Since 2004
UET
ĐẠI HỌC CÔNG NGHỆ, ĐHQGHN
VNU-University of Engineering and Technology

Since 1906
VNU
ĐẠI HỌC QUỐC GIA HÀ NỘI
Vietnam National University, Hanoi

INT3405 - Machine Learning

Lecture 5: Classification (P2) - SVM

Duc-Trong Le, Viet-Cuong Ta

Hanoi, 02/2023

Outline

- Problem and Intuition: Perceptron
- Formulation of Linear SVM
  - Hard Margin SVM
  - Soft Margin SVM
  - Primal/dual Problems
- Nonlinear SVM with Kernel
  - Kernel Tricks
  - SVM with Kernel
- Multi-class classification

Recap: Bayes Theorem & Decision Boundary

Posterior $\propto$ Likelihood Prior

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Decision Boundary

$$\ln \frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})} = b + \mathbf{x}^\top \mathbf{w}$$

Problem Settings

- **Problem Setting**
  - Training data

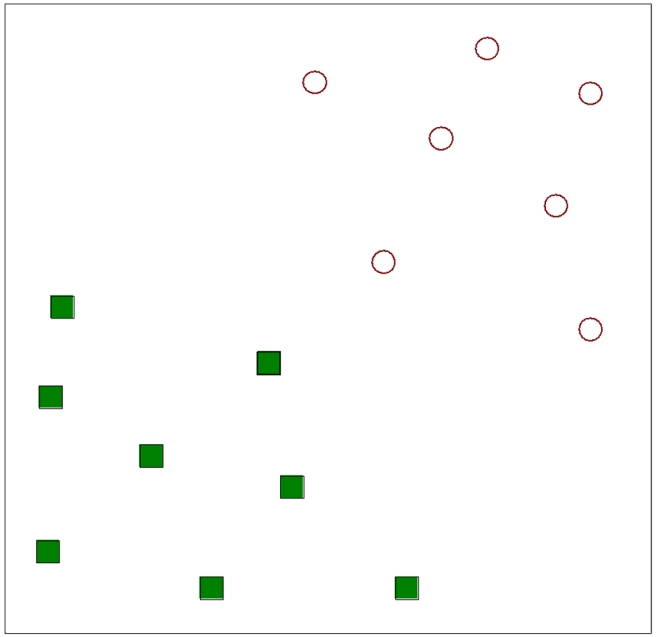
    $$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots$$

  - For two-class (binary) classification

    $$y_i \in \{+1, -1\}$$

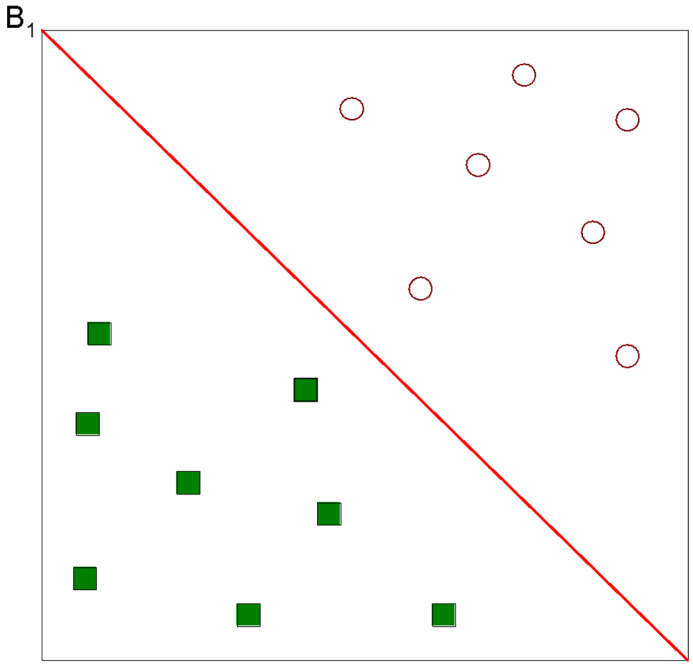
- **Goal**
  - To find an optimal linear hyperplane (decision boundary) that separates all the data

    $$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^\top \mathbf{x} + b)$$



Intuition (1)

- One possible solution



Perceptron

The Perceptron[1] model infers the weight vector $\mathbf{w} \in \mathbb{R}^d$ and the value b representing the hyperplane

$$(H) : \{\mathbf{x} : \mathbf{w}^T\mathbf{x} + b = 0\}$$

that separates the space $\mathbb{R}^d$ into two parts:
the positive class (positive, $y = +1$)

$$(+) : \{\mathbf{w}^T\mathbf{x} + b \geq 0\}$$

and the negative class (negative, $y = -1$)

$$(-) : \{\mathbf{w}^T\mathbf{x} + b < 0\}$$

Perceptron

A dataset $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ in $\mathbb{R}^d \times \{-1, +1\}$ is **linearly separable** if $\exists \mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that

$$s_i = y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0, \forall i = 1, 2, \ldots, n$$

in other words, the hyperplane H completely separate the dataset D into negative and positive classes according to the labels y_i.

Mini exercises:

- Use a linear function to represent the AND function.
- Use a linear function to represent the OR function.
- Use linear functions to represent the XOR function.

Perceptron

Assume that D is linearly separable by a hyperplane H, The minimum distance from a data point to H is called as the margin of that hyperplane:

$$\delta = \min_{i=1}^{n} \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|}$$

Clearly, we have:

- If $\|\mathbf{w}\| = 1$ then $\delta = \min_{i=1}^{n} |\mathbf{w}^T \mathbf{x}_i + b| = \min_{i=1}^{n} y_i(\mathbf{w}^T x_i + b)$.
- If $\min_i |\mathbf{w}^T \mathbf{x}_i + b| = 1$ then $\delta = 1/\|\mathbf{w}\|$.

