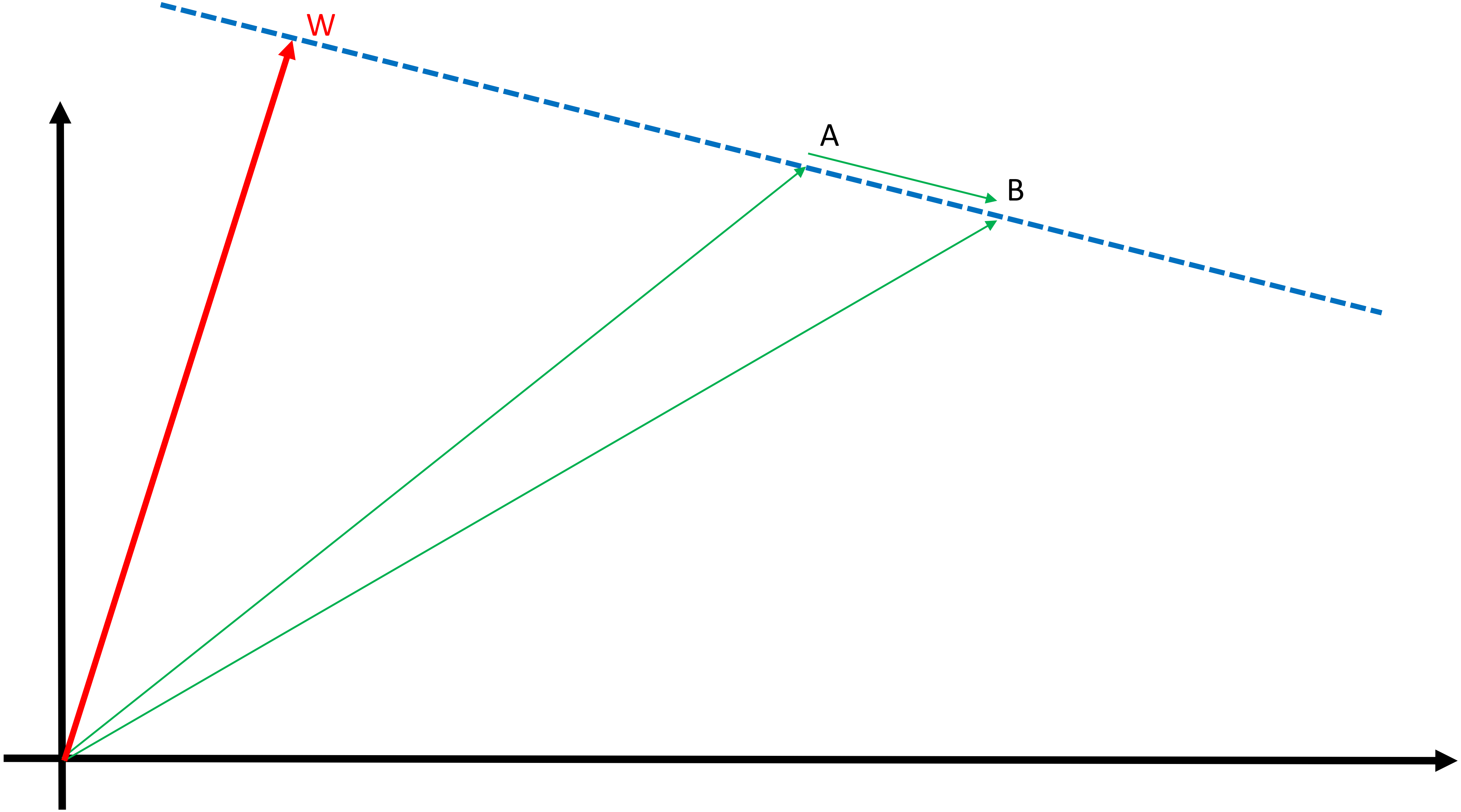
Linear Support Vector Machine

Dr. Kalidas Y., IIT Tirupati

SVM Formulation

- xi is a k dimensional input point
- yi is a number {-1,1}
- Model, $f(x) = sign(x \cdot w)$
- Data set {(xi,yi)} are given for i=[1..N]
- *Loss function??? … Lets design it!*

