ECEN 377: Engineering Applications of AI

Dr. Mahmoud Nabil Mahmoud
mnmahmoud@ncat.edu

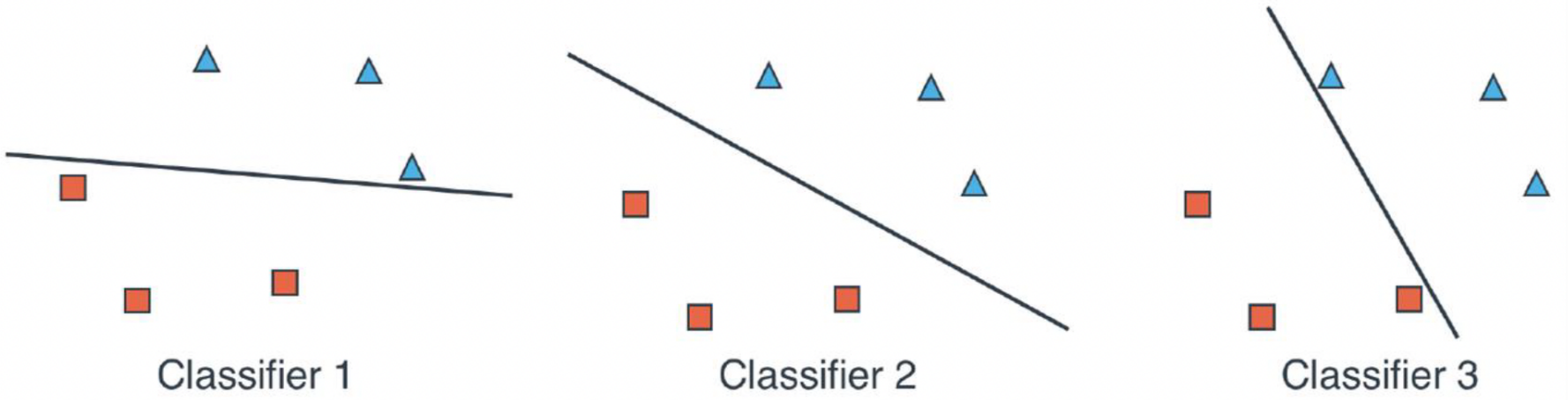
North Carolina A & T State University

October 3, 2024

Outline

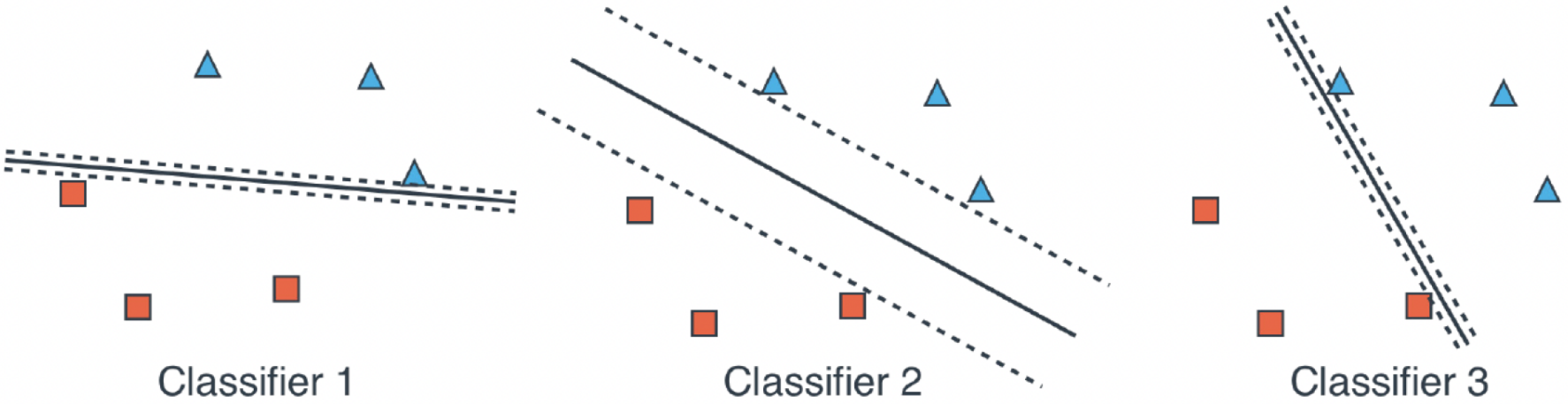
Introduction to SVM

Which of the following is the best classifier? and **why?**



Introduction to SVM

Which of the following is the best classifier? and **why?**



Classifier 2 is better because it has a **larger margin** between the classes. **Margin** is the distance between the decision boundary and the closest data points. (Support Vectors)

Outline

SVM Loss Function

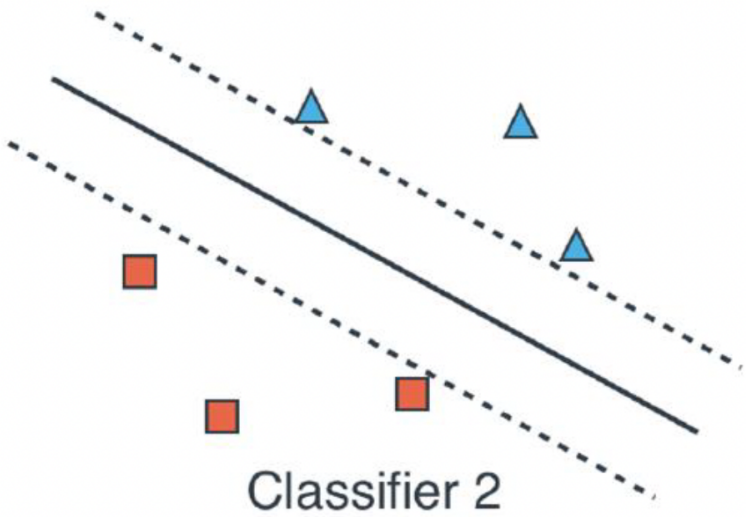
How we should penalize misclassified data points and maximize the margin?

