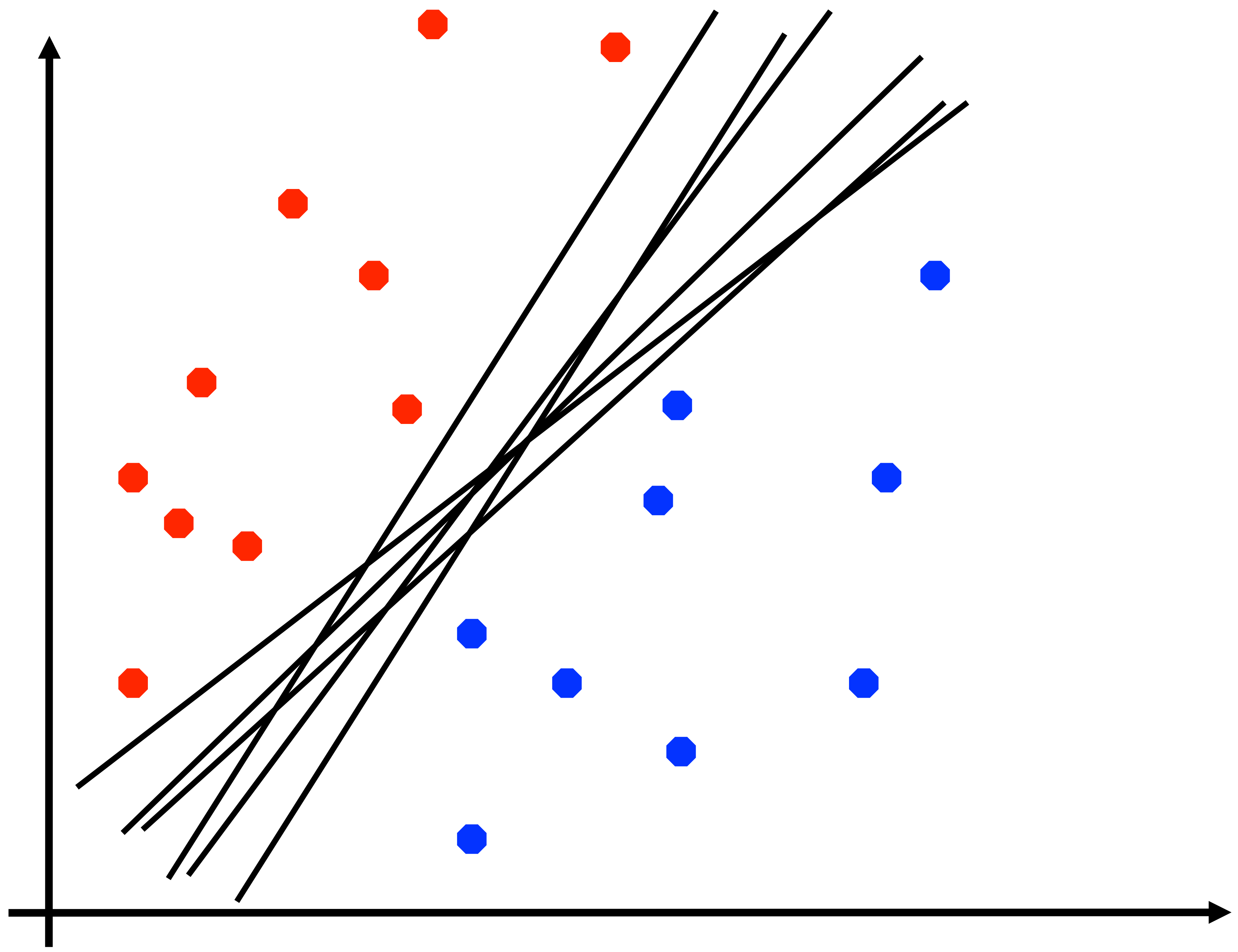
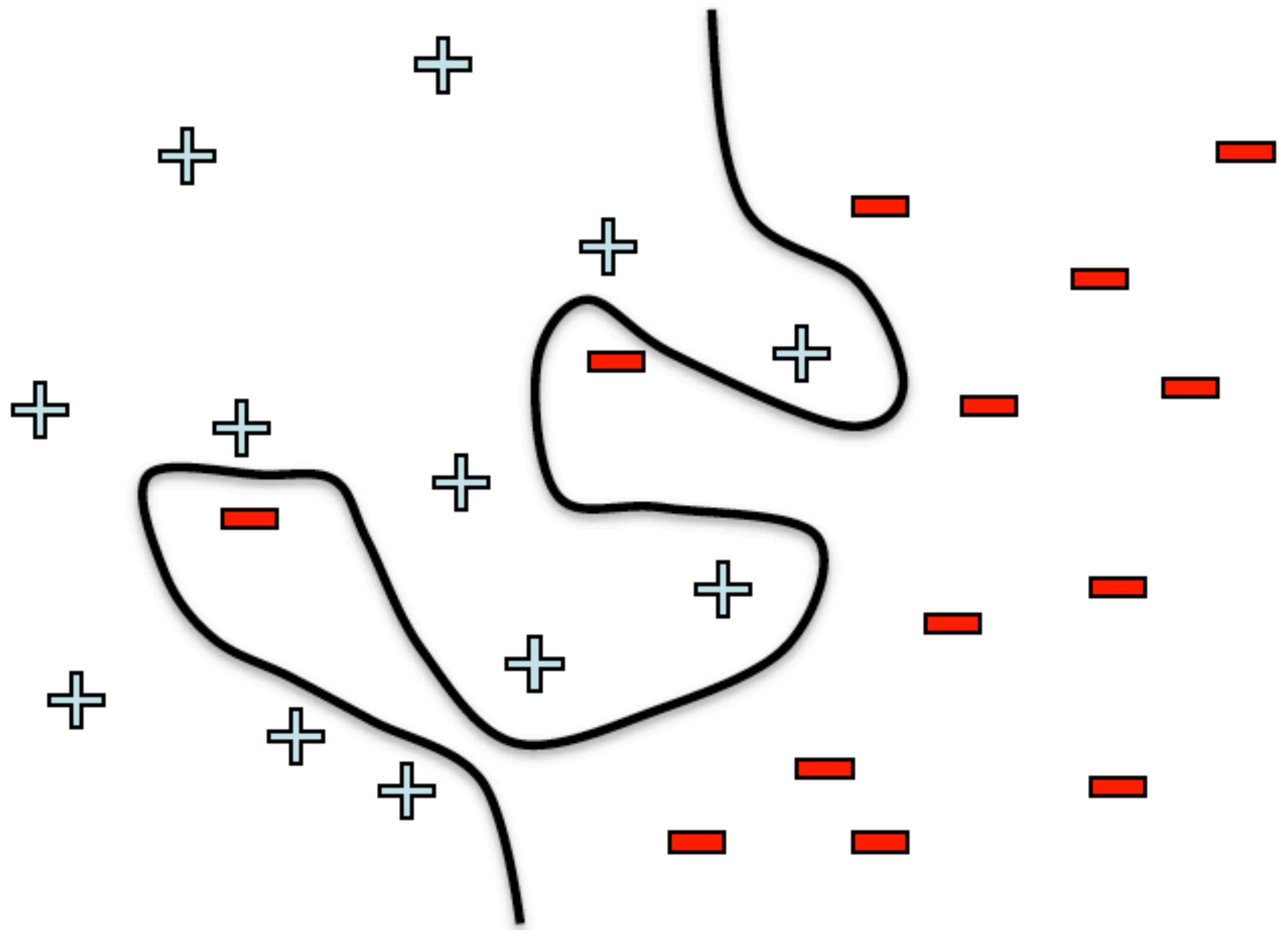
# Today