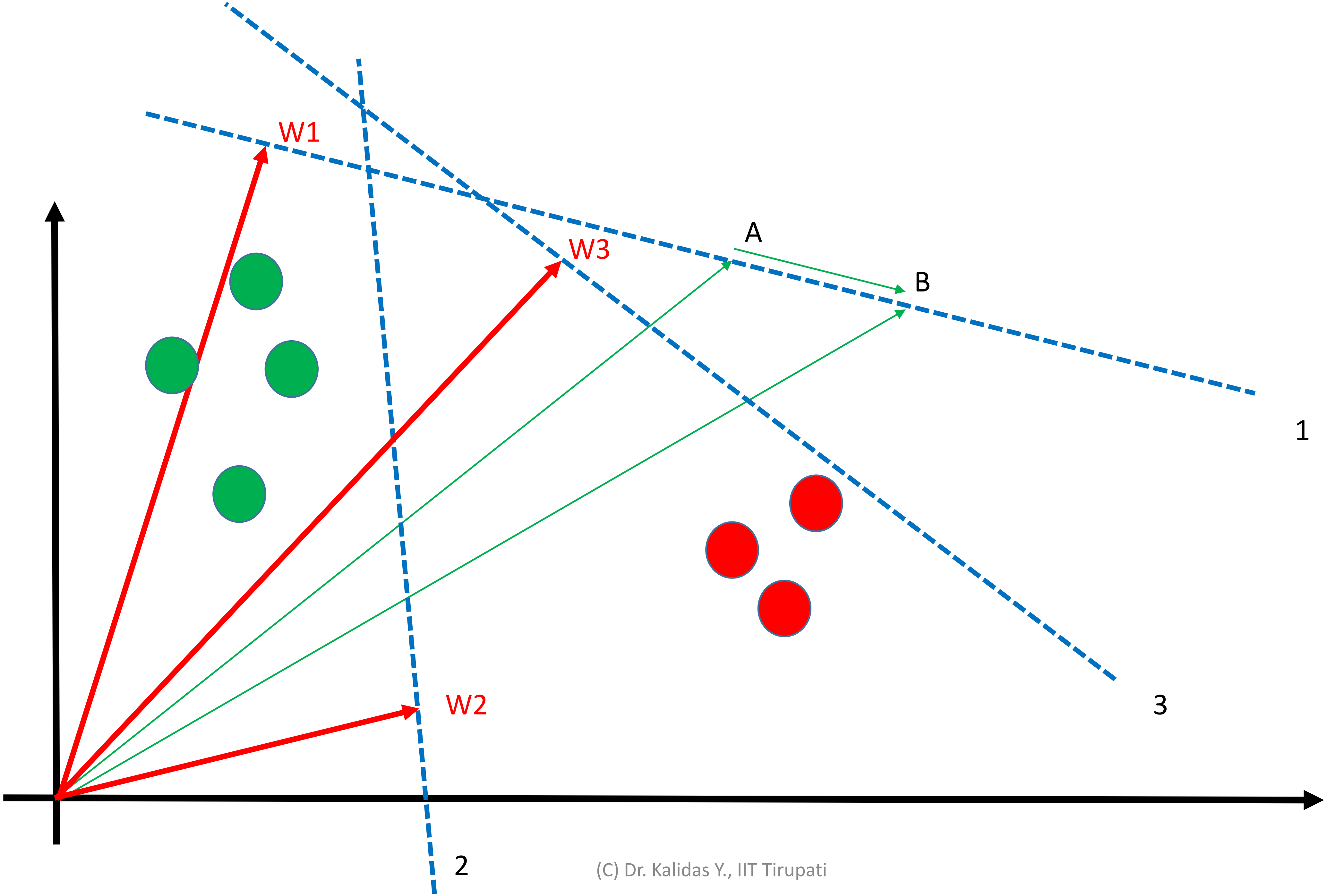

W
A
B

A 2-dimensional weight vector represents a line!