Loss Function =

Classification Loss

+

Distance Loss



SVM Loss Function

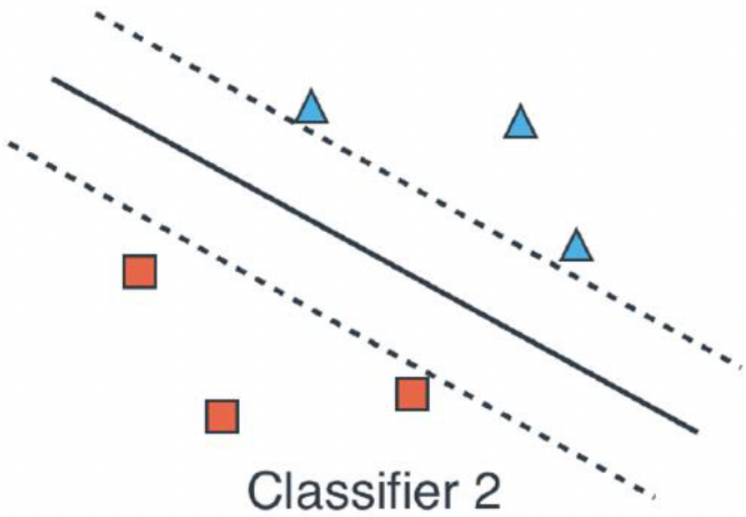
How we should penalize misclassified data points and maximize the margin?

Loss Function =

Classification Loss

+

Distance Loss



Assume we aim to train a linear SVM on 2D data points (x_1, x_2).

Outline

Classification Loss Function

In SVM, Instead of using only the decision boundary, we use two parallel **imaginary lines** shifted by bias.

Example:

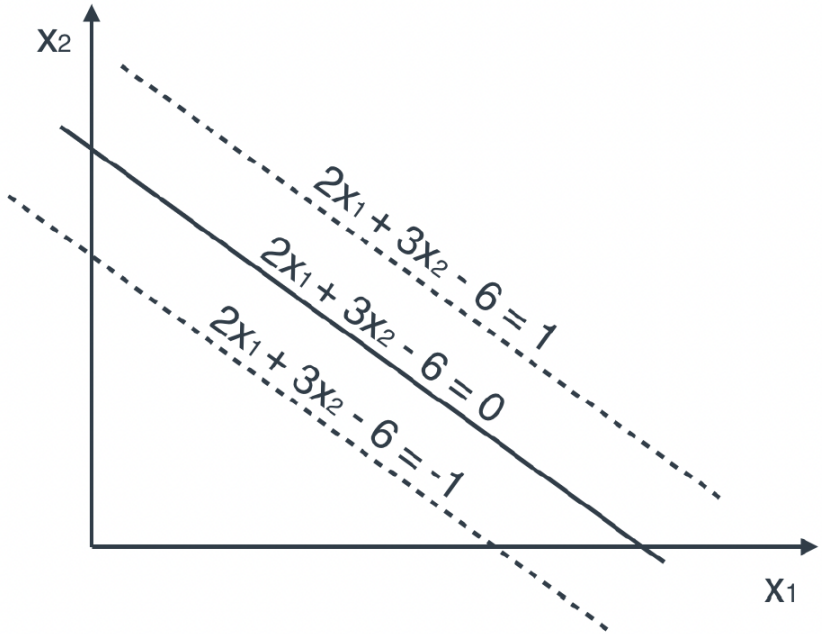
Decision Boundary: $2x_1 + 3x_2 - 6 = 0$

Imaginary Line (+ve):

$2x_1 + 3x_2 - 6 = 1$

Imaginary Line (-ve):

$2x_1 + 3x_2 - 6 = -1$



Classification Loss Function

In SVM, Instead of using only the decision boundary, we use two parallel **imaginary lines** shifted by bias.

Example:

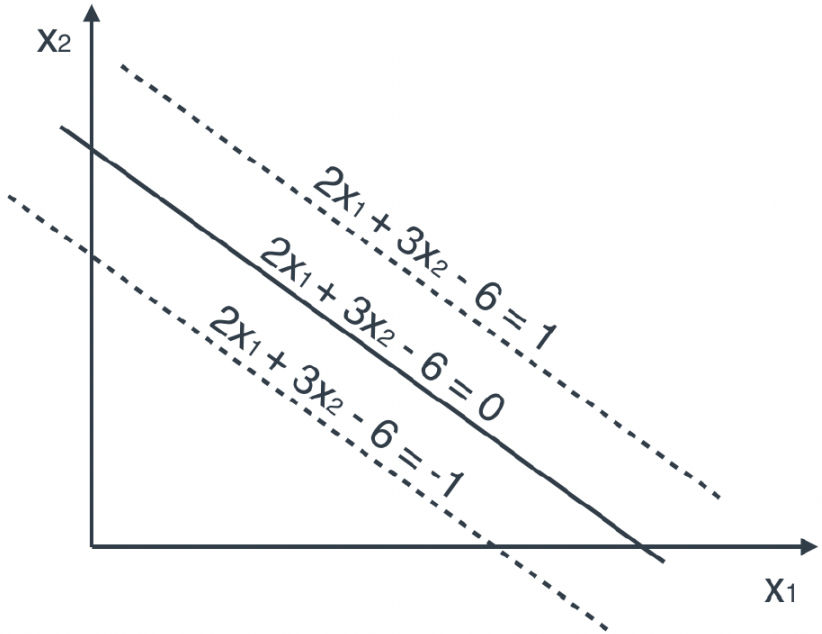
Decision Boundary: $2x_1 + 3x_2 - 6 = 0$

Imaginary Line (+ve):

$2x_1 + 3x_2 - 6 = 1$

Imaginary Line (-ve):

$2x_1 + 3x_2 - 6 = -1$



Note: Even correctly classified data points within the margin should be penalized! **How?**

Classification Loss

Requirement for Classification Loss:

- Correct classification above the +ve margin has a loss value of zero.