Perceptron: Algorithm

Find $\mathbf{w}$ and b subject to: $s_i = y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 0, \forall i = 1, \ldots, n$

$\text{Perceptron}(D)$

- Initialize $\mathbf{w}_{(0)} = 0, b_{(0)} = 0, t = 0$
- Iterate through the dataset multiple times, for each data sample $\mathbf{x}_i, y_i$
  - Calculate the score $s_i = y_i(\mathbf{w}_{(t)}^T\mathbf{x}_i + b_{(t)})$
  - If $s_i \geq 0$, skip (correctly classified)
  - If $s_i < 0$ (falsely classified), update $\mathbf{w}_{(t)}$ in the direction of the derivative $\frac{\partial s_i}{\partial \mathbf{w}}$ to increase s_i.

$$\mathbf{w}_{(t+1)} \leftarrow \mathbf{w}_{(t)} + y_i\mathbf{x}_i$$
$$b_{(t+1)} \leftarrow b_{(t)} + y_i$$
$$t \leftarrow t + 1$$

- Stop when correctly classifying all data: $s_i \geq 0, \forall i$.

Perceptron: Theory

Theorem: If there exists $\mathbf{w}_\star, b_\star$ such that $y_i(\mathbf{w}_\star^T \mathbf{x}_i + b) \geq \delta > 0$, then the maximum number of updates of Perceptron algorithm[2] is

$$t \leq \frac{R^2(\|\mathbf{w}^\star\|^2 + b^2)}{\delta^2}$$

where R is the radius of the dataset, $R^2 = \max_i \|\mathbf{x}_i\|^2 + 1$.

Question:

- Can you differ easy and diffcult dataset, given its linear seperation properties?

Perceptron: Proof

Proof: For convenience, let $\mathbf{v} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}, \mathbf{z} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$ then $\mathbf{v}^T\mathbf{z} = \mathbf{w}^T\mathbf{x} + b$. After each update we have

$$\|\mathbf{v}_{(t)}\|^2 = \|\mathbf{v}_{(t-1)} + y_i\mathbf{z}_i\|^2 \tag{1}$$

$$= \|\mathbf{v}_{(t-1)}\|^2 + \underbrace{y_i^2}_{1}\|\mathbf{z}_i\|^2 + 2\underbrace{y_i\mathbf{v}_{(t-1)}^T\mathbf{z}_i}_{s_i<0} \tag{2}$$

$$\leq \|\mathbf{v}_{(t-1)}\|^2 + R^2 \leq tR^2 \tag{3}$$

Furthermore,

$$\|\mathbf{v}_\star\| \cdot \|\mathbf{v}_{(t)}\| \geq \mathbf{v}_\star^T\mathbf{v}_{(t)} = \mathbf{v}_\star^T(\mathbf{v}_{(t-1)} + y_i\mathbf{z}_i) \tag{4}$$

$$\geq \mathbf{v}_\star^T\mathbf{v}_{(t-1)} + \delta \geq t\delta \tag{5}$$

$$\Rightarrow \|\mathbf{v}_\star\| \geq \frac{t\delta}{R\sqrt{t}} = \frac{\sqrt{t}\delta}{R} \qquad \text{Divide by sqrt of (3)} \tag{6}$$

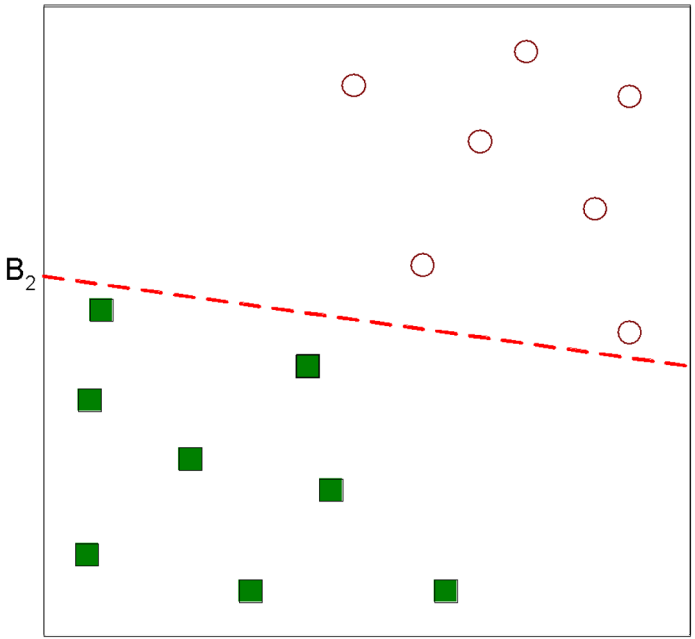
Exercises

Start with w = (w1=0, w2=0), b = 1, do three steps update and output the resulting (w, b). Could you compute the upper bound of the number of steps, which are required to learn the data.

x1	x2	y
1	9	-1
5	5	-1
1	1	-1
8	5	+1
13	1	+1
13	9	+1

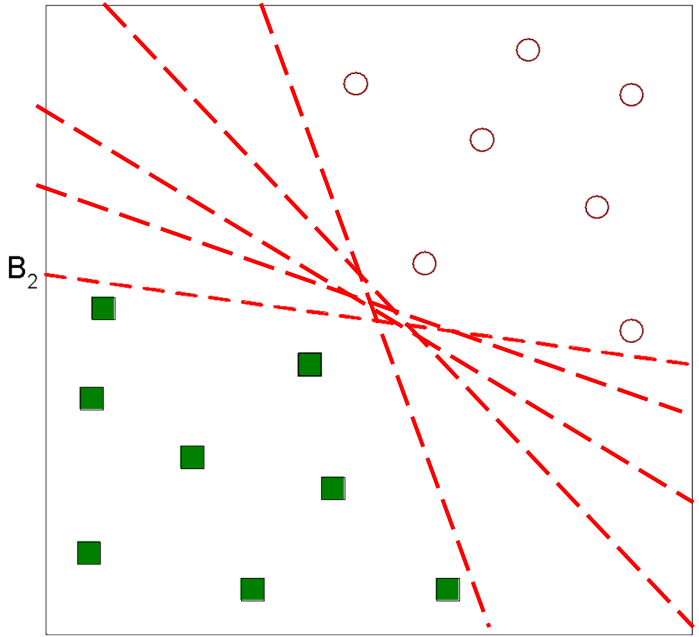
Intuition (2)

