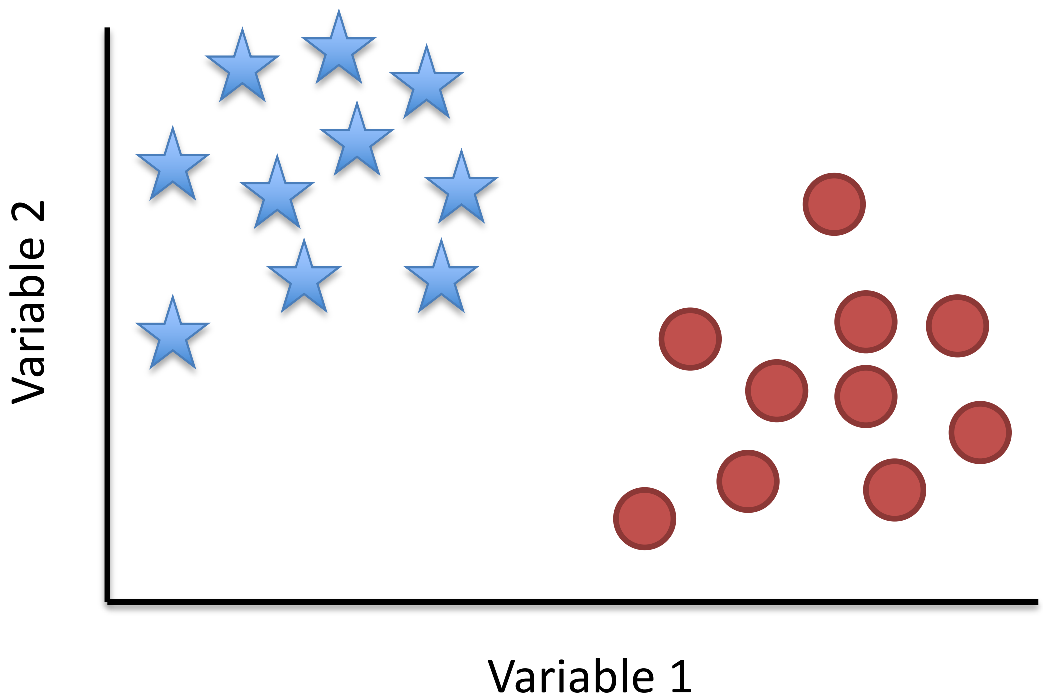
Data Mining: Classification Support Vector Machines (SVMs)

Laura Brown

Some slides adapted from P. Smyth; A. Moore, D. Klein, S. Russell, M. Wellman, Han, Kamber, & Pei; C.F. Aliferis Tan, Steinbach, & Kumar; and L. Kaebling

Some slide material from AMIA 2008 Tutorial, A. Statnikov, et al., "SVMs without Tears"

Support Vector Machines (SVMs)

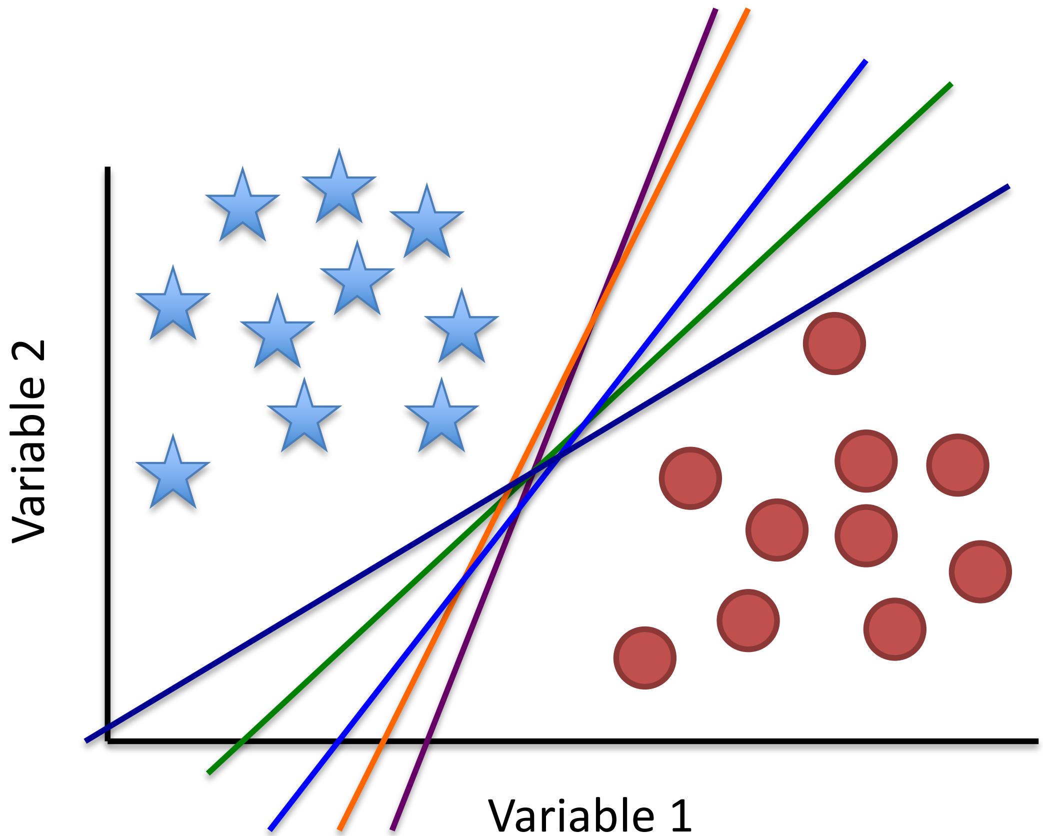


- How can you classify this data?

- Identify linear hyperplane (decision boundary) to separate data

$$f(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b)$$

Support Vector Machines (SVMs)



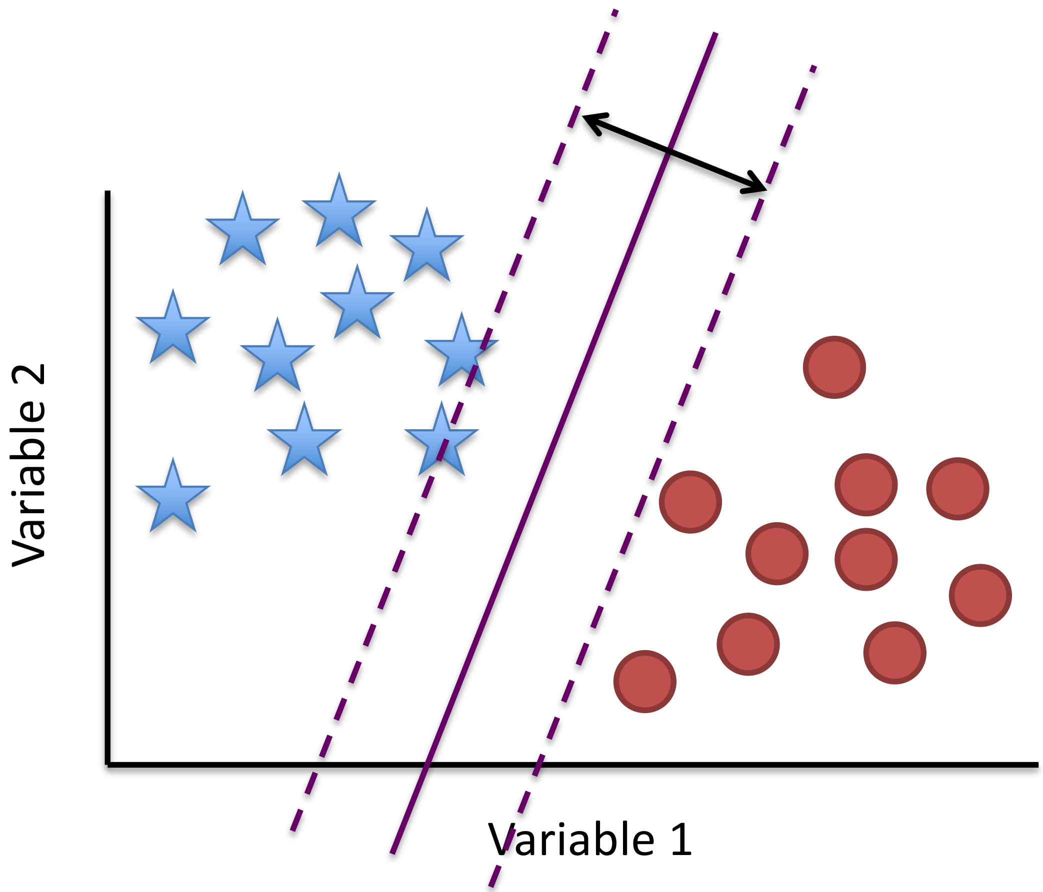
- How can you classify this data?

