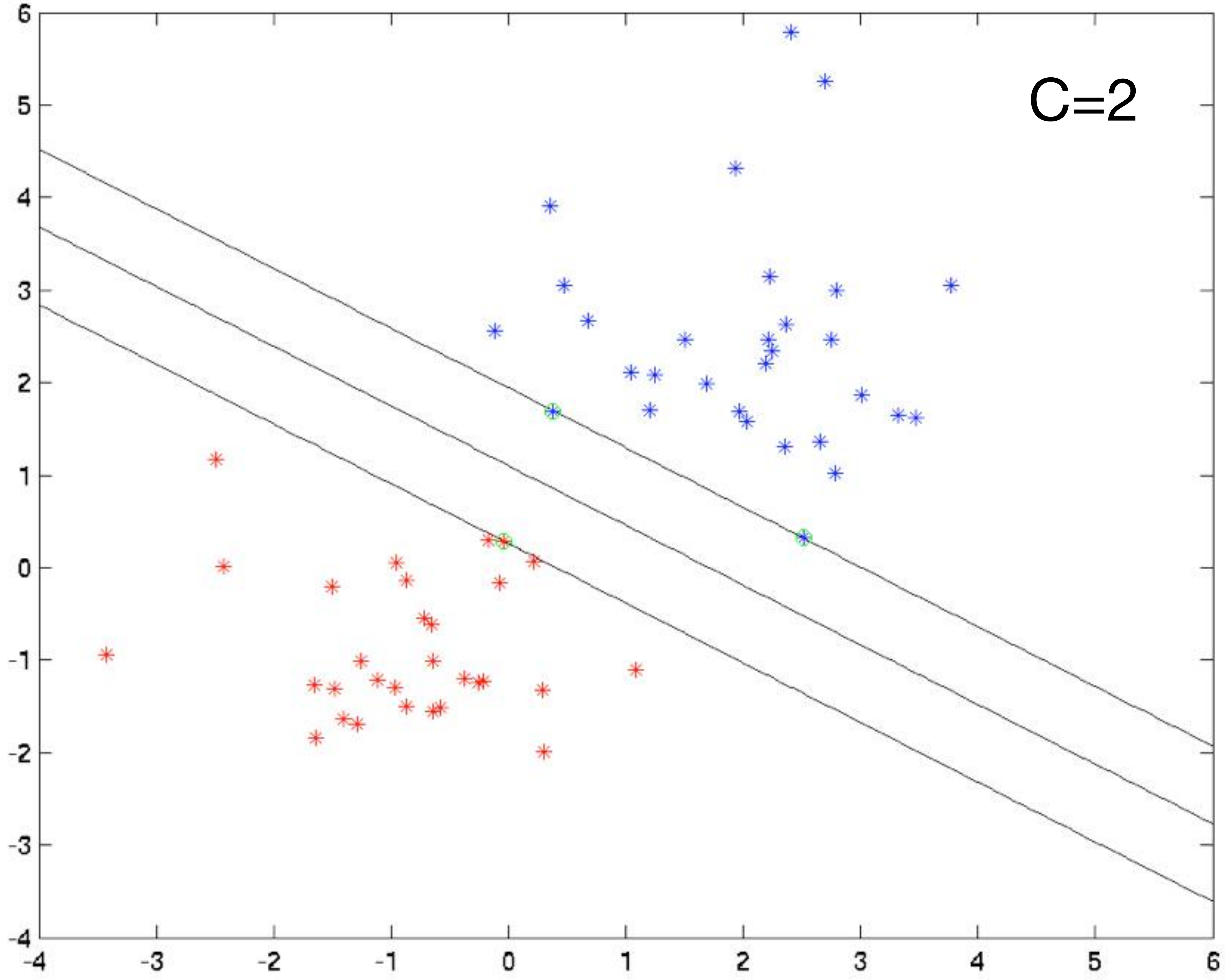
slide by Alex Smola



slide by Alex Smola

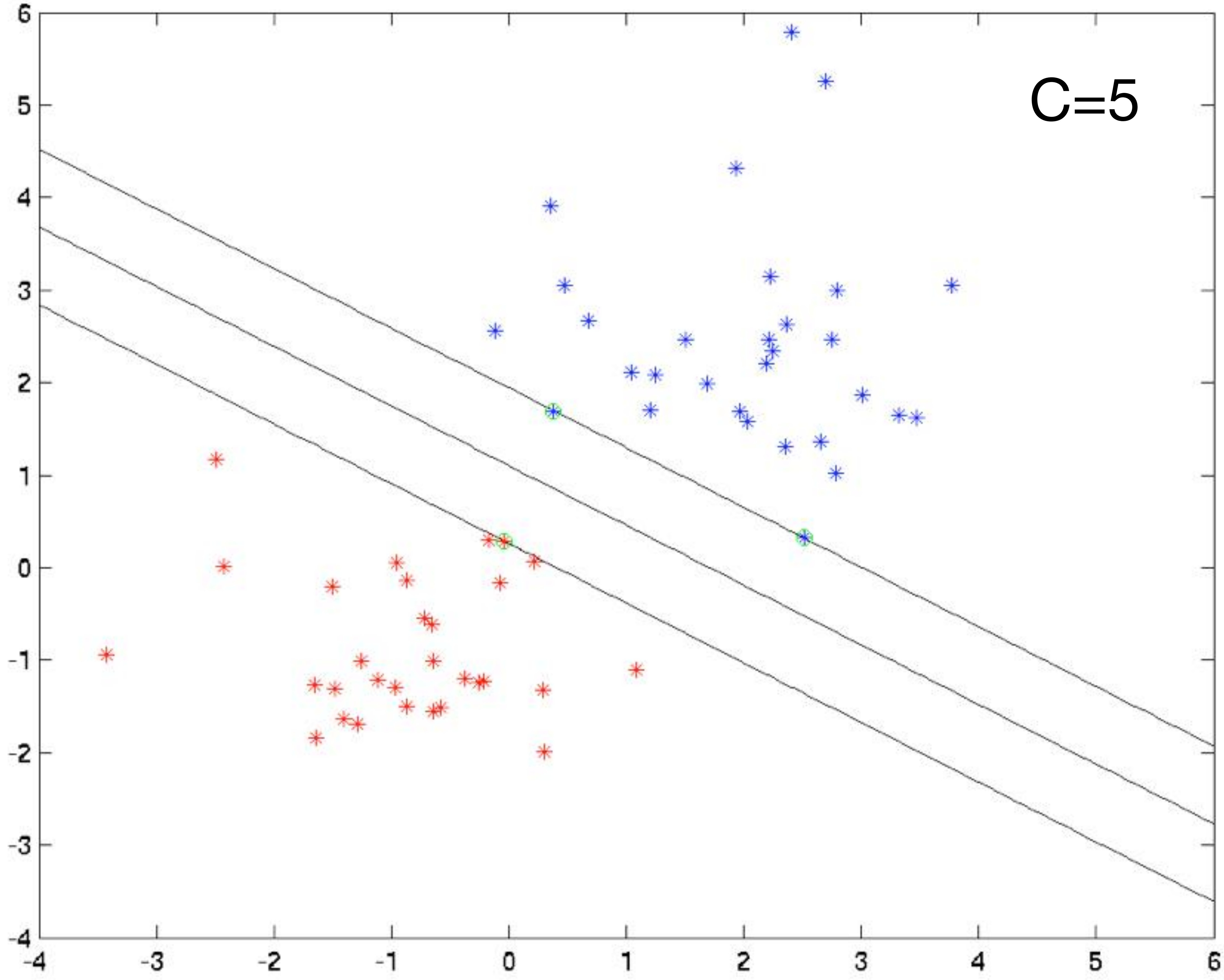


slide by Alex Smola

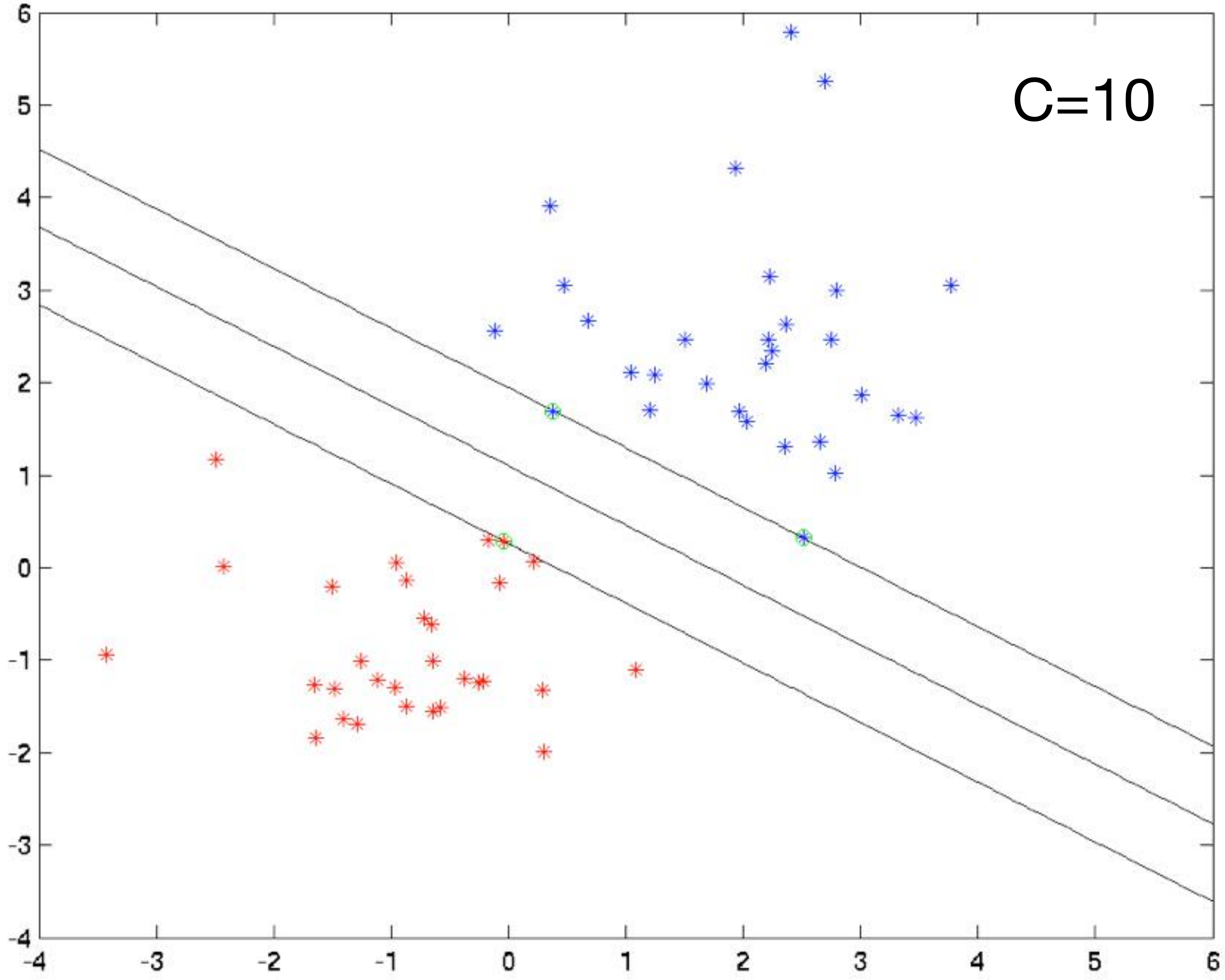


C=2

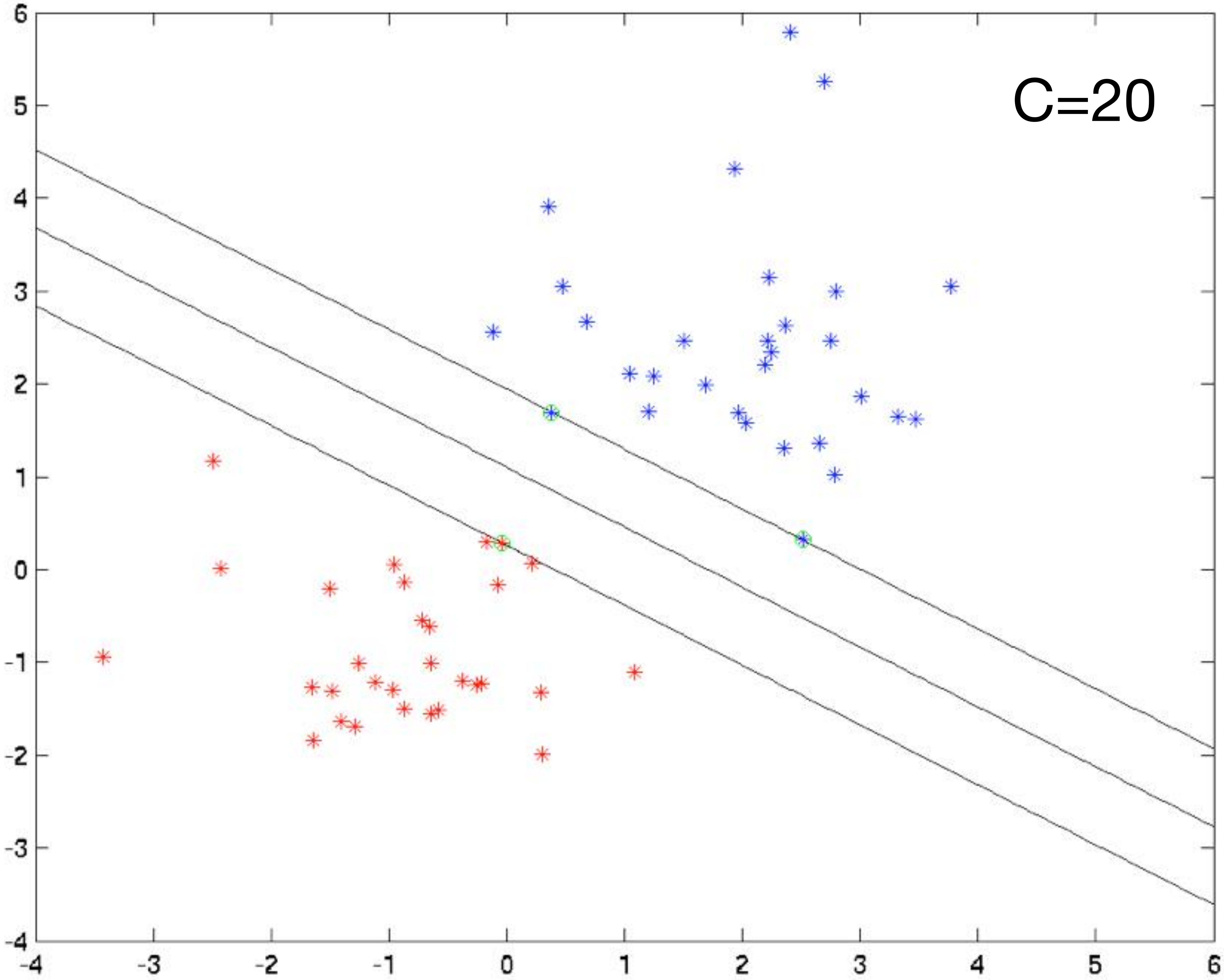
slide by Alex Smola



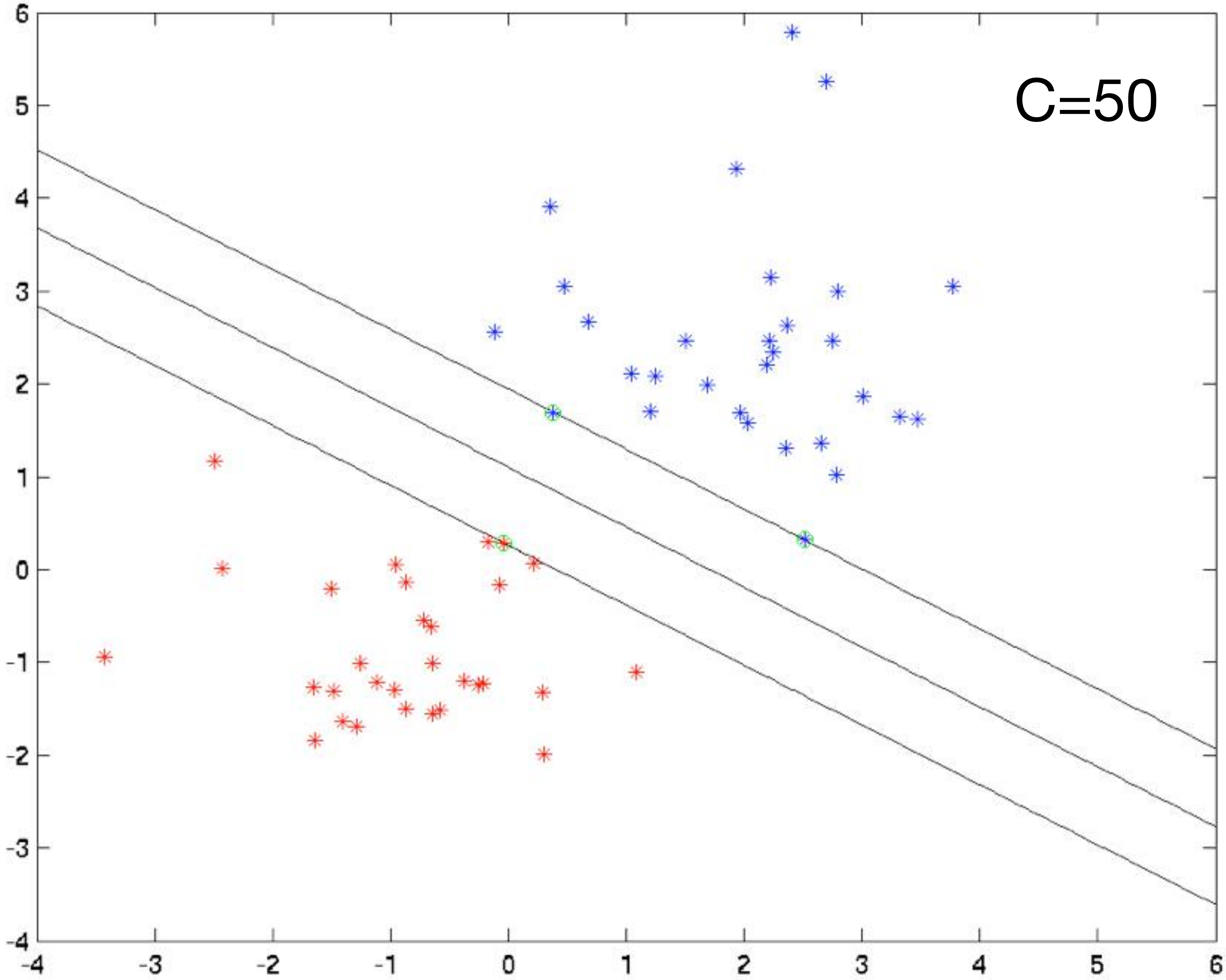
slide by Alex Smola