Classification Loss

Requirement for Classification Loss:

- Correct classification above the +ve margin has a loss value of zero.
- Correct classification below the -ve margin has a loss value of zero.

Classification Loss

Requirement for Classification Loss:

- Correct classification above the +ve margin has a loss value of zero.
- Correct classification below the -ve margin has a loss value of zero.
- Misclassified data points above the +ve margin have a loss value greater than zero.

Classification Loss

Requirement for Classification Loss:

- Correct classification above the +ve margin has a loss value of zero.
- Correct classification below the -ve margin has a loss value of zero.
- Misclassified data points above the +ve margin have a loss value greater than zero.
- Misclassified data points below the -ve margin have a loss value greater than zero.

Classification Loss

Requirement for Classification Loss:

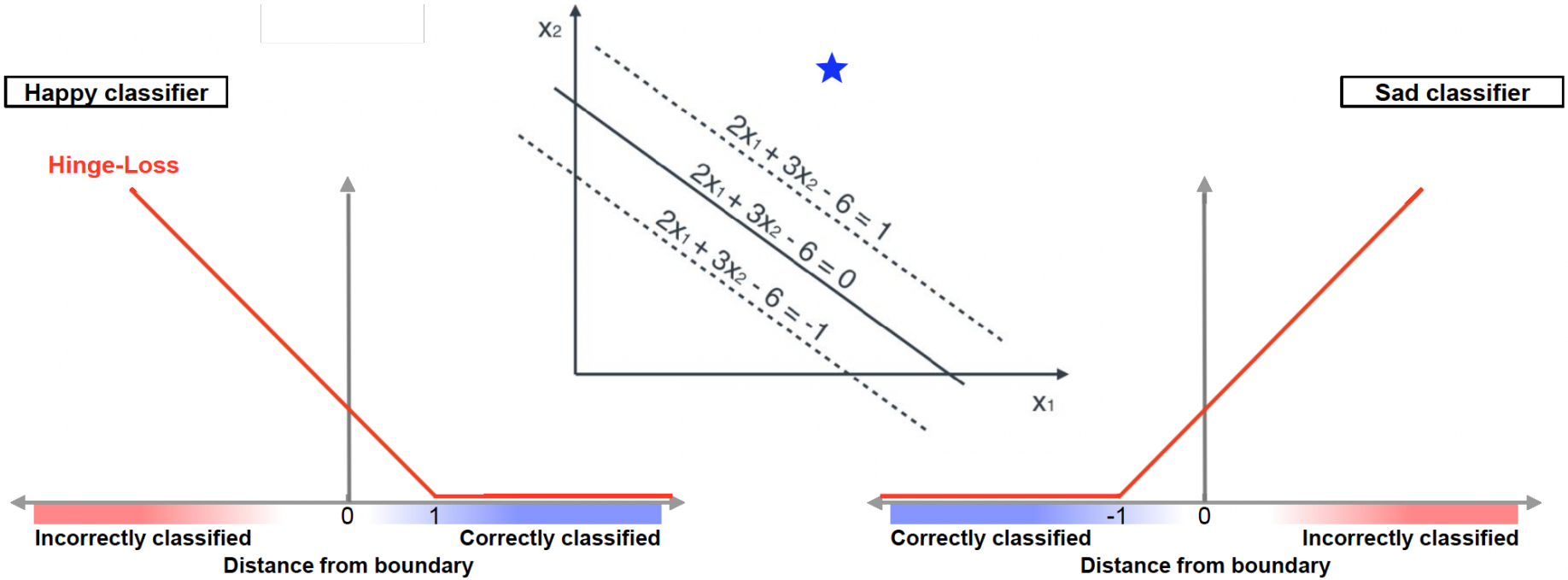
- Correct classification above the +ve margin has a loss value of zero.
- Correct classification below the -ve margin has a loss value of zero.
- Misclassified data points above the +ve margin have a loss value greater than zero.
- Misclassified data points below the -ve margin have a loss value greater than zero.
- Correctly classified data points within the margin have a loss value greater than zero.

Hinge Loss (Classification Loss)

The Hinge Loss function is defined as:

$$L_{\text{classification}}(y, \hat{y}) = \max(0, 1 - y \cdot y')$$

where y is the true label (either $+1$ or -1) and y' is the predicted label.