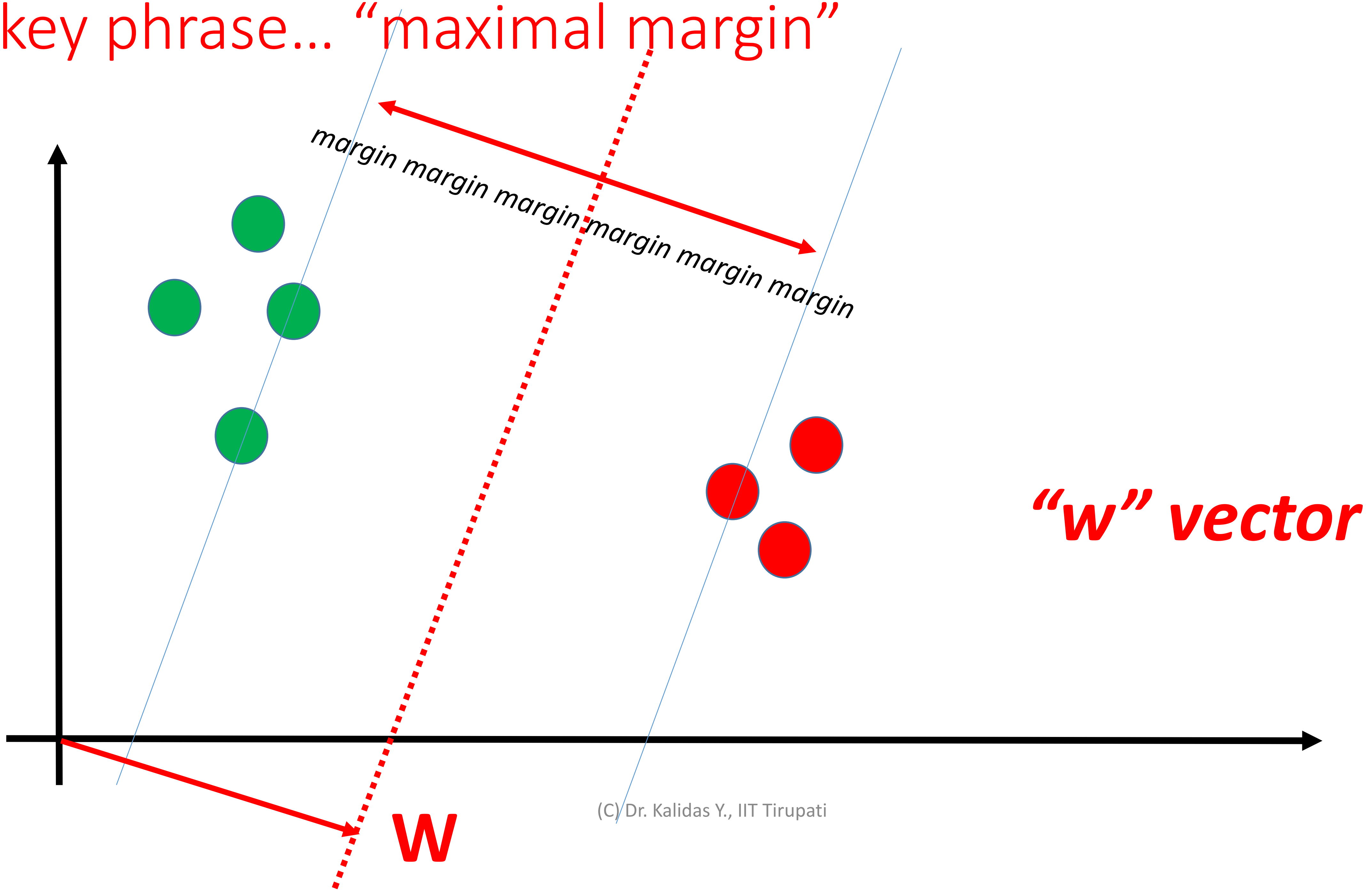
- Consider
  - OW
  - OA
  - OB
  - AB
- Consider
  - $AB = OB - OA$
  - $OW \perp AB$ for all A,B on the line
- Given $OW = (w_1, w_2)$ we can always construct all points along the perpendicular to the dotted line
- A 2-dimensional weight vector represents a line!

