Support Vector Machine (SVM)

Guowei Wei
Department of Mathematics
Michigan State University

References:
Duc D. Nguyen's lecture notes
Andrew Ng's notes
Wikipedia

Introduction

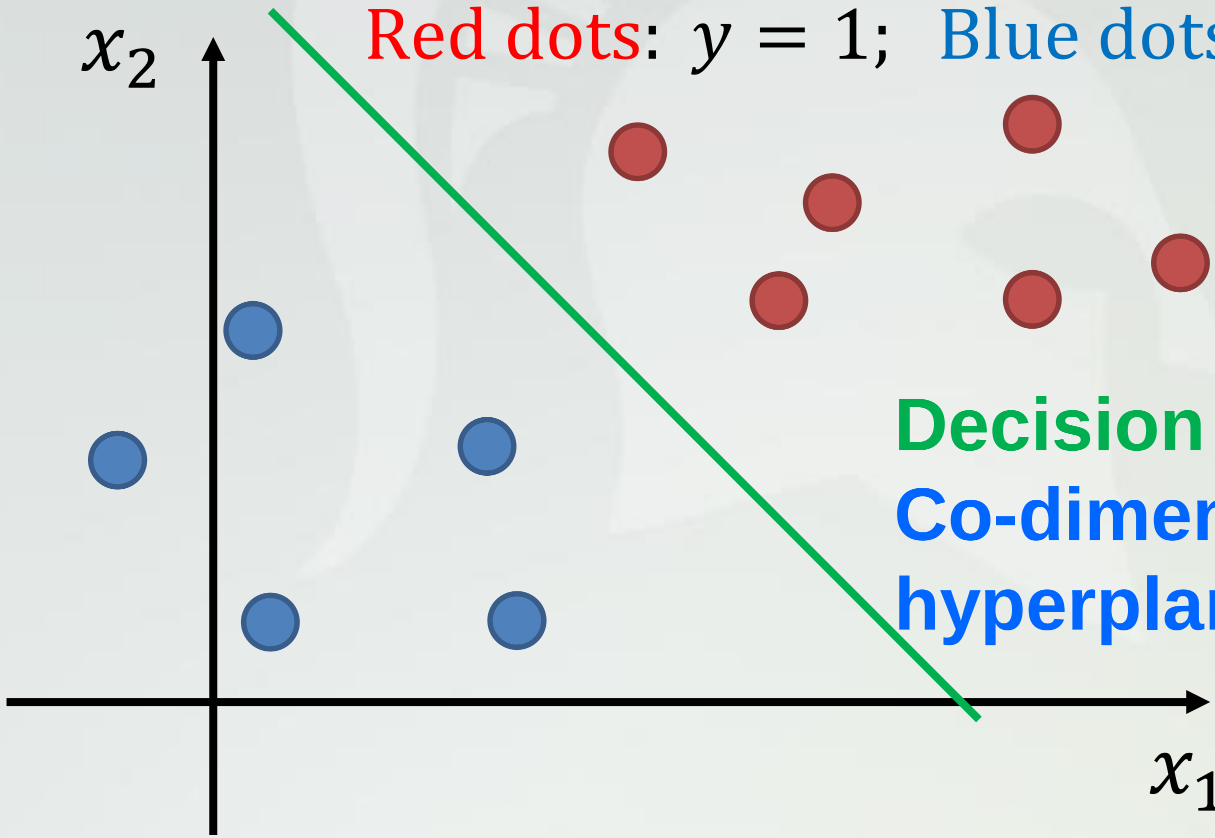
- One of top ten methods in data science
- Classification
- Regression, i.e., support vector regression (SVR)
- Supervised learning in general
- For unsupervised learning:

Support vector clustering (SVC) by Hava Siegelmann and Vladimir Vapnik

SVM for linear Classifiers
Decision Boundary

Training set: $\mathcal{D} = \left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \middle| \mathbf{x}^{(i)} \in \mathbb{R}^n, y^{(i)} \in \{-1,1\}\right\}_{i=1}^{M}$

Red dots: $y = 1$; Blue dots: $y = -1$



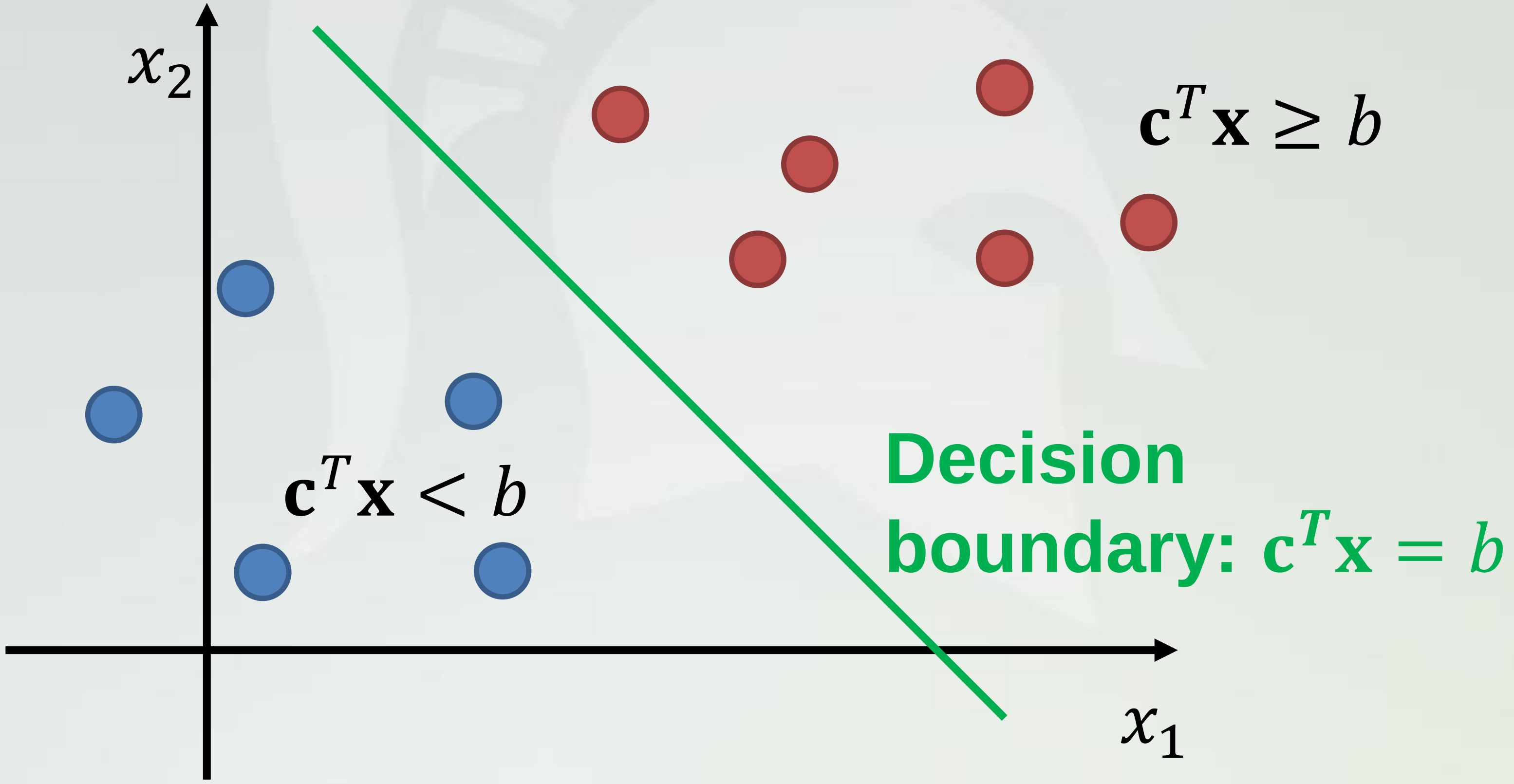
Decision boundary:
Co-dimension 1 hyperplane

Decision Boundary

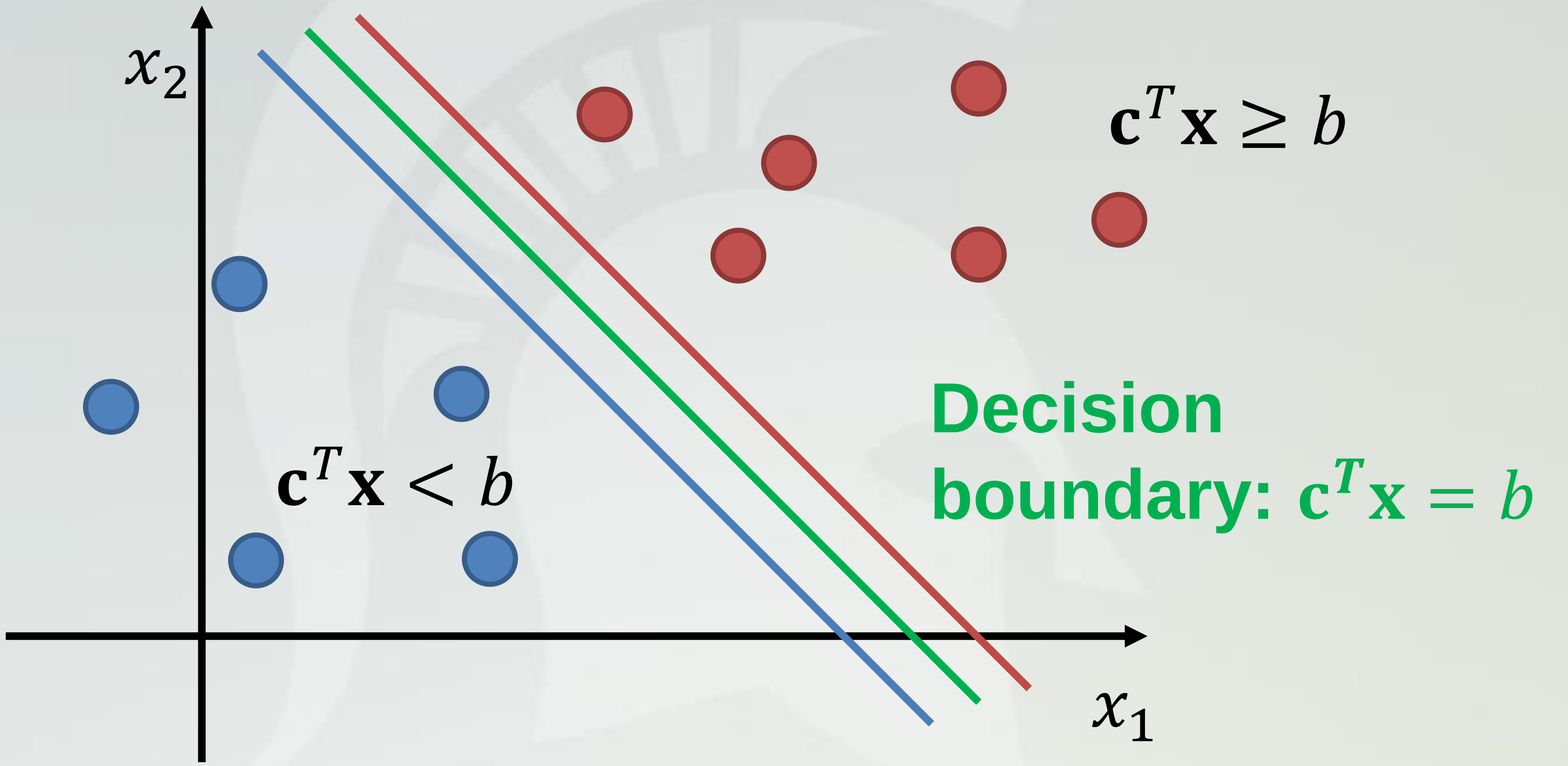
Recall:

$\mathbf{x} = (1, x_1, \ldots, x_n)^T$,

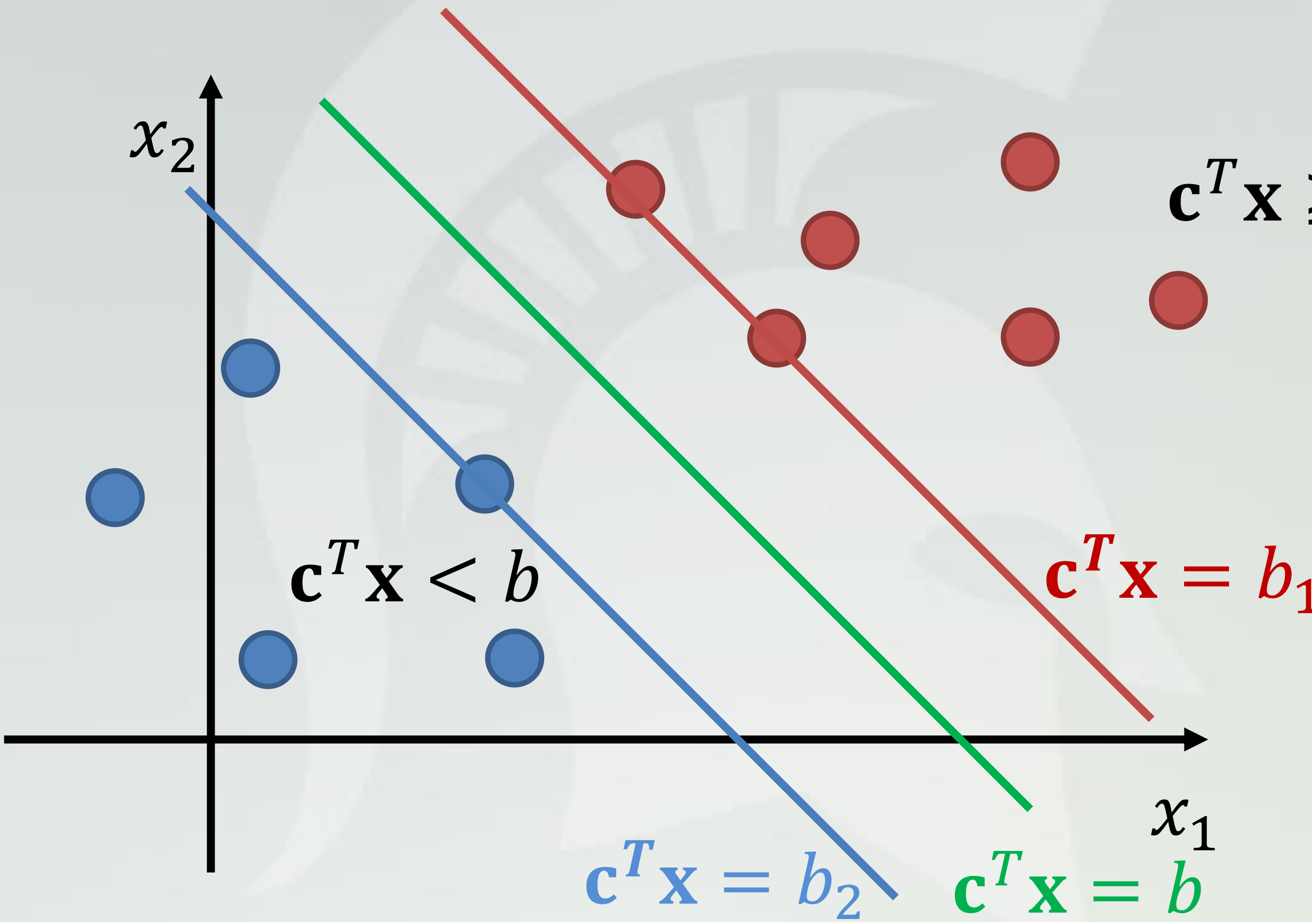
$\mathbf{c} = (c_0, c_1, \ldots, c_n)^T$



Decision Boundary

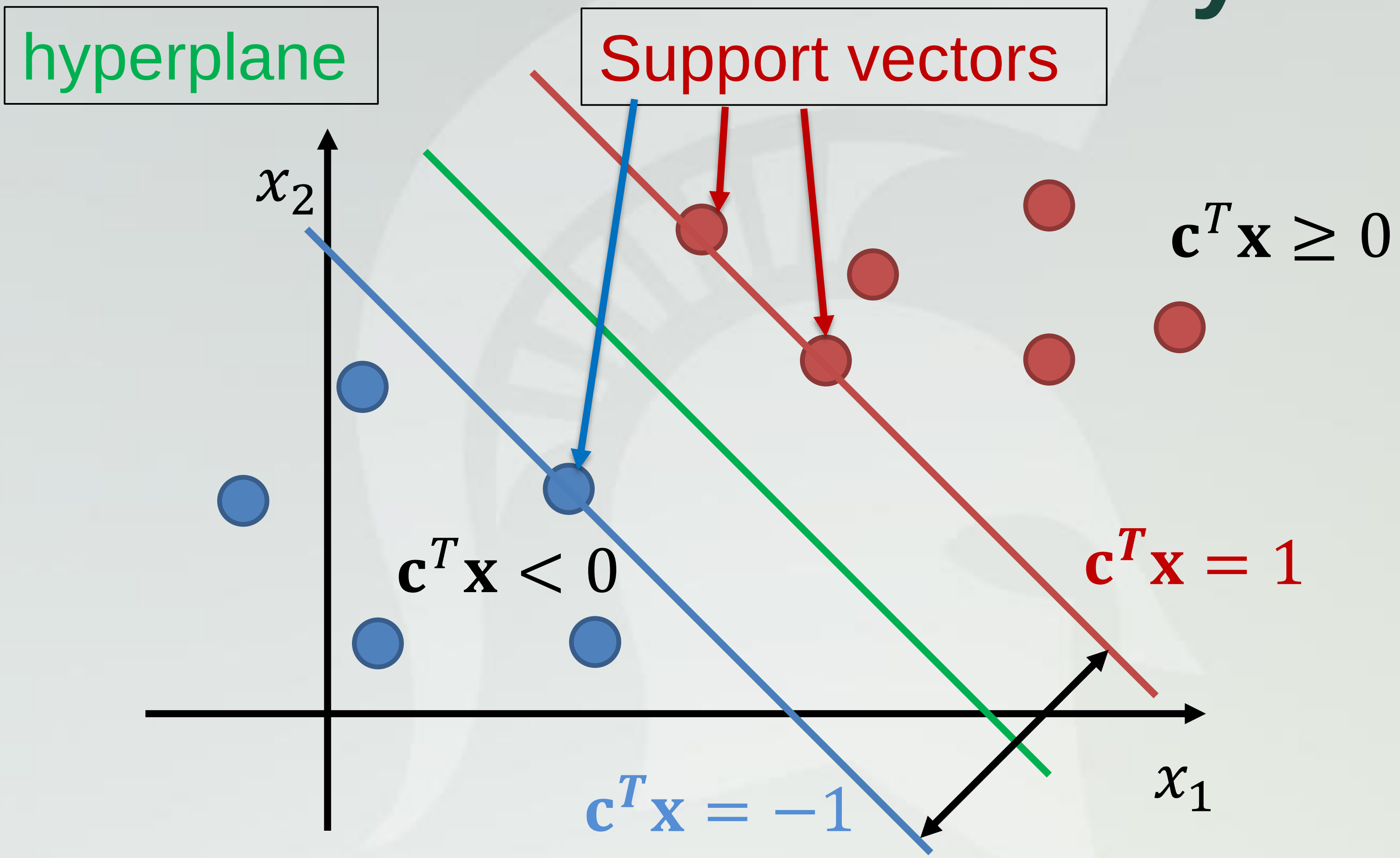


Decision Boundary



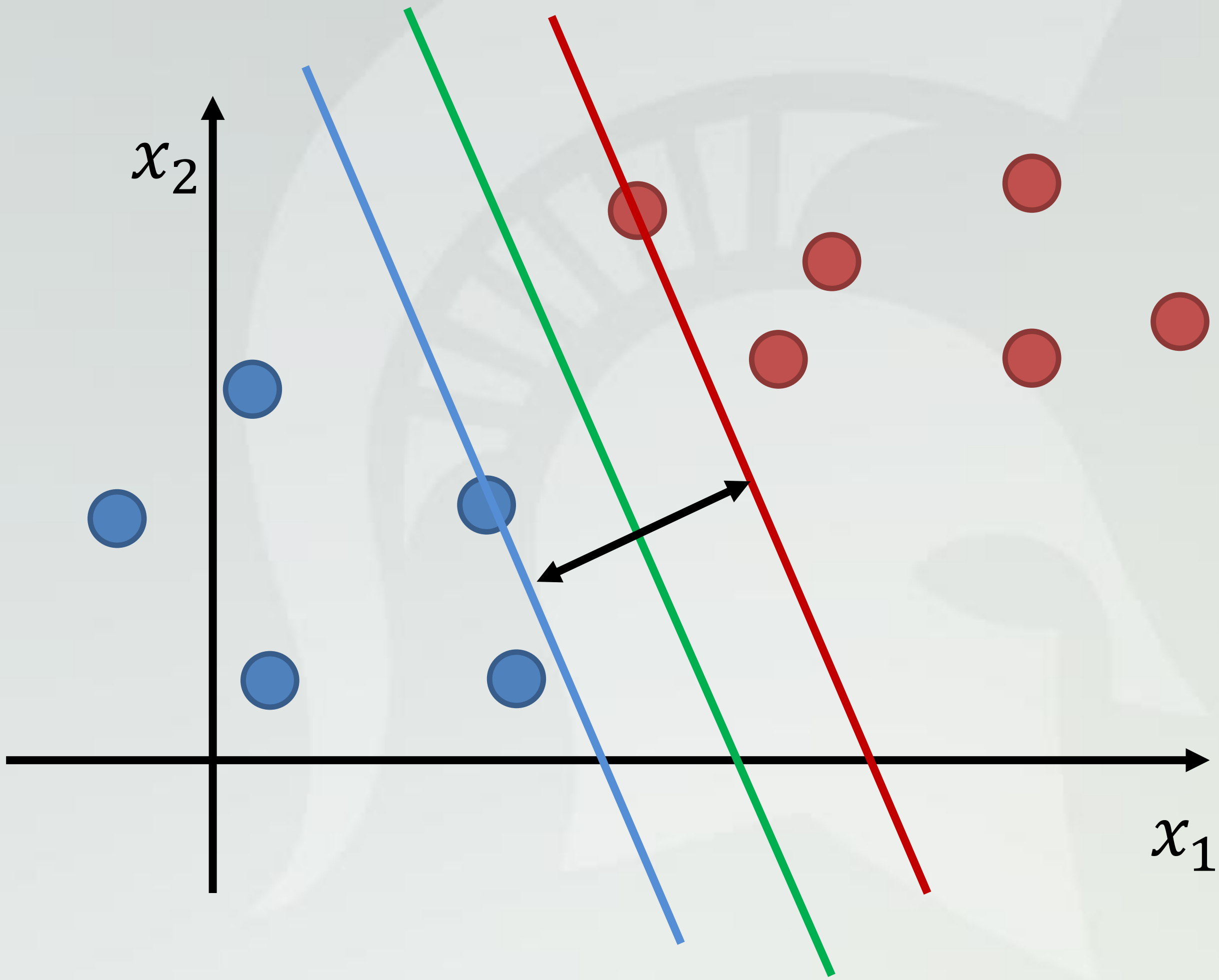
For simplicity: choose $b = 0$, $b_1 = 1$, $b_2 = -1$