- Another possible solution



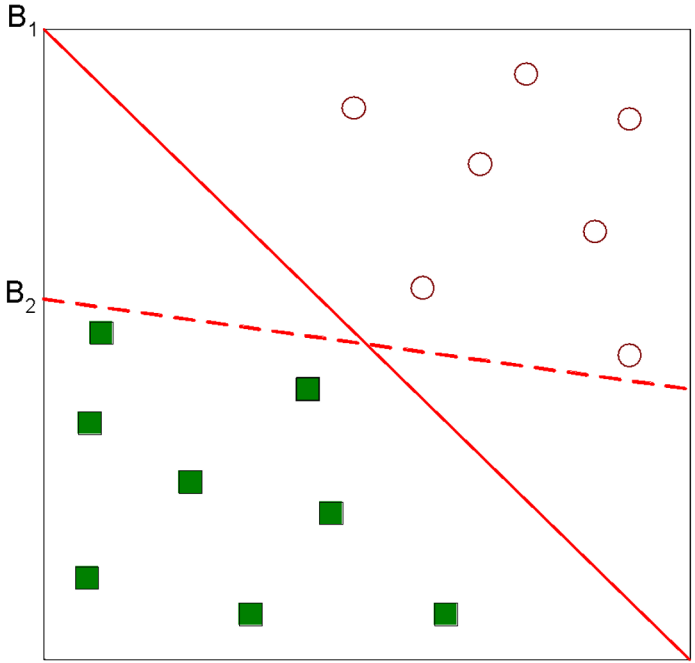
Intuition

- Too many other possible solutions



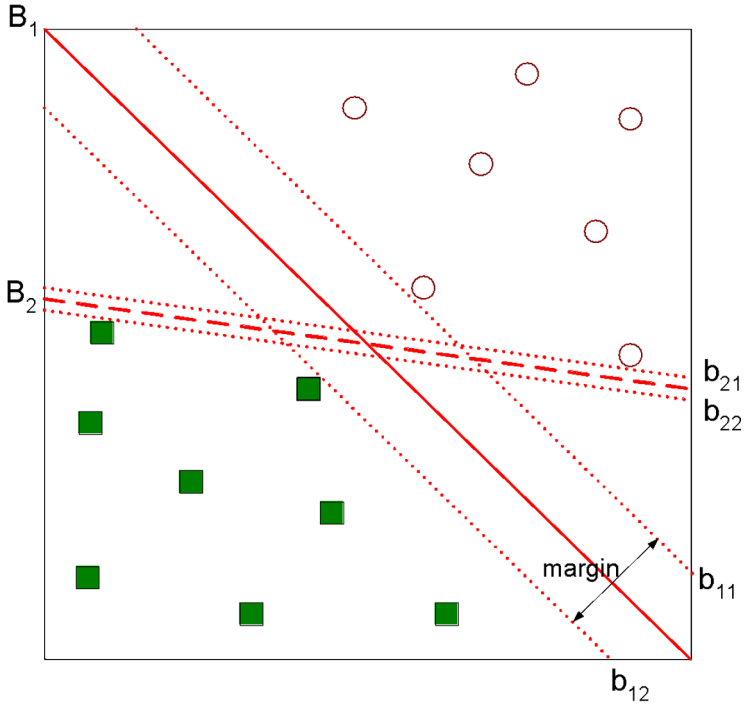
Intuition

- Which one is better than the other?
- How to define better?

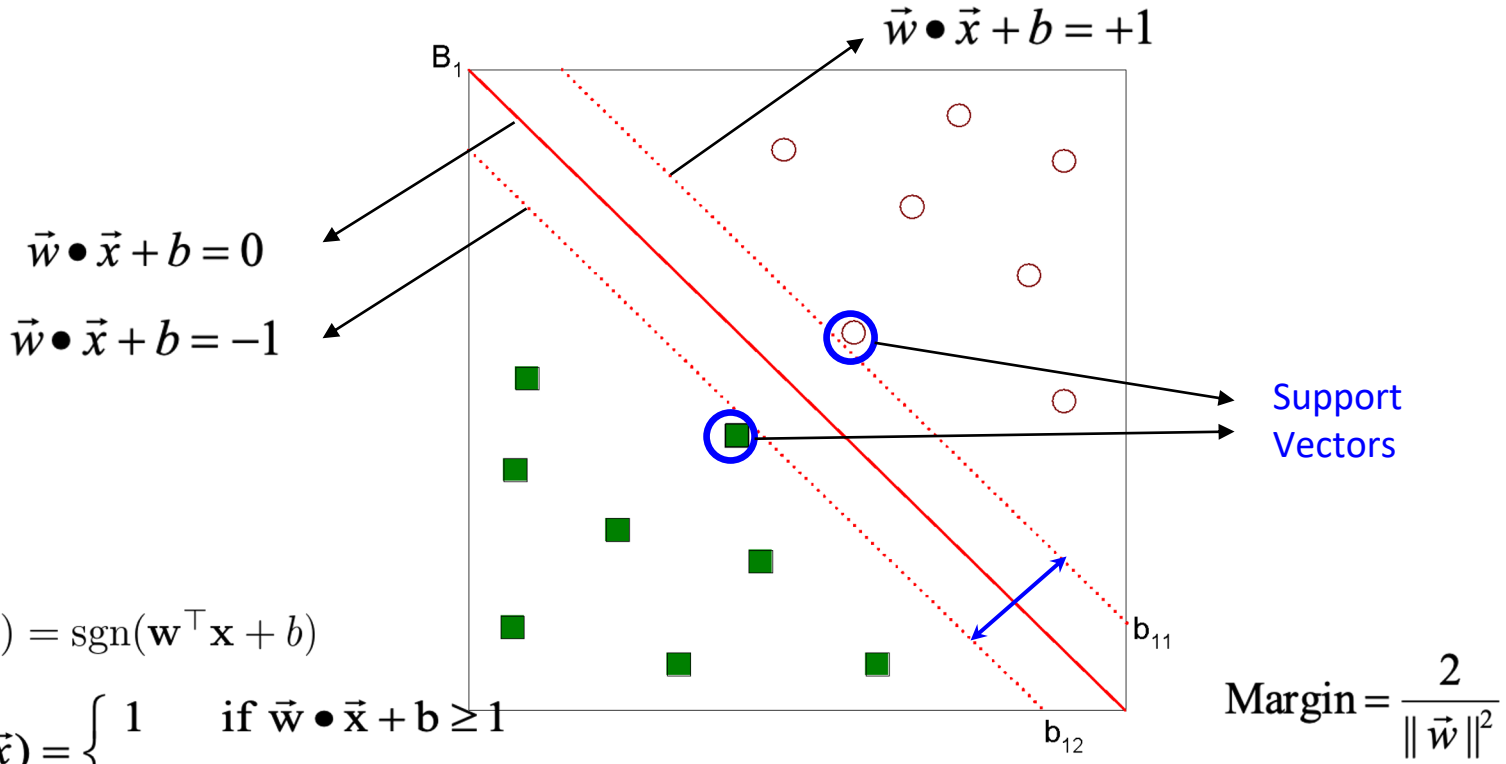


Intuition: Maximum Margin

- Intuition of “Margin”
  - The margin of a linear classifier as the width that the boundary could be increased by before hitting a data point.
- Idea of SVM
  - Find the separating hyperplane maximizing the margin