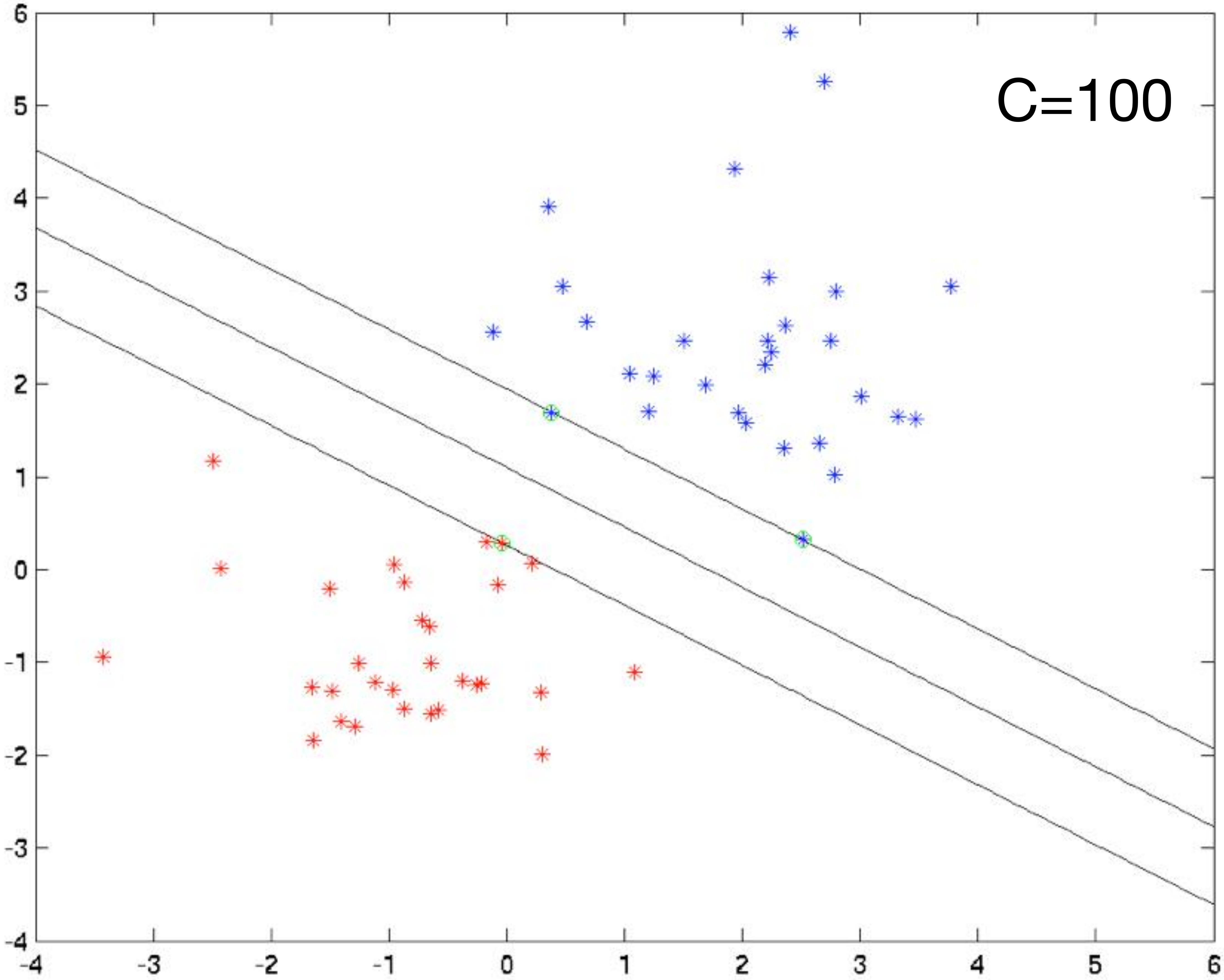
slide by Alex Smola



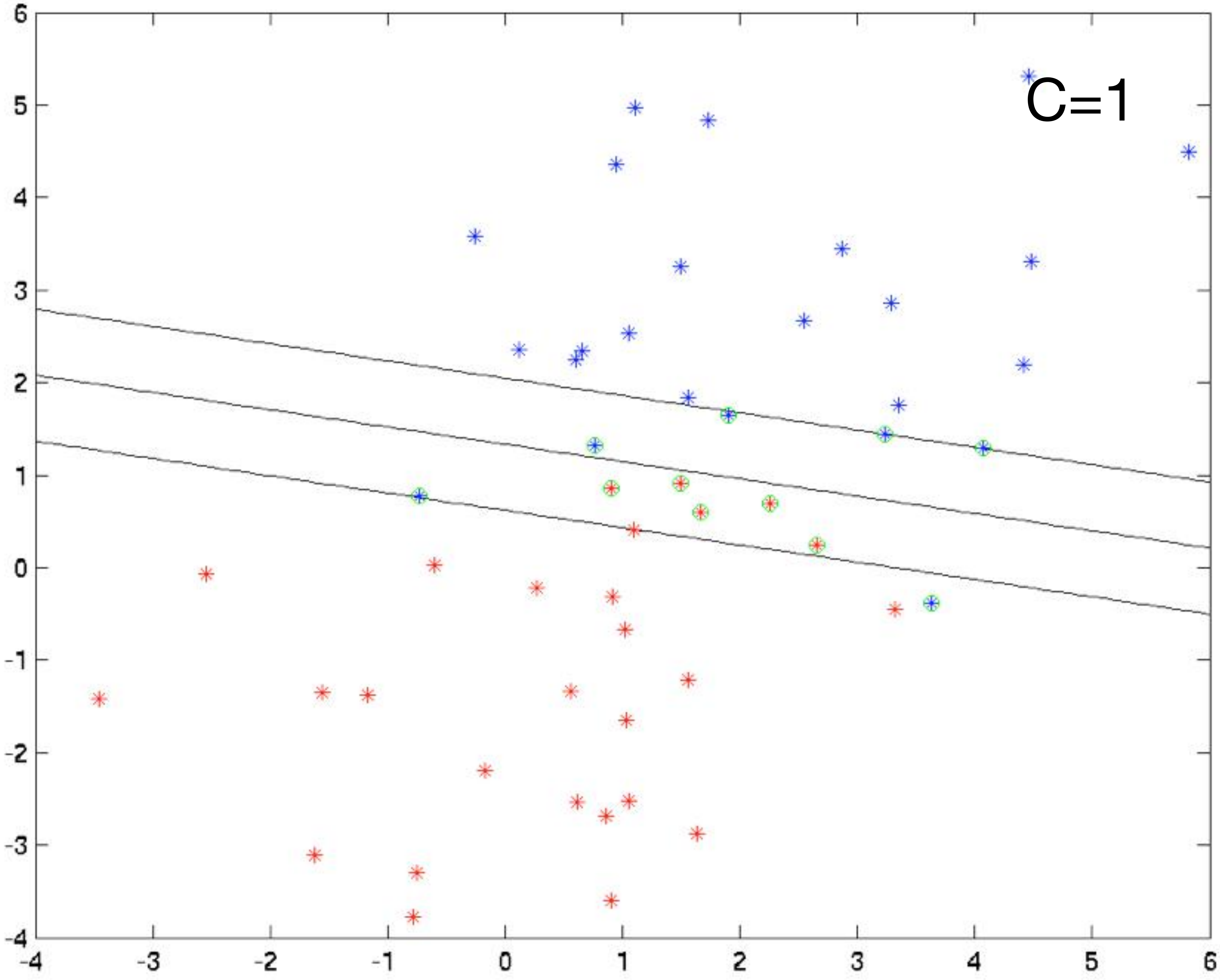
slide by Alex Smola



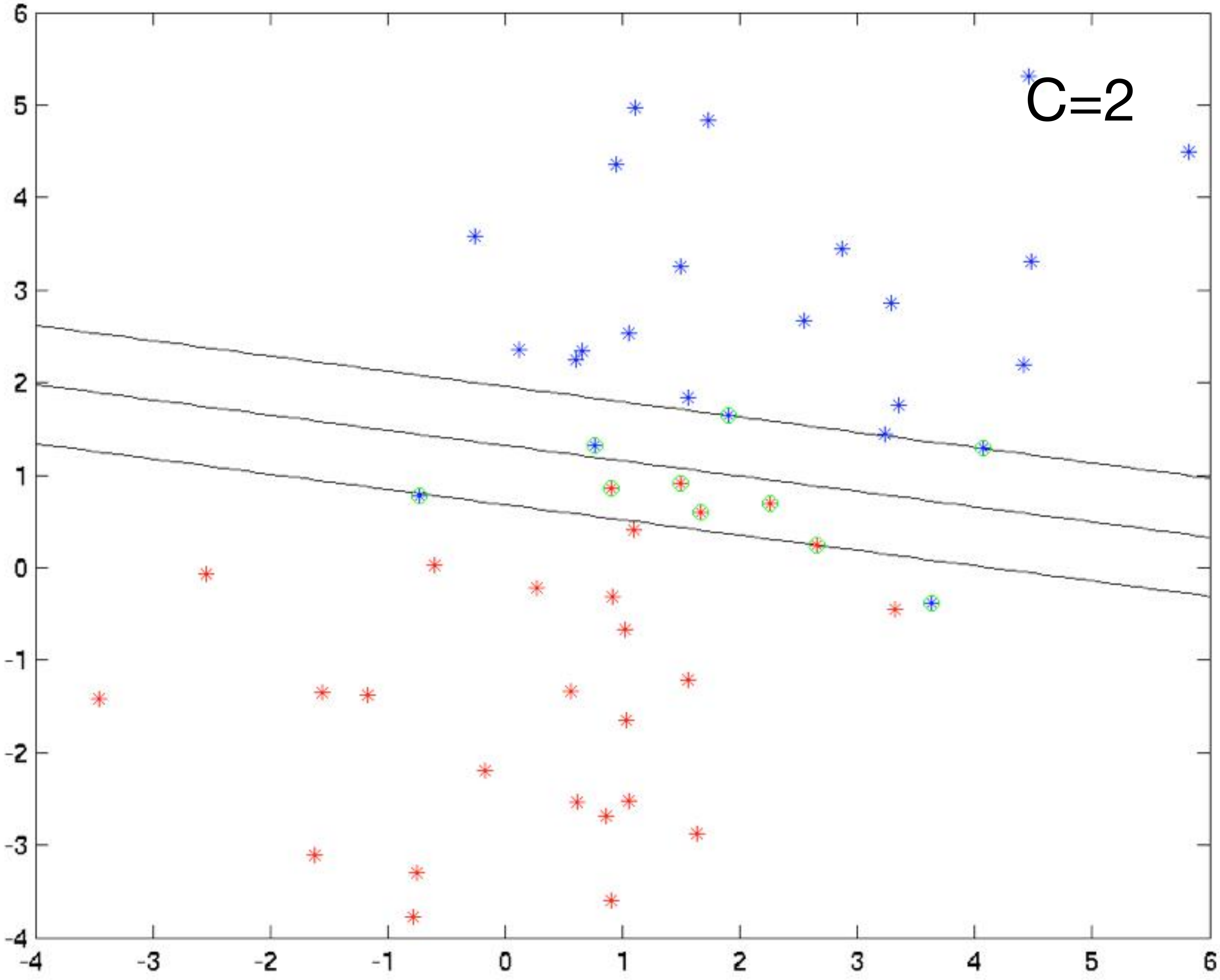
slide by Alex Smola



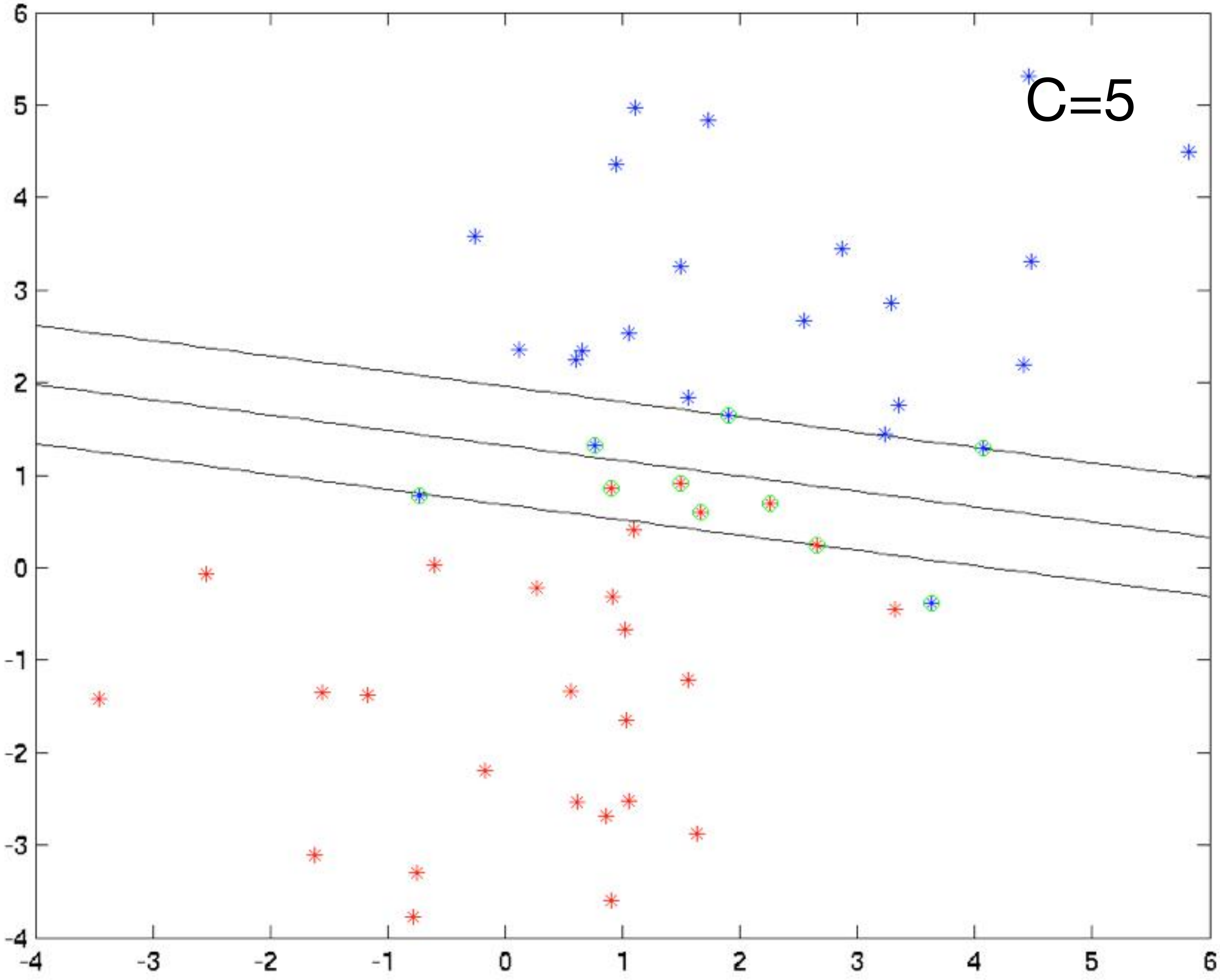
slide by Alex Smola



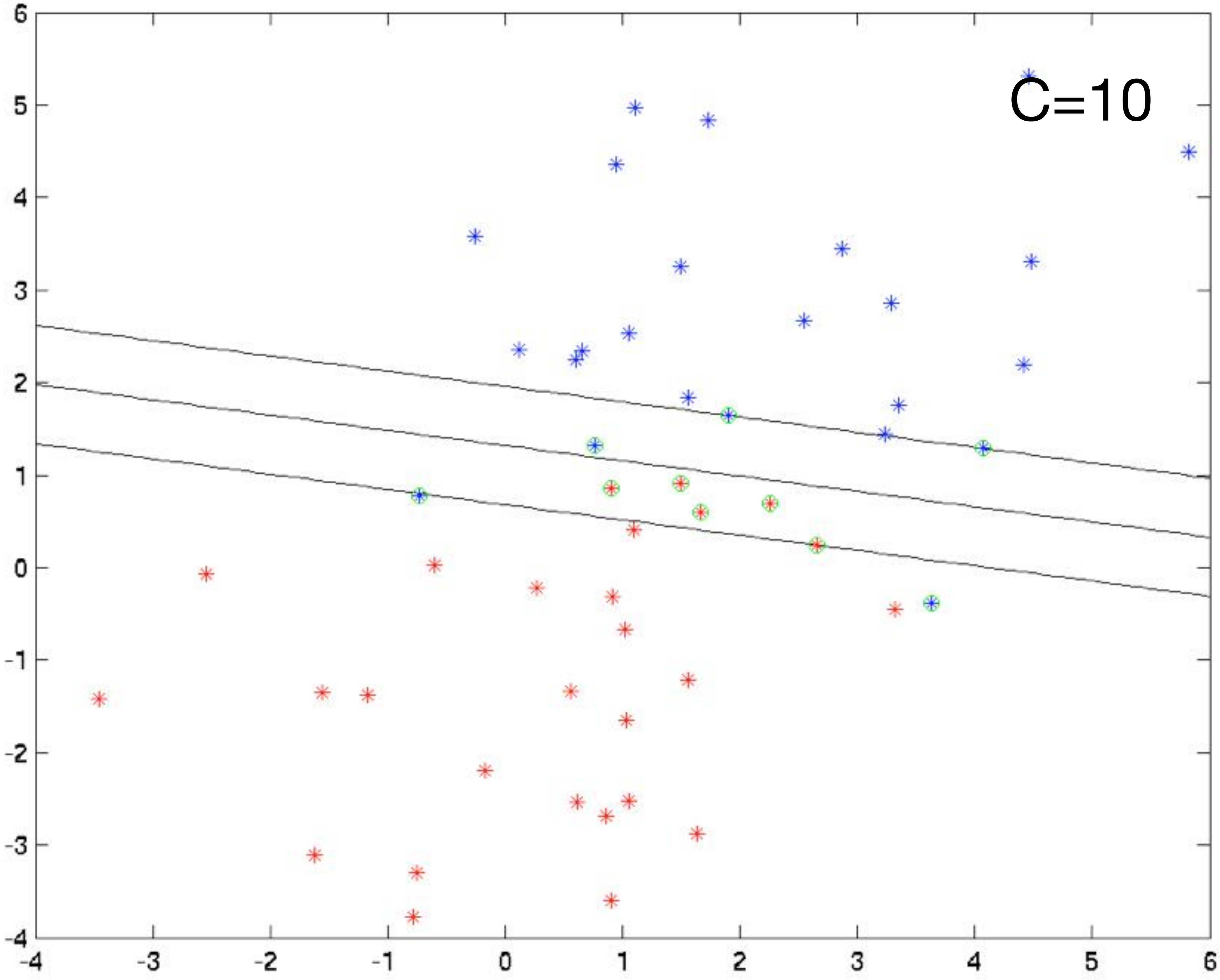
slide by Alex Smola