- Last time
  - SVMs with slack (reading: Bishop, Sec 7.1.1, Notes by Andrew Ng http://cs229.stanford.edu/notes/cs229-notes3.pdf )
- Today
  - SVMs with kernels (reading: Bishop Sec 7.1)
  - Multiclass classification (reading: Bishop, Sec 7.1.3)
- Announcements
  - HW 2 due on Wed, Oct 14
  - Midterm exam is coming up (Wed, Oct 21)
    - Time for Q&A on Mon, Oct 19 in lecture
    - Office hours: Mondays from 9-10am, Wednesdays 12.15-1.15pm
    - Current plan: Midterm online via GradScope (more details to come)
    - Tell me asap if you cannot take the exam on Wed, Oct 21 from 11am-12.15pm because of timezone conflicts

# SVMs with kernels

# From linear to nonlinear decision boundary





[David Sontag]

# Motivation

**board**

# Least-squares regression with polynomials

- First-order polynomial

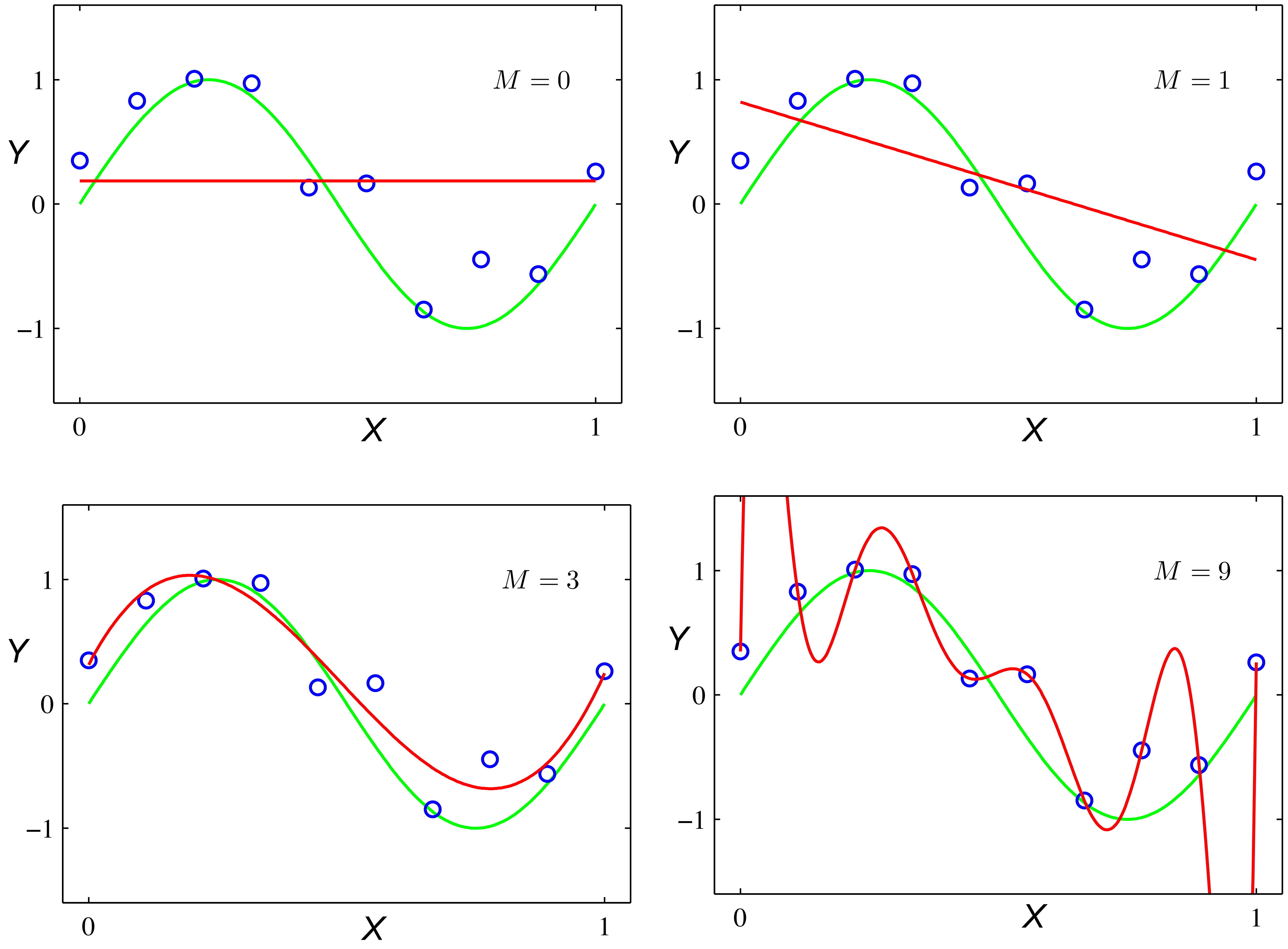
  $h_\theta(x) = \theta_1 x + \theta_0$
- Second-order poly