- Identify linear hyperplane (decision boundary) to separate data

$$f(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b)$$

- which is best?

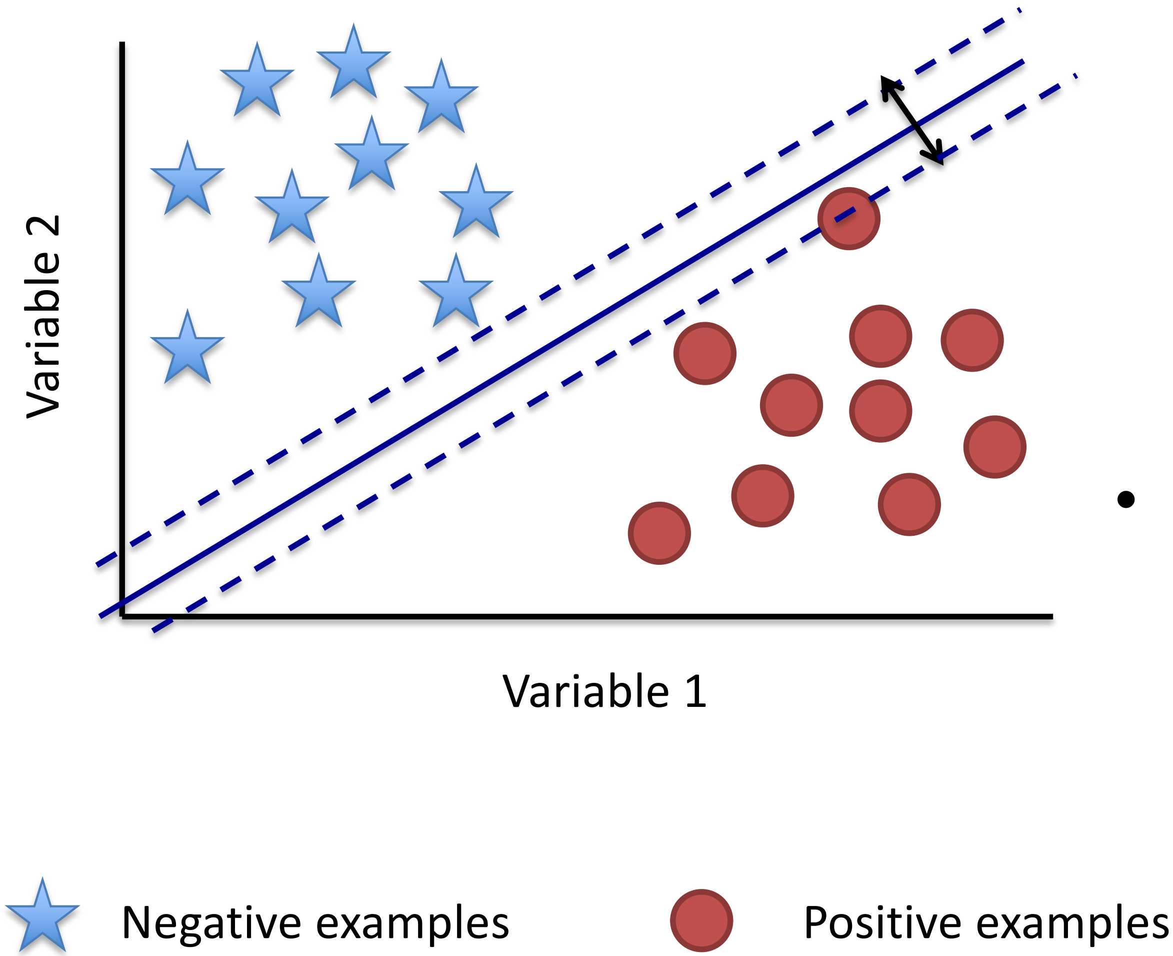
Support Vector Machines (SVMs)



- Define the margin of a linear classifier – the width that the boundary could be increased by before hitting a data sample

- which is best?

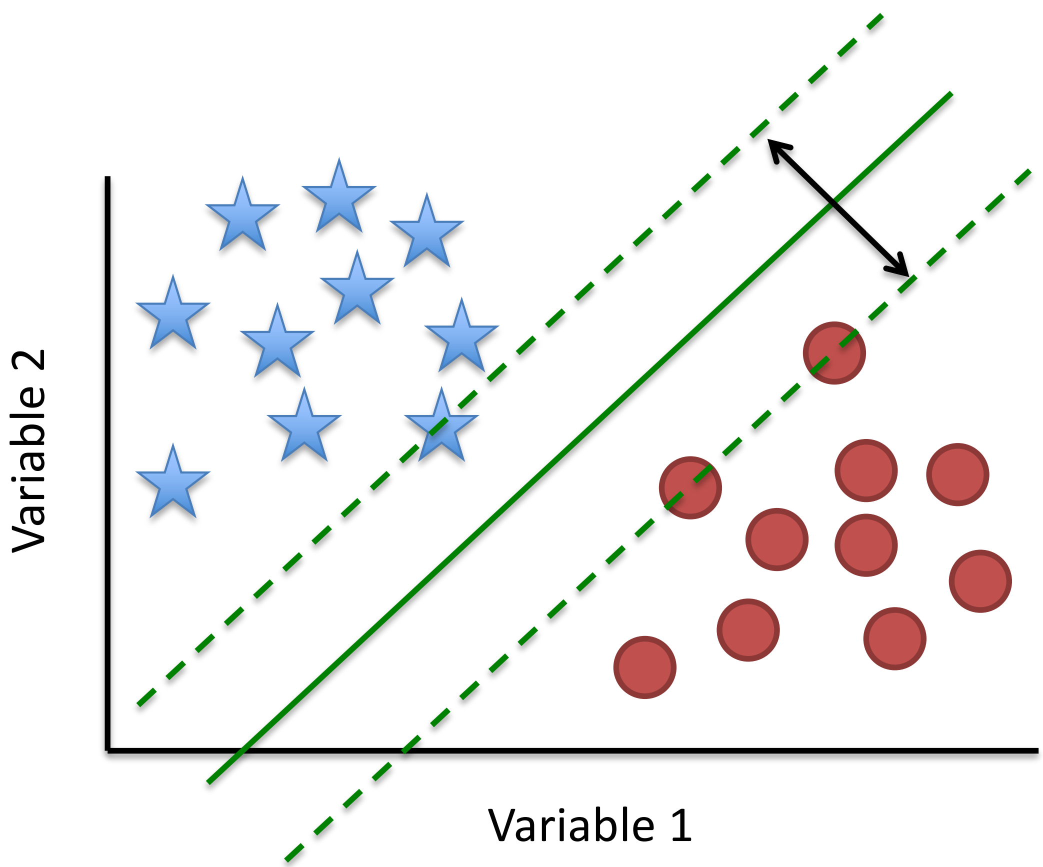
Support Vector Machines (SVMs)



- Define the margin of a linear classifier – the width that the boundary could be increased by before hitting a data sample