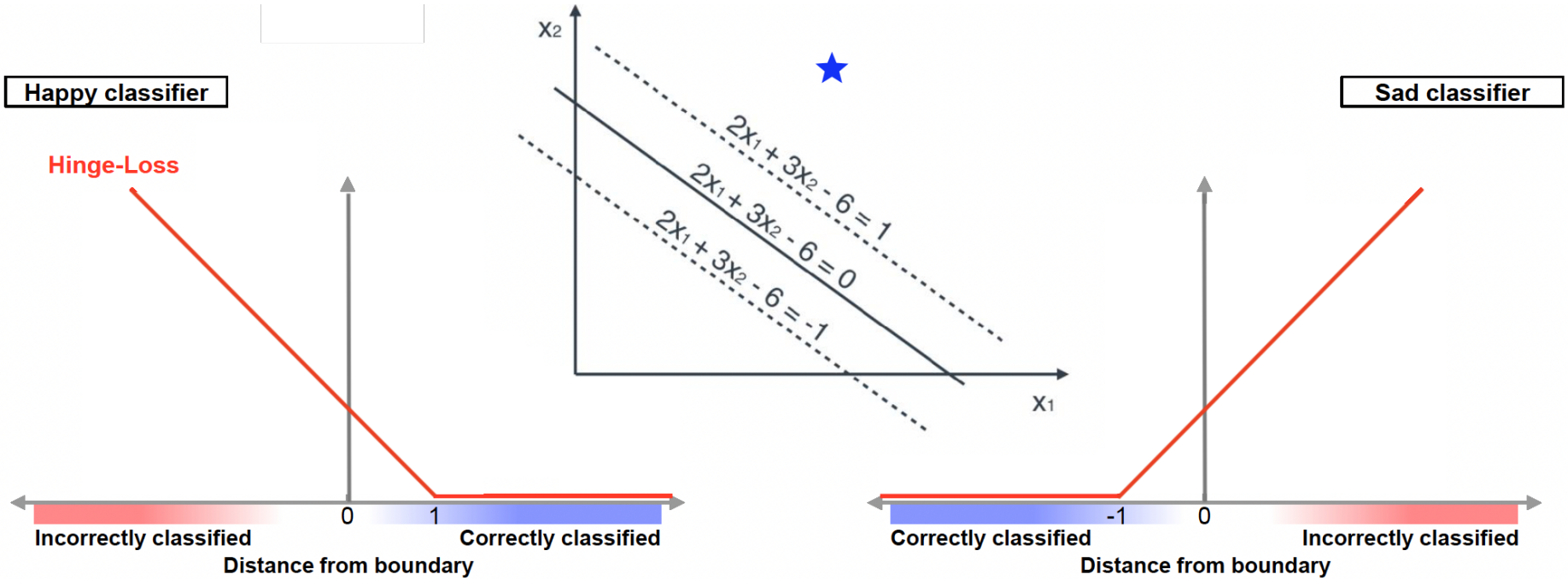


Hinge Loss (Classification Loss)

The Hinge Loss function is defined as:

$$L_{\text{classification}}(y, \hat{y}) = \max(0, 1 - y \cdot y')$$

where y is the true label (either $+1$ or -1) and y' is the predicted label.



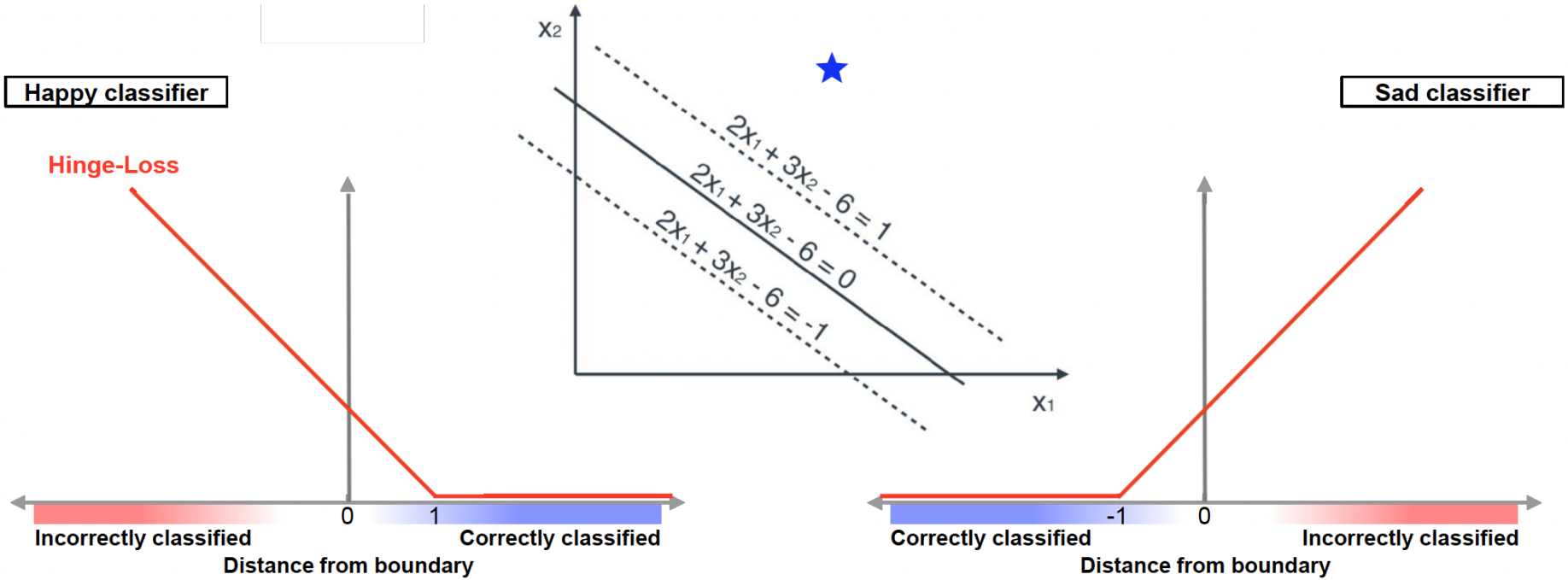
What is the loss for the blue data point if it is correctly classified? Which area is the loss in?

Hinge Loss (Classification Loss)

The Hinge Loss function is defined as:

$$L_{\text{classification}}(y, \hat{y}) = \max(0, 1 - y \cdot y')$$

where y is the true label (either $+1$ or -1) and y' is the predicted label.