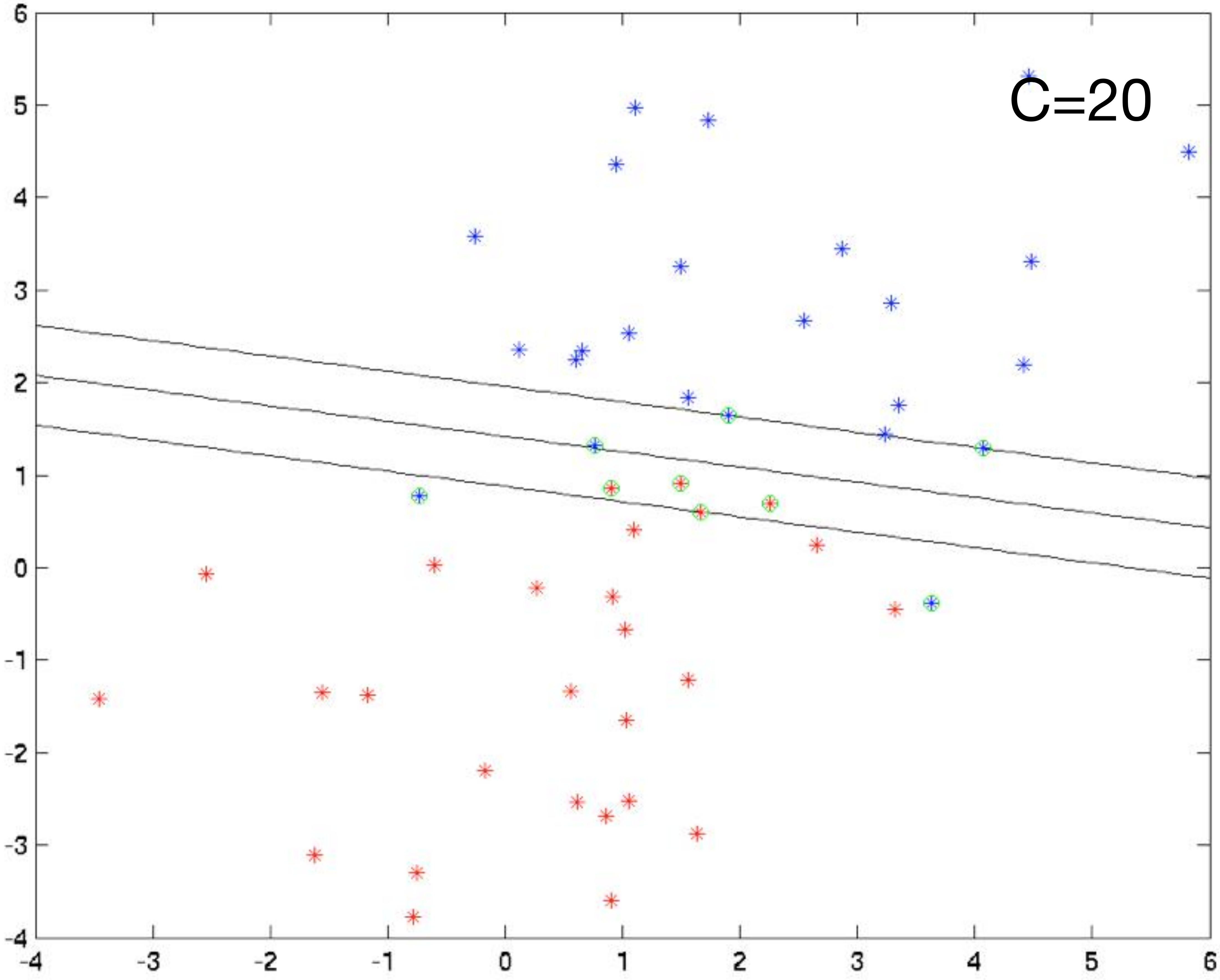
slide by Alex Smola



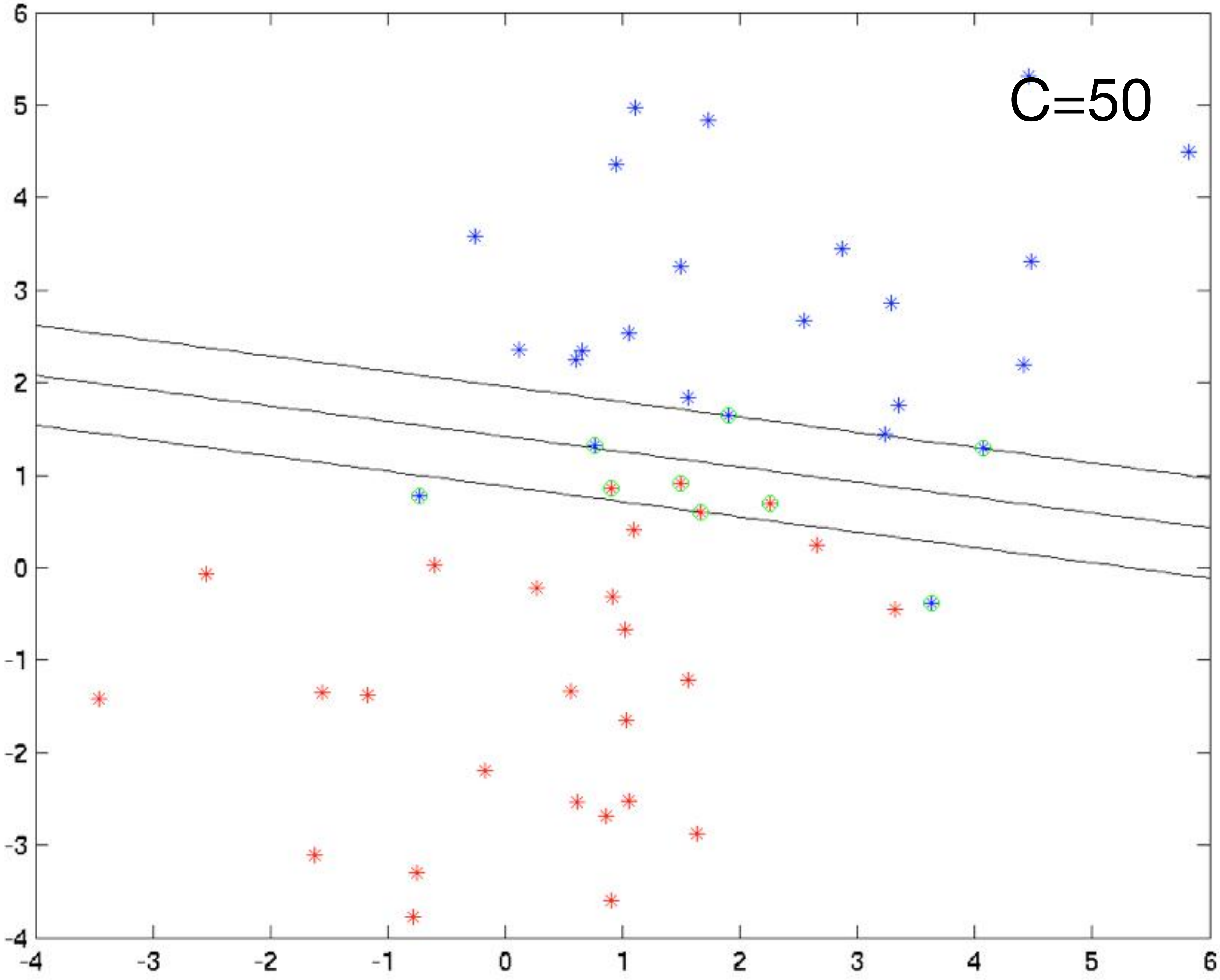
slide by Alex Smola



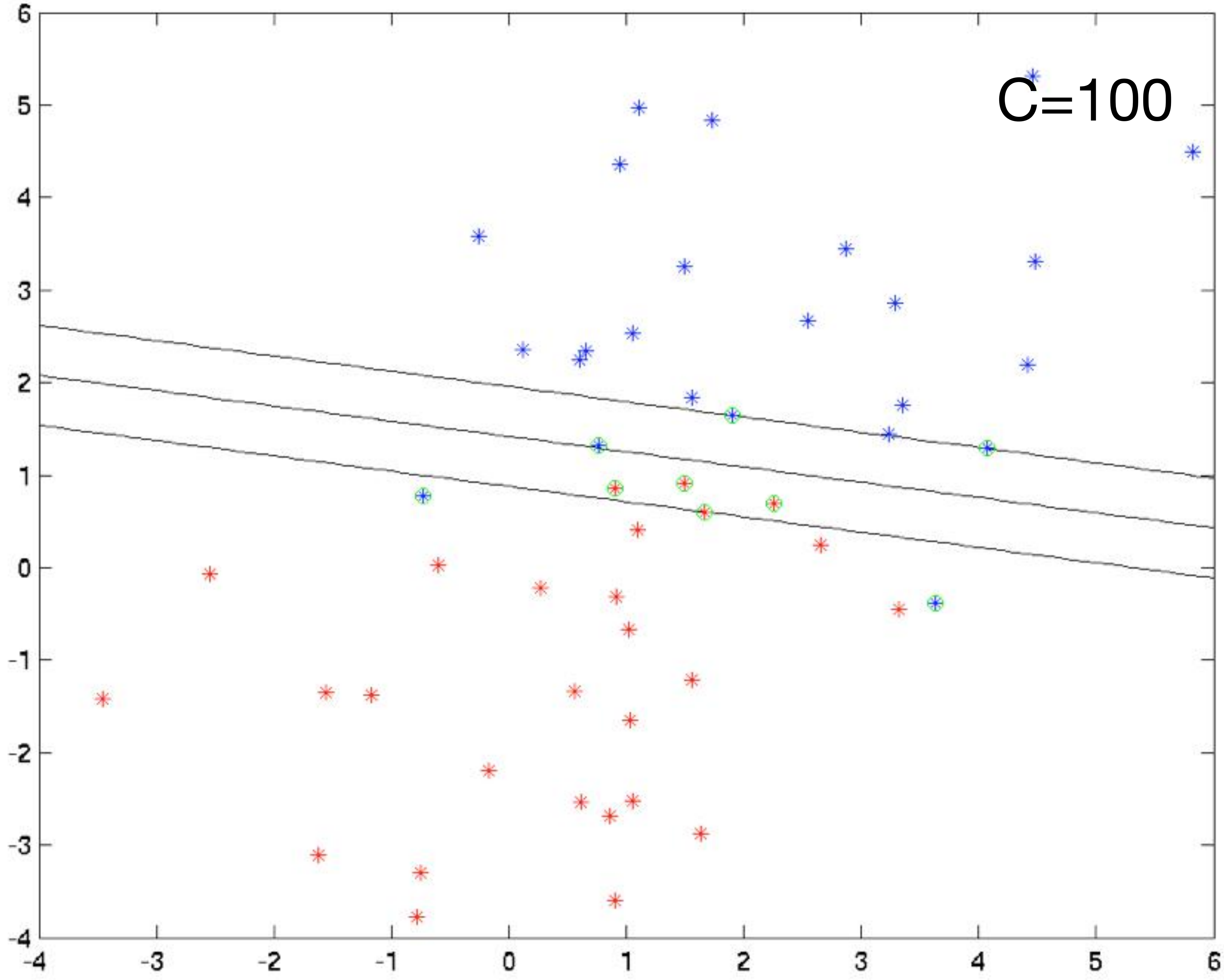
slide by Alex Smola



slide by Alex Smola



slide by Alex Smola



slide by Alex Smola

# Solving the optimization problem

- Dual problem