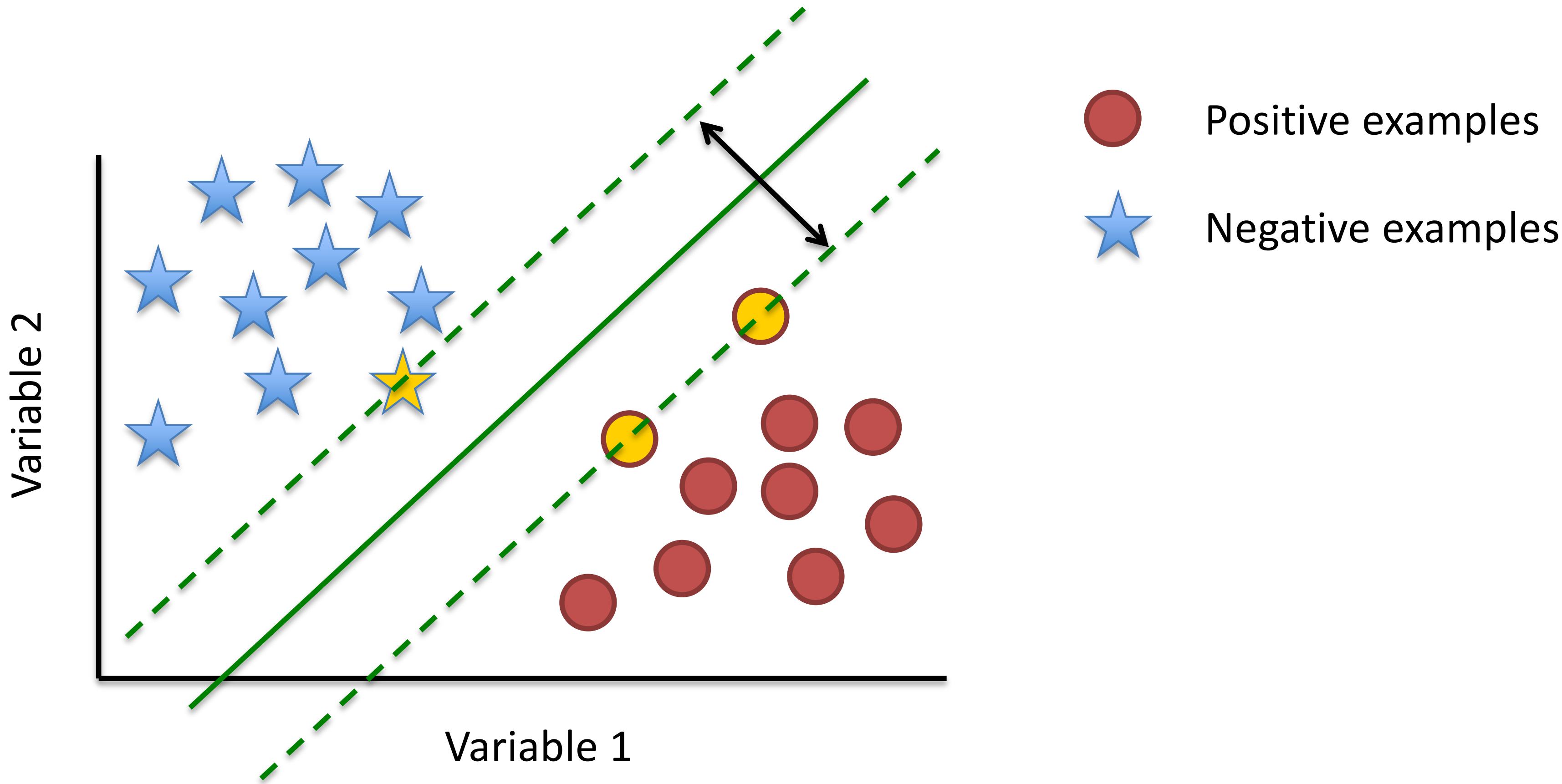
- which is best?

Support Vector Machines (SVMs)



- Which is best?
- The maximum margin linear classifier!
- This is the basic idea of linear SVMs

Main Idea for SVMs

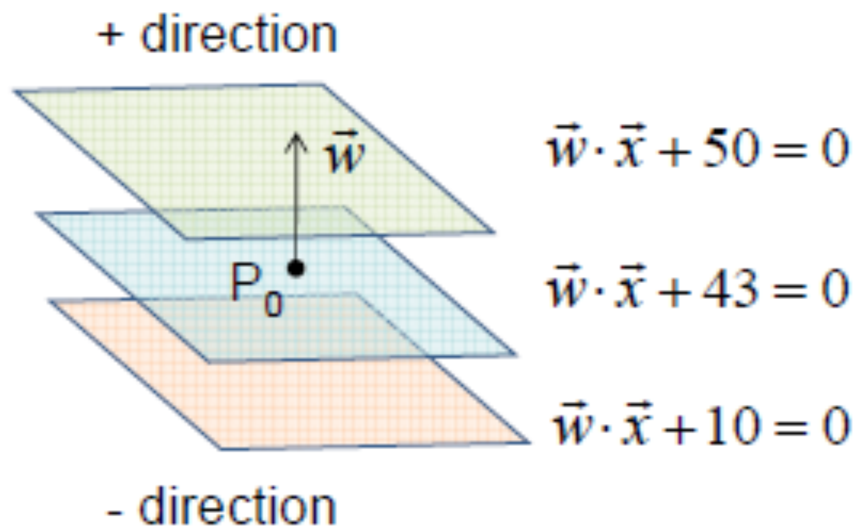


- Samples are vectors
- Find linear decision surface (hyperplane) to separate classes that has the largest distance (maximum margin) between border samples (support vectors)

Equation of Hyperplane

- Recall, interested in maximum margin
- In 2D, find maximum margin as distance between parallel lines from decision boundary
- In general, looking at parallel hyperplanes