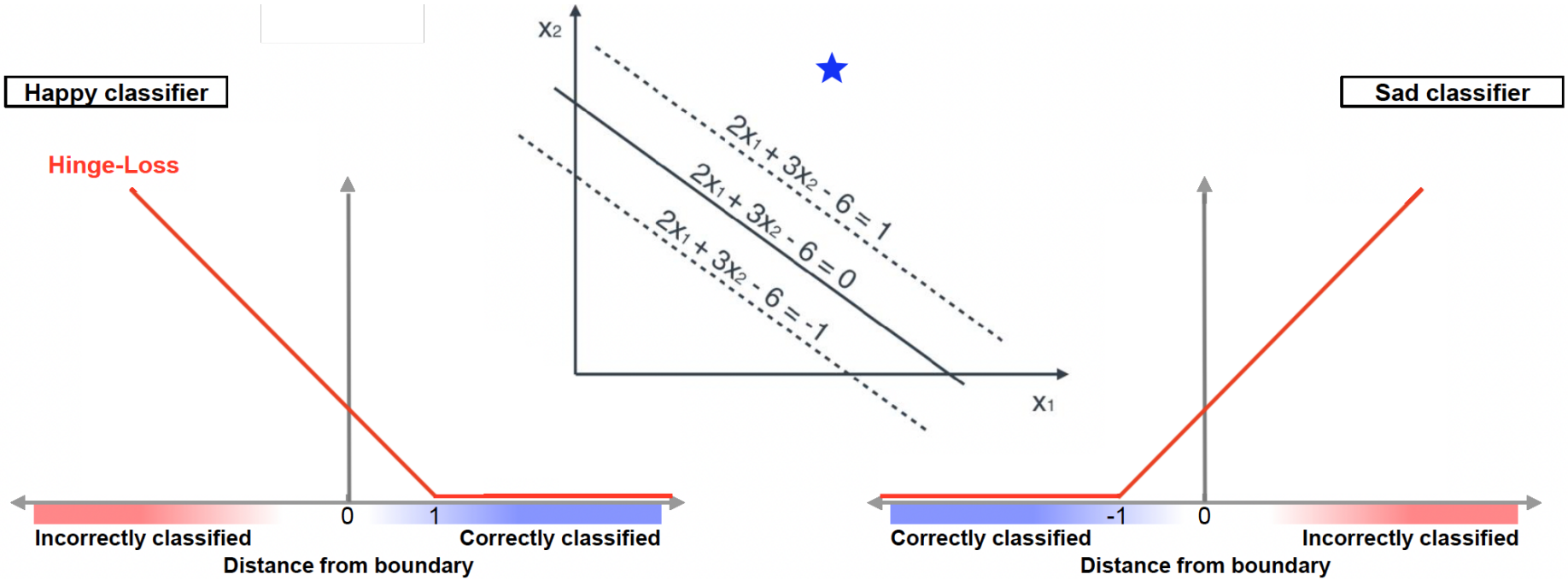


Hinge Loss (Classification Loss)

The Hinge Loss function is defined as:

$$L_{\text{classification}}(y, \hat{y}) = \max(0, 1 - y \cdot y')$$

where y is the true label (either $+1$ or -1) and y' is the predicted label.



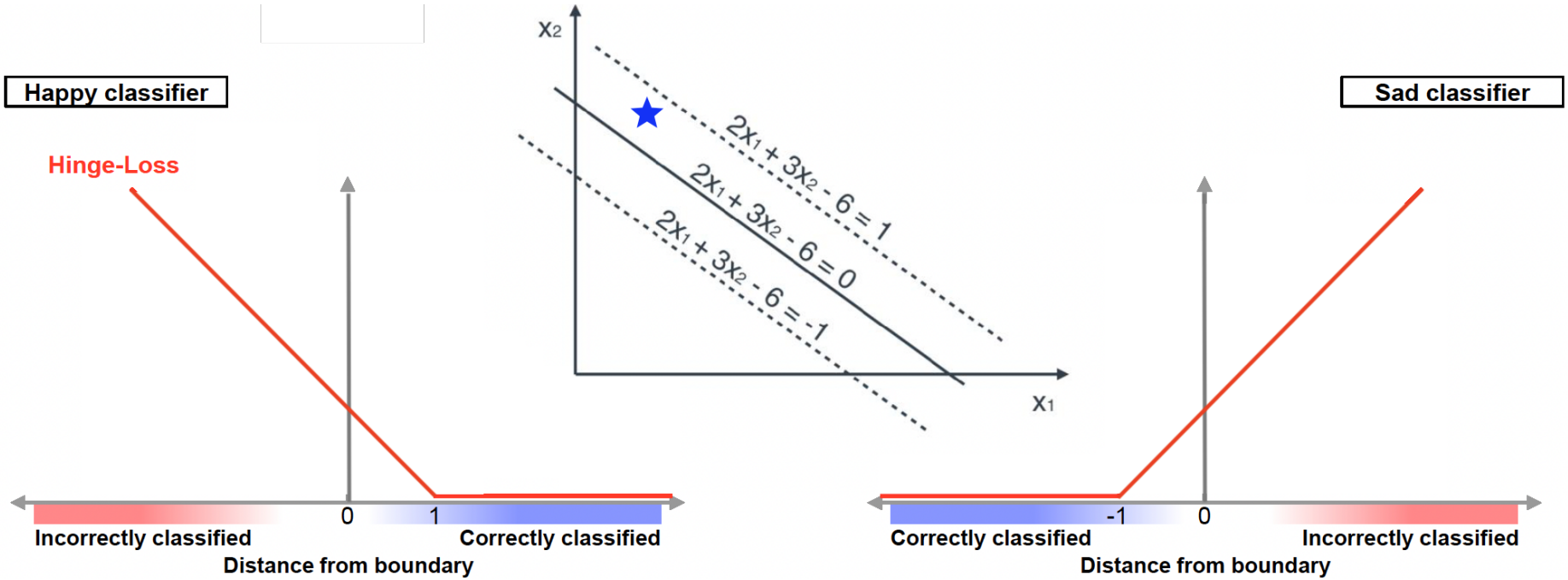
What is the loss for the blue data point if it is misclassified? Which area is the loss in?

Hinge Loss

The Hinge Loss function is defined as:

$$L_{\text{classification}}(y, \hat{y}) = \max(0, 1 - y \cdot y')$$

where y is the true label (either $+1$ or -1) and y' is the predicted label.