Decision Boundary

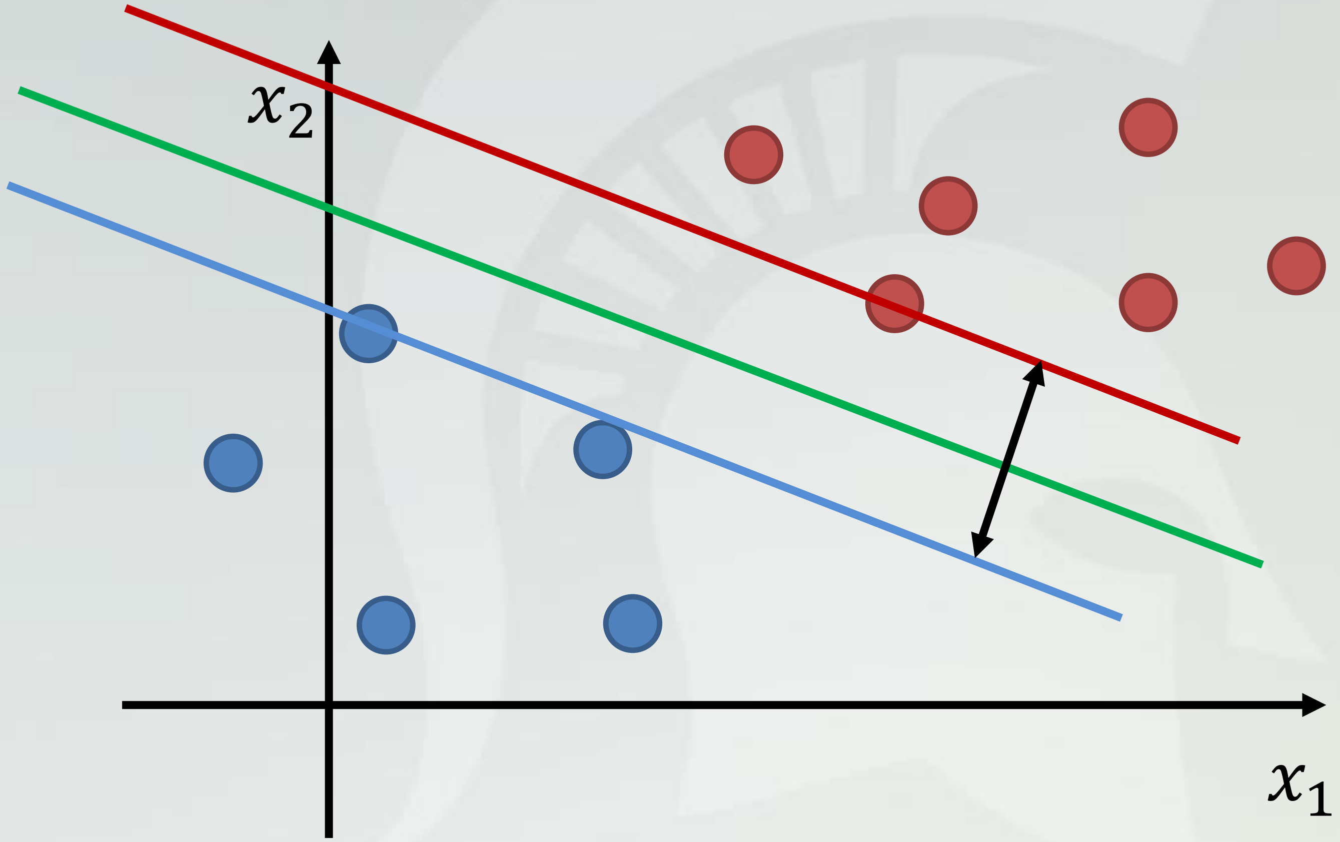


Margin: $\dfrac{2}{\sqrt{c_1^2 + c_2^2 + \cdots + c_n^2}}$

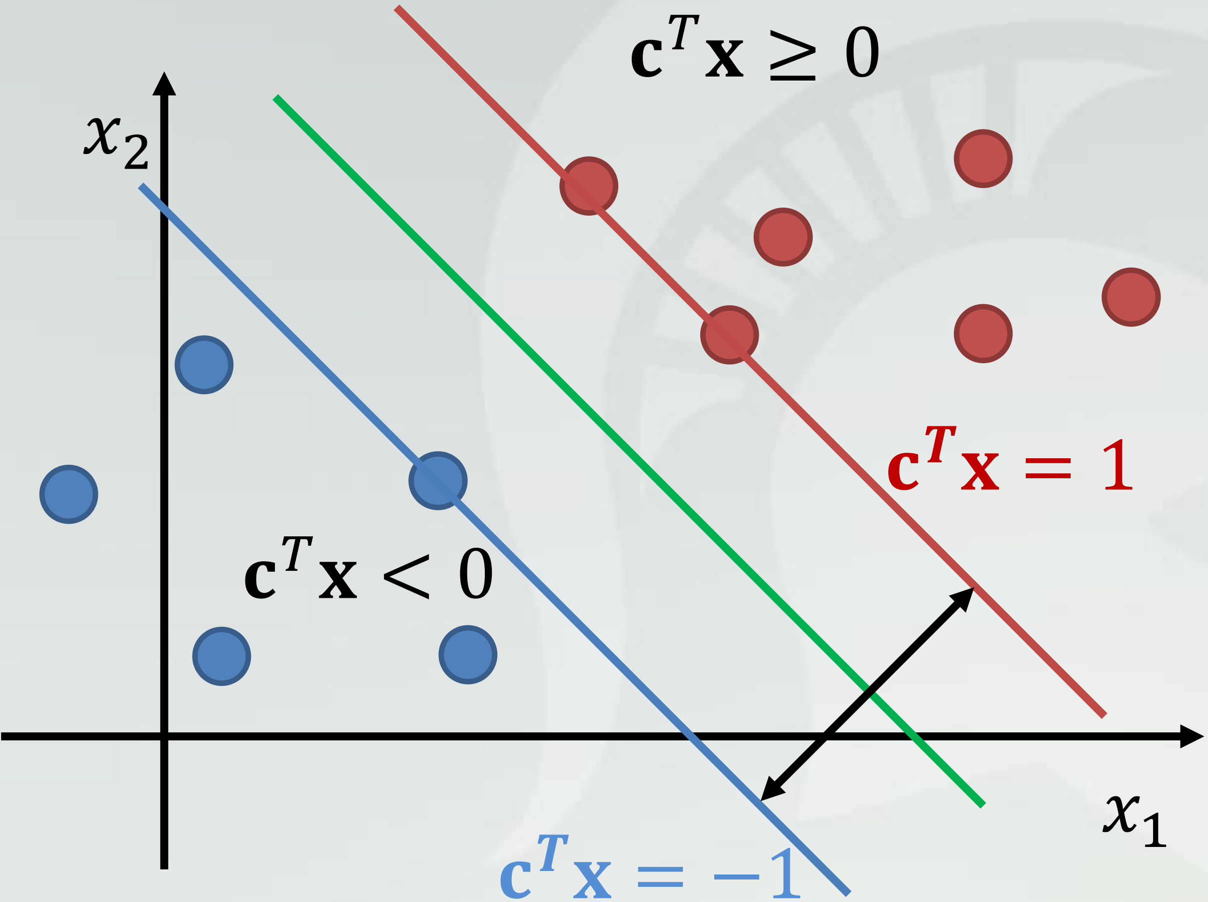
Decision Boundary



Decision Boundary



Optimization Objective



Margin: $\frac{2}{\sqrt{c_1^2+c_2^2+\cdots+c_n^2}}$

Maximize the margin:

$$\frac{2}{\sqrt{c_1^2+c_2^2+\cdots+c_n^2}}$$

Subject to

$\mathbf{c}^T\mathbf{x}^{(i)} \geq 1$ if $y^{(i)} = 1$

or

$\mathbf{c}^T\mathbf{x}^{(i)} \leq -1$ if $y^{(i)} = -1$

(in SVM, we label negative class as – 1 instead of 0)

Optimization Objective

- Maximize

$$\frac{2}{\sqrt{c_1^2 + c_2^2 + \cdots + c_n^2}}$$

Subject to

$$\mathbf{c}^T\mathbf{x}^{(i)} \geq 1 \text{ if } \; y^{(i)} = 1$$

or

$$\mathbf{c}^T\mathbf{x}^{(i)} \leq -1 \text{ if } \quad y^{(i)} = -1$$

- Equivalent to (dual problem):

Minimize:

$$\sqrt{c_1^2 + c_2^2 + \cdots + c_n^2}$$

Subject to $\mathbf{c}^T\mathbf{x}^{(i)} \geq 1$ if $y^{(i)} = 1$ or $\mathbf{c}^T\mathbf{x}^{(i)} \leq -1$ if $y^{(i)} = -1$

Optimization Objective

- Minimize:

$$\sqrt{c_1^2 + c_2^2 + \cdots + c_n^2}$$

Subject to $\mathbf{c}^T\mathbf{x}^{(i)} \geq 1$ if $y^{(i)} = 1$ or $\mathbf{c}^T\mathbf{x}^{(i)} \leq -1$ if $y^{(i)} = -1$

- Equivalent to

Minimize:

Loss function

$$\sqrt{c_1^2 + c_2^2 + \cdots + c_n^2}$$

Subject to $y^{(i)}\mathbf{c}^T\mathbf{x}^{(i)} \geq 1$

Predictor?

$$p_{\mathbf{c}}(\mathbf{x}) = \mathbf{c}^T\mathbf{x} = c_0 + c_1 x_1 + \cdots + c_n x_n$$ Predictor

Minimize: Loss function

$$L(\mathbf{c}) = L(c_0, c_1, \ldots, c_n) = \sqrt{c_1^2 + c_2^2 + \cdots + c_n^2}$$

Subject to $y^{(i)} p_{\mathbf{c}}(\mathbf{x}^{(i)}) = y^{(i)}\mathbf{c}^T\mathbf{x}^{(i)} \geq 1$

- Classifier: Take threshold=0

  if $p_{\mathbf{c}}(\mathbf{x}) \geq 0$ then $y = 1$

  if $p_{\mathbf{c}}(\mathbf{x}) < 0$ then $y = -1$

Loss Function

$$p_{\mathbf{c}}(\mathbf{x}) = \mathbf{c}^T\mathbf{x} = c_0 + c_1x_1 + \cdots + c_nx_n$$ Predictor

Minimize:

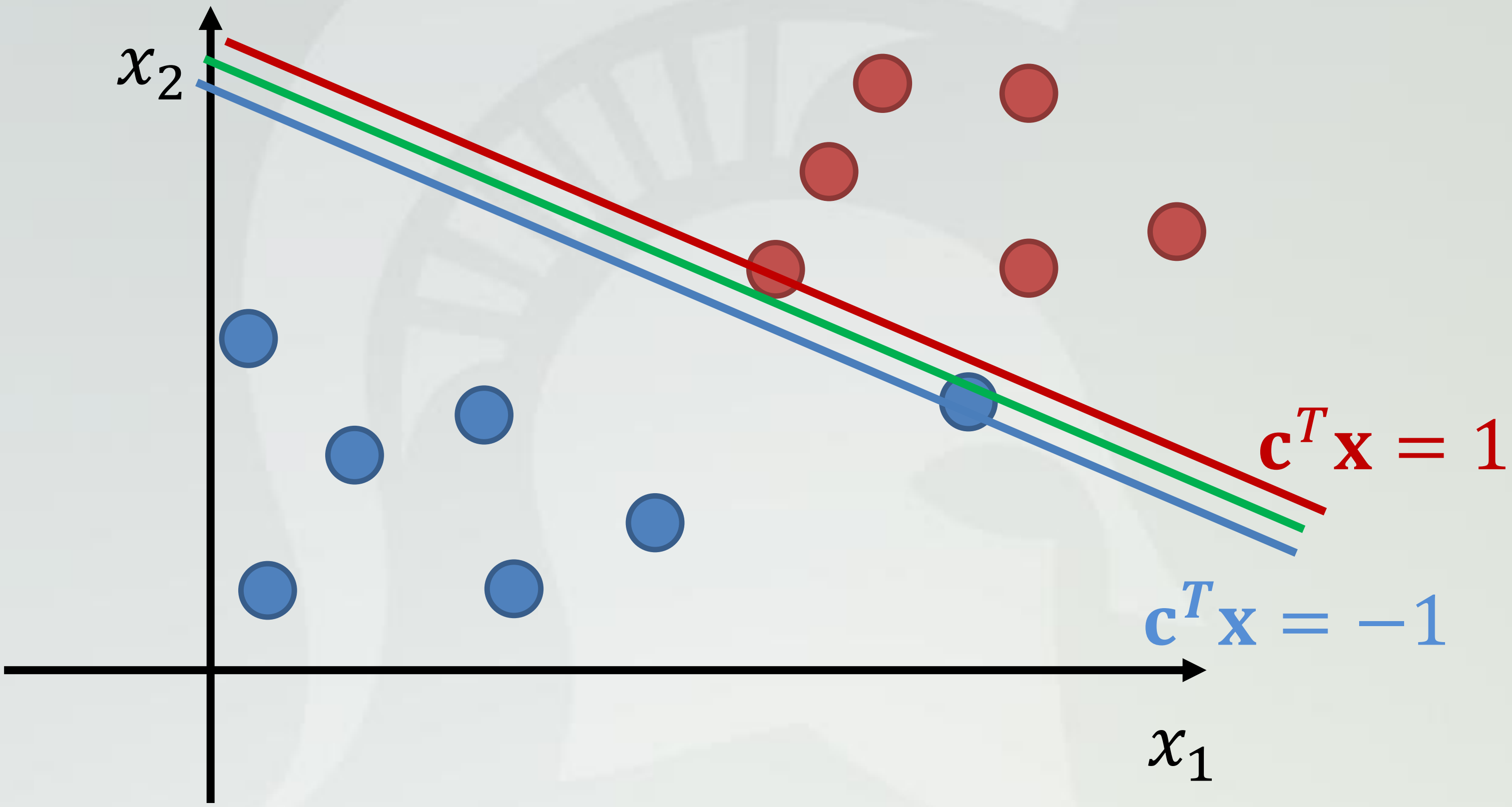
Loss function

$$L(\mathbf{c}) = L(c_0, c_1, \dots, c_n) = \sqrt{c_1^2 + c_2^2 + \cdots + c_n^2}$$