  $h_\theta(x) = \theta_2 x^2 + \theta_1 x + \theta_0$
  - Write as

    $h_\theta(z) = \boldsymbol{\theta}^T \boldsymbol{z}$

    with

    $\boldsymbol{z} = [x^2, x, 1]$
  - Interpretation of fitting higher-order polynomial: Perform regression on features $z = [x^M, \ldots, x, 1]$ rather than on $x$



# Feature map

- Define a feature map $\phi : \mathbb{R}^n \to \mathbb{R}^k, k \in \mathbb{N} \cup \{\infty\}$
- Polynomial regression with degree 3

$$\phi(x) = \begin{bmatrix} x^3 \\ x^2 \\ x \\ 1 \end{bmatrix}$$

- Let's apply **linear** SVM to $\phi(x)$ instead of $x$
- Potential problem?
  - The dimension $k$ potentially very (infinite) high
  - Goal: Avoid operating in $\mathbb{R}^k$

# Recall: Inner product and SVMs

Solve:

$$\max_{\alpha} \quad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)}, x^{(j)} >$$

s.t. $\quad \alpha_i \geq 0 \, , i = 1, \ldots, N$

$$\sum_{i=1}^{N} \alpha_i y^{(i)} = 0$$

Predict:

$$\sum_{i=1}^{N} \alpha_i^* y^{(i)} < (x^{(i)})^T, x > + b$$

- **Solve and predict "touch" training data only via inner products**
- This is key for using SVMs with kernels

# Kernel

- Define the kernel

$$K(x, z) = \phi(x)^T \phi(z) = \langle \phi(x), \phi(z) \rangle$$

- Walk through the SVM algorithm and replace $\langle x, z \rangle$ with

$$\langle \phi(x), \phi(z) \rangle$$

- **Kernel trick:** Compute $K(x, z) = \langle \phi(x), \phi(z) \rangle$ without computing $\phi(x)$ and $\phi(z)$ (high-dimensional quantities)

- Example
  - $K(x, z) = (x^T z)^2$
  - $K(x, z) = (x^T z + c)^2$

# Example: quadratic kernel

**board**

# Kernel matrix

- Data set $\mathscr{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$
- Kernel matrix
$$K_{ij} = K(x^{(i)}, x^{(j)})$$
- Properties of kernel matrix
  - Symmetric: $K_{ij} = K_{ji}$
  - Positive semi-definite: $\langle K\boldsymbol{x}, \boldsymbol{x}\rangle \geq 0\,, \boldsymbol{x} \in \mathbb{R}^N$
- Theorem (Mercer)
*Let $K : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ be given, then we call $K$ a kernel if the kernel matrix $\boldsymbol{K}$ for any set $\{x^{(1)}, \ldots, x^{(N)}\}, N < \infty$ is symmetric positive semi-definite*

# Common kernels

- Polynomials of degree exactly *d*

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

- Polynomials of degree up to *d*

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$$

**Note that kernels may depend on parameters, e.g., $\sigma$ in case of Gaussian Kernel**

- Gaussian kernels

$$K(\vec{u}, \vec{v}) = \exp\left(-\frac{||\vec{u} - \vec{v}||_2^2}{2\sigma^2}\right)$$