Support Vector Machines (SVM)



$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^\top \mathbf{x} + b)$$

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\| \vec{w} \|^2}$$

SVM: Optimization Formulation (1)

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^\top \mathbf{x} + b)$$

- **From Margin to Norm**
  - Margin: distance between $\mathbf{w}^T\mathbf{x} + b = 1$ and -1:

$$2/\|\mathbf{w}\| = 2/\sqrt{\mathbf{w}^T\mathbf{w}}$$

  - Maximizing margin is equivalent to minimizing $\frac{1}{2}\|\mathbf{w}\|^2$
- **Constraints**
  - Separation with margin, i.e.,

$$\begin{aligned} \mathbf{w}^\top \mathbf{x}_i + b &\geq 1 \quad \text{if} \quad y_i = +1 \\ \mathbf{w}^\top \mathbf{x}_i + b &\leq -1 \quad \text{if} \quad y_i = -1 \end{aligned}$$

  - Simplified as the equivalent constraint

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$$

SVM: Optimization Formulation (2)

- SVM as a Quadratic Programming (QP) problem

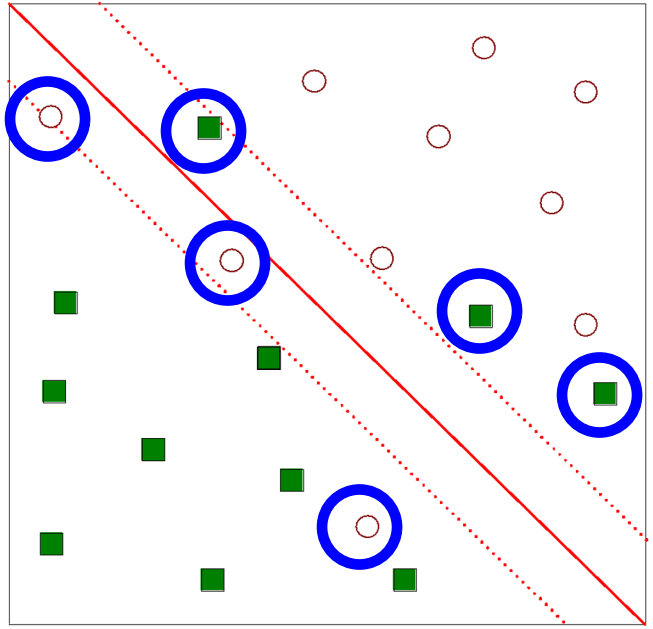
$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \\ & i = 1, \ldots, l. \end{aligned}$$

  - Convex problem, has unique minimum
  - Quadratic objective function
  - Linear equality and inequality constraints

Linearly Non-separable Cases

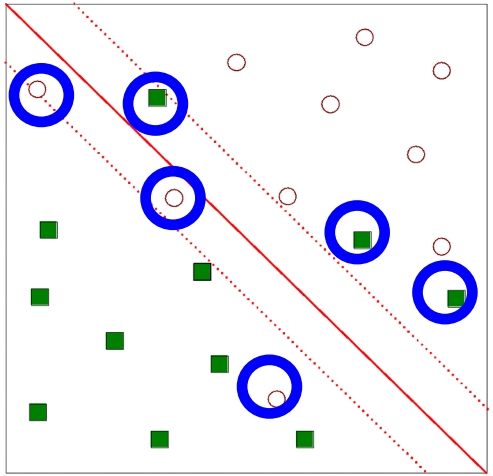
- What if the data cannot be linearly separable?

- For such case,
  Hard margin SVM cannot be applied directly



Soft Margin SVM

- Standard Linear SVM
  - Introduce slack variables
  - Relax the constraints
  - Penalize the relaxation



Primal Problem:

$$\min_{\mathbf{w},b,\xi} \quad \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to} \quad y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1 - \xi_i,$$
$$\xi_i \geq 0, i = 1, \ldots, N$$

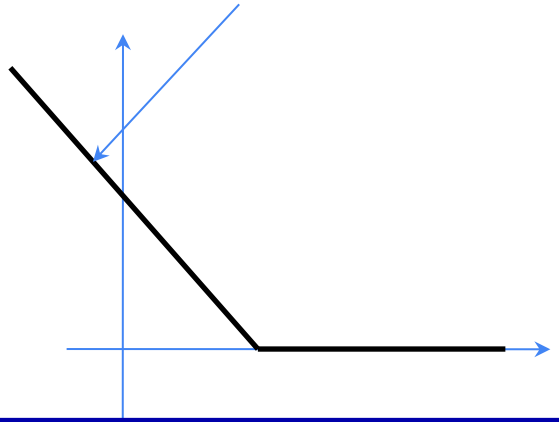
C is a regularization parameter. Soft margin SVM trade off between maximizing the margin and minimizing the misclassification error rate

Linearly Non-separable Case

- Re-written as an unconstrained optimization:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\sum_{j=1}^{d} w_j^2 + C\sum_{i=1}^{N} \ell(y_i[\mathbf{x}_i^\top \mathbf{w} + b])$$

$$\text{Hinge loss } \ell(z) = \max(0, 1 - z)$$



Linearly Non-separable Case

Model Complexity

Training Error

Support Vector Machine

$$\min_{\mathbf{w},b} \frac{1}{2}\sum_{j=1}^{d} w_j^2 + C\sum_{i=1}^{N} \max(0, 1 - y_i[\mathbf{x}_i^\top \mathbf{w} + b])$$

Regularized logistic regression

$$\min_{\mathbf{w},b} \frac{1}{2}\sum_{j=1}^{d} w_j^2 + C\sum_{i=1}^{N} \ln(1 + \exp(-y_i[\mathbf{x}_i^\top \mathbf{w} + b]))$$