69) key phrase… “hyper plane”

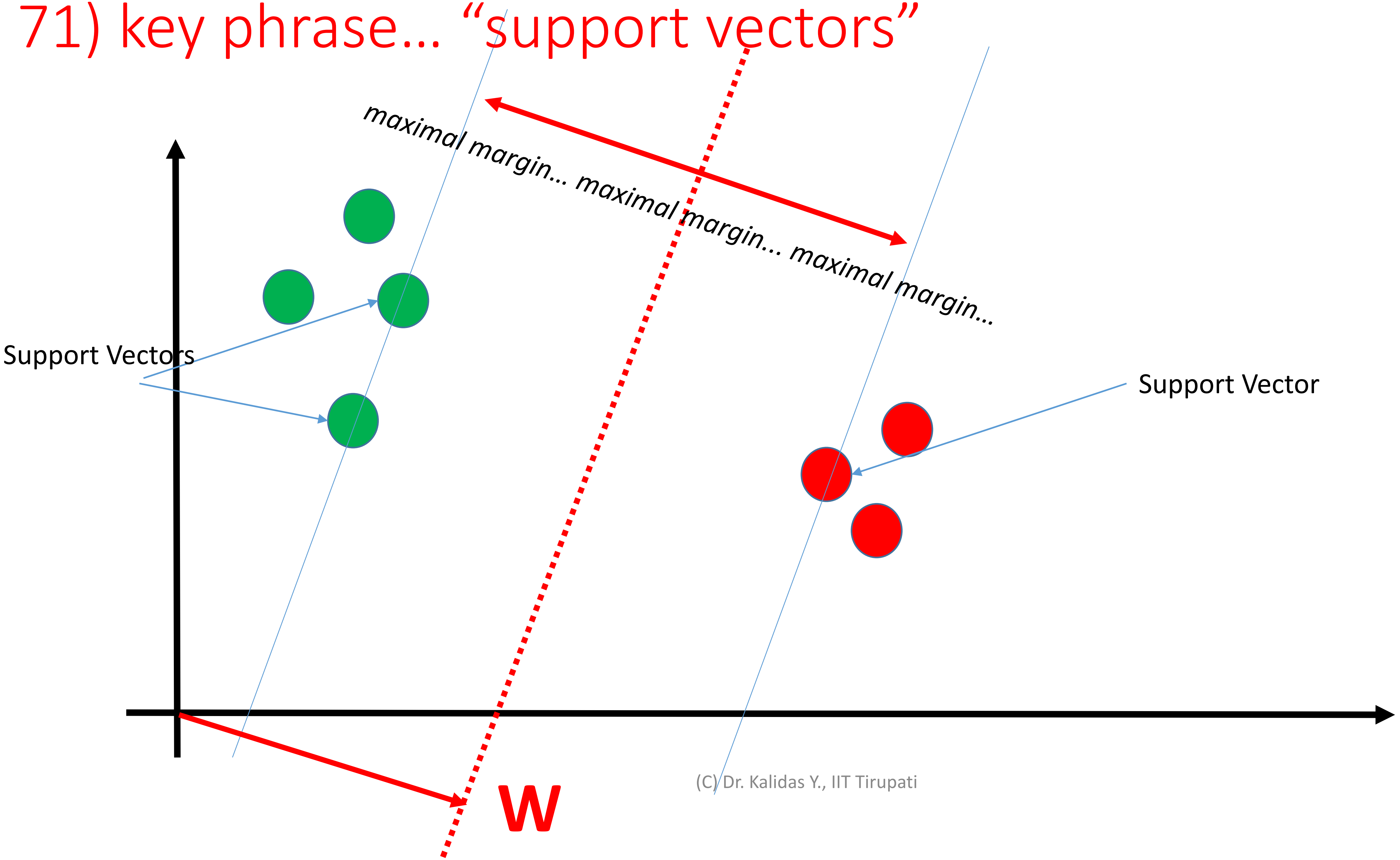
- A weight vector and its associated plane in high dimensions (typically assume 3 or more)
- You can think of it as a plane subtended by the weight vector from origin
- Imagine… weight vector is like a pillar holding the entire plane up!