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

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

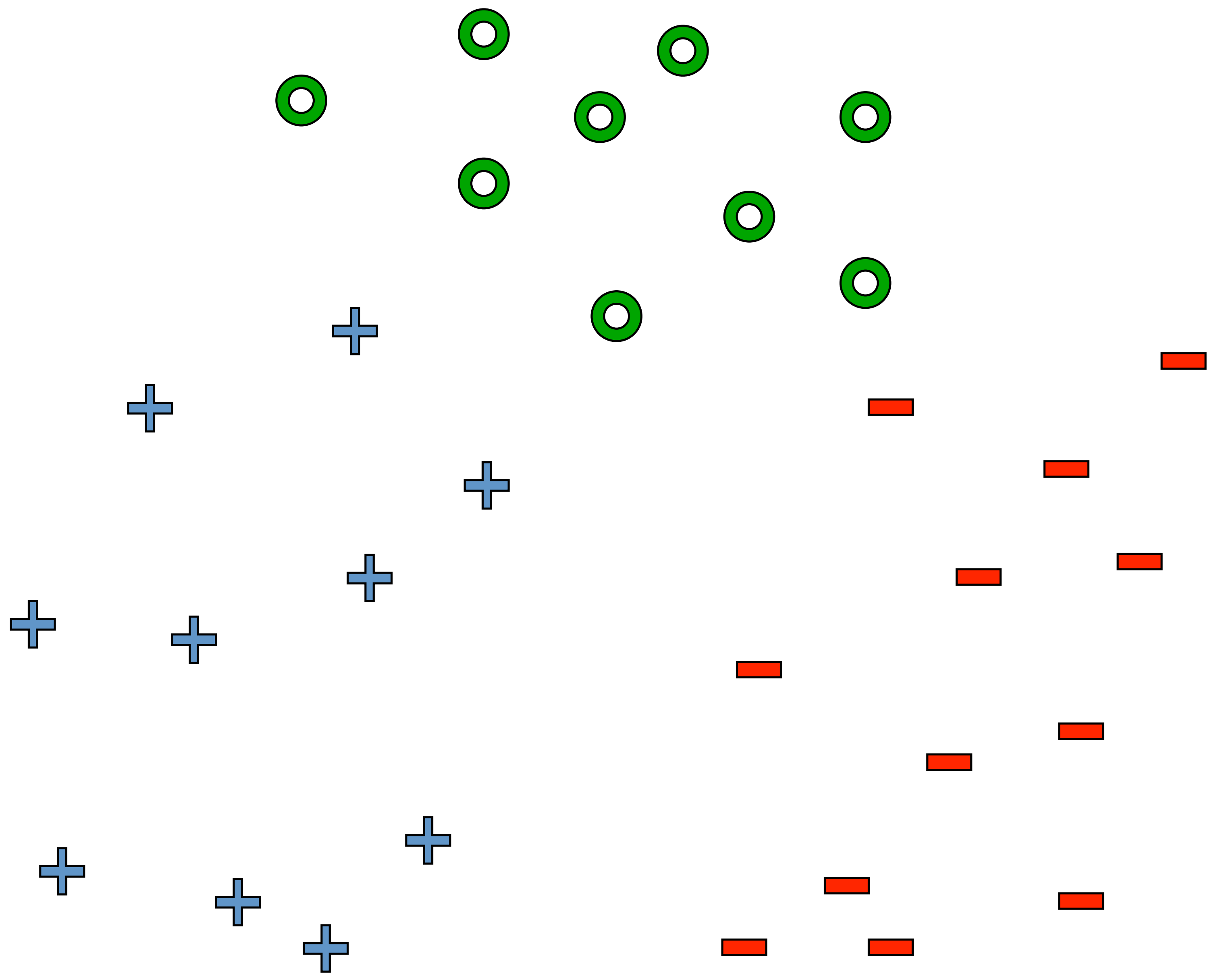
- If problem is small enough (1000s of variables) we can use off-the-shelf solver (CVXOPT, CPLEX, OOQP, LOQO)
- For larger problem use fact that only SVs matter and solve in blocks (active set method).