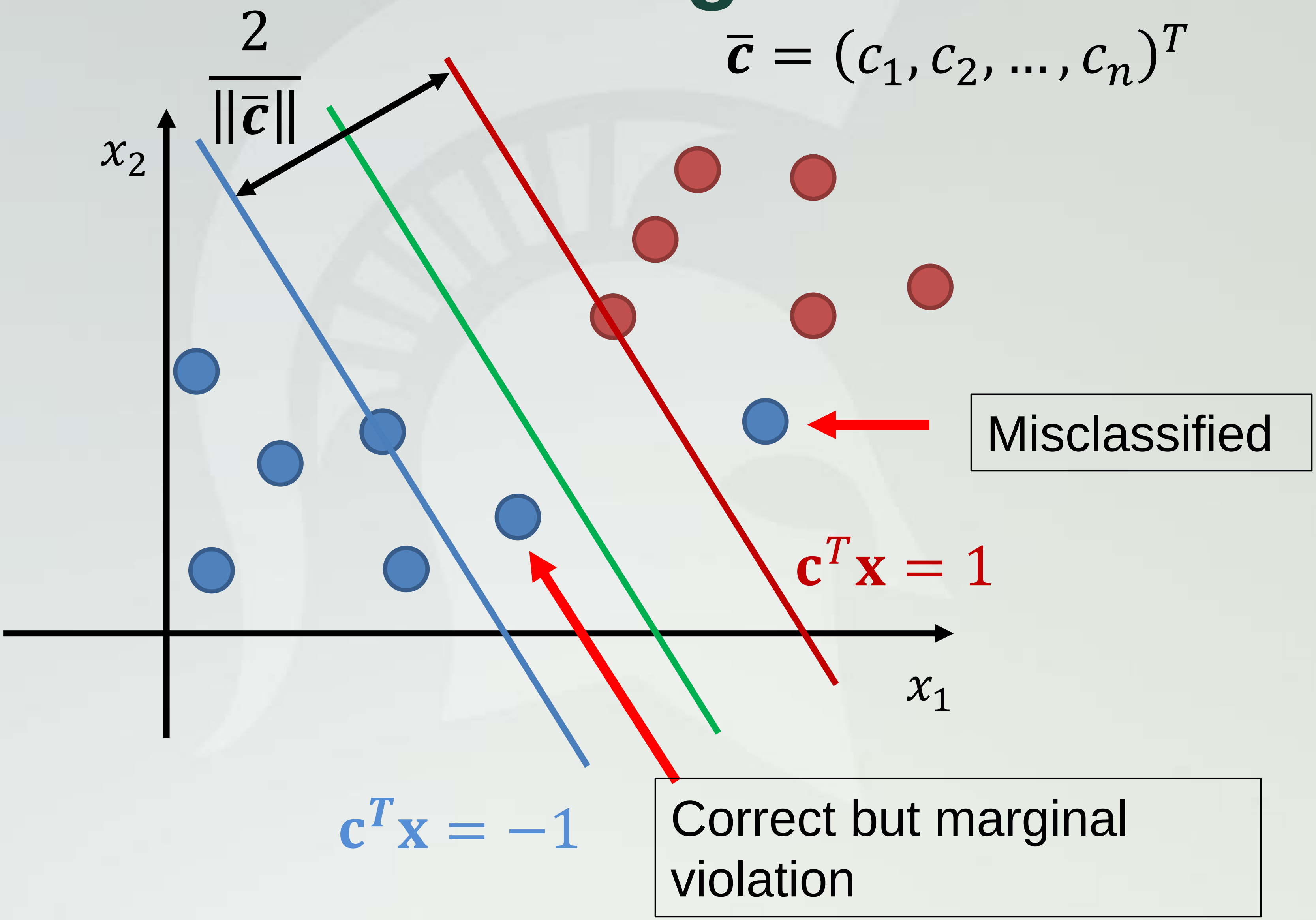
Subject to $y^{(i)}\mathbf{c}^T\mathbf{x}^{(i)} \geq 1$

Simplified condition

Hard Margin

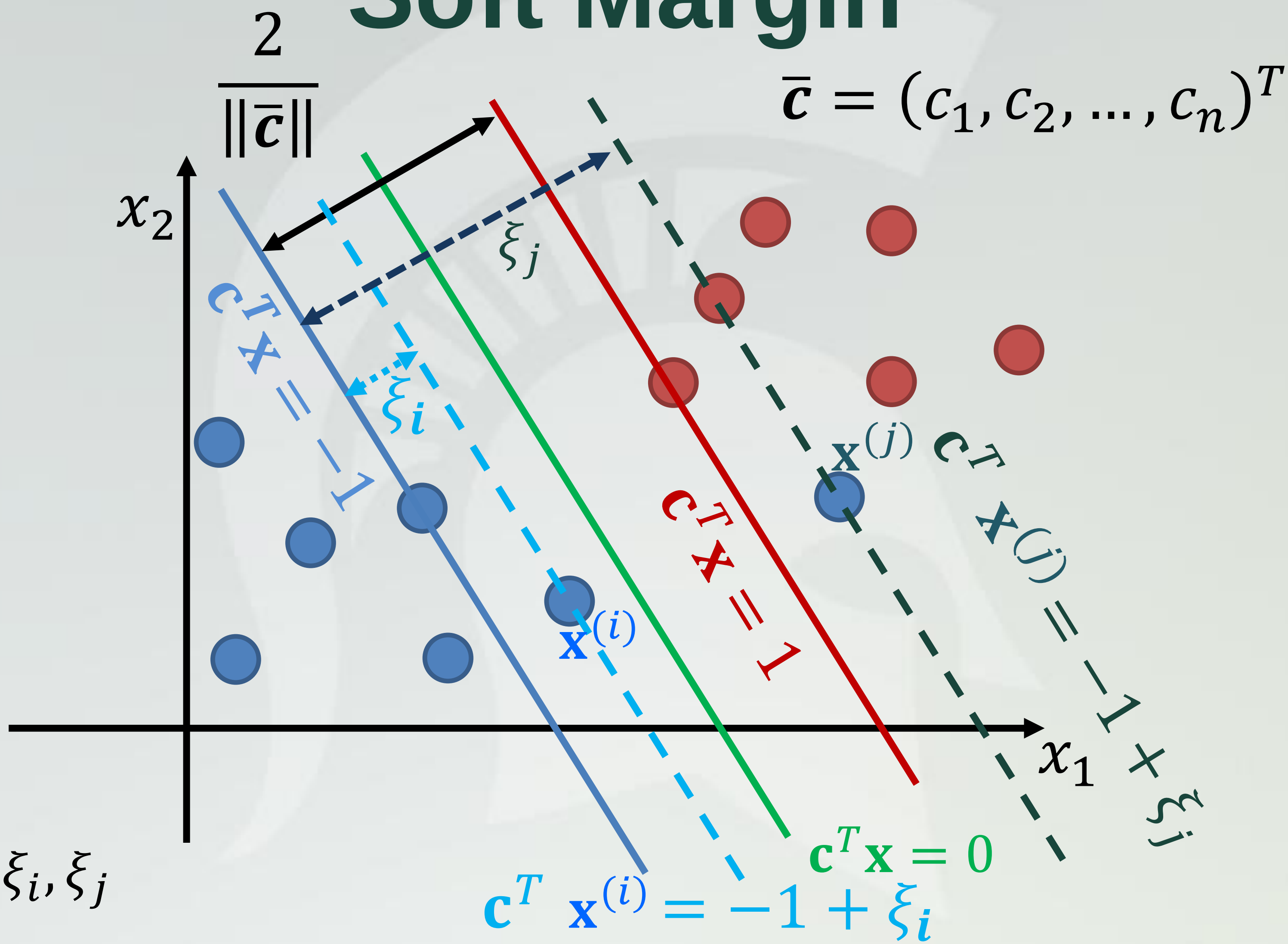


Soft Margin



$\frac{2}{\|\bar{c}\|}$
$\bar{c} = (c_1, c_2, \dots, c_n)^T$
x_2
x_1
Misclassified
$\mathbf{c}^T\mathbf{x} = 1$
$\mathbf{c}^T\mathbf{x} = -1$
Correct but marginal violation

Soft Margin



$0 \leq \xi_i, \xi_j$

$\xi_i < 1$ (Correct but marginal violation)

$\xi_j > 1$ (incorrect) If $\xi_k = 0$: perfect

Minimize ξ_k!

Loss Function for Soft Margin

Modified loss function

Predictor

$$p_{\mathbf{c}}(\mathbf{x}) = \mathbf{c}^T\mathbf{x} = c_0 + c_1x_1 + \cdots + c_nx_n$$

Minimize:

Loss function

$$L(\mathbf{c}) = L(c_0, c_1, \ldots, c_n) = \sqrt{c_1^2 + c_2^2 + \cdots + c_n^2}$$

Subject to $y^{(i)}\mathbf{c}^T\mathbf{x}^{(i)} \geq 1 - \xi_i$, with $\xi_i \geq 0$

Modified condition

Loss Function for Soft Margin

Predictor

$$p_{\mathbf{c}}(\mathbf{x}) = \mathbf{c}^T\mathbf{x} = c_0 + c_1x_1 + \cdots + c_nx_n$$

Minimize:

Loss function

$$L(\mathbf{c}, \boldsymbol{\xi}) = L(c_0, c_1, \dots, c_n, \xi_1, \xi_2, \dots, \xi_M) =$$

$$\sqrt{c_1^2 + c_2^2 + \cdots + c_n^2} + \sum_{i=1}^{M} \xi_i$$

Regularization

Subject to $y^{(i)}\mathbf{c}^T\mathbf{x}^{(i)} \geq 1 - \xi_i$, with $\xi_i \geq 0$

Loss Function for Soft Margin

Predictor

$$p_{\mathbf{c}}(\mathbf{x}) = \mathbf{c}^T\mathbf{x} = c_0 + c_1x_1 + \cdots + c_nx_n$$

Minimize: Loss function

$$L(\mathbf{c}, \boldsymbol{\xi}) = L(c_0, c_1, \ldots, c_n, \xi_1, \xi_2, \ldots, \xi_M) =$$

$$\sqrt{c_1^2 + c_2^2 + \cdots + c_n^2} + \lambda \sum_{i=1}^{M} \xi_i$$

Subject to $y^{(i)}\mathbf{c}^T\mathbf{x}^{(i)} \geq 1 - \xi_i$, with $\xi_i \geq 0$

λ: regularization parameter

If $\lambda \to \infty$?

then $\sum_{i=1}^{M} \xi_i \to 0 \Rightarrow \xi_i = 0 \Rightarrow$ hard margin

Simplify Loss Function

$$p_{\mathbf{c}}(\mathbf{x}) = \mathbf{c}^T\mathbf{x} = c_0 + c_1x_1 + \cdots + c_nx_n$$

Minimize:

$$L(\mathbf{c}, \boldsymbol{\xi}) = L(c_0, c_1, \ldots, c_n, \xi_1, \xi_2, \ldots, \xi_M) =$$

$$\sqrt{c_1^2 + c_2^2 + \cdots + c_n^2} + \lambda \sum_{i=1}^{M} \xi_i$$

Subject to $y^{(i)}\mathbf{c}^T\mathbf{x}^{(i)} \geq 1 - \xi_i$, with $\xi_i \geq 0$