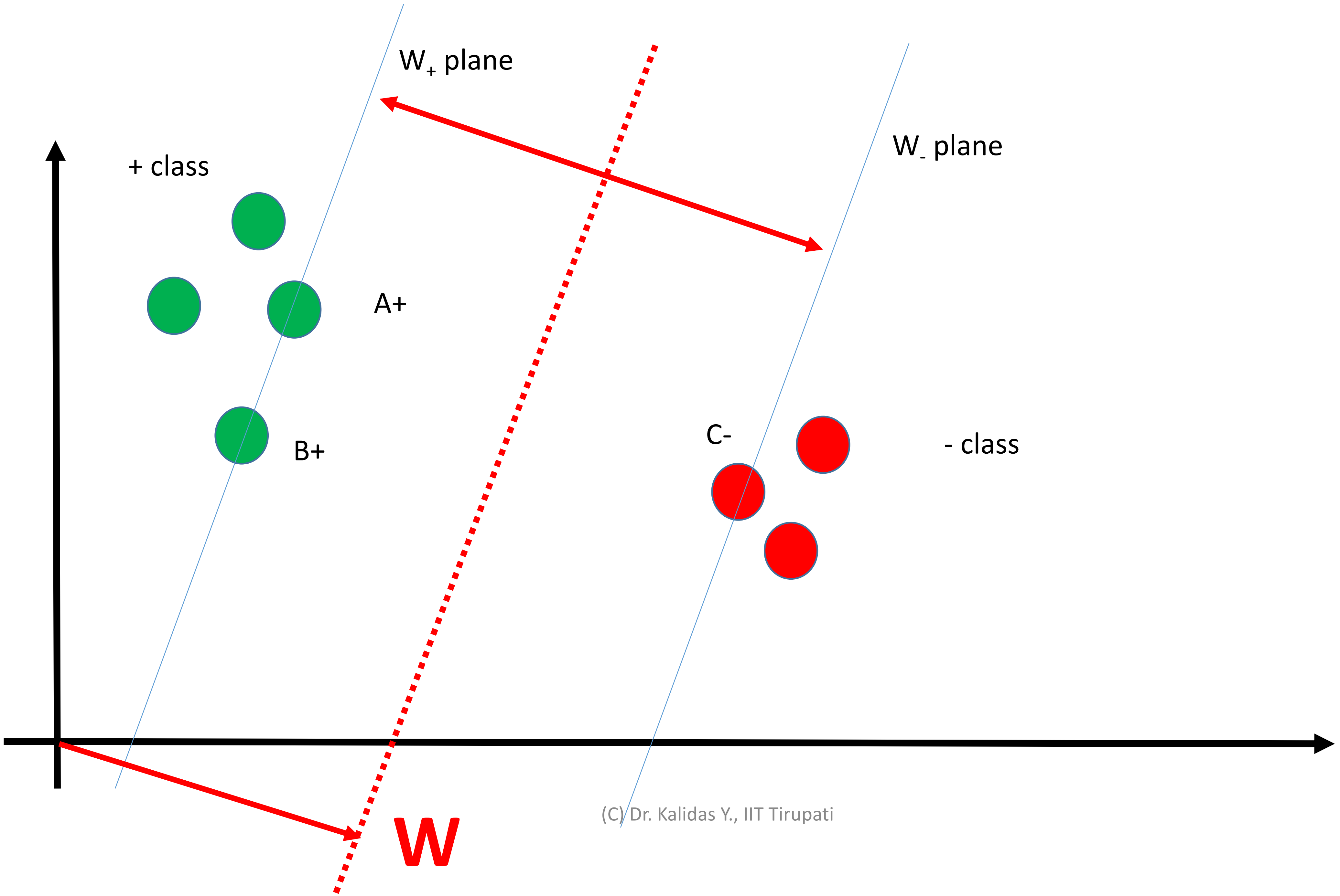

W1
W3
A
B
1
W2
3
2

70) key phrase… “maximal margin”

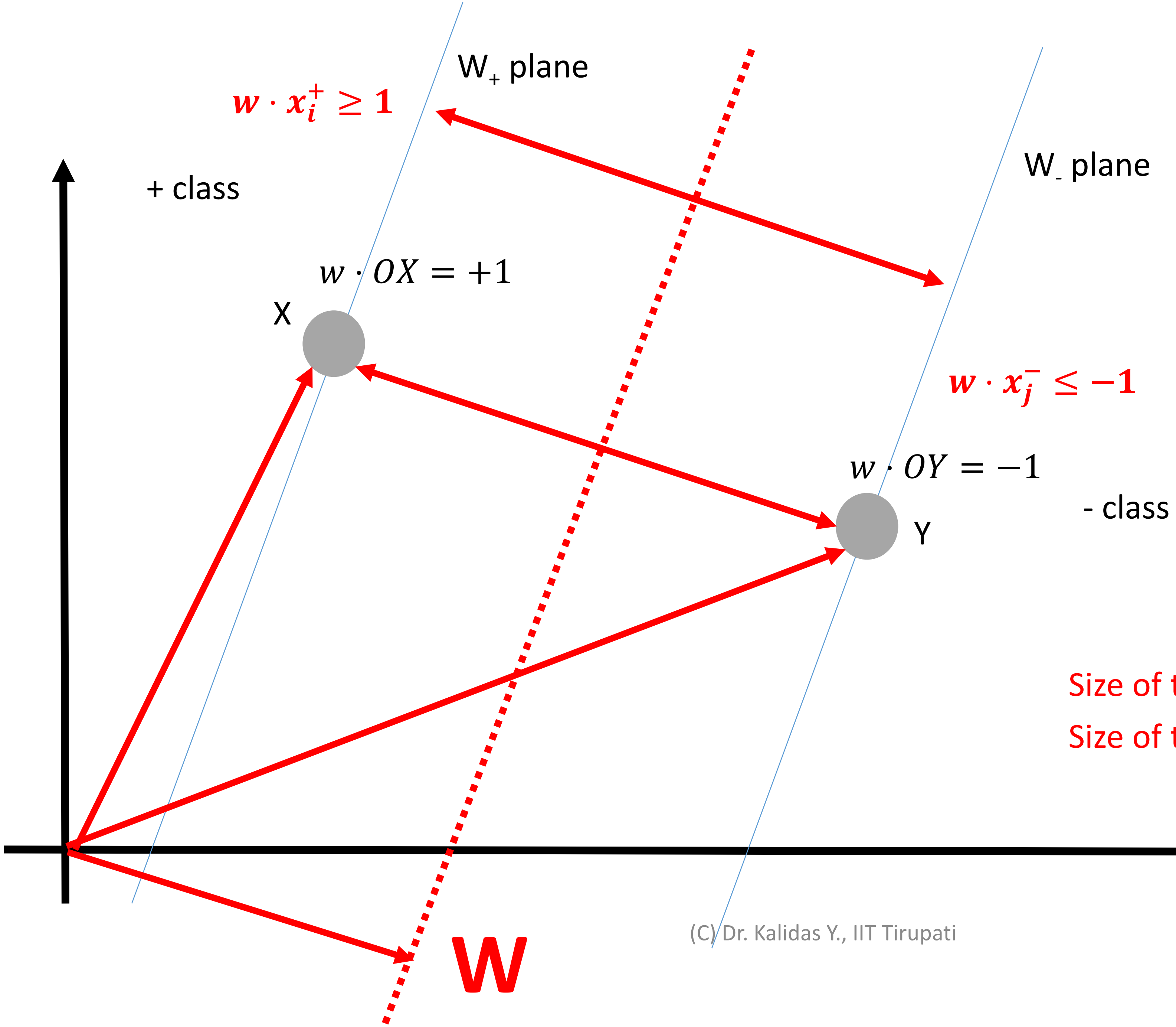


71) key phrase… “support vectors”