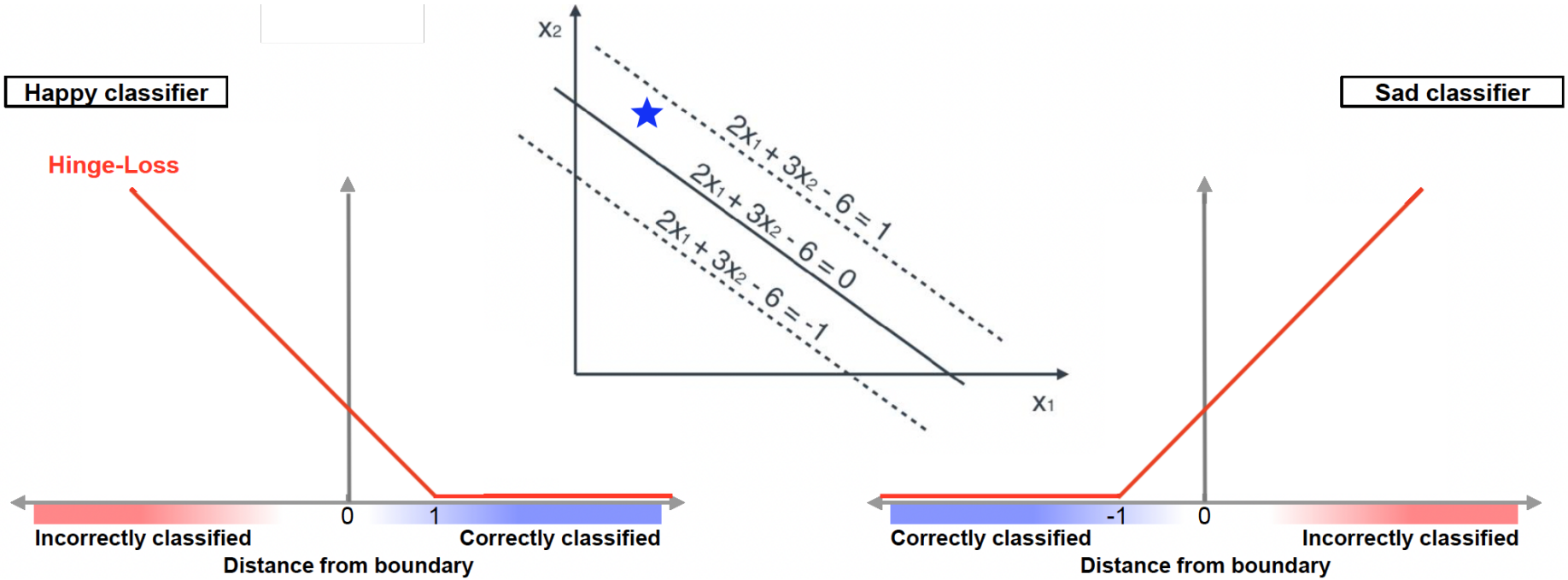
What is the loss for the blue data point if it is correctly classified? Which area is the loss in?

Hinge Loss

The Hinge Loss function is defined as:

$$L(y, \hat{y}) = \max(0, 1 - y \cdot y')$$

where y is the true label (either $+1$ or -1) and y' is the predicted label.



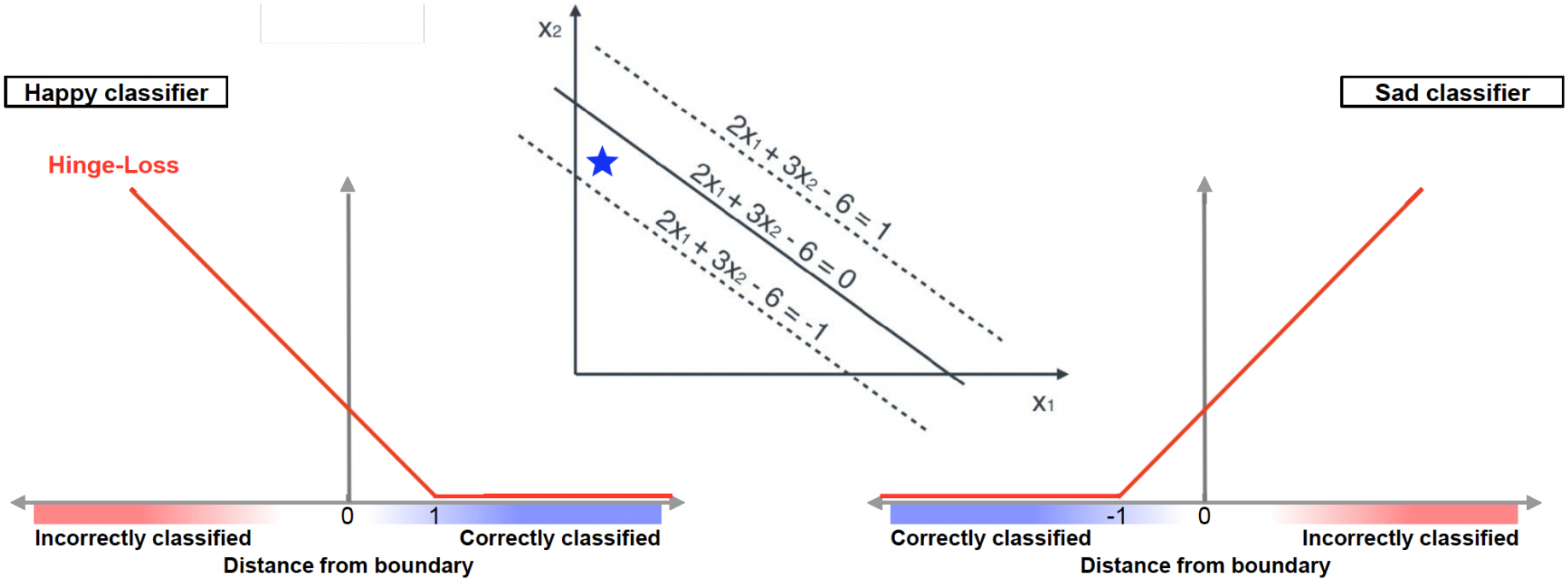
What is the loss for the blue data point if it is misclassified? Which area is the loss in?