slide by Alex Smola

## Multi-class classification

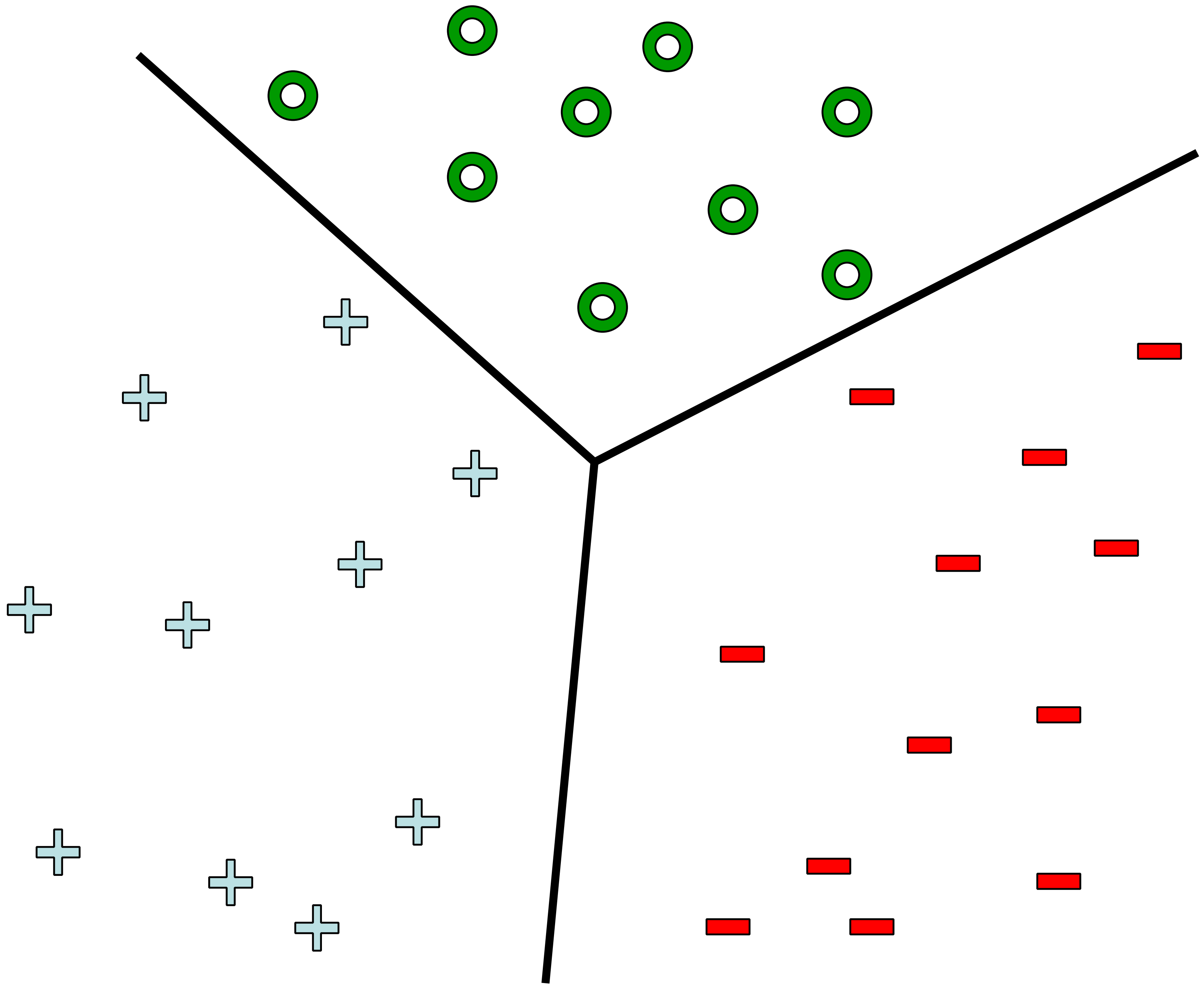
slide by Eric Xing

# Multi-class classification



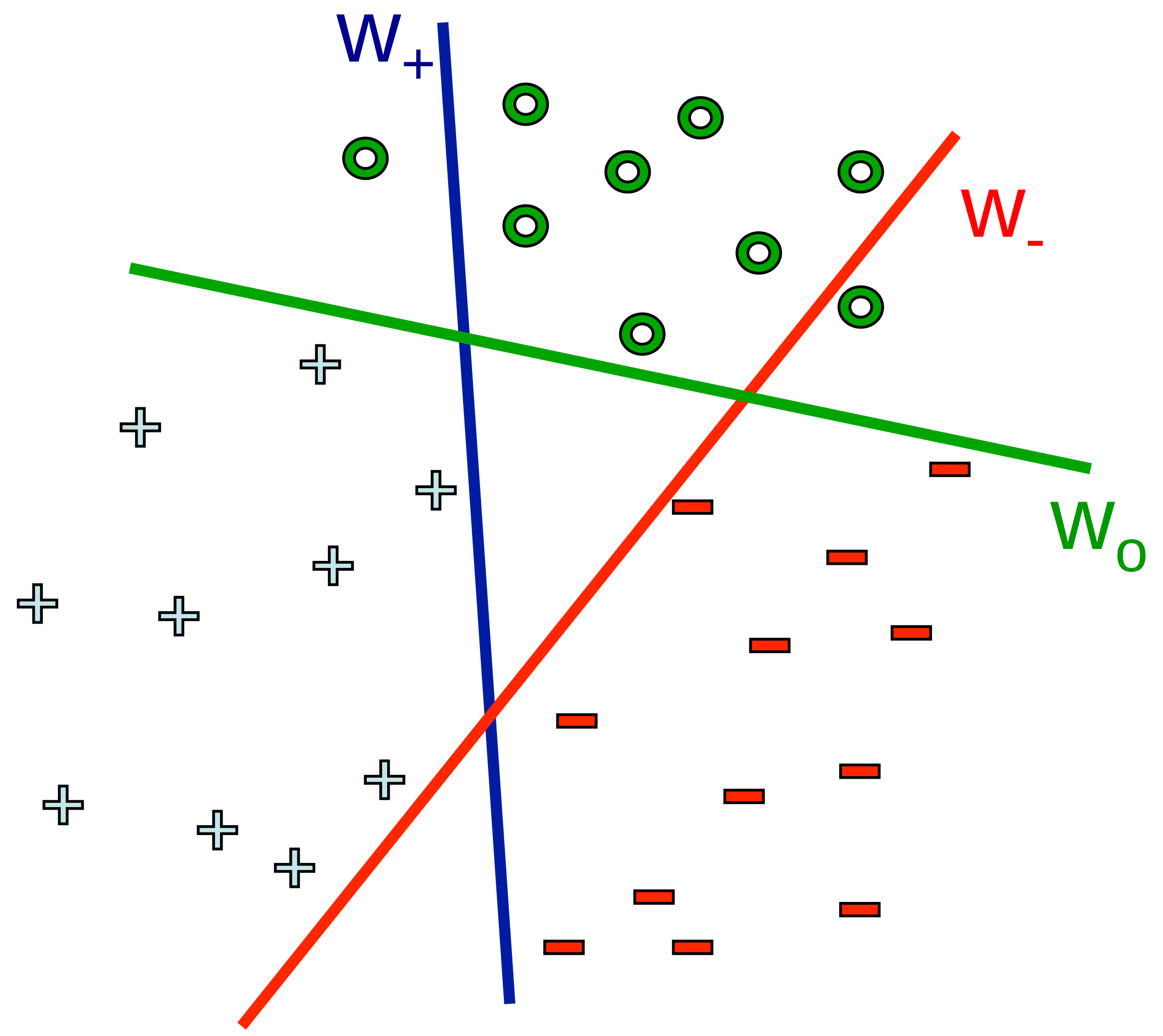
slide by Eric Xing

# Multi-class classification



slide by Eric Xing

# One versus all classification



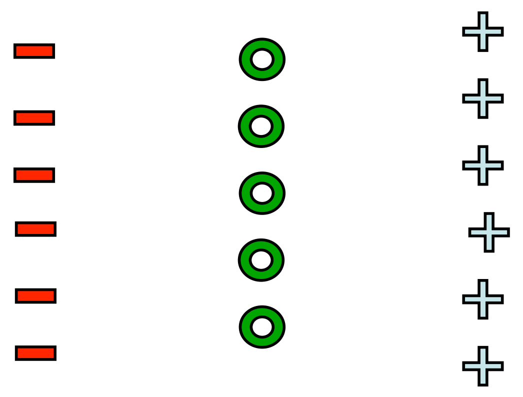
- Learn 3 classifiers:
  - - vs. {o,+}, weights $w_-$
  - + vs. {o,-}, weights $w_+$
  - o vs. {+,-}, weights $w_o$

- Predict label using:

$$\hat{y} \leftarrow \arg\max_k \; w_k \cdot x + b_k$$

- Any problems?

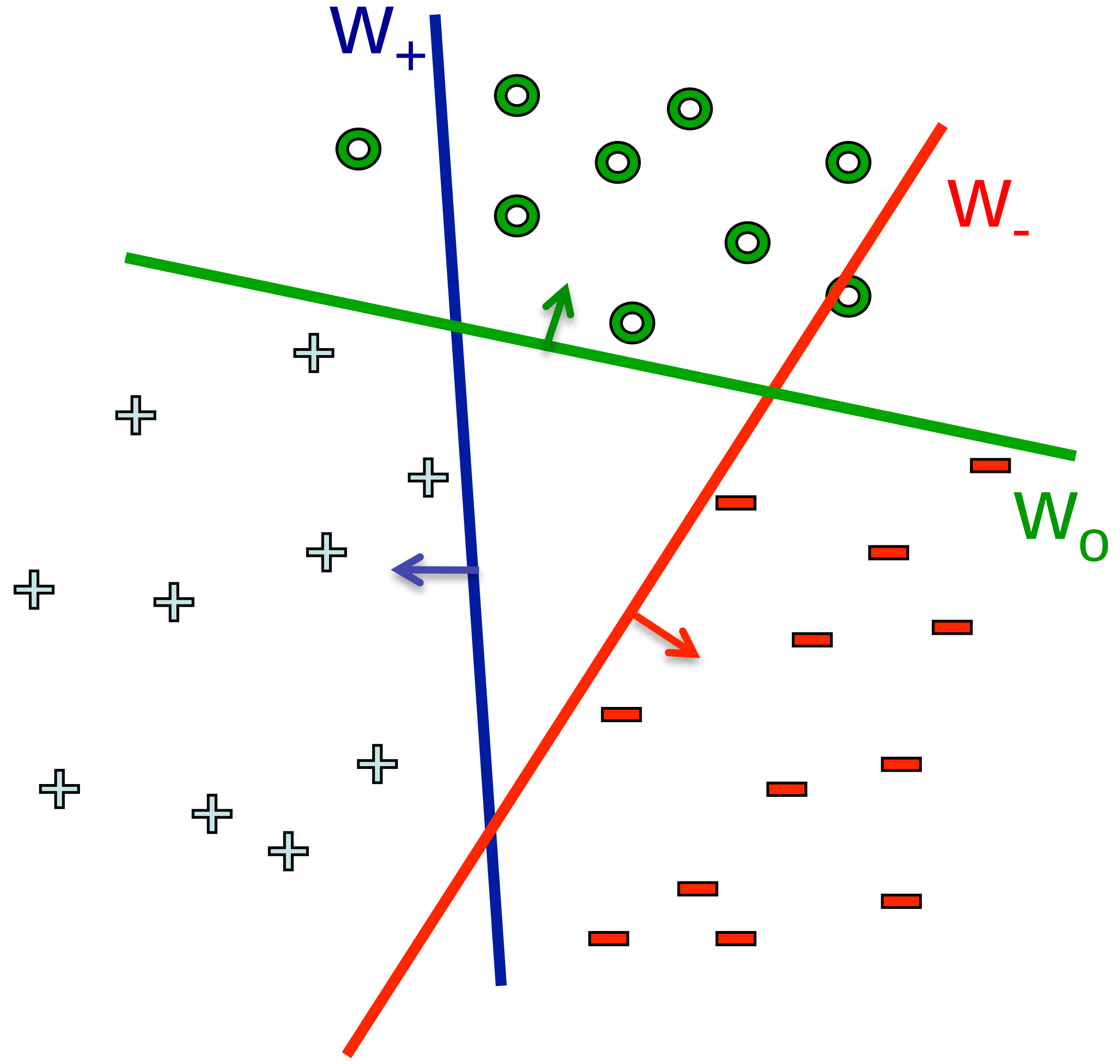
- Could we learn this dataset?



slide by Eric Xing

# Multi-class SVM

- Simultaneously learn 3 sets of weights:
- How do we guarantee the correct labels?
- Need new constraints!



The "score" of the correct class must be better than the "score" of wrong classes:

$$w^{(y_j)} \cdot x_j + b^{(y_j)} > w^{(y)} \cdot x_j + b^{(y)} \qquad \forall j,\ y \neq y_j$$

slide by Eric Xing

# Multi-class SVM

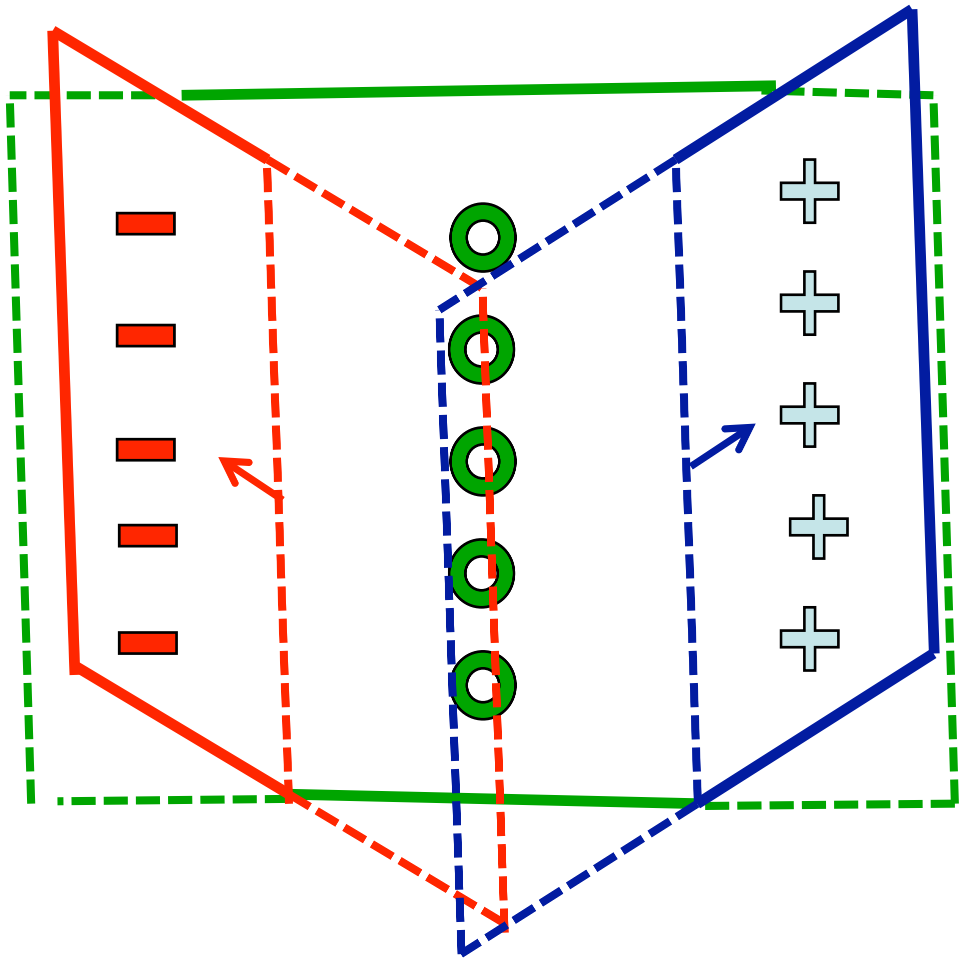
- As for the SVM, we introduce slack variables and maximize margin:

$$\begin{aligned} &\text{minimize}_{\mathbf{w},b} \quad \textstyle\sum_y \mathbf{w}^{(y)}.\mathbf{w}^{(y)} + C\sum_j \xi_j \\ &\mathbf{w}^{(y_j)}.\mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y')}.\mathbf{x}_j + b^{(y')} + 1 - \xi_j, \ \forall y' \neq y_j, \ \forall j \\ &\hfill \xi_j \geq 0, \ \forall j \end{aligned}$$