Changing *b* coefficients results in parallel hyperplanes



Distance between Hyperplanes

- Distance between two parallel hyperplanes

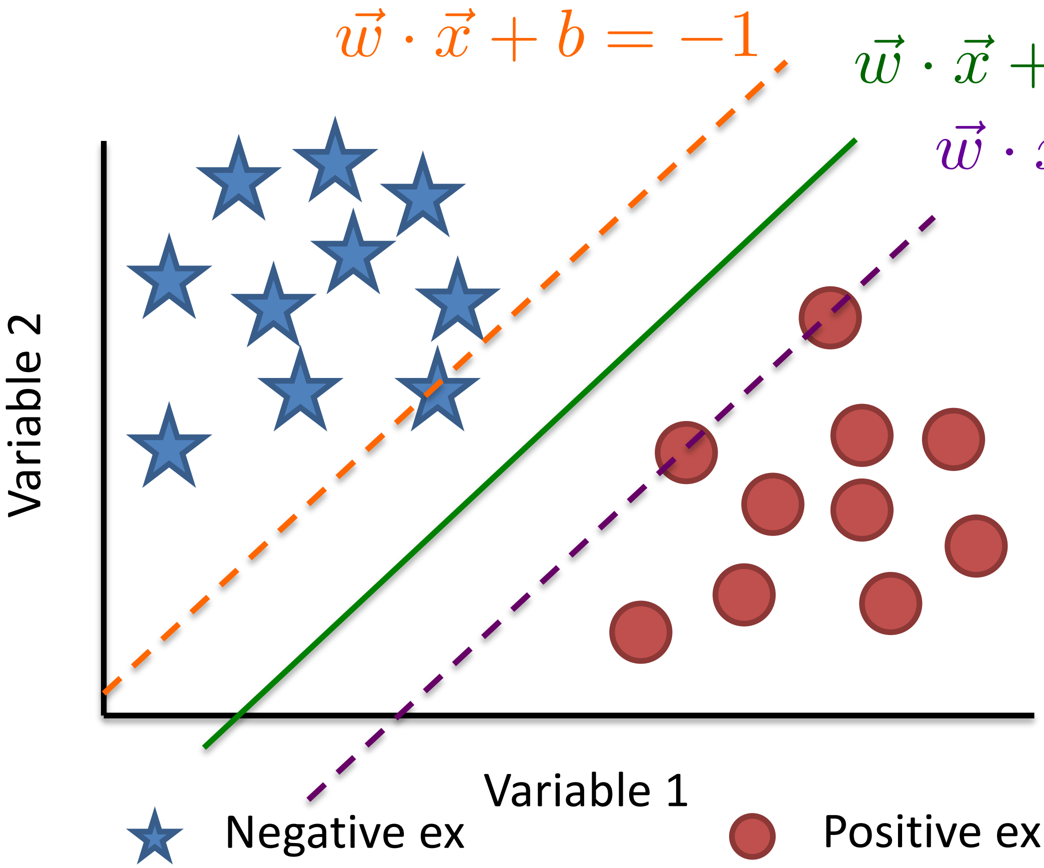
$$\vec{w} \cdot \vec{x} + b_1 = 0$$

$$\vec{w} \cdot \vec{x} + b_2 = 0$$

$$D = \frac{|b_1 - b_2|}{\|\vec{w}\|}$$

- Find $\vec{w}$ to maximize the margin

Case 1: "Hard-margin" Linear SVM



- Find "best" classifier; identify $\vec{w}$
- Gap between hyperplanes

$$\vec{w} \cdot \vec{x} + b = -1$$
$$\vec{w} \cdot \vec{x} + b = +1$$

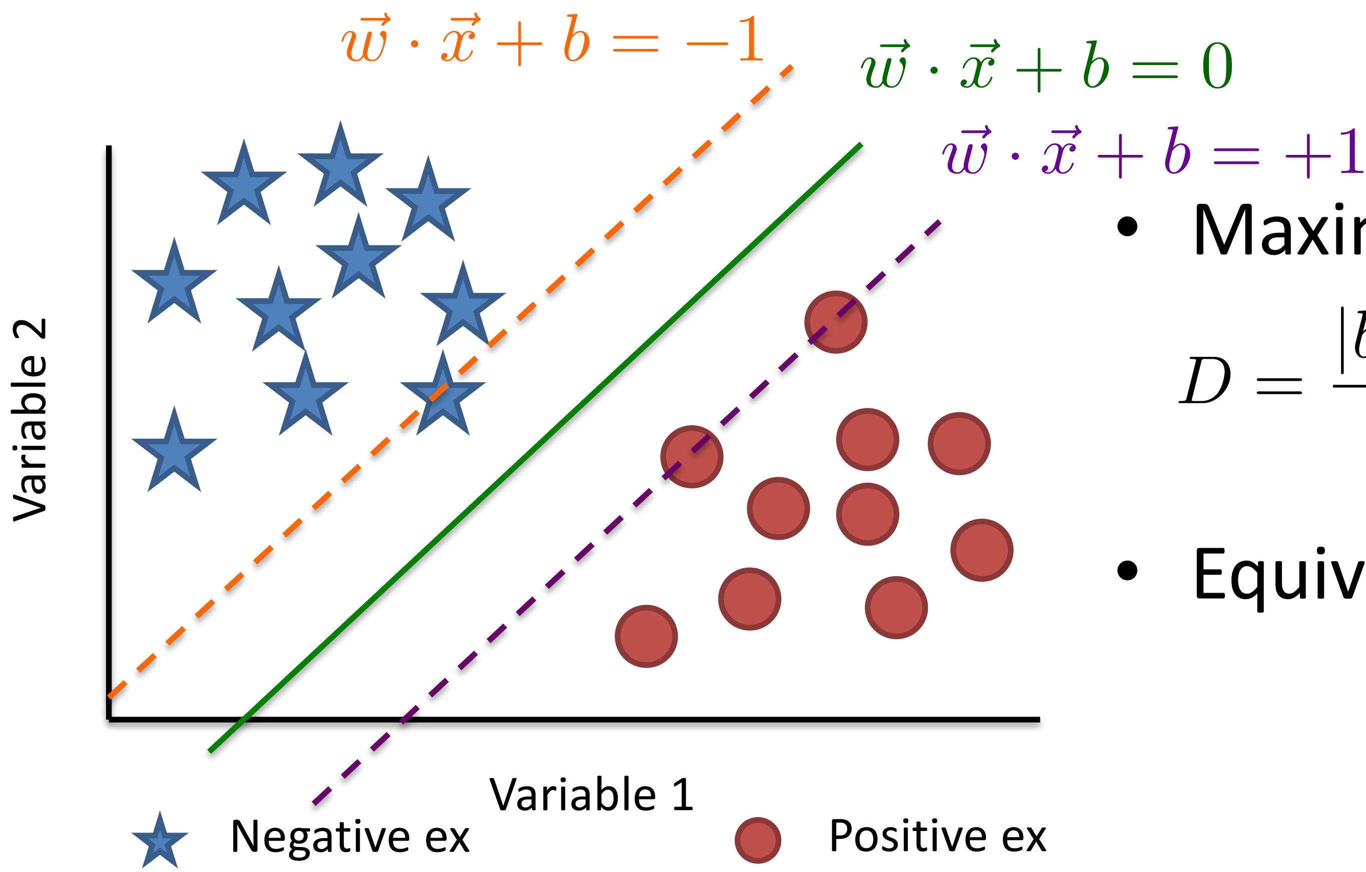
- Margin:

$$D = \frac{|b_1 - b_2|}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|}$$

- Training Data

$$\mathcal{D} = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), \ldots (\vec{x_n}, y_n)\}$$
$$\text{where } \vec{x_i} \in \mathbb{R}^m, y_i \in \{-1, +1\}$$

Case 1: "Hard-margin" Linear SVM



- Maximize Margin

$$D = \frac{|b_1 - b_2|}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|}$$

- Equivalently Minimize

$$\frac{1}{2}\|\vec{w}\|^2$$

$$s.t. \;\; y_i(\vec{w} \cdot \vec{x} + b) \geq 1$$

this is optimization problem

- Subject to constraints

$$\vec{w} \cdot \vec{x} + b \leq -1, \text{if } y_i = -1$$
$$\vec{w} \cdot \vec{x} + b \geq +1, \text{if } y_i = +1$$

Quadratic Programming (QP)

- QP optimization
  - function to be optimized ("objective") is quadratic, subject to linear constraints
- Problems are solved efficiently with greedy methods (for convex problems)
- Example:

$$\min \ \frac{1}{2}\|\vec{x}\|^2 \ \text{ subject to } \ x_1 + x_2 - 1 \geq 0$$

SVM optimization problem: Primal formulation

Problem

$$\min \boxed{\frac{1}{2}\sum_{j=1}^{m} w_j^2}$$

Objective function

$$\text{s.t.} \quad \boxed{y_i(\vec{w} \cdot \vec{x} + b) \geq 1} \quad \text{for } i = 1, \ldots, n$$

Constraints

- Primal formulation of linear SVM
- A convex quadratic programming optimization problem with *m* variables

SVM optimization problem: dual formulation

- Recast problem to "dual form"
- Also, a convex quadratic optimization problem with *n* variables, where *n* is the number of samples

$$\max \boxed{\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \, \alpha_j \, y_i \, y_j \, \vec{x_i} \cdot \vec{x_j}}$$

Objective function

$$\text{s.t.} \quad \boxed{\alpha_i \geq 0 \text{ and } \sum_{i=1}^{n} \alpha_i \, y_i = 0}$$

Constraints

where

$$\vec{w} = \sum_{i=1}^{n} \alpha_i \, y_i \, \vec{x_i} \qquad f(\vec{x}) = sign(\sum_{i=1}^{n} \alpha_i \, y_i \, \vec{x_i} \cdot \vec{x} + b)$$

Benefits of dual formulation

- No need to access original data, only need to access dot products

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \, \alpha_j \, y_i \, y_j \boxed{\vec{x_i} \cdot \vec{x_j}}$$