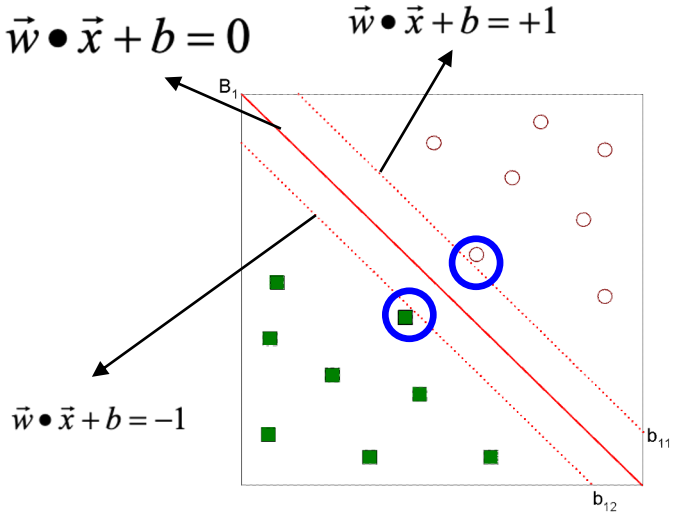
Choice of Parameter C:

Large C: Lower bias, high variance

Small C: Higher bias, low variance

Dual Form of SVM

$$\max_{\alpha_i \in [0,C]} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^\top \mathbf{x}_j) : \sum_{i=1}^{N} \alpha_i y_i = 0 \right\}$$



$$\alpha_i (1 - y_i (\langle w, x_i \rangle + b)) = 0.$$

Support vectors: $\alpha_i > 0$

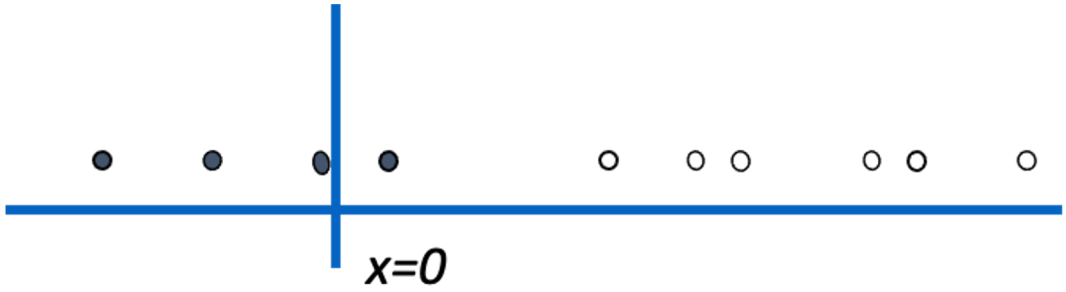
$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) = 1$$

$$\mathbf{w} \cdot \mathbf{x_i} - b = 1/y_i = y_i \iff b = \mathbf{w} \cdot \mathbf{x_i} - y_i$$

$$b = \frac{1}{N_{SV}} \sum_{i=1}^{N_{SV}} (\mathbf{w} \cdot \mathbf{x_i} - y_i)$$

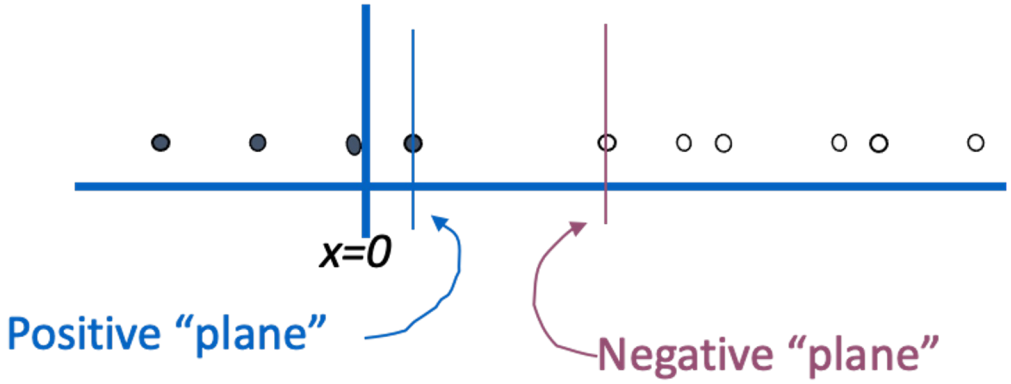
Suppose we're in 1-dimension

What would SVMs do with this data?