Hinge loss

$$\xi_i = \max(0,1 - y^{(i)}\mathbf{c}^T\mathbf{x}^{(i)})$$

Simplify Loss Function

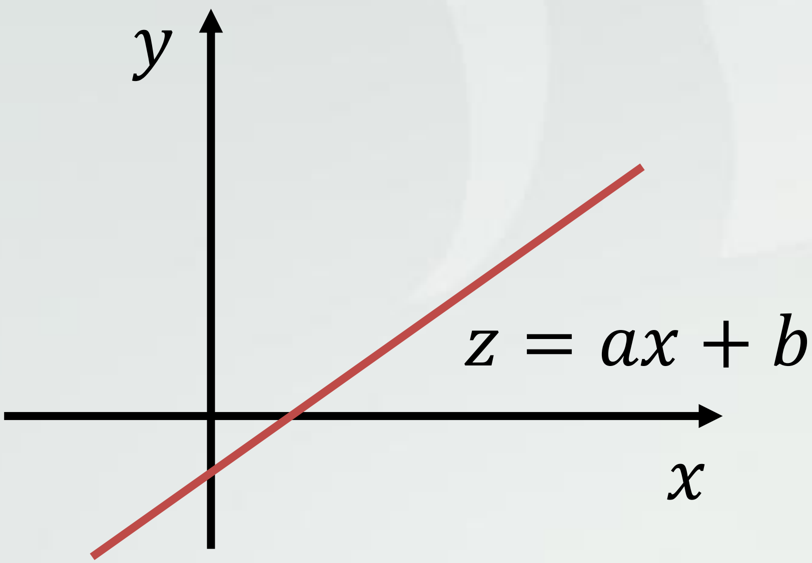
$$p_{\mathbf{c}}(\mathbf{x}) = \mathbf{c}^T\mathbf{x} = c_0 + c_1 x_1 + \cdots + c_n x_n$$

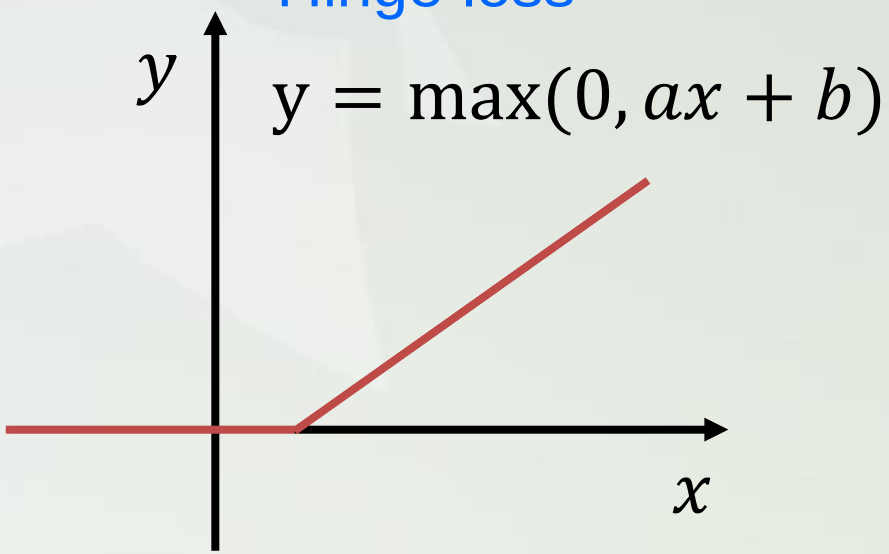
Minimize:

$$L(\mathbf{c}, \boldsymbol{\xi}) = L(c_0, c_1, \ldots, c_n, \xi_1, \xi_2, \ldots, \xi_M) =$$

$$\sqrt{c_1^2 + c_2^2 + \cdots + c_n^2} + \lambda \sum_{i=1}^{M} \underline{\max(0, 1 - y^{(i)}\mathbf{c}^T\mathbf{x}^{(i)})}$$

Hinge loss





How to Minimize Loss Function

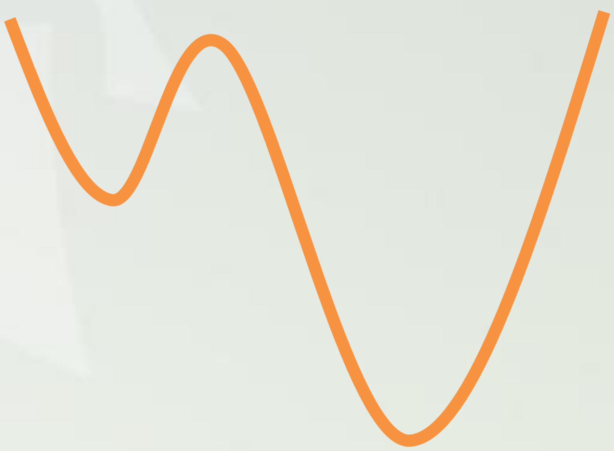
Minimize:

$$L(\mathbf{c}) = L(c_0, c_1, \dots, c_n) = \sqrt{c_1^2 + c_2^2 + \cdots + c_n^2} + \lambda \sum_{i=1}^{M} \max(0, 1 - y^{(i)}\mathbf{c}^T\mathbf{x}^{(i)})$$

- Our loss function is convex

Convex

Convex



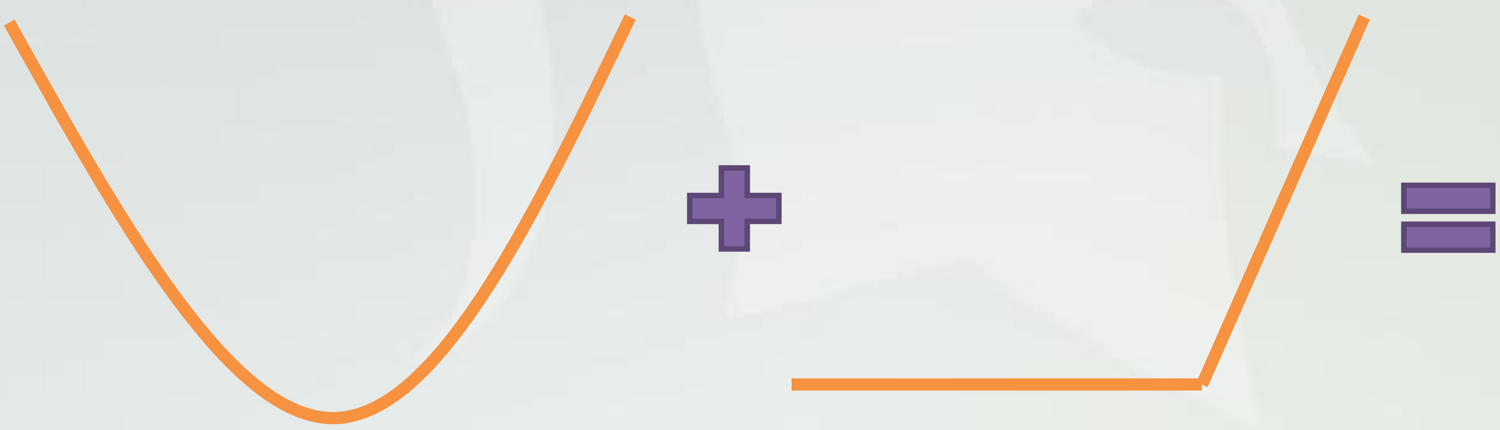
Not Convex

How to Minimize Loss Function

Minimize:

$$L(\mathbf{c}) = L(c_0, c_1, \dots, c_n) =$$

$$\sqrt{c_1^2 + c_2^2 + \cdots + c_n^2} + \lambda \sum_{i=1}^{M} \max(0, 1 - y^{(i)} \mathbf{c}^T \mathbf{x}^{(i)})$$



Convex

Convex

Convex

How to Minimize Loss Function

Minimize:

$$L(\mathbf{c}) = L(c_0, c_1, \dots, c_n) =$$

$$\sqrt{c_1^2 + c_2^2 + \cdots + c_n^2} + \lambda \sum_{i=1}^{M} \max(0, 1 - y^{(i)} \mathbf{c}^T \mathbf{x}^{(i)})$$

- The loss function is convex
- In convex function, local minimum is the global minimum
- Loss function can be optimized by
  - Quadratic optimization method
  - Gradient descent (continuity condition)?

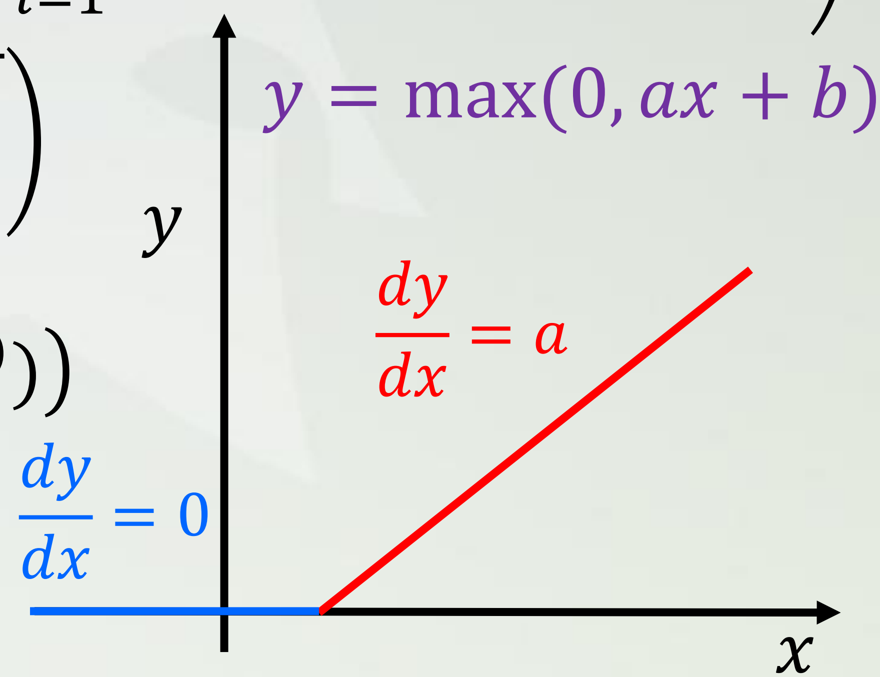
Sub-gradient descent

For non-differentiable objective functions

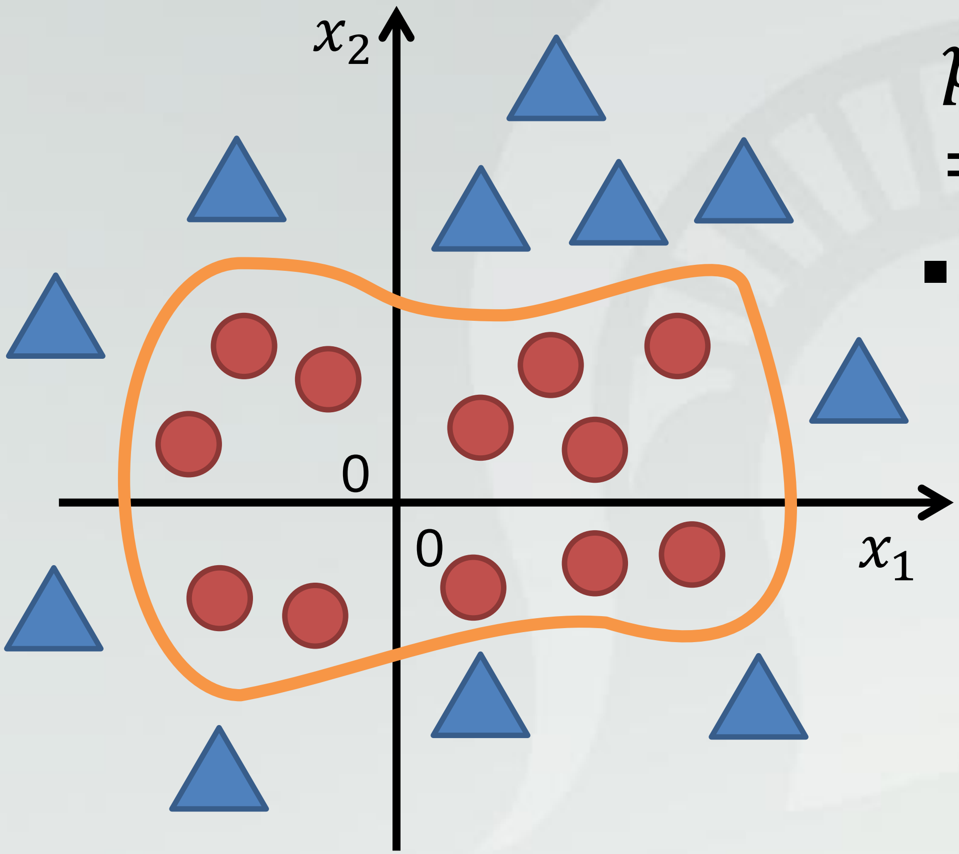
$$\mathbf{c} := \mathbf{c} - \alpha \nabla_{\mathbf{c}} L(\mathbf{c})$$

$$:= \mathbf{c} - \alpha \nabla_{\mathbf{c}} \left(\sqrt{c_1^2 + c_2^2 + \cdots + c_n^2} + \lambda \sum_{i=1}^{M} \max(0, 1 - y^{(i)} \mathbf{c}^T \mathbf{x}^{(i)}) \right)$$

$$:= \mathbf{c} - \alpha \nabla_{\mathbf{c}} \left(\sqrt{c_1^2 + c_2^2 + \cdots + c_n^2} \right) - \lambda \sum_{i=1}^{M} \nabla_{\mathbf{c}} \left(\max(0, 1 - y^{(i)} \mathbf{c}^T \mathbf{x}^{(i)}) \right)$$