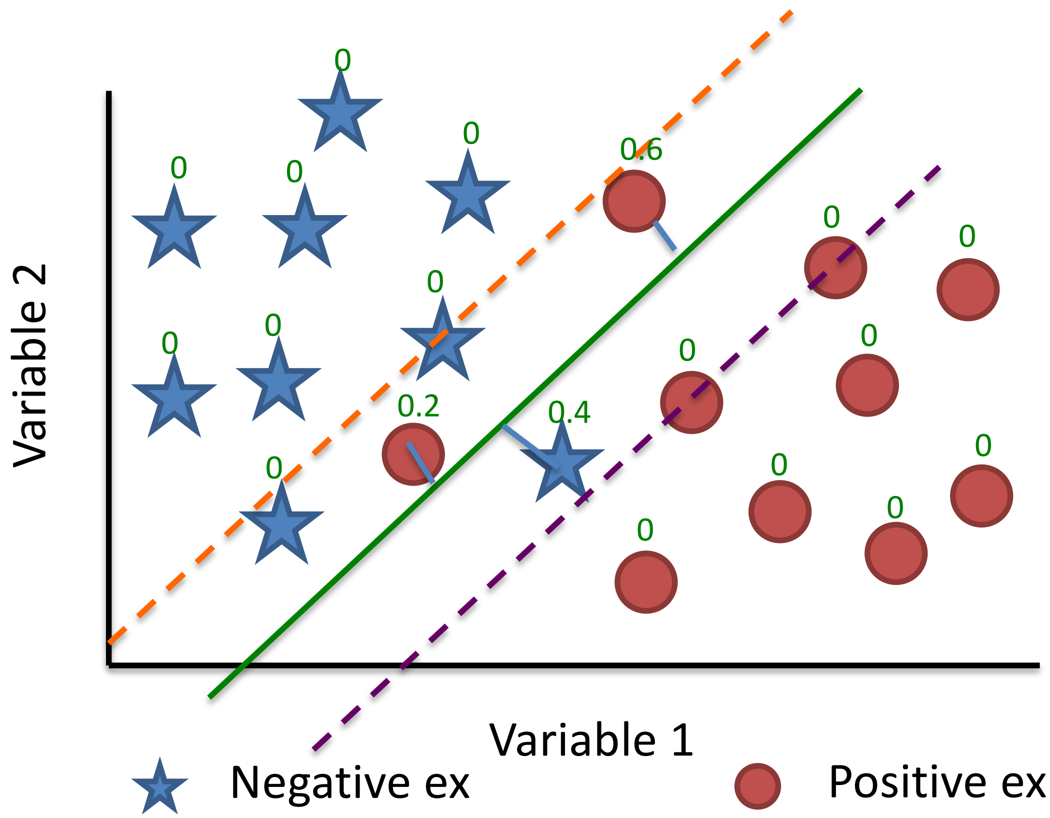
Objective function

$$f(\vec{x}) = sign(\sum_{i=1}^{n} \alpha_i \, y_i \boxed{\vec{x_i} \cdot \vec{x}} + b)$$

Classifier

- Number of free parameters bounded by the number of support vectors and not number of variables

Case 2: “Soft-margin” Linear SVM



- Data that is not linearly separable (noise, outliers, etc.)
- Use “slack” variable ξ_i
- distance from separating hyperplane if a sample is misclassified

$$\min \ \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad y_i(\vec{w}\cdot\vec{x}+b) \geq 1-\xi_i \quad \text{for } i = 1,\ldots,n$$

Soft-margin Linear SVM formulations

- Primal formulation

$$\min \boxed{\frac{1}{2}\sum_{j=1}^{m} w_j^2 + C\sum_{i=1}^{n}\xi_i}$$

Objective function

$$\text{s.t.} \boxed{y_i(\vec{w}\cdot\vec{x}+b) \geq 1-\xi_i} \text{ for } i=1,\ldots,n$$

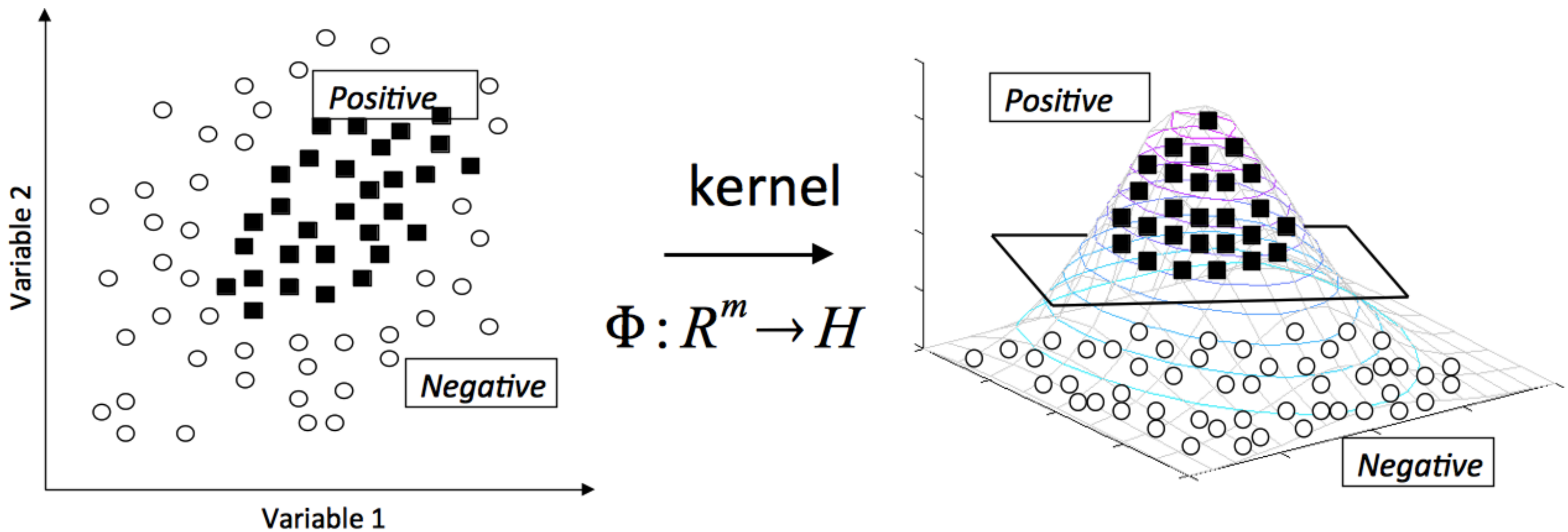
Constraints

- Dual formulation

$$\max \boxed{\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\,\alpha_j\,y_i\,y_j\,\vec{x_i}\cdot\vec{x_j}}$$

$$\text{s.t.} \boxed{0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{n}\alpha_i\,y_i = 0} \text{ for } i=1,\ldots,n$$

Case 3: Kernel Trick



Data is not linearly separable in the input space

Data is linearly separable in the feature space obtained by a kernel

Kernel Trick

- Input Space
  original data $\vec{x}$

- Feature Space
  data in higher dim. $\Phi(\vec{x})$

$$f(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b)$$

$$f(\vec{x}) = sign(\vec{w} \cdot \Phi(\vec{x}) + b)$$

$$\vec{w} = \sum_{i=1}^{n} \alpha_i \, y_i \, \vec{x_i}$$

$$\vec{w} = \sum_{i=1}^{n} \alpha_i \, y_i \, \Phi(\vec{x_i})$$

We do not need to know Φ explicitly; use Kernel function

$$f(\vec{x}) = sign(\sum_{i=1}^{n} \alpha_i \, y_i \boxed{\Phi(\vec{x_i}) \cdot \Phi(\vec{x})} + b)$$

$$f(\vec{x}) = sign(\sum_{i=1}^{n} \alpha_i \, y_i \boxed{K(\vec{x_i}, \vec{x})} + b)$$

Popular Kernels

- Kernel is dot product in some feature space $K(\vec{x_i}, \vec{x_j}) = \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$

- Examples

$$K(\vec{x_i}, \vec{x_j}) = \vec{x_i} \cdot \vec{x_j} \qquad \text{(Linear kernel)}$$

$$K(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + p)^d \qquad \text{(Polynomial kernel)}$$

$$K(\vec{x_i}, \vec{x_j}) = e^{-\frac{\|\vec{x_i} - \vec{x_j}\|^2}{2\sigma^2}} \qquad \text{(RBF kernel)}$$

$$K(\vec{x_i}, \vec{x_j}) = tanh(\kappa \vec{x_i} \cdot \vec{x_j} - \sigma) \qquad \text{(Sigmoidal kernel)}$$