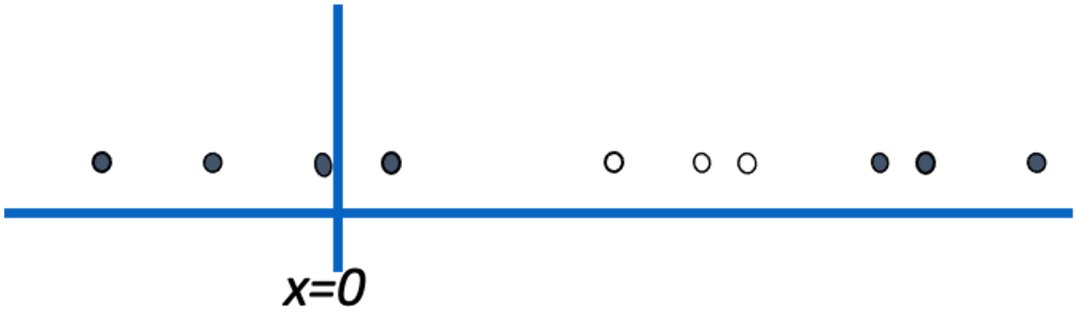


Suppose we're in 1-dimension

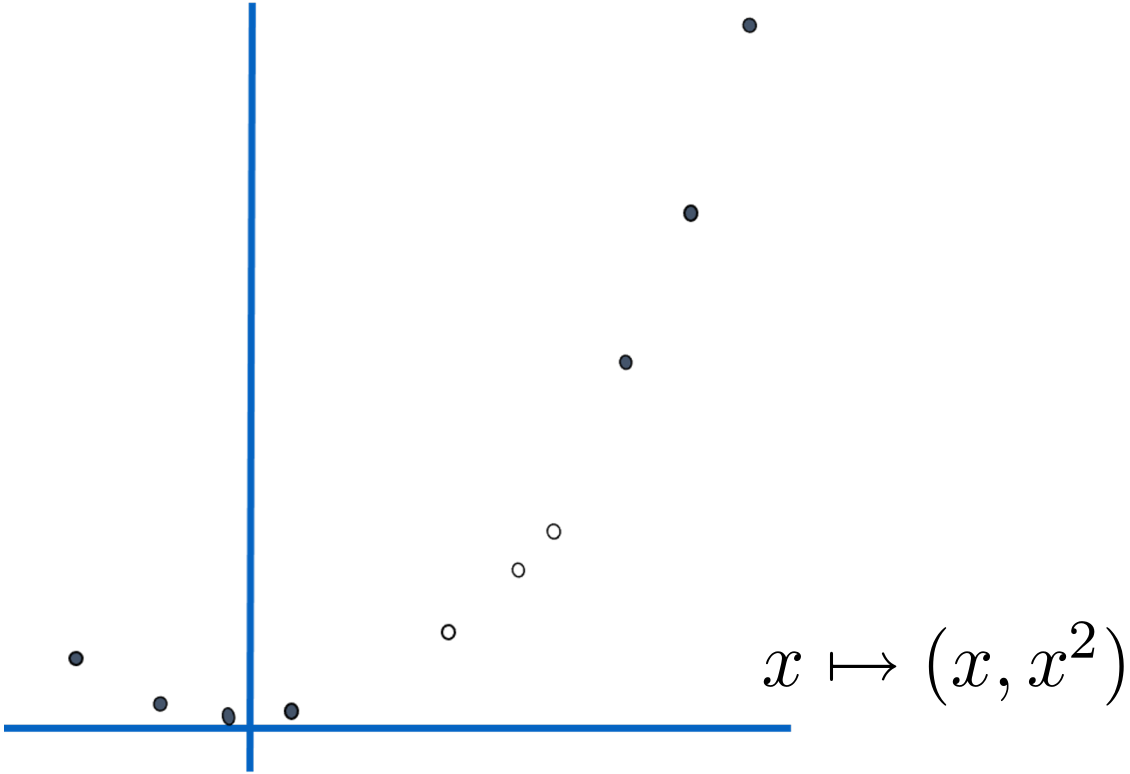
Not a big surprise



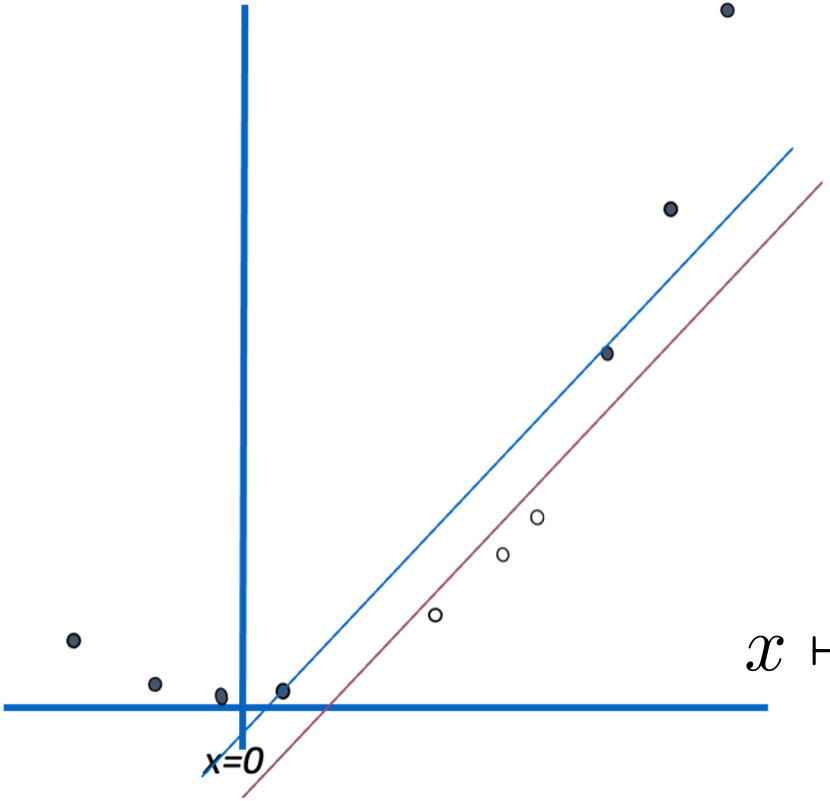
Harder 1-dimensional Dataset



Harder 1-dimensional Dataset



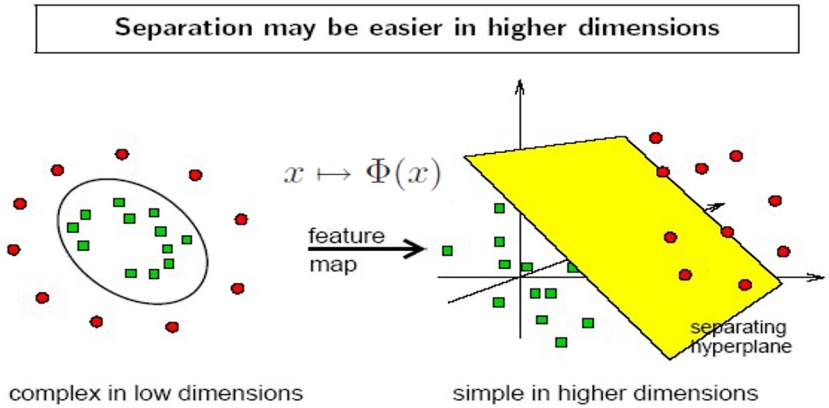
Harder 1-dimensional Dataset



$$x \mapsto (x, x^2)$$

SVM: Nonlinear Case

- Limitation of linear SVM
  - Linear SVM classifiers sometimes are restricted for some complex classification tasks where data are not linearly separable in input space
- Basic Idea of Nonlinear SVM
  - Map data into a richer feature space including nonlinear features, then construct a linear hyperplane in that space (using the same way)



SVM: Nonlinear Case

- First, define a feature mapping

$$x \mapsto \Phi(x)$$

- Then learns a hyperplane in the feature space

$$f(x) = w \cdot \Phi(x) + b$$

- Almost the same Primal form of SVM

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, N \end{aligned}$$

SVM: Nonlinear Case

- The dual problem

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \le \alpha_i \le C, i = 1, \ldots, l \\ & \mathbf{y}^T\boldsymbol{\alpha} = 0, \end{aligned}$$

where $Q_{ij} = y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ and $\mathbf{e} = [1, \ldots, 1]^T$