To predict, we use:

$$\hat{y} \leftarrow \arg\max_k \ w_k \cdot x + b_k$$

Now can we learn it? →



slide by Eric Xing

# Kernels

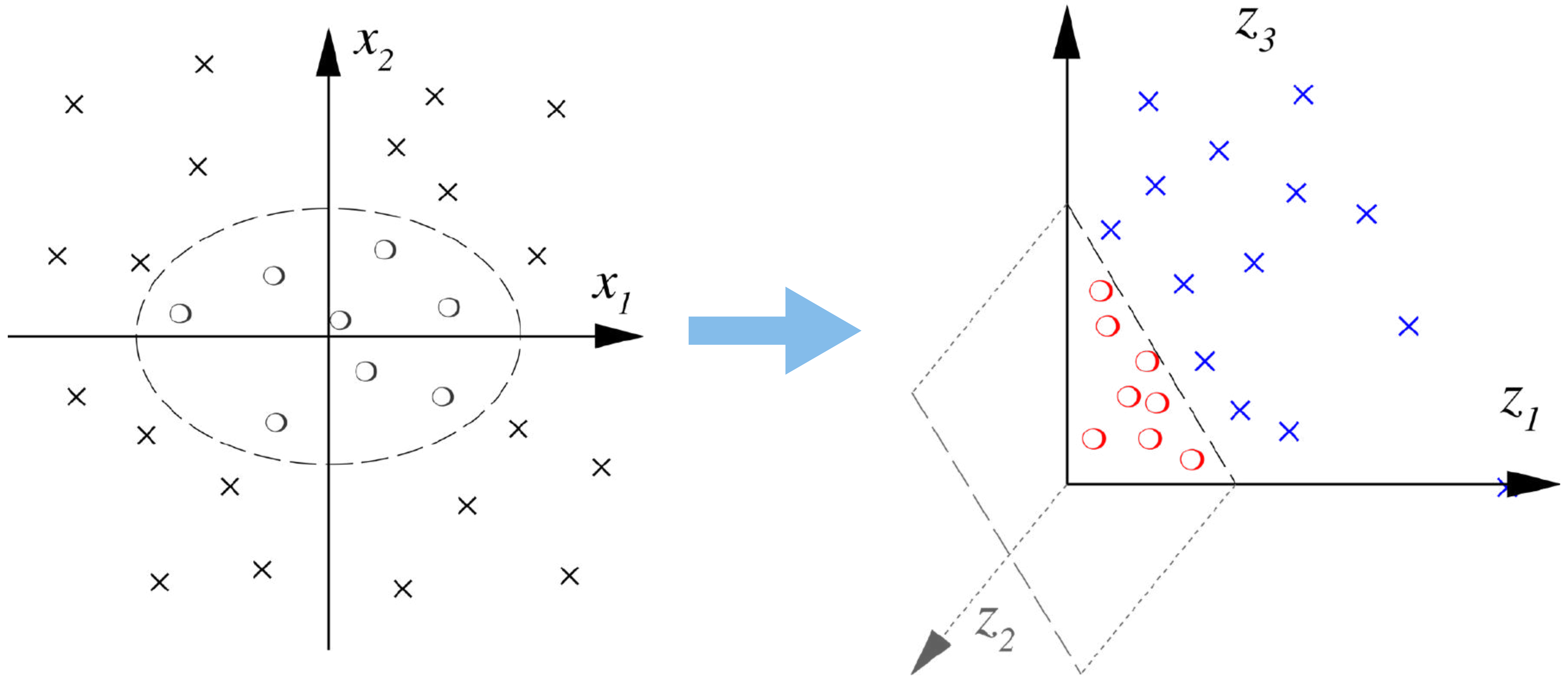
slide by Alex Smola

# Non-linear 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 $\phi(x_i)$

slide by Alex Smola

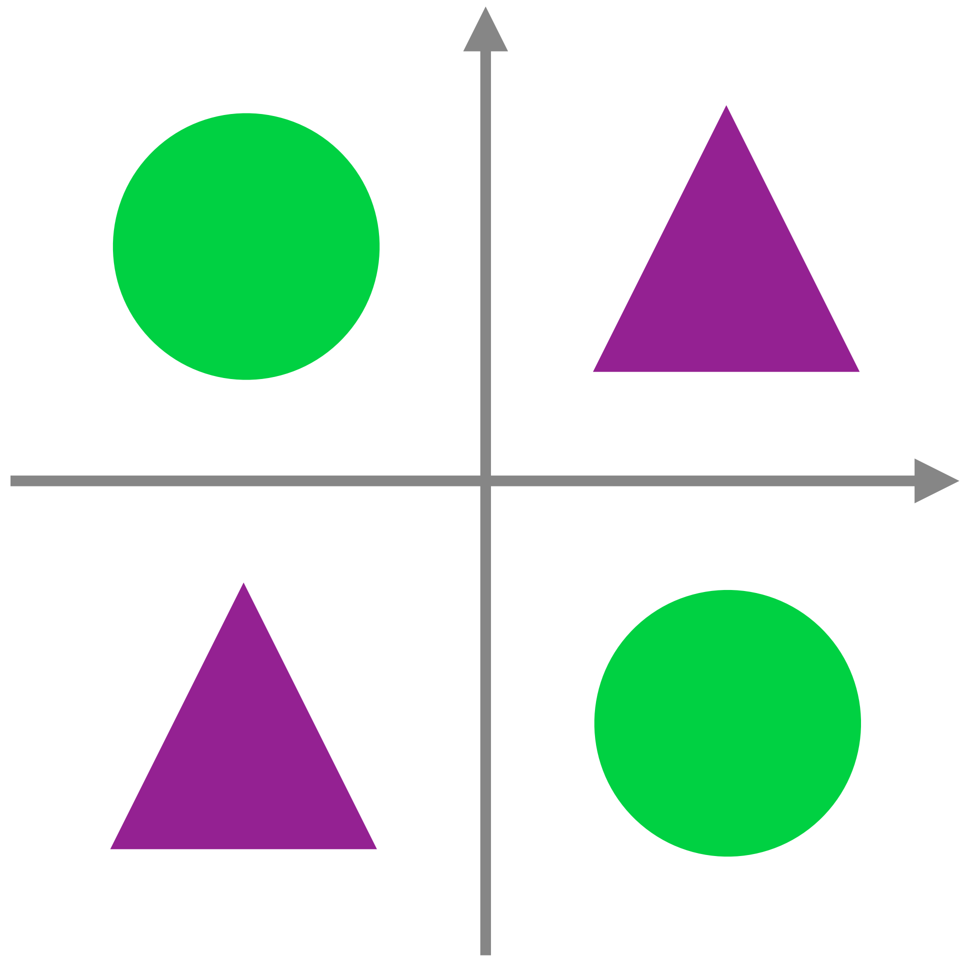
# Non-linear features

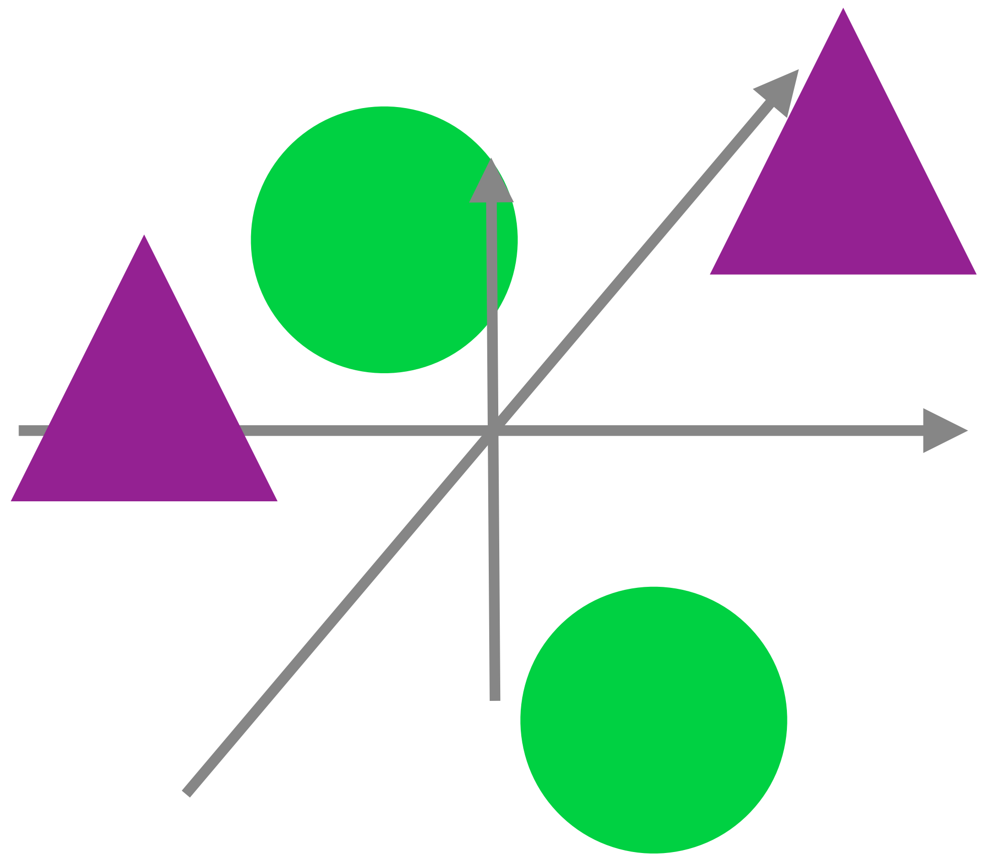


- Separating surfaces are Circles, hyperbolae, parabolae

slide by Alex Smola

# 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

slide by Alex Smola

# Quadratic Features

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

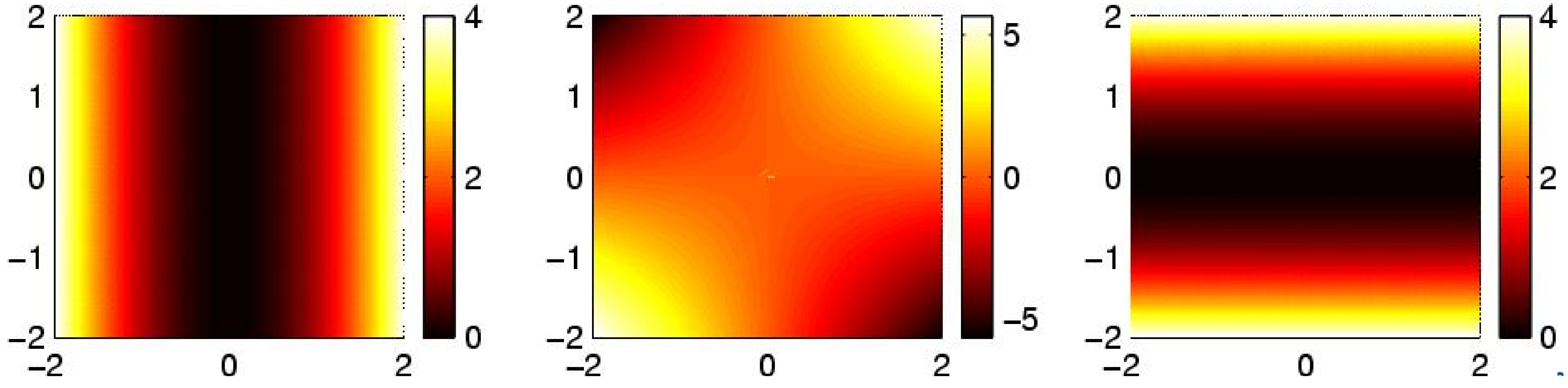
$$\Phi(x) := \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$

## Dot Product

$$\begin{aligned}\langle \Phi(x), \Phi(x')\rangle &= \left\langle \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right), \left({x'_1}^2, \sqrt{2}x'_1x'_2, {x'_2}^2\right)\right\rangle \\ &= \langle x, x'\rangle^2.\end{aligned}$$