Understanding the Polynomial Kernel

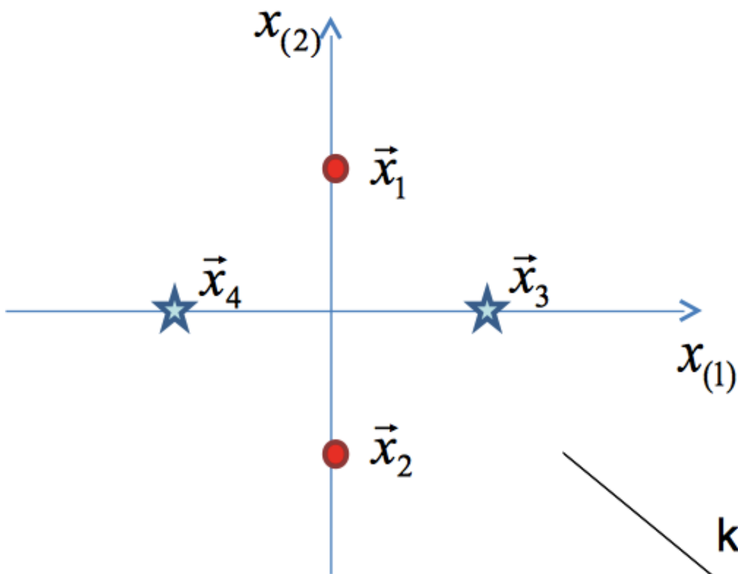
- Polynomial Kernel: parameter degree, *d*=3

$$K(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^3$$

$$\begin{pmatrix} x_{(1)} \\ x_{(2)} \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ x_{(1)} \\ x_{(2)} \\ x_{(1)}^2 \\ x_{(2)}^2 \\ x_{(1)}x_{(2)} \\ x_{(1)}^3 \\ x_{(2)}^3 \\ x_{(1)}x_{(2)}^2 \\ x_{(1)}^2 x_{(2)} \end{pmatrix}$$

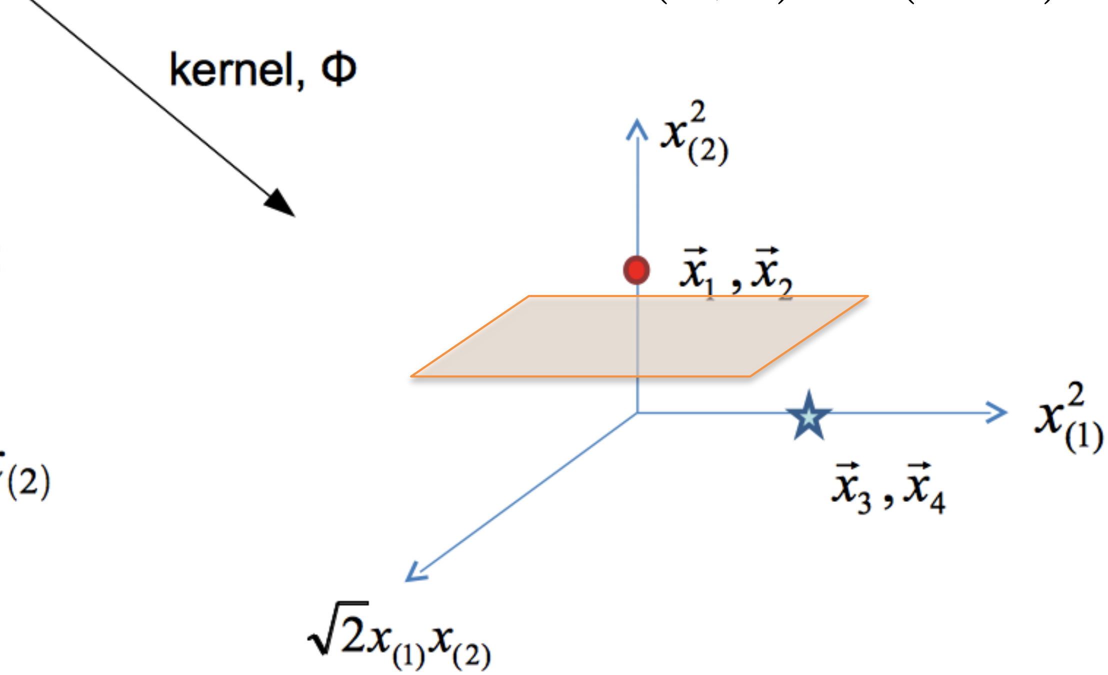
Number of components in feature space $\binom{m+d}{d}$

Benefits of Kernel



- Data that is not linearly separable, can be shifted to higher dimension where it is
- Apply kernel to map to feature space

$$K(\vec{x}, \vec{z}) = (\vec{x} \cdot \vec{z})^2$$



The explicit mapping is:

$$\Phi(\vec{x}) = \begin{matrix} x_{(1)}^2 \\ \sqrt{2}\, x_{(1)} x_{(2)} \\ x_{(2)}^2 \end{matrix}$$

SVMs – Loss + Penalty Paradigm

- SVMs build the following classifiers: $f(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b)$
- Consider the soft-margin linear formulation:

$$\min \ \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

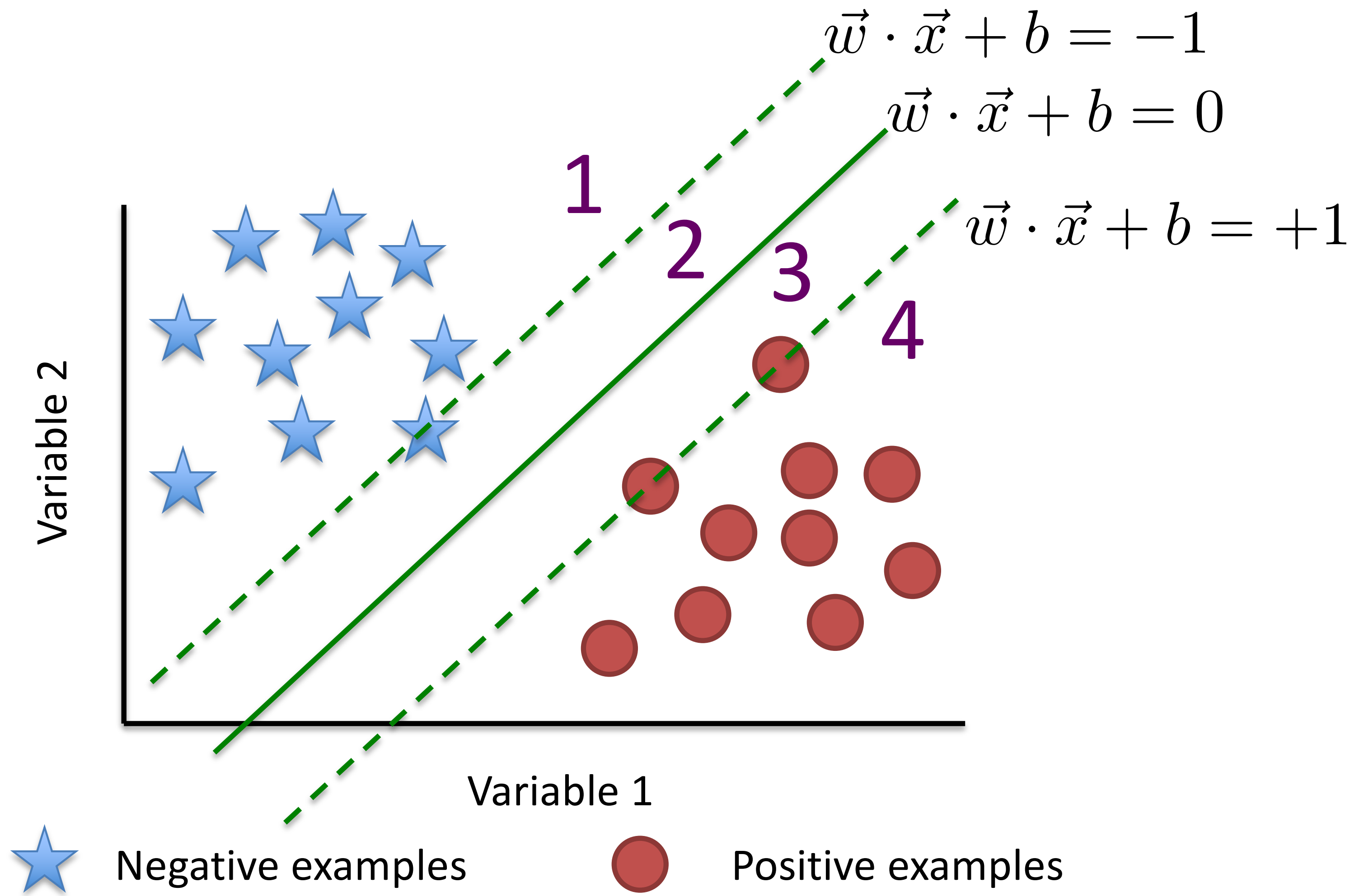
$$\text{s.t.} \quad y_i(\vec{w} \cdot \vec{x} + b) \geq 1 - \xi_i \quad \text{for } i = 1, \ldots, n$$