Hinge Loss

The Hinge Loss function is defined as:

$$L_{\text{classification}}(y, y') = \max(0, 1 - y \cdot y')$$

where y is the true label (either $+1$ or -1) and y' is the predicted label.



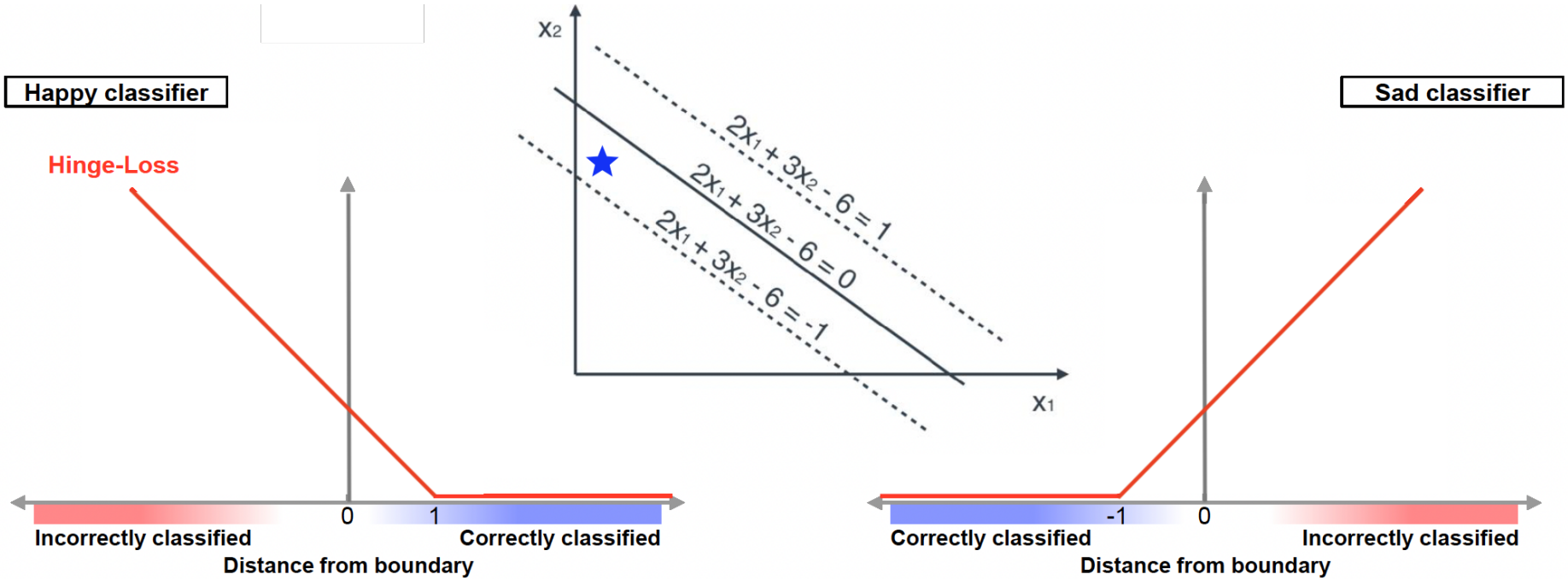
What is the loss for the blue data point if it is correctly classified? Which area is the loss in?

Hinge Loss

The Hinge Loss function is defined as:

$$L_{\text{classification}}(y, y') = \max(0, 1 - y \cdot y')$$

where y is the true label (either $+1$ or -1) and y' is the predicted label.



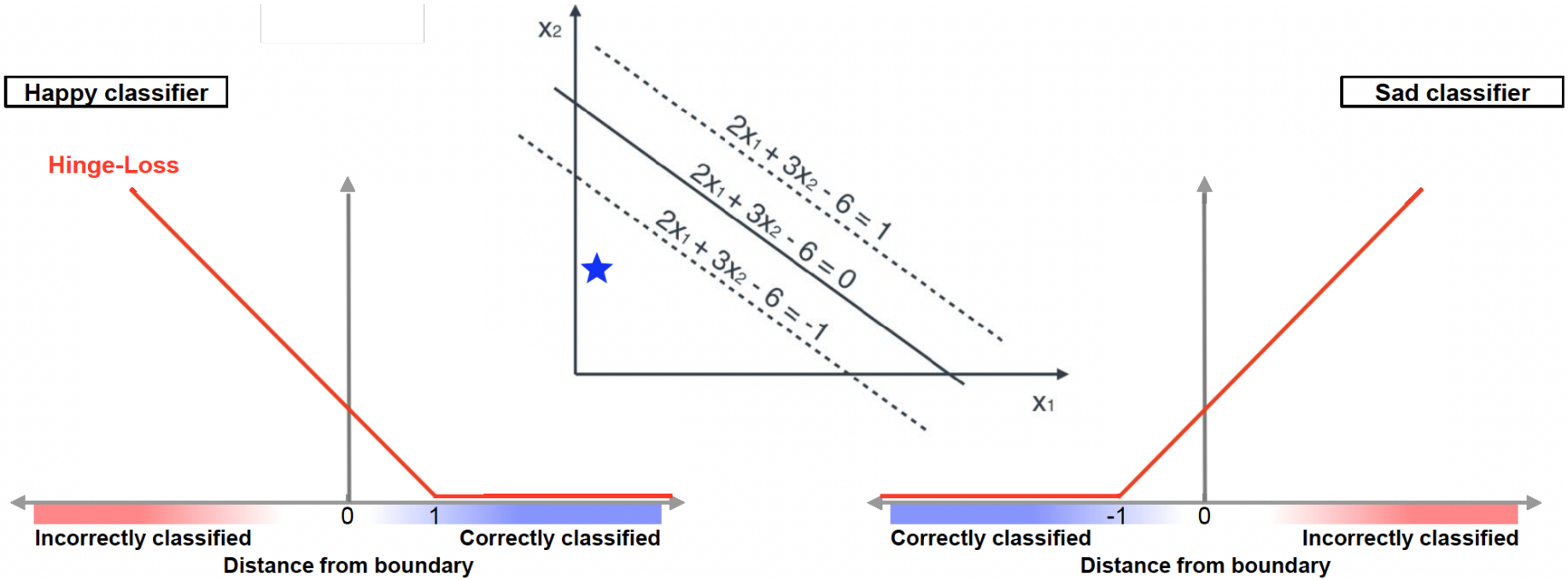
What is the loss for the blue data point if it is misclassified? Which area is the loss in?