- The optimal solution

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)$$

$$f(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}) + b = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)\Phi(\mathbf{x}) + b$$

How to choose the feature mapping?

$$x \mapsto \Phi(x)$$

- Polynomial mapping
- Example:

$$\mathbf{x} \in R^3, \phi(\mathbf{x}) \in R^{10}$$

$$\phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3)$$

- Problem of using explicit feature mapping:
  - The dimensionality of $\Phi(\mathbf{x})$ can be very large, making $\boldsymbol{w}$ hard to represent explicitly in memory, and hard for the QP to solve

Kernel Tricks

- **Idea:** Replacing dot product with a kernel function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$$

- Not all functions are kernel functions
- A function could be a kernel if it is
  - Symmetric: $\kappa(x, y) = \kappa(y, x)$
  - Positive semi-definite (PSD): $\forall x_1, \ldots, x_n \in \mathcal{X}$ the "Gram matrix" K defined by $K_{ij} = \kappa(x_i, x_j)$ is PSD
    (the PSD means $\forall a \in \mathbb{R}^n, a^\top K a \geq 0$)
- **Benefits**
  - **Efficiency**: Computing kernel is often more efficient than compute Φ and the dot product
  - **Flexibility**: can choose various kernel functions as long as the existence of Φ is guaranteed (Mercer' condition)

Kernel Functions

- Linear Kernel

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^\top \mathbf{x}_j$$

- Polynomial Kernel (degree d)

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j / a + b)^d$$

- Gaussian / RBF Kernel

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right)$$

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(- \gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2 \right)$$

Kernel Functions

- Example: Polynomial Kernels

$$x, y \in \mathbb{R}^2, d = 2, a = 1$$

$$\kappa(x, y) = (x_1 y_1 + x_2 y_2 + b)^2$$

$$= \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \\ \sqrt{2b}x_1 \\ \sqrt{2b}x_2 \\ b \end{bmatrix} \cdot \begin{bmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2}y_1 y_2 \\ \sqrt{2b}y_1 \\ \sqrt{2b}y_2 \\ b \end{bmatrix}$$

Gaussian/RBF Kernel

- The kernel can be inner product in the infinite dimensional space. Assume x∈R. $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$

$$e^{-\gamma \|x_i - x_j\|^2} = e^{-\gamma (x_i - x_j)^2} = e^{-\gamma x_i^2 + 2\gamma x_i x_j - \gamma x_j^2}$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2} \left(1 + \frac{2\gamma x_i x_j}{1!} + \frac{(2\gamma x_i x_j)^2}{2!} + \frac{(2\gamma x_i x_j)^3}{3!} + \cdots\right)$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2} \left(1 \cdot 1 + \sqrt{\frac{2\gamma}{1!}} x_i \cdot \sqrt{\frac{2\gamma}{1!}} x_j + \sqrt{\frac{(2\gamma)^2}{2!}} x_i^2 \cdot \sqrt{\frac{(2\gamma)^2}{2!}} x_j^2 \right.$$

$$\left. + \sqrt{\frac{(2\gamma)^3}{3!}} x_i^3 \cdot \sqrt{\frac{(2\gamma)^3}{3!}} x_j^3 + \cdots\right) = \phi(x_i)^T \phi(x_j),$$

$$\phi(x) = e^{-\gamma x^2} \left[1, \sqrt{\frac{2\gamma}{1!}} x, \sqrt{\frac{(2\gamma)^2}{2!}} x^2, \sqrt{\frac{(2\gamma)^3}{3!}} x^3, \cdots\right]^T.$$

Nonlinear SVM with Kernel (1)

- Introducing nonlinearity into the model
- Computationally efficient
- The dual form

$$\max_{\alpha_i \in [0,C]} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) : \sum_{i=1}^{N} \alpha_i y_i = 0 \right\}$$

- The decision function

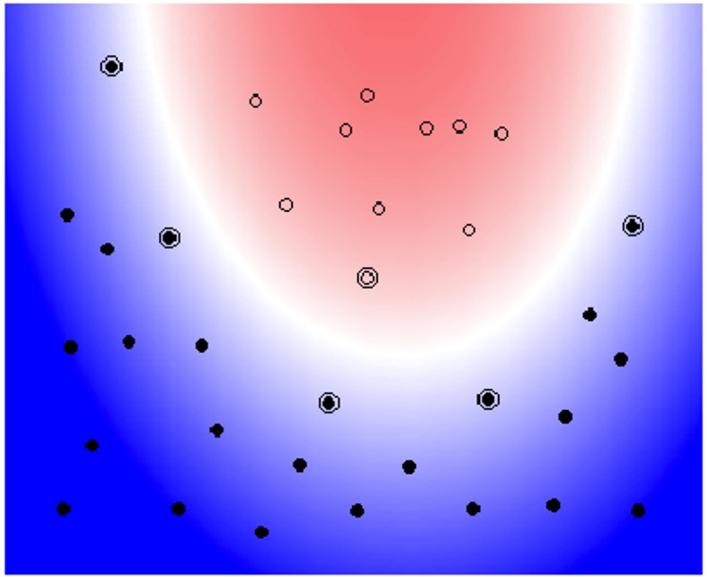
$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \kappa(\mathbf{x}_i, \mathbf{x}) + b$$

Nonlinear SVM with Kernel (2)

Example: SVM with Polynomial of Degree 2

Kernel: $K(\vec{x}_i, \vec{x}_j) = [\vec{x}_i \cdot \vec{x}_j + 1]^2$

plot by Bell SVM applet

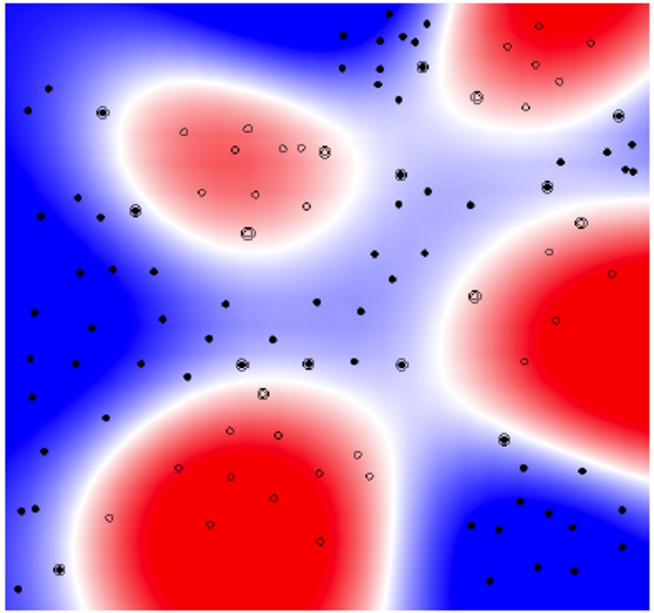


Nonlinear SVM with Kernel (3)