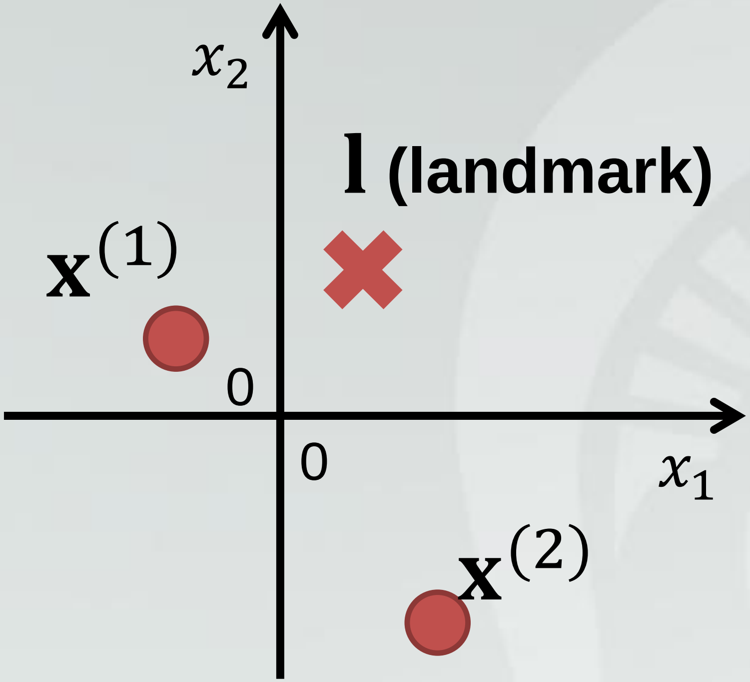
SVM for Nonlinear Classifiers



- Linear predictor:
  $p_{\mathbf{c}}(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$
  $= c_0 + c_1x_1 + \cdots + c_nx_n$
- Nonlinear predictor => nonlinear decision boundary:
  - $p_{\mathbf{c}}(\mathbf{x}) = c_0 + c_{11}x_1 + \cdots + c_{1k}x_1^k + c_{21}x_2 + \cdots$
  - $p_{\mathbf{c}}(\mathbf{x}) = c_0 + c_1x_1 + c_2x_1^2 + c_3x_1x_2 + \cdots + c_mx_1x_2 \ldots x_n$

Drawback: High risk of overfitting

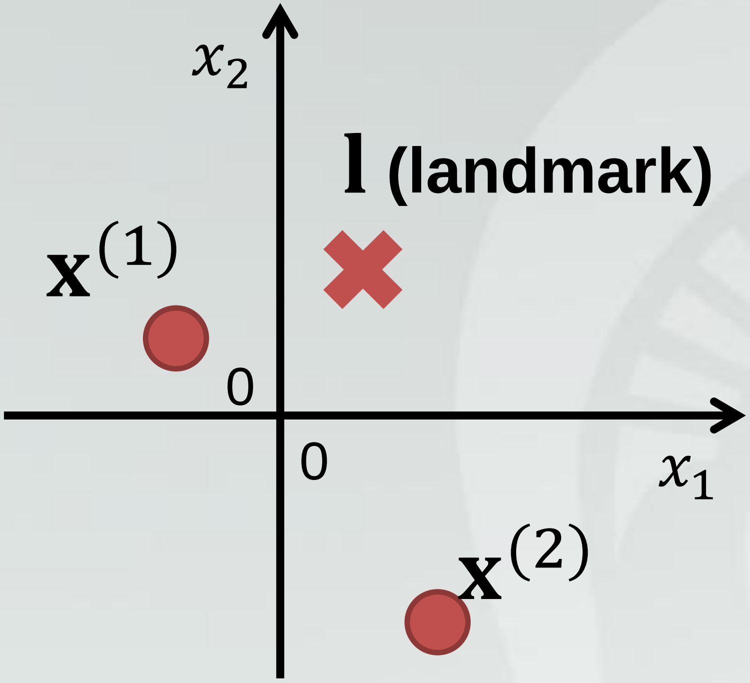
SVM for Nonlinear Classifiers



- Use kernel (Kernel method, Vapnik 1963)
  - A similarity function $k(\mathbf{x}, \mathbf{l})$
  - $k(\mathbf{x}, \mathbf{l})$ define how similar a given data point $\mathbf{x}$ to the pre-defined landmark $\mathbf{l}$
  - $\mathbf{x}^{(1)}$ is more similar (or close) to $\mathbf{l}$ than $\mathbf{x}^{(2)}$ if

$$k(\mathbf{x}^{(1)}, \mathbf{l}) > k(\mathbf{x}^{(2)}, \mathbf{l})$$

SVM for Kernel Classifiers



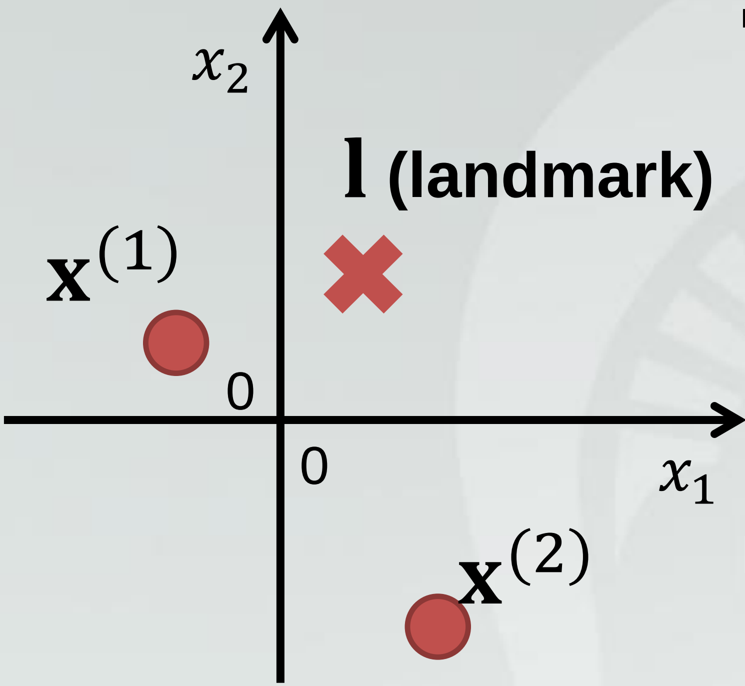
- Use kernel
  - A similarity function $k(\mathbf{x}, \mathbf{l})$
  - $k(\mathbf{x}, \mathbf{l})$ define how similar a given data point $\mathbf{x}$ to the pre-defined landmark $\mathbf{l}$
  - $\mathbf{x_1}$ is more similar (or close) to $\mathbf{l}$ than $\mathbf{x_2}$ if $k(\mathbf{x_1}, \mathbf{l}) > k(\mathbf{x_2}, \mathbf{l})$

Kernel functions:

$$k(\mathbf{x}, \mathbf{l}) = \frac{1}{1 + \|\mathbf{x} - \boldsymbol{l}\|}$$

$$k(\mathbf{x}, \mathbf{l}) = \frac{1}{1 + \left(\frac{\|\mathbf{x} - \boldsymbol{l}\|}{\eta}\right)^{\nu}} \quad \text{(lorentz)}$$

SVM for kernel Classifiers



- Use kernel
  - $\mathbf{x}^{(1)}$ is more similar (or close) to $\mathbf{l}$ than $\mathbf{x}^{(2)}$ if $k(\mathbf{x}^{(1)}, \mathbf{l}) > k(\mathbf{x}^{(2)}, \mathbf{l})$
  - Kernel functions:

$$k(\mathbf{x}, \mathbf{l}) = \frac{1}{1 + \|\mathbf{x} - \mathbf{l}\|}$$

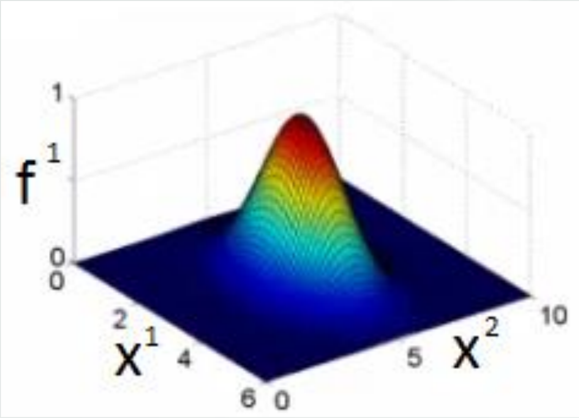
$$k(\mathbf{x}, \mathbf{l}) = \frac{1}{1 + \left(\frac{\|\mathbf{x} - \mathbf{l}\|}{\sigma}\right)^{\nu}} \quad \text{(Lorentz)}$$

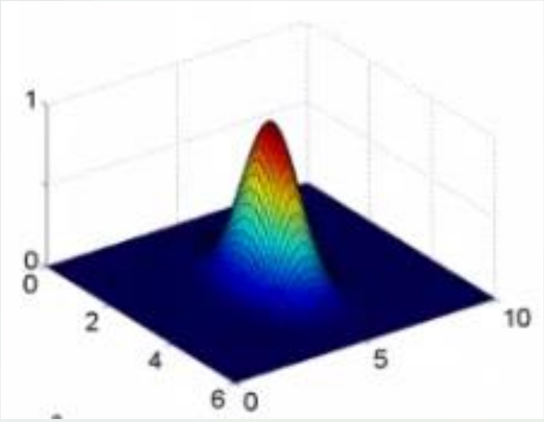
$$k(\mathbf{x}, \mathbf{l}) = e^{-\left(\frac{\|\mathbf{x} - \mathbf{l}\|}{\sigma}\right)^{\nu}} \quad \text{(exponential)}$$

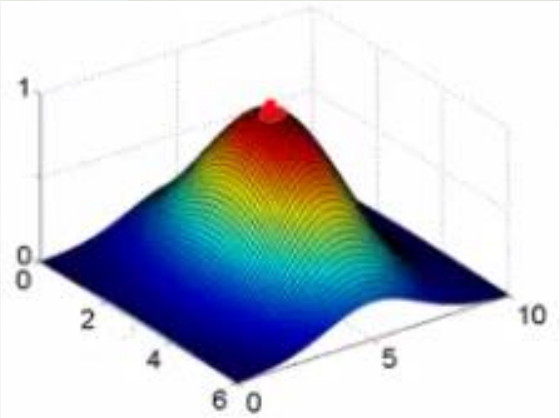
- if $\mathbf{x}$ very close to $\mathbf{l} \Rightarrow \|\mathbf{x} - \mathbf{l}\| \to 0 \Rightarrow k(\mathbf{x}, \mathbf{l}) \to 1$
- if $\mathbf{x}$ far away from $\mathbf{l} \Rightarrow \|\mathbf{x} - \mathbf{l}\| \to \infty \Rightarrow k(\mathbf{x}, \mathbf{l}) \to 0$
- In exponential kernel, when $\nu = 2$ we get Gauss kernel $e^{-\left(\frac{\|\mathbf{x} - \mathbf{l}\|}{\sigma}\right)^{2}}$