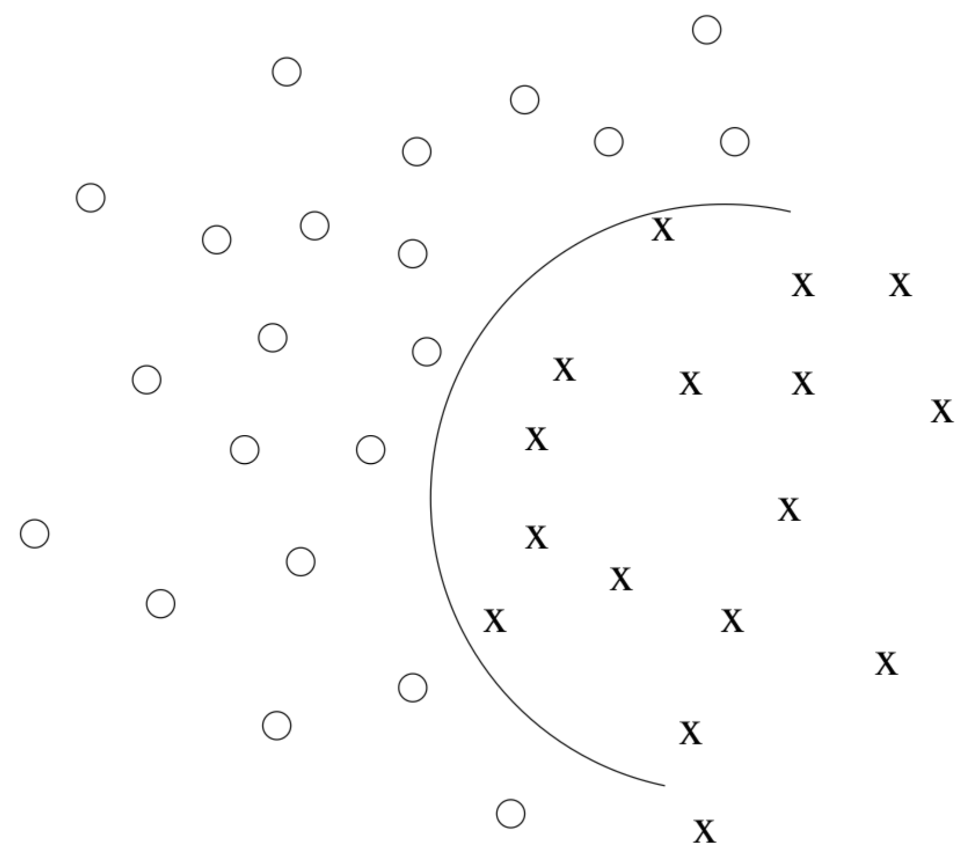
- Sigmoid

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

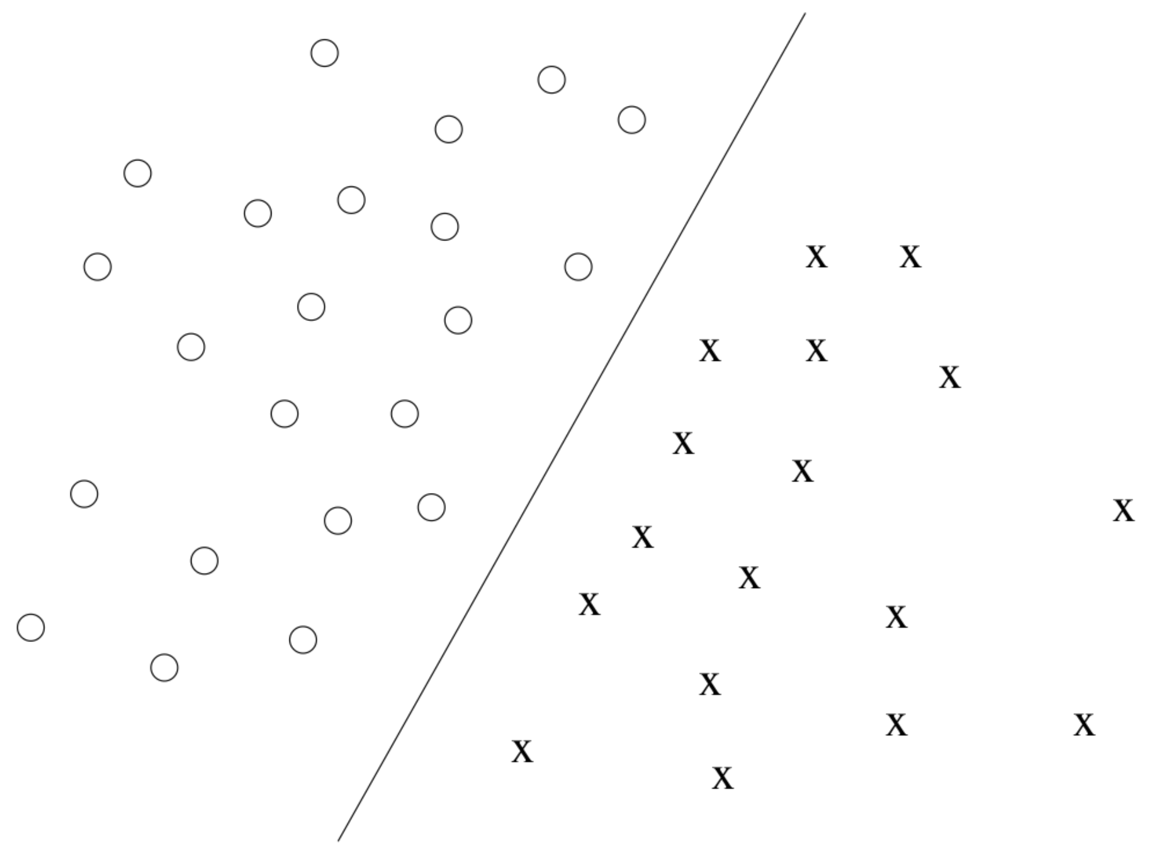
- And many others: very active area of research!

[David Sontag]