Example: SVM with RBF-Kernel

Kernel: $K(\vec{x}_i, \vec{x}_j) = \exp(-|\vec{x}_i - \vec{x}_j|^2 / \sigma^2)$

plot by Bell SVM applet



Nonlinear SVM with Kernel (4)

- The inner product in the feature space (similarity score) is performed implicitly
- Any linear classification method can be easily extended to nonlinear feature space (e.g., kernelized logistic regression)
- Non-vectorial data can be utilized (as long as kernel matrix is PSD)
- **Questions:**
  - Which kernel to use? How to set the parameters?
  - One kernel for each feature type or for all?

Curse of Kernalization

- **Challenge**
  - Training kernel classifiers is often much more computationally expensive
  - For kernel SVM, if one solves it by typical QP solvers, it will need O(N^3). Even for faster solvers (SMO) or others, it typically needs at least O(N^2) time cost.
  - But linear classifiers can be trained in much fasters, typically in linear time O(N)
- **Question**
  - How to train kernel machines for large-scale datasets?

Kernel Approximation

- Our goal
  - To construct a new representation $\mathbf{z}(\mathbf{x}) \in \mathbb{R}^D$ so that $\kappa(\mathbf{x}_i, \mathbf{x}_j) \approx \mathbf{z}(\mathbf{x}_i)^\top \mathbf{z}(\mathbf{x}_j)$
- Linear model
  - The hypothesis can be rewritten:

$$f(\mathbf{x}) = \sum_{i=1}^{B} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}) \approx \sum_{i=1}^{B} \alpha_i \mathbf{z}(\mathbf{x}_i)^\top \mathbf{z}(\mathbf{x}) = \mathbf{w}^\top \mathbf{z}(\mathbf{x})$$

  - where $\mathbf{w}^\top = \sum_{i=1}^{B} \alpha_i \mathbf{z}(\mathbf{x}_i)$
  - Then apply the new representation $\mathbf{z}$
- Two methods
  - Kernel Functional Approximation: Fourier method
  - Kernel Matrix Approximation: Nyström method

Multi-class Classification

- Consider **k** classes
- **One-against-the rest**: Train k binary SVMs:
  - 1st class vs. (2 – k)th class
  - 2nd class vs. (1, 3 – k)th class
  - ...
- k decision functions

$$(\mathbf{w}^1)^T \phi(\mathbf{x}) + b_1$$
$$\vdots$$
$$(\mathbf{w}^k)^T \phi(\mathbf{x}) + b_k$$

Multi-class Classification

- Prediction

$$\arg\max_j \ (\mathbf{w}^j)^T \phi(\mathbf{x}) + b_j$$

- Reason: If it's the 1st class, then we should have

$$\begin{aligned}
(\mathbf{w}^1)^T \phi(\mathbf{x}) + b_1 &\geq +1 \\
(\mathbf{w}^2)^T \phi(\mathbf{x}) + b_2 &\leq -1 \\
&\vdots \\
(\mathbf{w}^k)^T \phi(\mathbf{x}) + b_k &\leq -1
\end{aligned}$$

Multi-class Classification

- One-against-one: train k(k − 1)/2 binary SVMs

  (1,2), (1,3), . . . , (1,k), (2,3), (2,4), . . . , (k−1,k)

- Example: if 4 classes 6 binary SVMs

$y_i = 1$	$y_i = -1$	Decision functions
class 1	class 2	$f^{12}(\mathbf{x}) = (\mathbf{w}^{12})^T\mathbf{x} + b^{12}$
class 1	class 3	$f^{13}(\mathbf{x}) = (\mathbf{w}^{13})^T\mathbf{x} + b^{13}$
class 1	class 4	$f^{14}(\mathbf{x}) = (\mathbf{w}^{14})^T\mathbf{x} + b^{14}$
class 2	class 3	$f^{23}(\mathbf{x}) = (\mathbf{w}^{23})^T\mathbf{x} + b^{23}$
class 2	class 4	$f^{24}(\mathbf{x}) = (\mathbf{w}^{24})^T\mathbf{x} + b^{24}$
class 3	class 4	$f^{34}(\mathbf{x}) = (\mathbf{w}^{34})^T\mathbf{x} + b^{34}$

Multi-class Classification

- For a testing data, predict all binary SVMs

Classes		winner
1	2	1
1	3	1
1	4	1
2	3	2
2	4	4
3	4	3

- Select the one with the largest vote

class	1	2	3	4
# votes	3	1	1	1

- May use decision values as well

Multi-class Classification

- There are many other methods
- A comparison in [Hsu and Lin, 2002]
- Accuracy similar for many problems
- But 1-against-1 fastest for training
- Assume the SVM optimization with size n is $\mathcal{O}(n^d)$
- 1 vs. all
  - k problems, each has N data $k\mathcal{O}(N^d)$
- 1 vs. 1
  - k(k − 1)/2 problems, each 2N/k data on average $\frac{k(k-1))}{2}\mathcal{O}((\frac{2N}{k})^d)$

Chih-Wei Hsu and Chih-Jen Lin, "A comparison of methods for multiclass support vector machines," in *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415-425, March 2002

Summary

- Problem and Intuition
- Formulation of Linear SVM
  - Hard Margin SVM
  - Soft Margin SVM
  - Primal/dual Problems
- Nonlinear SVM with Kernel
  - Kernel Tricks
  - SVM with Kernel
- Multi-class classification

Bài tập

Cho 6 điểm data dùng cho bài toán phân lớp {+1,-1} như bảng. Vẽ các điểm, tìm các support vectors, độ rộng đường biên lớn nhất theo phương pháp SVM tuyến tính

x1	x2	y
1	9	-1
5	5	-1
1	1	-1
8	5	+1
13	1	+1
13	9	+1