## Insight

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



slide by Alex Smola

## 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 $\Phi(x)$ ...

slide by Alex Smola

# Recap: The Perceptron

```
**initialize** $w = 0$ and $b = 0$
**repeat**
  **if** $y_i [\langle w, x_i \rangle + b] \leq 0$ **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 $f(x) = \sum_{i \in I} y_i \langle x_i, x \rangle + b$

slide by Alex Smola

# Recap: 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
        $w' = w + y_i \Phi(x_i)$
        $b' = b + y_i$
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 $f(x) = \sum_{i \in I} y_i \langle \phi(x_i), \phi(x) \rangle + b$

slide by Alex Smola

# The Kernel Perceptron

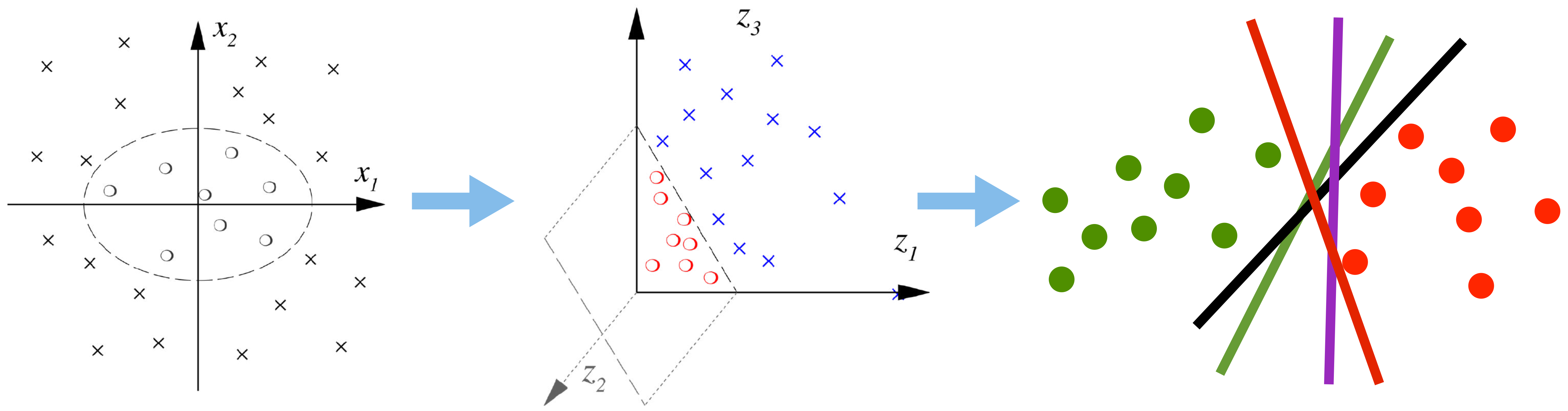
```
initialize $f = 0$
repeat
    Pick $(x_i, y_i)$ from data
    if $y_i f(x_i) \leq 0$ then
        $f(\cdot) \leftarrow f(\cdot) + y_i k(x_i, \cdot) + y_i$
until $y_i f(x_i) > 0$ for all $i$
```

- 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

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

slide by Alex Smola

# Processing Pipeline



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

slide by Alex Smola

# 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 $\Phi_i(x)$.

slide by Alex Smola

## 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 $\Phi$ such that $k$ really is a dot product?

slide by Alex Smola

# 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

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

slide by Alex Smola