- This can be restated as:

$$\min \ \underbrace{\sum_{i=1}^{n}[1 - y_i\, f(\vec{x_i})]_+}_{\text{Loss}} + \underbrace{\lambda\|\vec{w}\|_2^2}_{\text{Penalty}}$$

SVMs – Loss + Penalty paradigm

- Hinge loss $\sum_{i=1}^{n}[1 - y_i\, f(\vec{x_i})]_+$



Loss + Penalty Framework

Minimize (Loss + λPenalty)

Loss function	Penalty function	Method
Hinge loss $\sum_{i=1}^{n}[1 - y_i\, f(\vec{x}_i)]_+$	$\lambda\, \|\vec{w}\|_2^2$	SVMs
Mean squared error $\sum_{i=1}^{n}(y_i - f(\vec{x}_i))^2$	$\lambda\, \|\vec{w}\|_2^2$	Ridge regression
Mean squared error $\sum_{i=1}^{n}(y_i - f(\vec{x}_i))^2$	$\lambda\, \|\vec{w}\|_1$	Lasso
Mean squared error $\sum_{i=1}^{n}(y_i - f(\vec{x}_i))^2$	$\lambda_1\, \|\vec{w}\|_1 + \lambda_2 \|\vec{w}\|_2^2$	Elastic net
Hinge loss $\sum_{i=1}^{n}[1 - y_i\, f(\vec{x}_i)]_+$	$\lambda\, \|\vec{w}\|_1$	1-norm SVM

Other Considerations

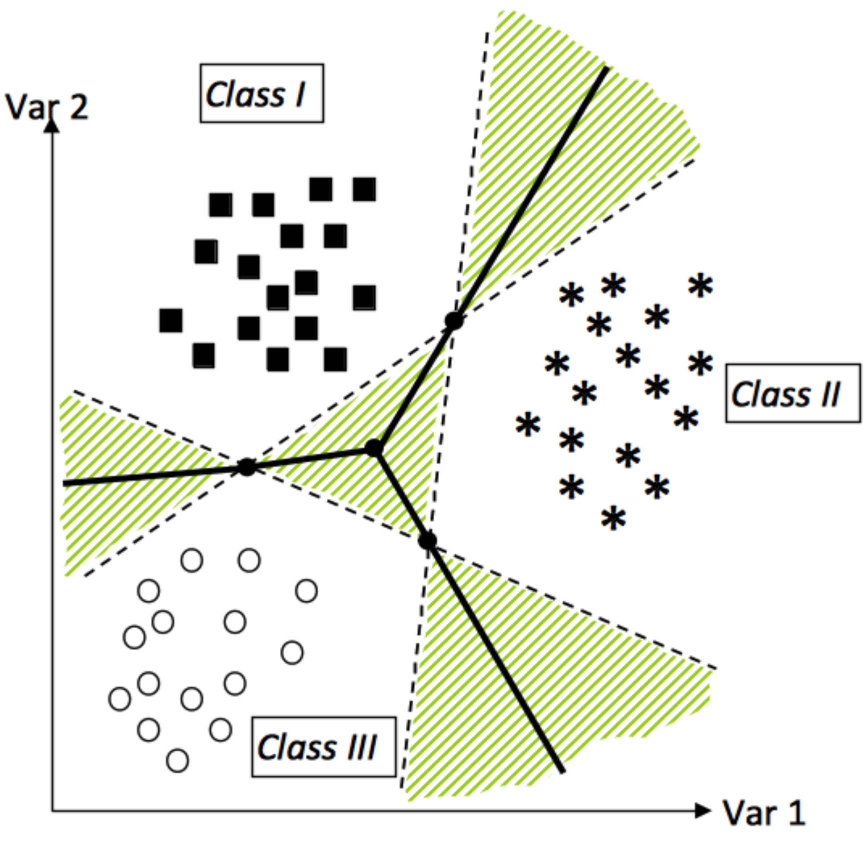
- Extensions to SVM model
  - model selection for SVMs
    - parameter selection: cost, kernel parameters
  - Multicategorical data
  - Support Vector Regression
  - Theory for constructing dual formulation

Model selection for SVMs

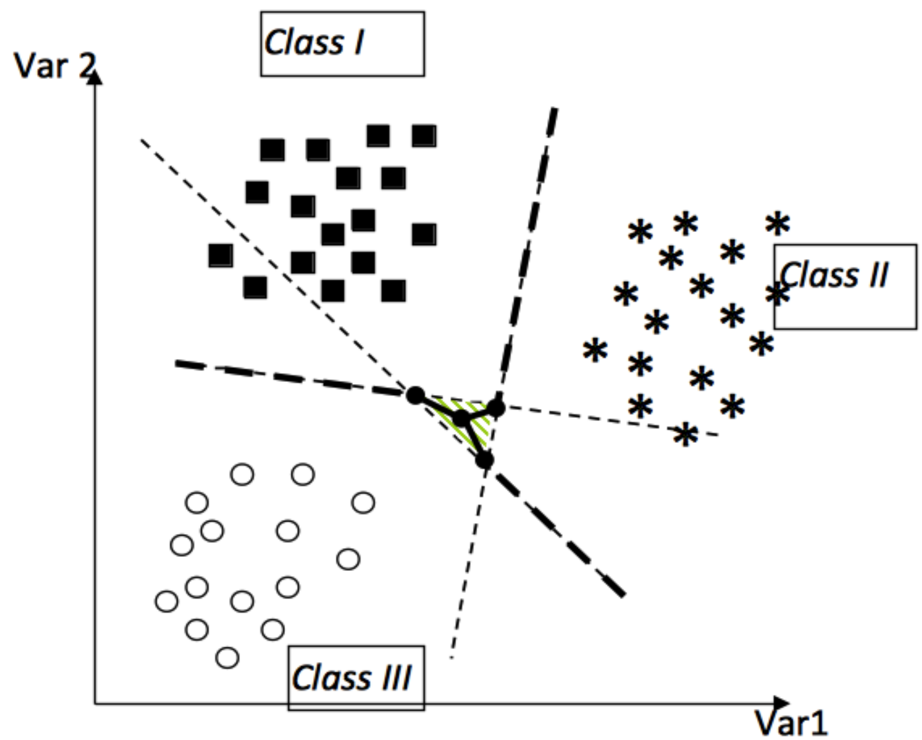
- Do not know *a priori* which kernel and kernel parameters are best for a given data set
- Examine combinations of parameters, search a grid of parameters

	Polynomial degree, *d*			
Cost parameter *C*	0.1, 1	0.1, 2	0.1, 3	0.1, 4
	1, 1	1, 2	1, 3	1, 4
	10, 1	10, 2	10, 3	10, 4
	100, 1	100, 2	100, 3	100, 4

SVMs for multicategory data



One vs. rest



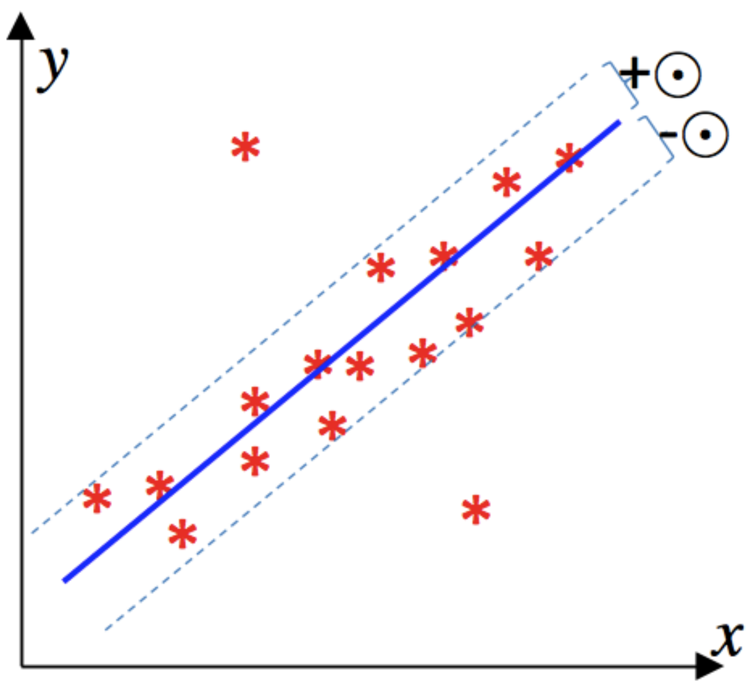
One vs. One

Support Vector Regression (SVR)