# Quadratic kernel



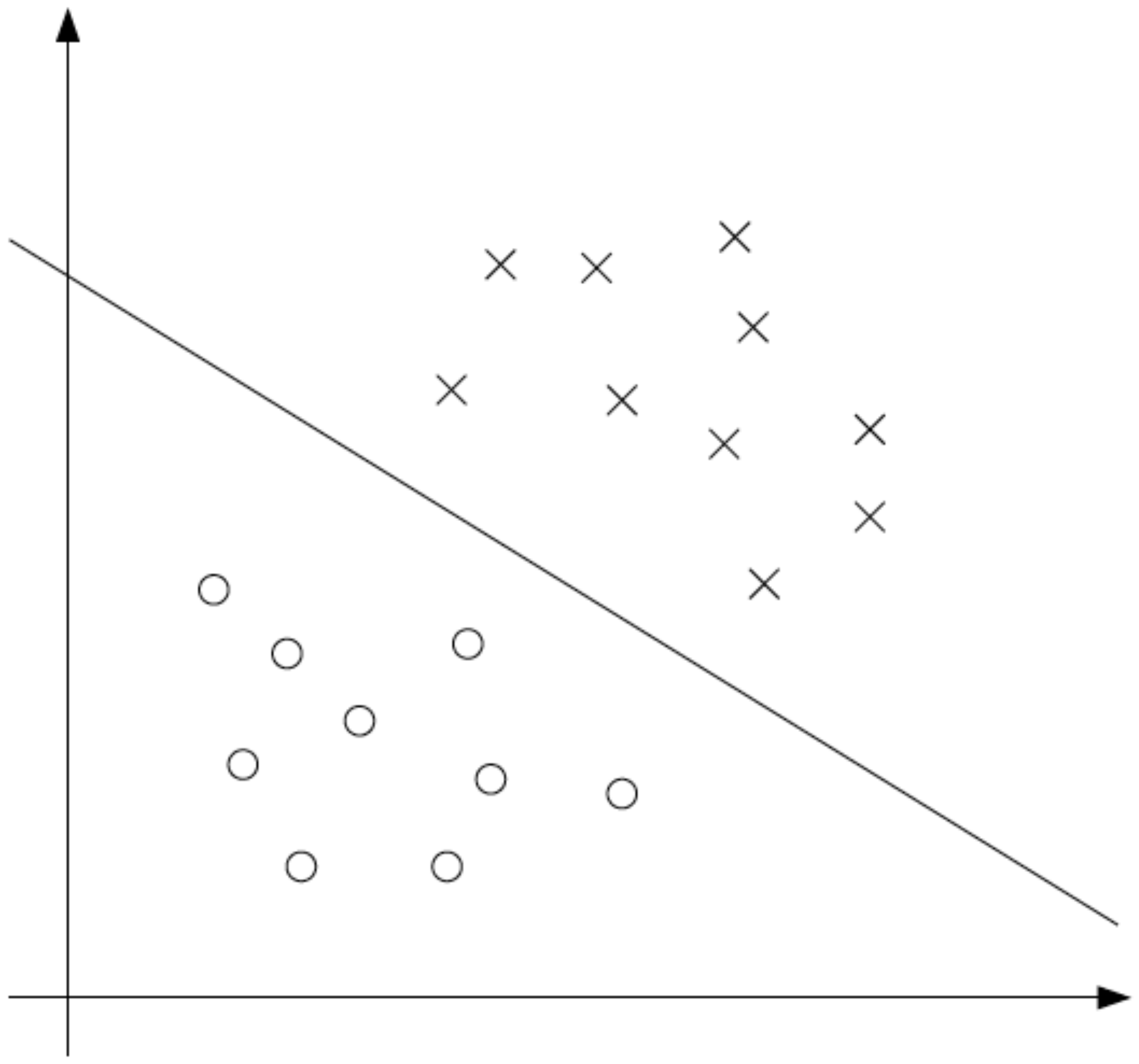
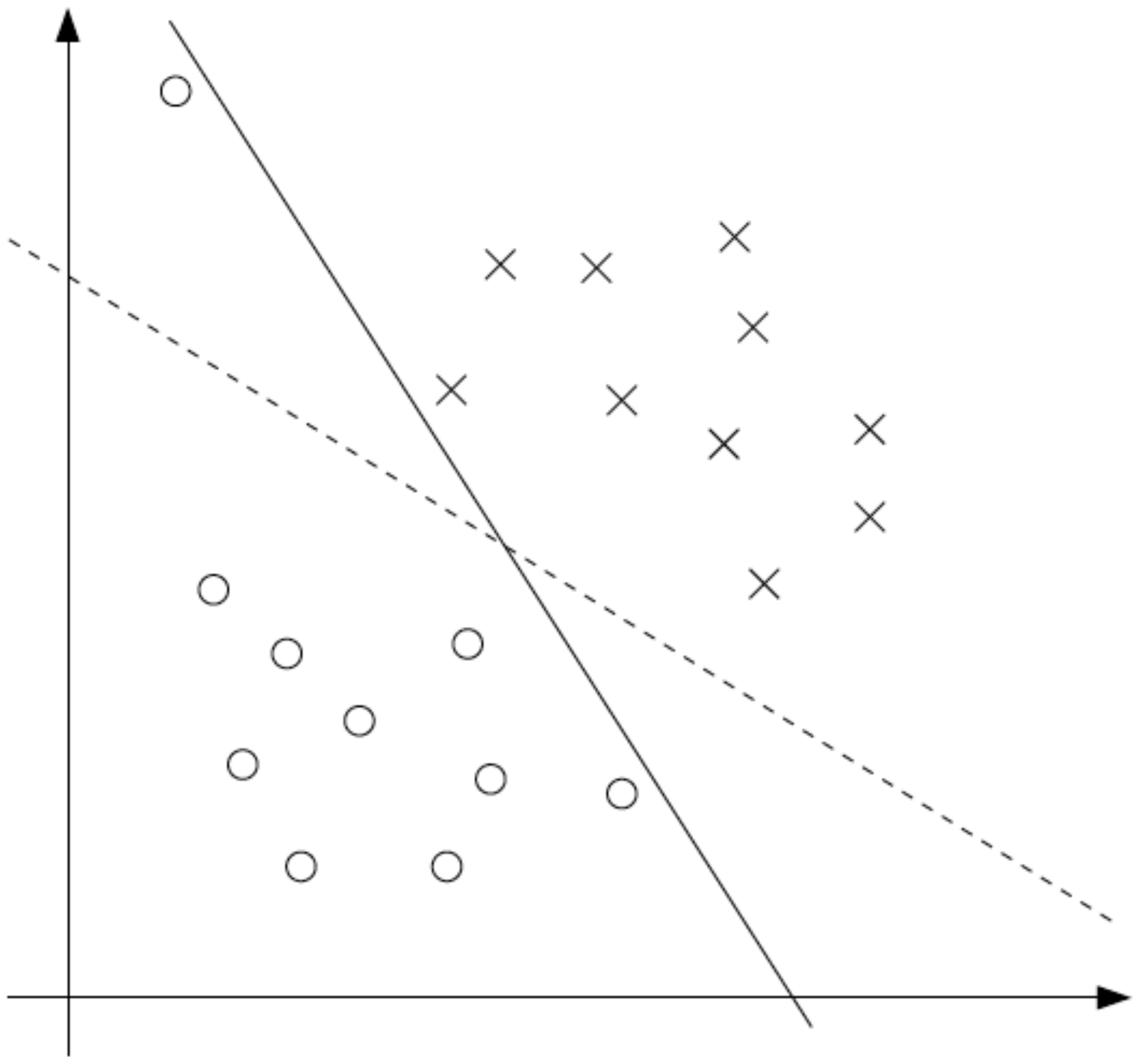
Non-linear separator in the original $\mathbf{x}$-space



Linear separator in the feature $\phi$-space

[Tommi Jaakkola]

# Why slack variables if we have kernels?

- No guarantee that separable in high-dimensional feature space
- One outlier can lead to dramatic change in decision boundary
  - Slack variables (cf. hinge loss) helps to prevent this