SVM for Nonlinear Classifiers

- Gaussian kernel: $\mathrm{k}(\mathbf{x}, \mathbf{l}) = e^{-\left(\frac{\|\mathbf{x}-\mathbf{l}\|}{\sigma}\right)^2}$
- σ: standard deviation
- σ^2: variance, define how steep from the landmark (the top) to the ground
- $\mathbf{l} = (3,5)^T$ with three σ^2 values: 1, 0.5, and 3.0

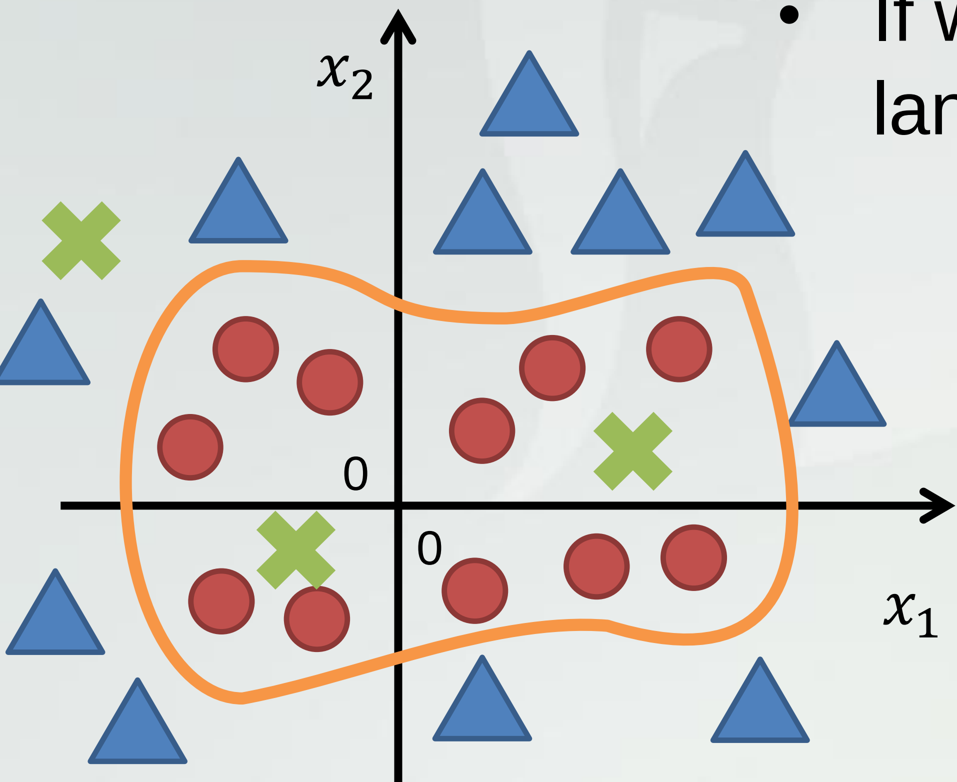






Predictor with Kernels

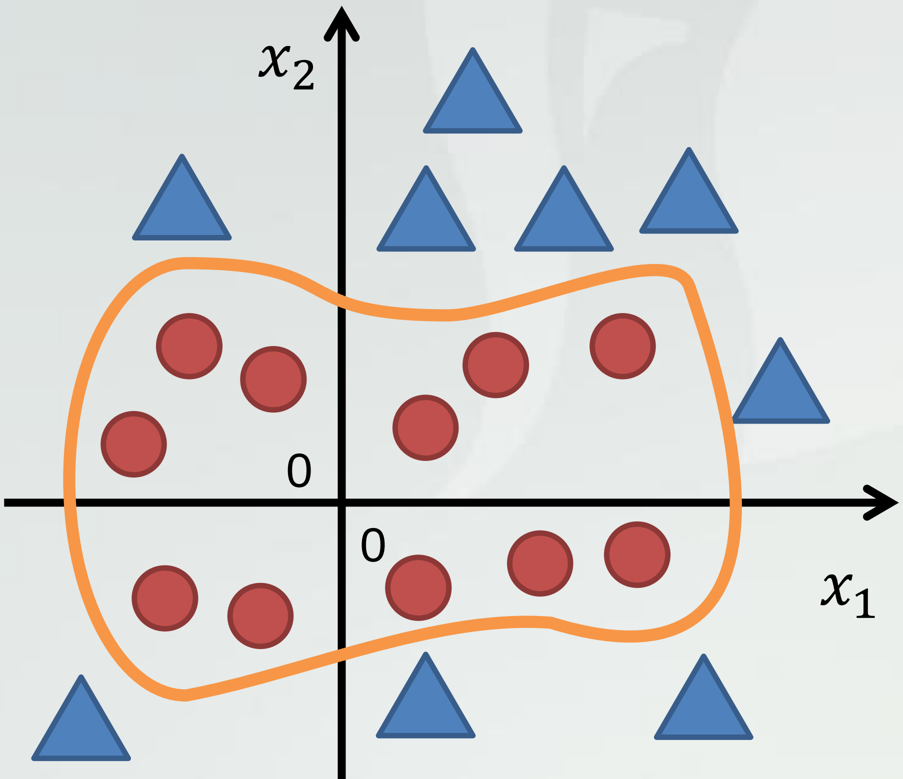
- Make use the predictor for linear classifier
$$p_{\mathbf{c}}(\mathbf{x}) = c_0 + c_1 x_1 + \cdots + c_n x_n$$
- If we use landmarks = use similarity functions = use kernels



- If we use three landmarks: $\mathbf{l}^{(1)}, \mathbf{l}^{(2)}, \mathbf{l}^{(3)}$
$$p_{\mathbf{c}}(\mathbf{x}) = c_0 + c_1 k\left(\mathbf{x}, \mathbf{l}^{(1)}\right) + c_2 k\left(\mathbf{x}, \mathbf{l}^{(2)}\right) + c_3 k\left(\mathbf{x}, \mathbf{l}^{(3)}\right)$$

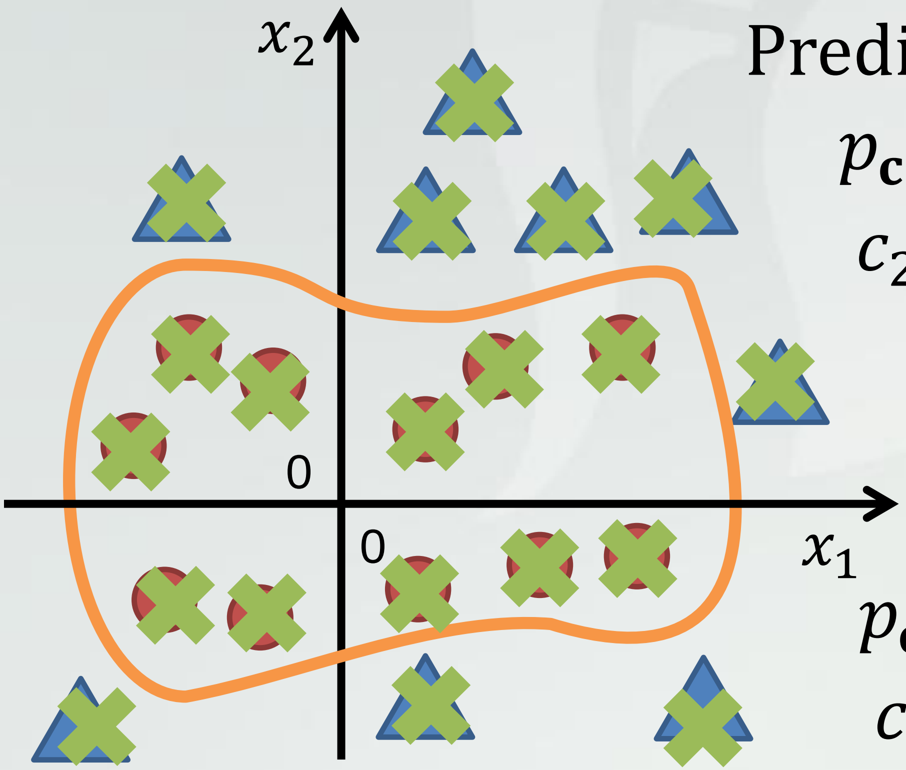
How to Choose Landmarks?

- Assume our training data is
$$\left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right), \ldots, \left(\mathbf{x}^{(M)}, y^{(M)}\right)$$
- How to choose landmarks for a given training data? (kernel trick, Guyon and Vapnik, 1992)



How to Choose Landmarks?

- Assume our training data is
$$\left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right), \ldots, \left(\mathbf{x}^{(M)}, y^{(M)}\right)$$
- How to choose landmarks for a given training data? $\mathbf{l}^{(1)} = \mathbf{x}^{(1)}, \mathbf{l}^{(2)} = \mathbf{x}^{(2)}, \ldots, \mathbf{l}^{(M)} = \mathbf{x}^{(M)}$



Predictor for M landmarks

$$p_{\mathbf{c}}(\mathbf{x}) = c_0 + c_1 k\left(\mathbf{x}, \mathbf{l}^{(1)}\right) + c_2 k\left(\mathbf{x}, \mathbf{l}^{(2)}\right) + \cdots + c_M k\left(\mathbf{x}, \mathbf{l}^{(M)}\right)$$

We have

$$p_{\mathbf{c}}(\mathbf{x}) = c_0 + c_1 k\left(\mathbf{x}, \mathbf{x}^{(1)}\right) + c_2 k\left(\mathbf{x}, \mathbf{x}^{(2)}\right) + \cdots + c_M k\left(\mathbf{x}, \mathbf{x}^{(M)}\right)$$

Loss Function with Kernels

- Predictor

$$p_{\mathbf{c}}(\mathbf{x}) = c_0 + c_1 k\left(\mathbf{x}, \mathbf{x}^{(1)}\right) + \cdots + c_M k\left(\mathbf{x}, \mathbf{x}^{(M)}\right)$$

- Loss function without kernel

$$L(\mathbf{c}) = \sqrt{c_1^2 + c_2^2 + \cdots + c_n^2} + \lambda \sum_{i=1}^{M} \max(0, 1 - y^{(i)} \mathbf{c}^T \mathbf{x}^{(i)})$$

- Loss function with kernels

$$L(\mathbf{c}) = \sqrt{c_1^2 + c_2^2 + \cdots + c_M^2} + \lambda \sum_{i=1}^{M} \max(0, 1 - y^{(i)} \mathbf{c}^T \mathbf{K}(\mathbf{x}^{(i)}))$$

Loss Function with Kernels

- Predictor

$$p_{\mathbf{c}}(\mathbf{x}) = c_0 + c_1 k\left(\mathbf{x}, \mathbf{x}^{(1)}\right) + \cdots + c_M k(\mathbf{x}, \mathbf{x}^{(M)})$$

- Loss function with kernels

$$L(\mathbf{c})$$

$$= \sqrt{c_1^2 + c_2^2 + \cdots + c_M^2} + \lambda \sum_{i=1}^{M} \max(0, 1 - y^{(i)} \mathbf{c}^T \mathbf{K}(\mathbf{x}^{(i)}))$$

$$\mathbf{K}\left(\mathbf{x}^{(i)}\right) \equiv \left(1, k\left(\mathbf{x}^{(1)}, \mathbf{x}^{(i)}\right), k\left(\mathbf{x}^{(2)}, \mathbf{x}^{(i)}\right), \ldots, k\left(\mathbf{x}^{(M)}, \mathbf{x}^{(i)}\right)\right)^T$$

Kernel Selections for SVM

- Not all similarity kernels are valid. Must satisfy Mercer's theorem

$$k: X \times X \rightarrow \mathbb{R}$$

$$k(\mathbf{x}, \mathbf{z}) = k(\mathbf{z}, \mathbf{x}) \quad \text{(symmetric)}$$

$$\int \int g(\mathbf{x}) k(\mathbf{x}, \mathbf{y}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0$$

(positive semidefinite)

for all vector $g \in \mathcal{H}$ and k (Hilbert–Schmidt integral operator)

$$\int \int |k(\mathbf{x}, \mathbf{y})|^2 d\mathbf{x} d\mathbf{y} < \infty$$

Mercer's requirement ensures that the loss function is convex in the dual form when using quadratic optimization method.