W+ plane
W- plane
+ class
A+
B+
C-
- class
W



1) $OX - OY = \lambda \, OW$
2) $w \cdot OX = +1$
3) $w \cdot OY = -1$
4) $OX - OY = \lambda w$
5) $w \cdot (OX - OY) = \lambda w \cdot w$
6) $w \cdot OX - w \cdot OY = \lambda ||w||_2^2$
7) $2 = \lambda ||w||_2^2$

Size of the margin $\propto \lambda$

Size of the margin $\propto \frac{1}{||w||_2^2}$

Maximize margin = Minimize $\frac{||w||_2^2}{2}$

Minimizing Error… $\xi_i = 1 - y_i \times w \cdot x_i$

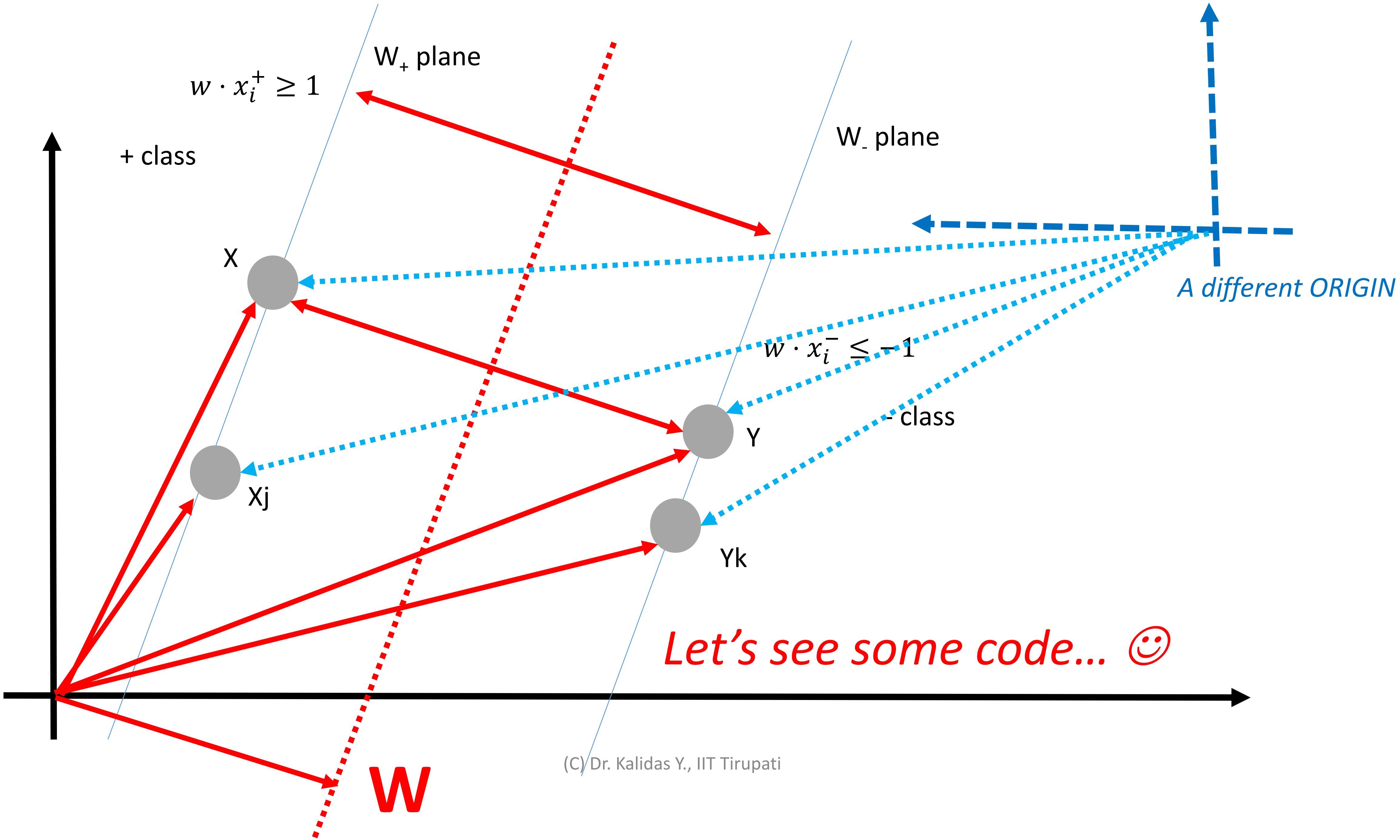
- For all, $\forall\ y_i = -1: w \cdot x_i \leq -1$
- For all, $\forall\ y_i = +1: w \cdot x_i \geq +1$
- For all, $\forall\ y_i:\ y_i \times w \cdot x_i \geq 1$
- Error per i: $\xi_i = 1 - y_i \times w \cdot x_i$
- Sum of all error values, $\sum_{i=1}^{i=N} \xi_i$