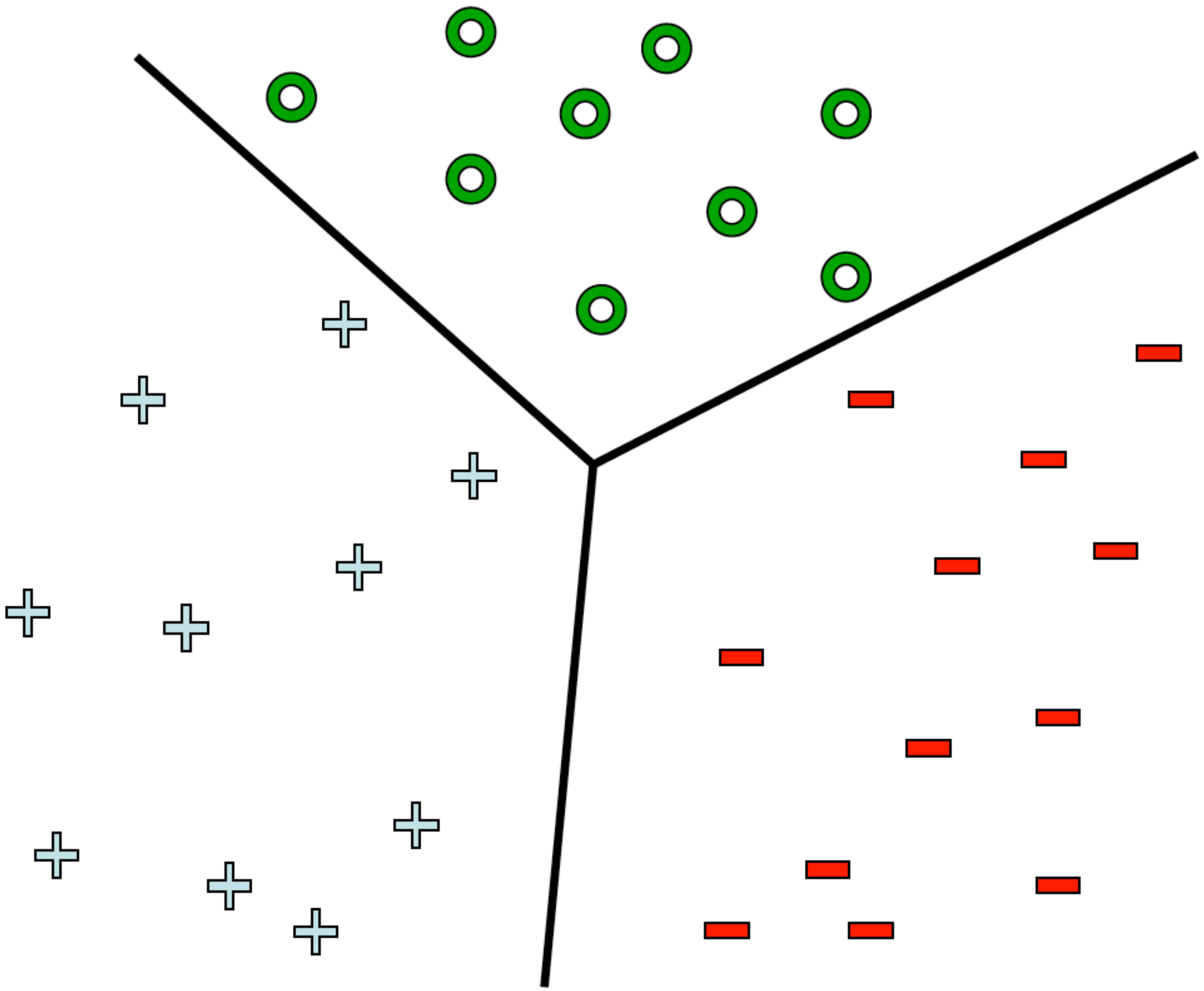
[Andrew Ng]

# SVM with kernels

- Given is training data $\mathscr{D}$
- Choose a feature map $\phi$ with kernel $K$
- Apply linear SVM (w/out slack) to $\phi(\mathscr{D})$ by exploiting dual formulation and that $K(x, y)$ is inner product in feature space given by $\phi$

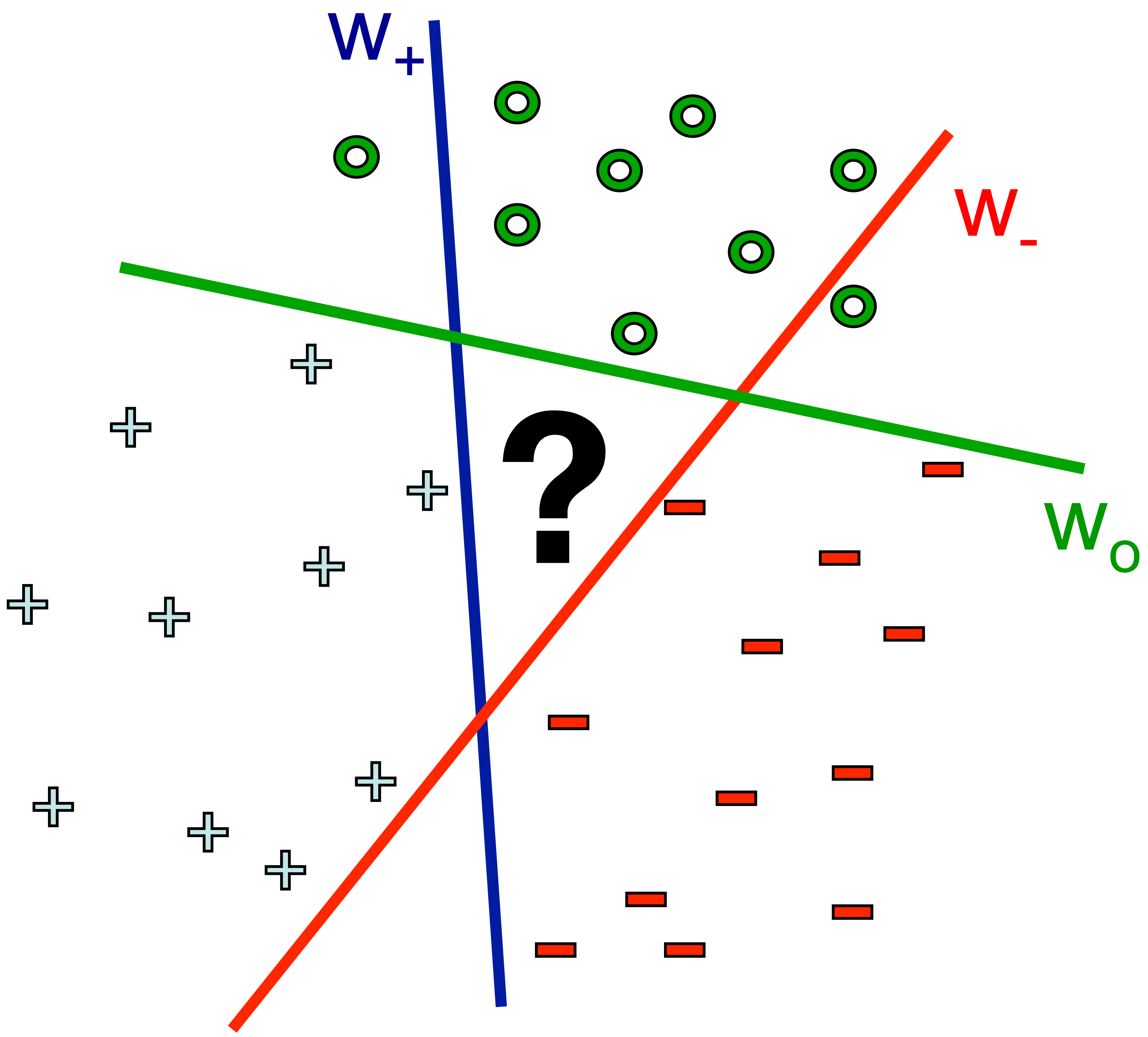
**-> homework HW3**

# Multi-class classification



[David Sontag]