Kernel Selections for SVM

- If kernel does not meet the Mercer conditions, no global minimum is guarantee, but one can use gradient descent to find a local minimum.

Commonly used Kernels

- Linear kernel (or dot product kernel)

$$k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$$

- Polynomial

$$k(\mathbf{x}, \mathbf{z}) = (\alpha \mathbf{x}^T \mathbf{z} + r)^d$$

- Radial basic function (RBF)

$$k(\mathbf{x}, \mathbf{z}) = e^{-\left(\frac{\|\mathbf{x}-\mathbf{z}\|}{\sigma}\right)^\nu}$$

- Sigmoid

$$\frac{1}{1 + e^{-\gamma \mathbf{x}^T \mathbf{z}}} \text{ or } \tanh(\gamma \mathbf{x}^T \mathbf{z} + r)$$

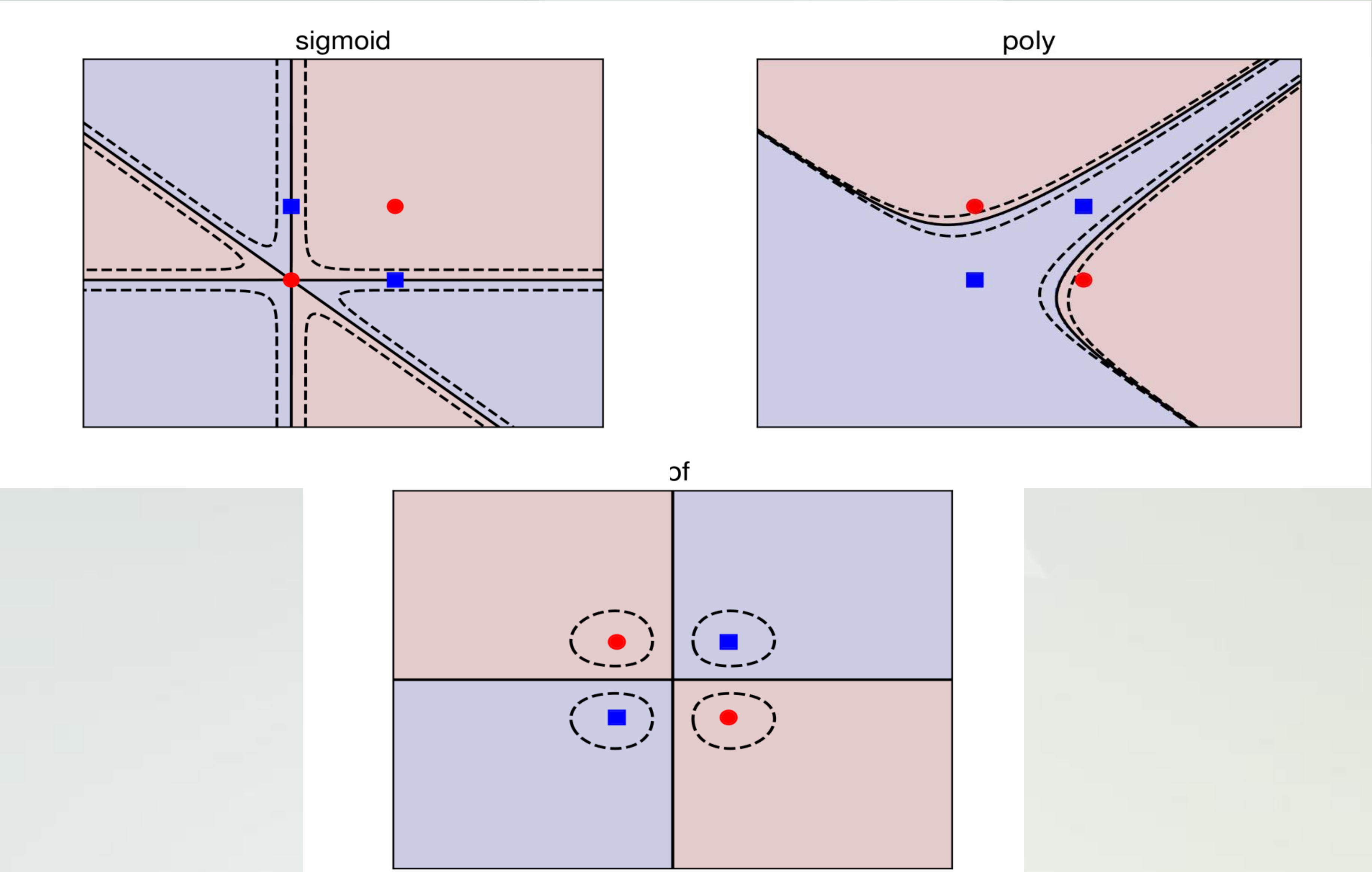
- Are these kernels Hilbert-Schmidt?

Discussions

How to Choose Kernel?

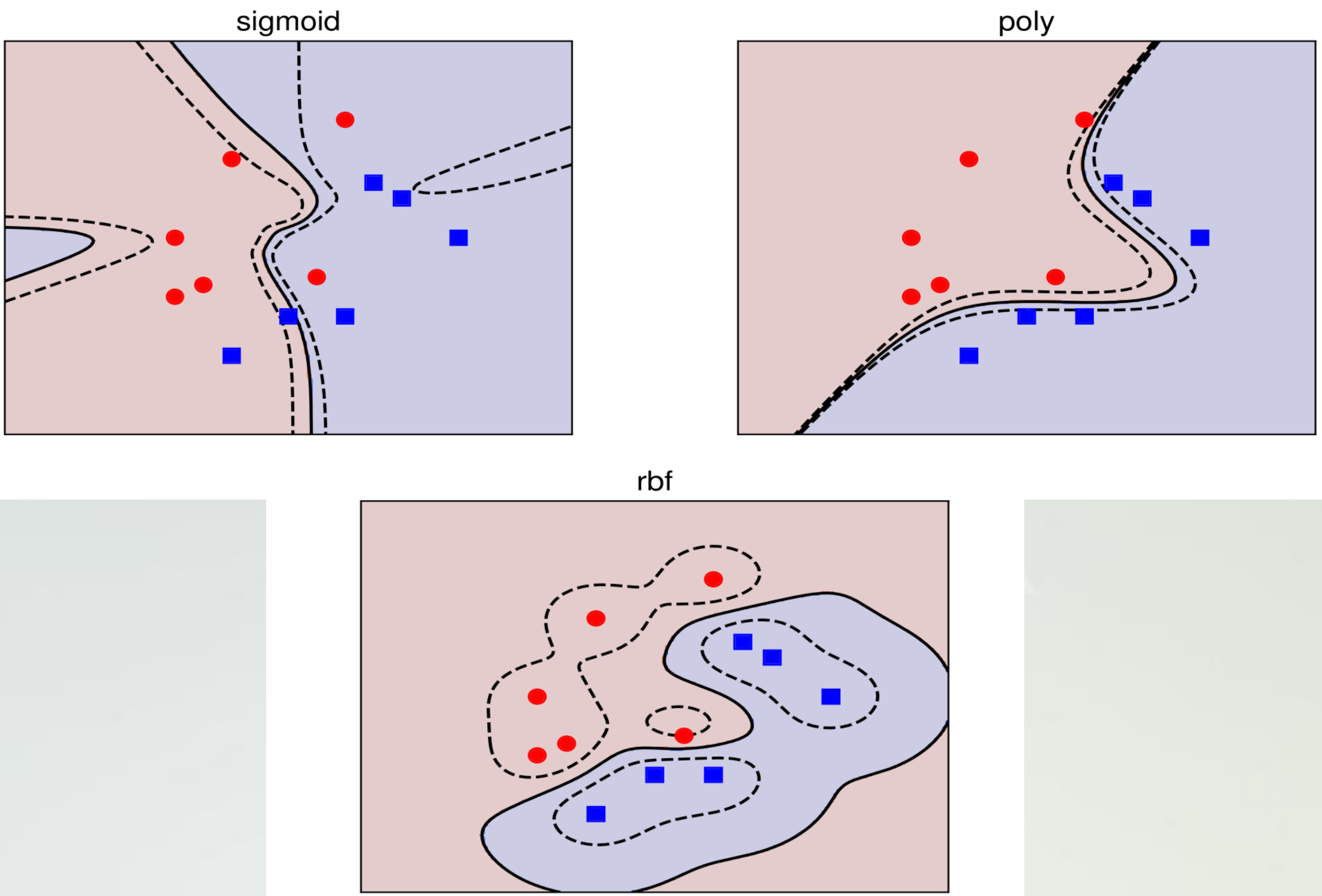
- Radial basic functions are commonly used
- Use polynomial for linear separation
- Sigmoid often performs worst
- Should try a variety of kernels for a given problem

Discussions -- Examples



(Machinelearningcoban.com)

Discussions -- Examples



(Machinelearningcoban.co

Discussions

- Support vector clustering (for unsupervised learning), a fundamental method in data science
- Multiclass SVM:
  - ❖ multiple binary classification problems: https://link.springer.com/chapter/10.1007%2F11494683_28.
  - ❖ single optimization problem: http://jmlr.csail.mit.edu/papers/volume2/crammer01a/crammer01a.pdf

Discussions

- **Support vector regression (SVR) (**Vladimir N. Vapnik)

Minimize $\frac{1}{2}||\bar{\boldsymbol{c}}||^2$

subject to $\begin{cases} y^{(i)} - \mathbf{c}^T\mathbf{x}^{(i)} \le \varepsilon \\ \mathbf{c}^T\mathbf{x}^{(i)} - y^{(i)} \le \varepsilon \end{cases}$ (where $\varepsilon \ge 0$)

- **Least squares support vector machine (LS-SVM): (**Suykens and Vandewalle)

Discussions

- **Mathematical issues?**
  1. Kernels (Reproducing kernels, Frames, Separable, etc.)
  2. Regularization and stability (Tikhonov)

$$\arg\min_{f \in \mathcal{H}} L(\mathbf{c}) + \mathcal{R}(\mathbf{K}), \quad \text{where } \mathcal{R}(f) = \gamma_{\boldsymbol{A}} \|f\|_{\mathcal{H}}^2$$

$$L(\mathbf{c}) = \sqrt{c_1^2 + \cdots + c_M^2} + \lambda \sum_{i=1}^{M} \max(0, 1 - y^{(i)} \mathbf{c}^T \mathbf{K}(\mathbf{x}^{(i)}))$$

$$f = \sum_{i=1}^{M} \mathbf{c}^T \mathbf{K}(\mathbf{x}^{(i)})$$

Discussions

- **Transductive support vector machines (semi-supervised learning):** The training and test sets are minimized together.

$$\text{Training set: } \mathcal{D} = \left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \middle| \mathbf{x}^{(i)} \in \mathbb{R}^n, y^{(i)} \in \{-1,1\}\right\}_{i=1}^{M}$$

$$\text{Test set: } \mathcal{D}^* = \left\{\mathbf{x}^{(i)} \middle| \mathbf{x}^{(i)} \in \mathbb{R}^n \right\}_{i=1}^{N}$$

- **Manifold learning for semi-supervised learning:**

$$\underset{f \in \mathcal{H}}{\arg\min}\, L(\mathbf{c}) + \mathcal{R}(f),$$

$$\mathcal{R}(f) = \gamma_{\boldsymbol{A}} \|f\|_{\mathcal{H}}^2 + \gamma_I \|f\|_I^2$$

$$\|f\|_I^2 = \frac{1}{(M+N)^2} \sum_{i,j=1}^{M+N} W_{ij} \left(f(\mathbf{x}_i) - f\left(\mathbf{x}_j\right)\right)$$