Focus only on Error... $\max(0, 1 - y_i \times w \cdot x_i)$

- For all, $\forall\, y_i = -1: w \cdot x_i \leq -1$
- For all, $\forall\, y_i = +1: w \cdot x_i \geq +1$
- For all, $\forall\, y_i:\ y_i \times w \cdot x_i \geq 1$
- Error per i: $\xi_i = 1 - y_i \times w \cdot x_i$
- Sum of all error values, $\sum_{i=1}^{i=N} \xi_i$
- If $y_i \times w \cdot x_i = 1000.0$ say, Then $\xi_i < 0$
  - That means, for those data points where prediction is correct
  - Correct predictions may outweigh incorrect predictions
  - We do not want that to happen!
- Be humble, and reduce incorrect predictions, even those for some points, predictions are right
  - $\max(0, 1 - y_i \times w \cdot x_i)$

72) key phrase… "Linear SVM"

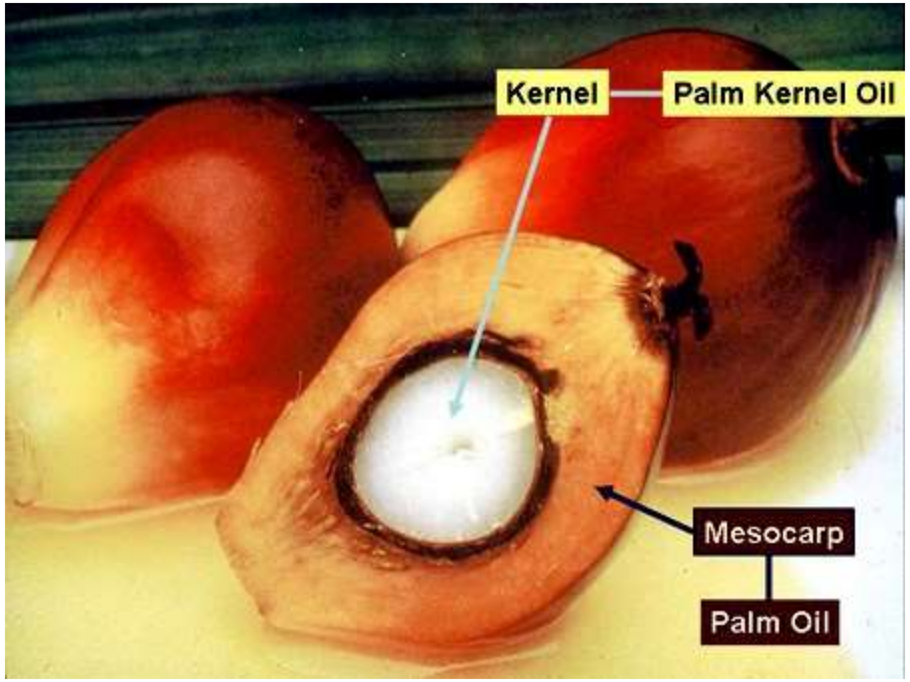
$$L(w) = \frac{1}{2}||w||_2^2 + \sum_{i=1}^{i=N} \max(0, 1 - y_i \times (w \cdot x_i))$$



W+ plane
$w \cdot x_i^+ \geq 1$
+ class
W- plane
X
A different ORIGIN
$w \cdot x_i^- \leq -1$
- class
Y
Xj
Yk
Let's see some code… ☺
W

73) key phrase… "Kernel SVM"

- ***dot product*** based formulation
- Each dot product can be replaced by a wide variety of similarity functions or kernel functions
  - polynomial
  - radial basis
  - etc.



Perceptron learning law (*error function*)

- xi is a k-dimensional input point
- yi is a univariate scalar
- Model, $f(x) = w^T x$
- Data set, $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$
- Loss function, $L(w) = \max(0,1 - y_i * f(x_i))$
- Weight update, $\mathrm{w}_{new} = w_{old} - \nabla L_{\mathrm{w}}|_{w=w_{old}}$
- *Gradient ???*

Gradient of max()???

- Consider $\mathrm{y} = f(x) = \max(0, x)$
- *What is,* $\frac{dy}{dx} = 0$ *??? OR,* $\frac{dy}{dx} = 1$ *???*

Gradient of max()???