- ε – support vector regression

$$\mathcal{D} = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), \ldots (\vec{x_n}, y_n)\}$$

$$\text{where } \vec{x_i} \in \mathbb{R}^m, y_i \in \mathbb{R}$$



<u>Main Idea:</u>

Find a function $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$
that approximates $y_1, \ldots, y_N$
with at most ε deviation from the true values of y

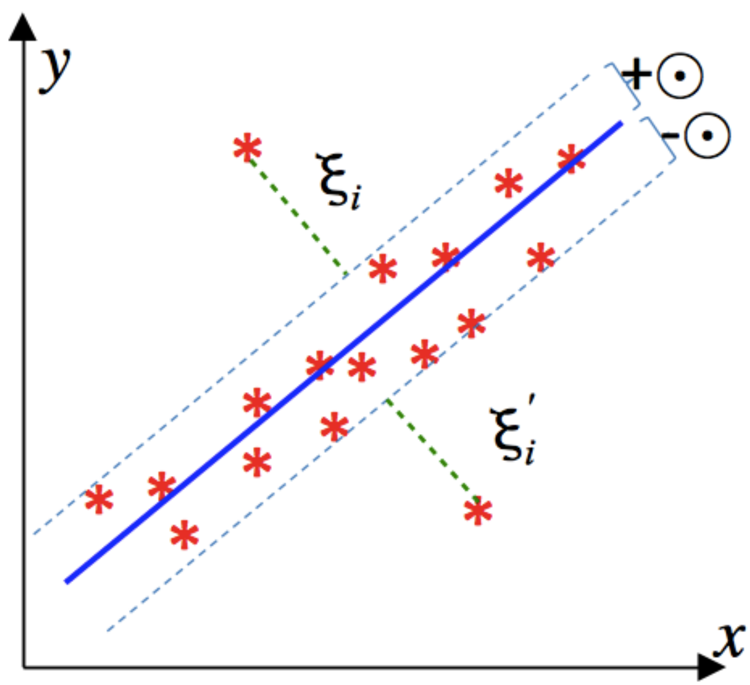
Minimize $\frac{1}{2}\|\vec{w}\|^2$
Subject to constraints

$$y_i - (\vec{w} \cdot \vec{x} + b) \leqslant \epsilon$$
$$y_i - (\vec{w} \cdot \vec{x} + b) \geqslant -\epsilon$$
$$for\ i = 1, \ldots, N$$

Support Vector Regression (SVR)

- ε – support vector regression

$$\mathcal{D} = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), \ldots (\vec{x_n}, y_n)\}$$
$$\text{where } \vec{x_i} \in \mathbb{R}^m, y_i \in \mathbb{R}$$



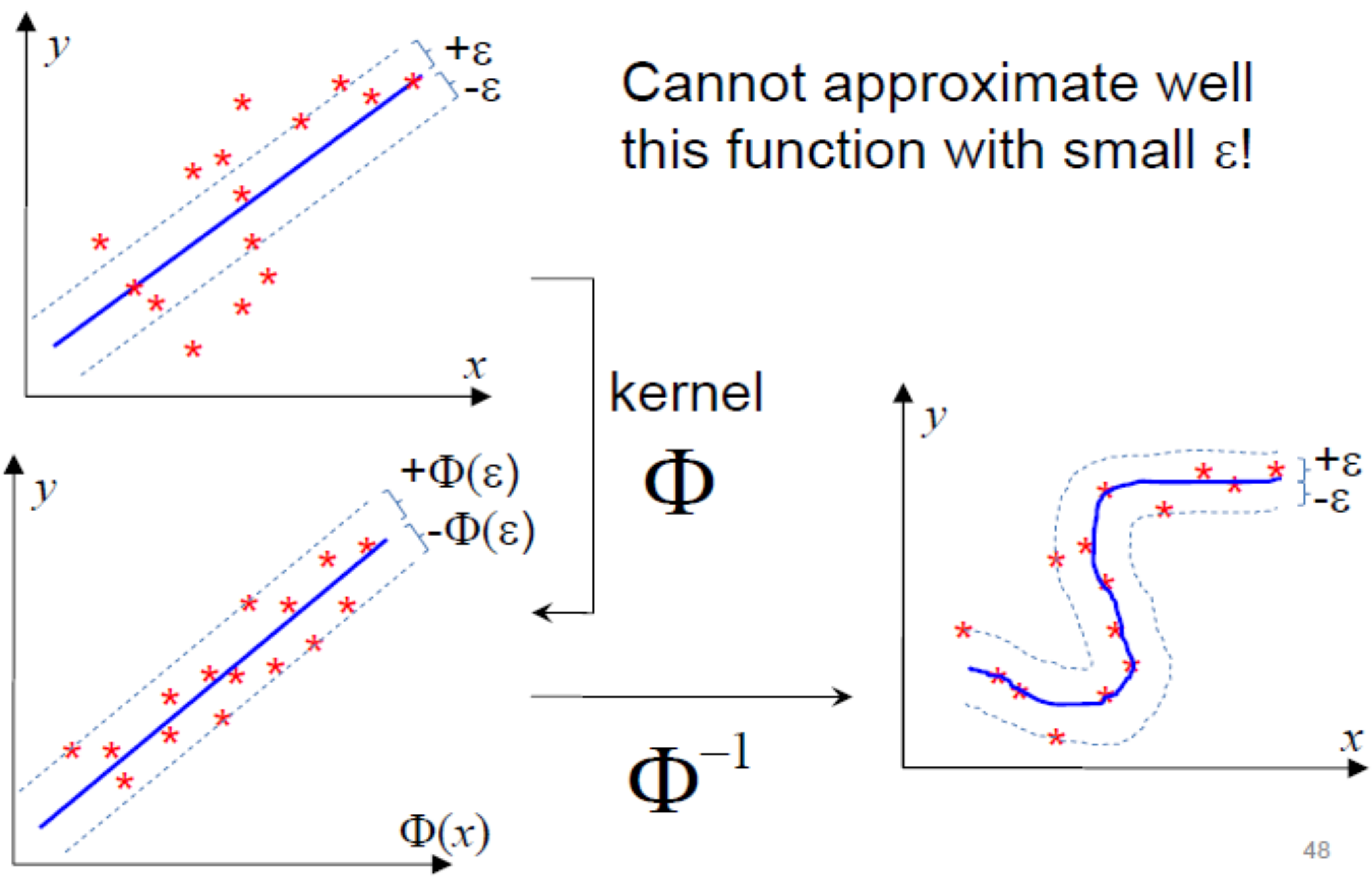
<u>Main Idea:</u>

Find a function $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$
that approximates $y_1, \ldots, y_N$
with at most ε deviation from
the true values of y

Minimize $\frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i')$
Subject to constraints

$$y_i - (\vec{w} \cdot \vec{x} + b) \leqslant \epsilon + \xi_i$$
$$y_i - (\vec{w} \cdot \vec{x} + b) \geqslant -\epsilon - \xi_i'$$
$$\xi_i, \xi_i' \geqslant 0$$
$$for\, i = 1, \ldots, N$$

Non-linear SVR



SVMs Summary

- Support Vector Machines work very well in practice on high-dimensional data
  - the number of support vectors can be used to compute an upper bound on the expected error rate
  - thus, a SVM with few support vectors can have good generalization, even with high dimensional data
- How to get SVM model
  - optimization packages: MINOS, LOQO, Matlab
  - software: Weka, SVMlight, LibSVM
  - libraries: sklearn, e1071