# 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.

slide by Alex Smola

# 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

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

# Examples

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

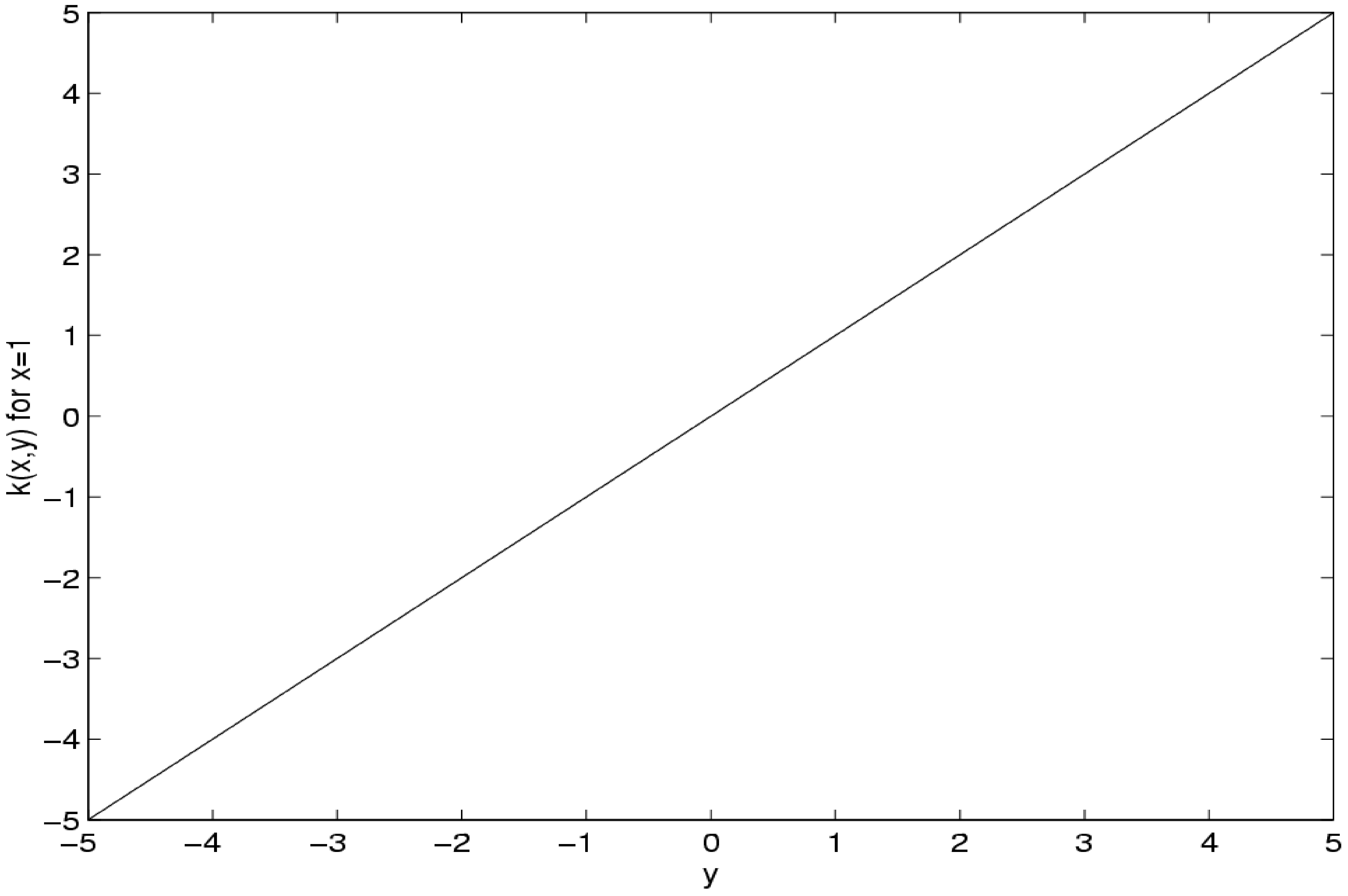
| | |
|---|---|
| Linear | $\langle x, x' \rangle$ |
| Laplacian RBF | $\exp\left(-\lambda \lVert x - x' \rVert\right)$ |
| Gaussian RBF | $\exp\left(-\lambda \lVert x - x' \rVert^2\right)$ |
| Polynomial | $\left(\langle x, x' \rangle + c\rangle\right)^d, c \geq 0,\ d \in \mathbb{N}$ |
| B-Spline | $B_{2n+1}(x - x')$ |
| 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.

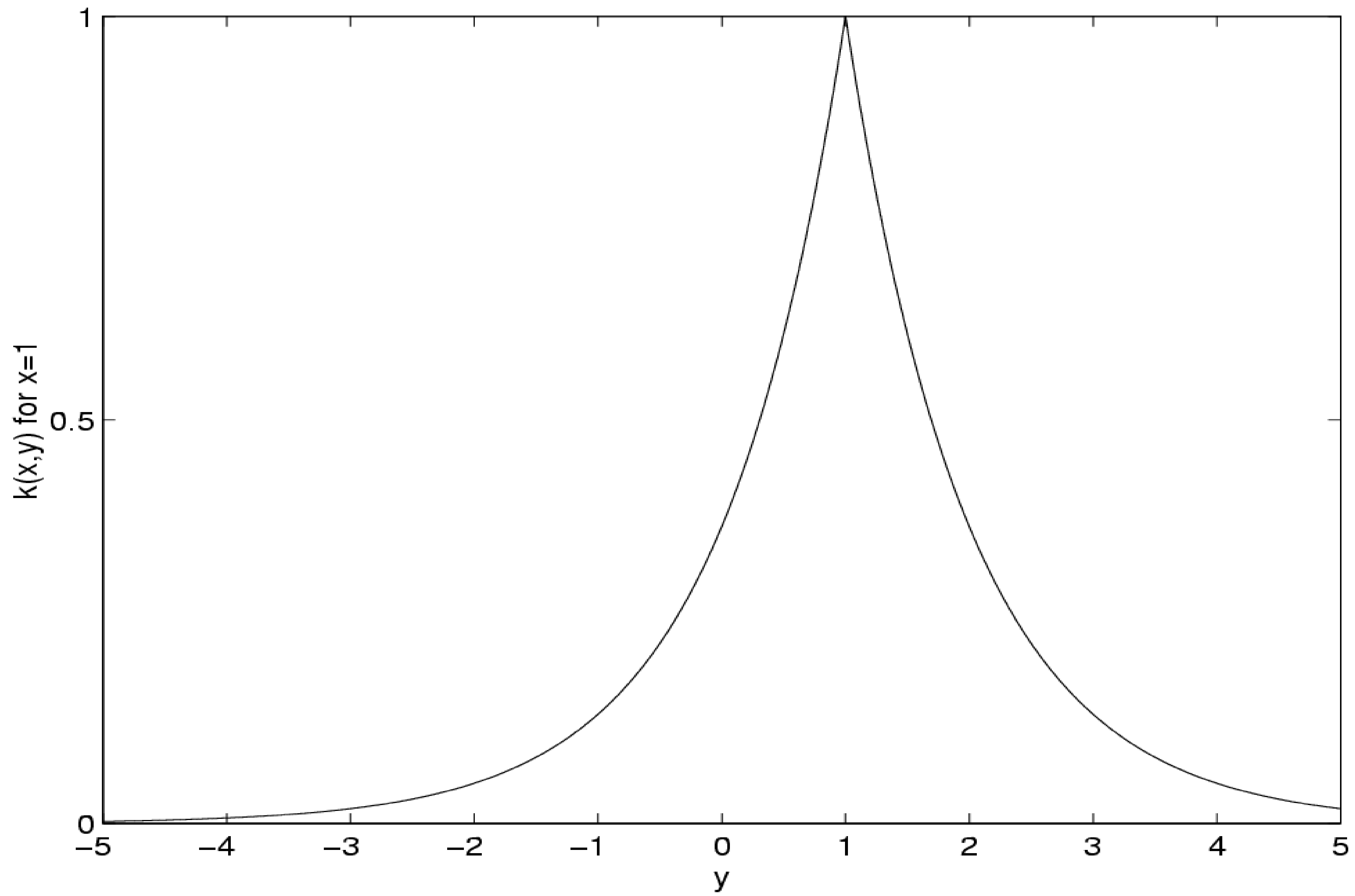
slide by Alex Smola

# Linear Kernel



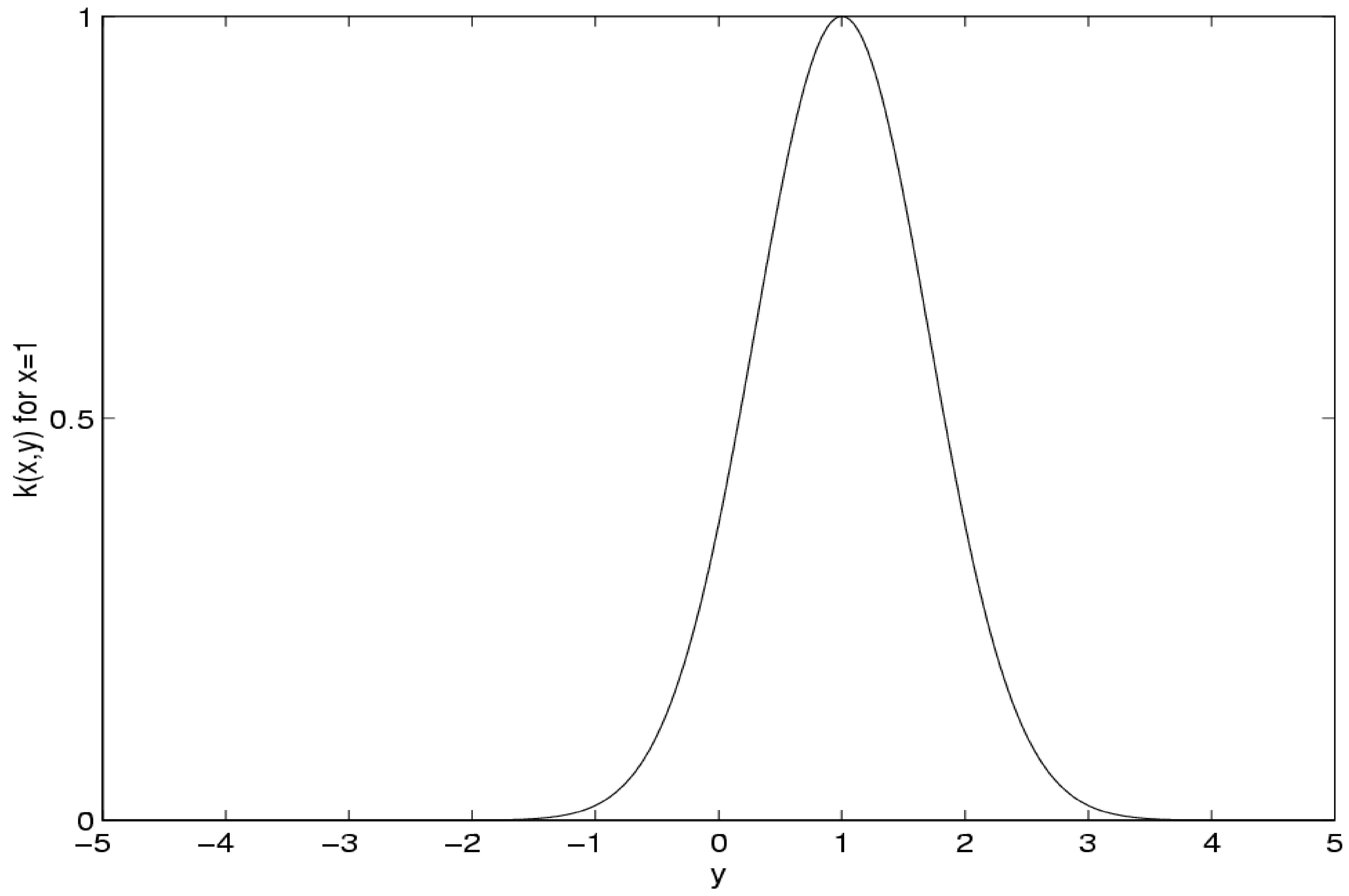
slide by Alex Smola

# Laplacian Kernel



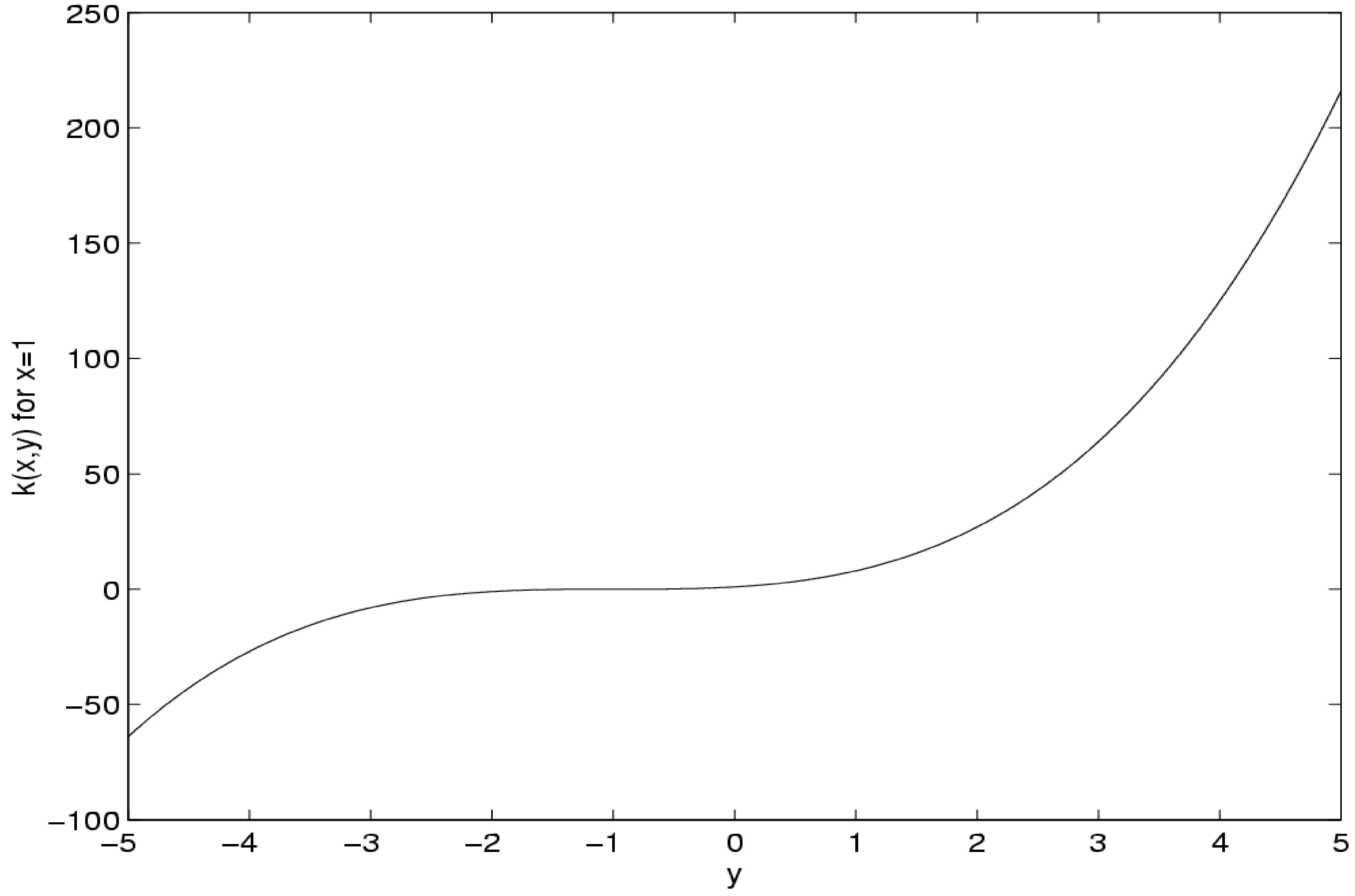
slide by Alex Smola

# Gaussian Kernel



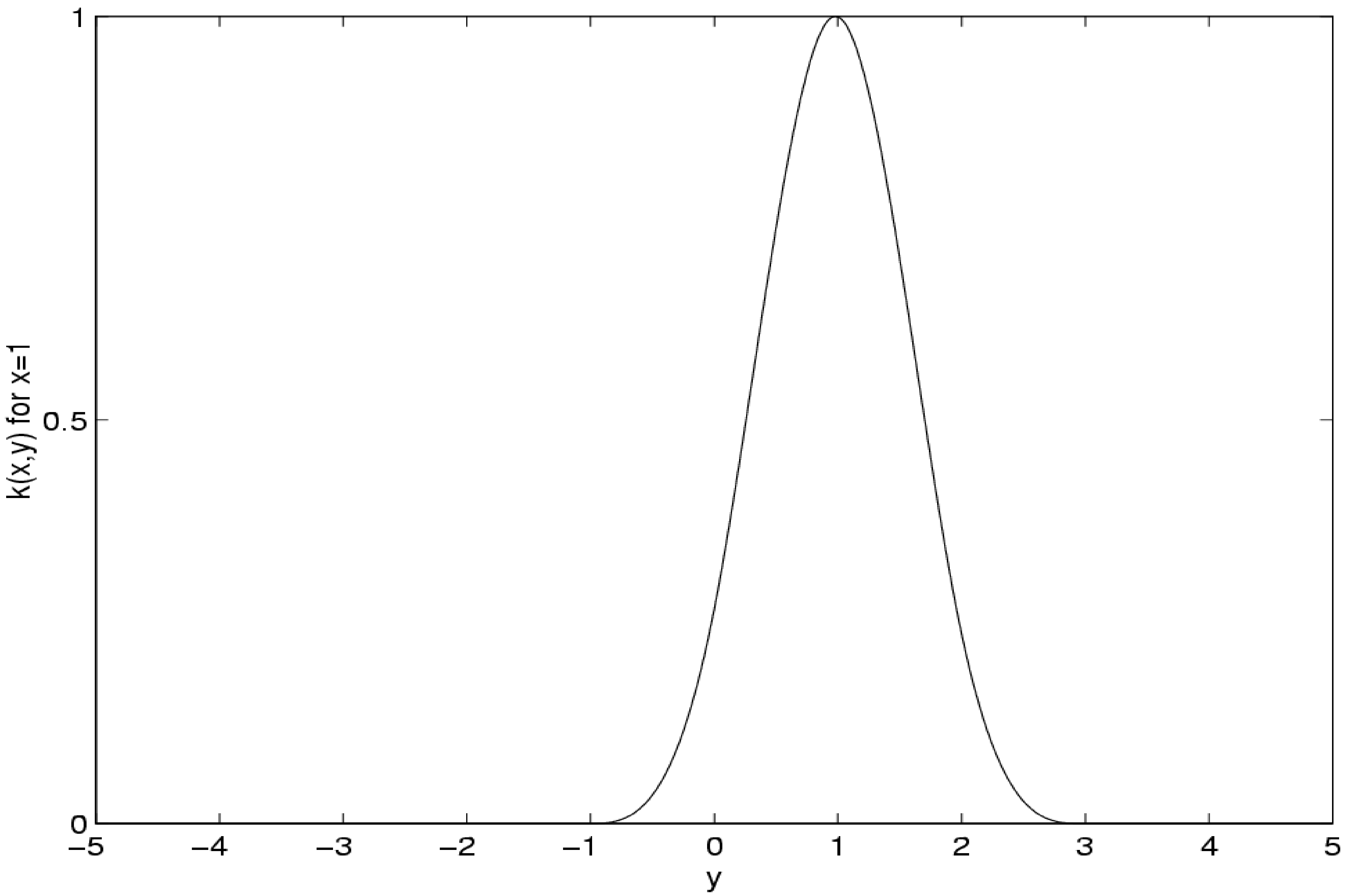
slide by Alex Smola

# Polynomial of order 3



slide by Alex Smola

# $B_3$ Spline Kernel



slide by Alex Smola

# Kernels in Computer Vision

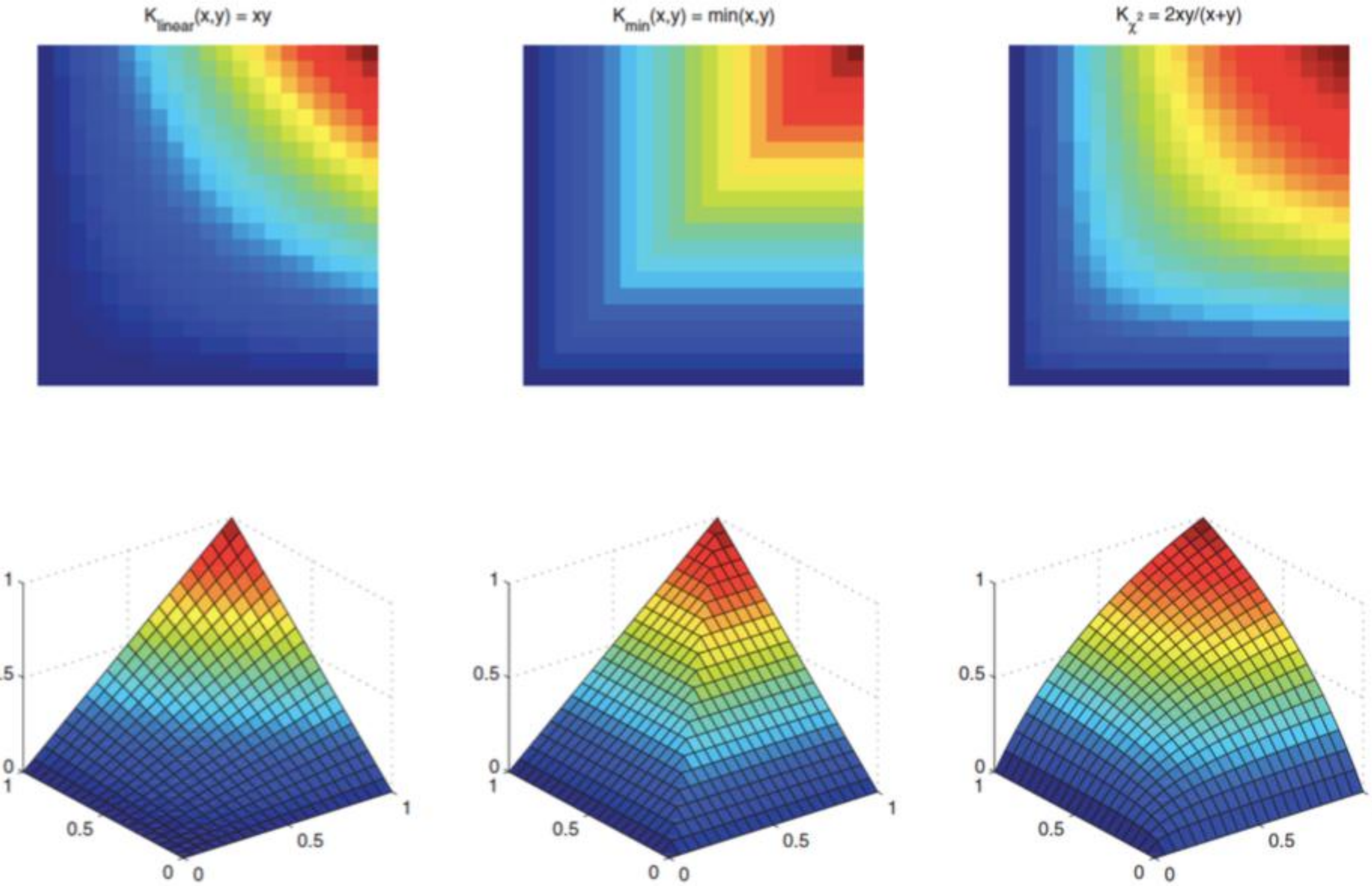
- Features x = histogram (of color, texture, etc)

- Common Kernels
  - Intersection Kernel
  - Chi-square Kernel

$$K_{\text{intersect}}(\boldsymbol{u}, \boldsymbol{v}) = \sum_i \min(u_i, v_i)$$

$$K_{\chi^2}(\boldsymbol{u}, \boldsymbol{v}) = \sum_i \frac{2u_i v_i}{u_i + v_i}$$

slide by Dhruv Batra



slide by Dhruv Batra

Image credit: Subhransu Maji

## **Next Lecture:**

Kernel Trick for SVMs,
Support Vector Regression