- Consider $\mathrm{y} = f(x) = \max(0, x)$
- What is, $\frac{\mathrm{dy}}{\mathrm{dx}} = 0$??? OR, $\frac{\mathrm{dy}}{\mathrm{dx}} = 1$???
- *Rule: Conditional derivative*
- IF $y == x$
  - $\frac{\mathrm{dy}}{\mathrm{dx}} = 1$
- ELSE
  - $\frac{\mathrm{dy}}{\mathrm{dx}} = 0$

- Consider $L(w) = \max(0{,}1 - y_i \times w^T x_i)$
- What is, $\frac{\partial L}{\partial w} = 0$ ***OR*** $\frac{\partial L}{\partial w} = -y_i\, x_i$
- *Rule: Conditional derivative*
- IF $y_i == w^T x_i$
  - $\frac{\partial L}{\partial w} = 0$
- ELSE
  - $\frac{\partial L}{\partial w} = -y_i x_i$

Gradient update of 'w' vector of max() based loss function

- $\mathrm{w}_{new} = w_{old} - \nabla L_w|_{w=w_{old}}$
- IF matched
  - $\mathrm{w}_{new} = w_{old} - 0$ ***OR***
- ELSE //not match
  - $\mathrm{w}_{new} = w_{old} - (-y_i\, x_i) = w_{old} + y_i\, x_i$

$$\mathrm{w}_{new} = w_{old} + \eta\,(y_i - \hat{y})x_i$$

This is called Perceptron Learning Law

Re-formulating "w" vector, $\mathrm{w} = \sum_{i=1}^{i=N} \alpha_i y_i x_i$

- $\mathrm{w}_{new} = w_{old} - \nabla L_w|_{w=w_{old}}$
- IF matched
  - $\mathrm{w}_{new} = w_{old} - 0$ ***OR***
- ELSE //not match
  - $\mathrm{w}_{new} = w_{old} - (-y_i\, x_i) = w_{old} + y_i\, x_i$
- By the end of iterations, "w" would have converged
  - Let α_i denote number of times mismatch occurred for i^{th} point
- Contribution from the i^{th} point to w is, $\alpha_i y_i x_i$
- Total w vector is, $\mathrm{w} = \sum_{i=1}^{i=N} \alpha_i y_i x_i$

… "Kernel SVM"

- The w vector is a weighted combination of input data points
  - $w = \alpha_1 y_1\, x_1 + \cdots + \alpha_N y_N x_N$
  - Each of the x_i is a k-dimensional input data point
  - There are N points
  - Each of the α_i is a scalar
- The loss function now takes a different form!
  - The term, $w \cdot w = \sum_{j=1}^{j=k} \sum_{p=1}^{p=k} \alpha_j \alpha_p y_j\, y_p\, (x_j \cdot x_p)$
  - The term, $w \cdot x_i = \sum_{j=1}^{j=k} w_j \times x_{i,j}$
- Prediction, $f(x) = sign(w \cdot x) = \sum_{i=1}^{i=N} \alpha_i y_i\, (x_i \cdot x)$

- *Why this dot product, $a \cdot b$ is important!???*

74) key phrase… “kernel function”

- kernel = inside key element
- Without actually having to do feature expansion, the kernel function will mimic the effect
- $x_i \vdash (x_i^0, \ldots, x_i^k)$ remember? k-dimensional expansion
- $x_i \cdot x_j = \sum_{p=0}^{p=k} x_{i,p} \times x_{j,p}$
- $\equiv \left(1 + x_i \times x_j\right)^p$

75) key phrase… “Linear kernel”

- $k(x_i, x_j) = x_i \cdot x_j$

76) key phrase… “Polynomial kernel”

- $k(x_i, x_j) = (1 + x_i \cdot x_j)^d$

- Equivalent to ***IMPLICIT FEATURE EXPANSION (polynomial)***
  - $x_i \vdash (x_i^0, \ldots, x_i^d)$
  - $x_i' \cdot x_j' = k(x_i, x_j)$

77) key phrase… "Radial Basis kernel"

- $k(x_i, x_j) = e^{-\frac{\left\|x_i - x_j\right\|^2}{\sigma^2}}$
- Equivalent to ***IMPLICIT FEATURE EXPANSION (exponential infinite..)***
  - $e^z = \frac{z^0}{0!} + \frac{z^1}{1!} + \cdots \frac{z^n}{n!} + \cdots \infty \text{ terms}$
  - $x_i \vdash (\frac{x_i^0}{0!}, \frac{x_i^1}{1!}, \frac{x_i^2}{2!} \ldots, \infty)$ //infinite number of terms!!!!
  - $x_i' \cdot x_j' = k(x_i, x_j)$

78) key phrase… “Kernel SVM”

- $L(w) = \frac{1}{2}||w||_2^2 + C \times \sum_{i=1}^{i=N} \max(0,1 - y_i \times (w \cdot x_i))$
- $L(w) = \sum_{i=1}^{i=N} \sum_{j=1}^{j=N} \alpha_i \alpha_j y_i y_j k(x_i, x_j) - \sum_{i=1}^{i=N} \alpha_i$
  Subject to constraints:
  - $0 \leq \alpha_i \leq C$
  - $\sum_{i=1}^{i=N} \alpha_i y_i = 0$
- Margin: Space between -1 and +1
- **Error Penalty C** = low → Larger margin and allows points in the margin
- **Error Penalty C** = high → Thinner margin, does not allow points in the margin
- RBF, σ = high → Loose margin about +1 or -1
- RBF, σ = low → Tight margin about +1 or -1