# 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?

**What is the class of points in the triangle in the middle?**

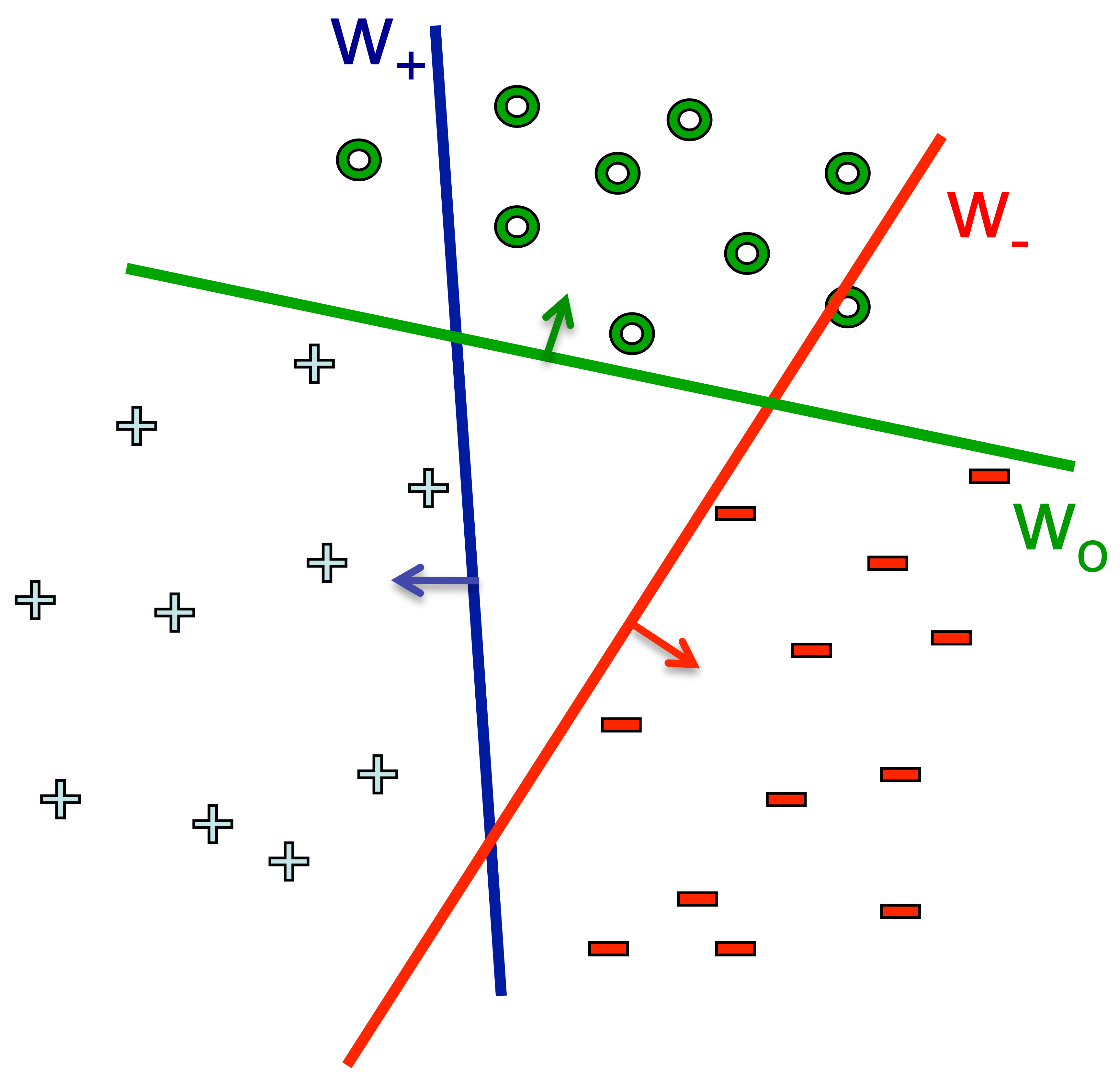
[David Sontag]

# 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:

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

[David Sontag]

# Multi-class SVM

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

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

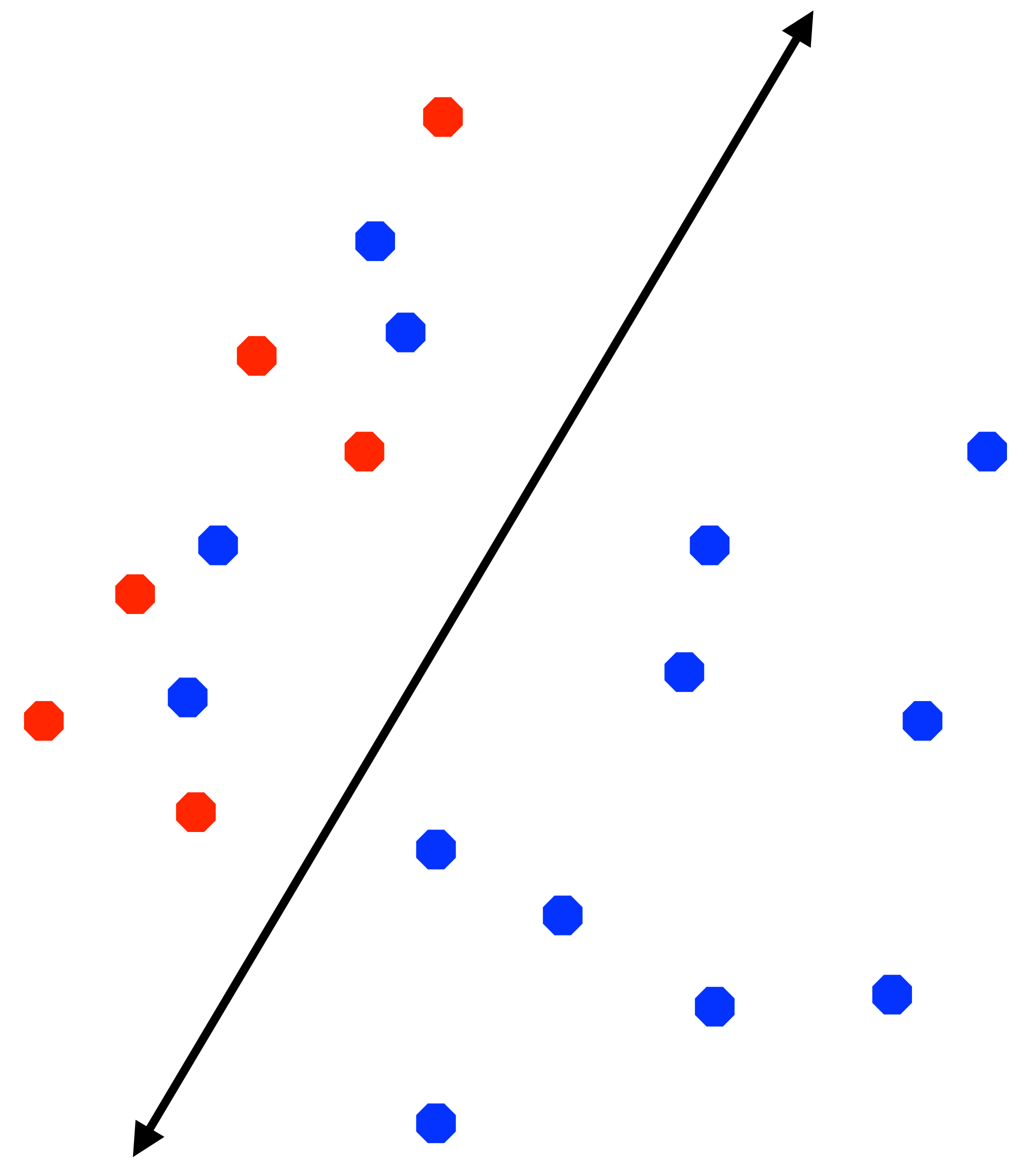
To predict, we use:

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

**Note: number of unknowns in optimization problem grows with number of classes $K$ because learn multiple weight vectors and biases**

[David Sontag]

# How to deal with imbalanced data?



- In many practical applications we may have **imbalanced** data sets
- We may want errors to be equally distributed between the positive and negative classes
- A slight modification to the SVM objective does the trick!

$$N = N_+ + N_-$$

$$\min_{w,b} \ ||w||_2^2 + \frac{CN}{2N_+} \sum_{j:y_j=+1} \xi_j + \frac{CN}{2N_-} \sum_{j:y_j=-1} \xi_j$$

**Note: #classes = 2**
**can be ignored if $C$ is scaled instead**

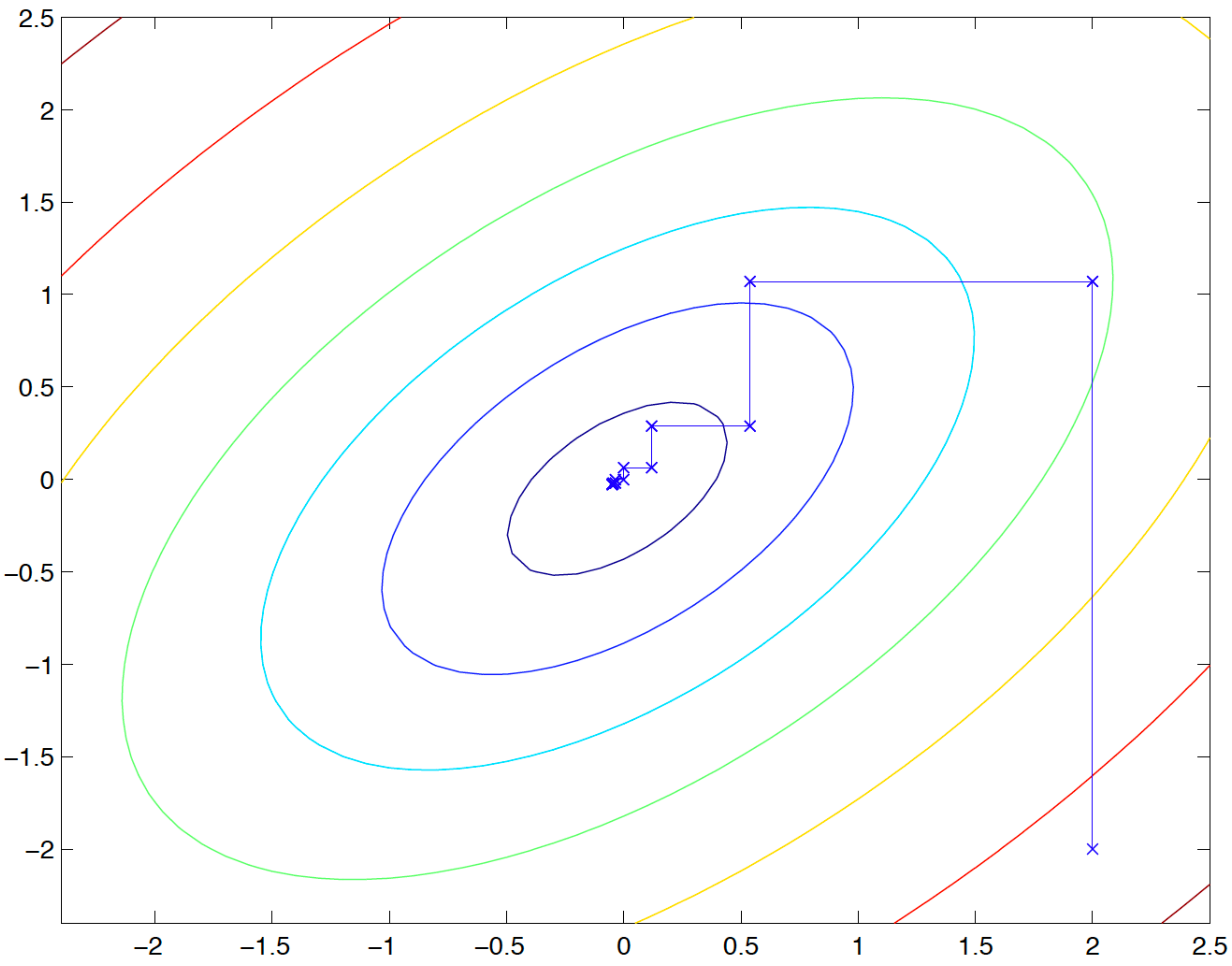
Class-specific weighting of the slack variables

[David Sontag]

# Algorithm for solving SVM problems

**board**

# Coordinate descent




[Andrew Ng]

# SMO algorithm

**board**