Hinge Loss

The Hinge Loss function is defined as:

$$L_{\text{classification}}(y, y') = \max(0, 1 - y \cdot y')$$

where y is the true label (either $+1$ or -1) and y' is the predicted label.



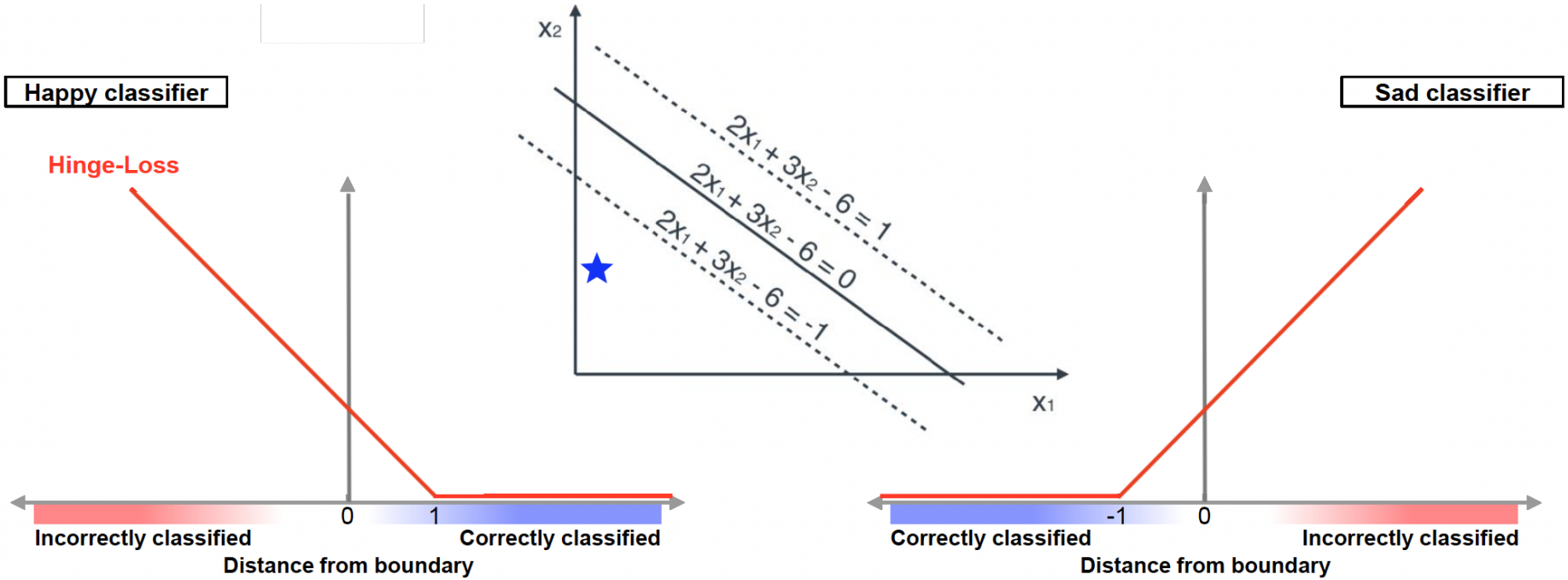
What is the loss for the blue data point if it is correctly classified? Which area is the loss in?

Hinge Loss

The Hinge Loss function is defined as:

$$L_{\text{classification}}(y, y') = \max(0, 1 - y \cdot y')$$

where y is the true label (either $+1$ or -1) and y' is the predicted label.



What is the loss for the blue data point if it is misclassified? Which area is the loss in?

Outline

Distance Loss Function (Large Margin)

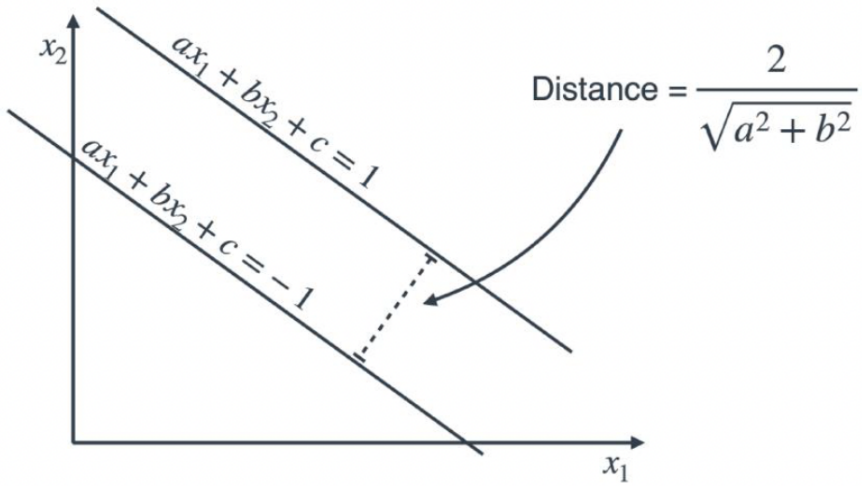
What is the distance between two parallel straight lines?

Distance Loss Function (Large Margin)

What is the distance between two parallel straight lines?

More general form:

$$\frac{2}{\|w\|} = \frac{2}{\sqrt{w_1^2 + w_2^2 + w_3^2 + \cdots + w_n^2}}$$



Note:

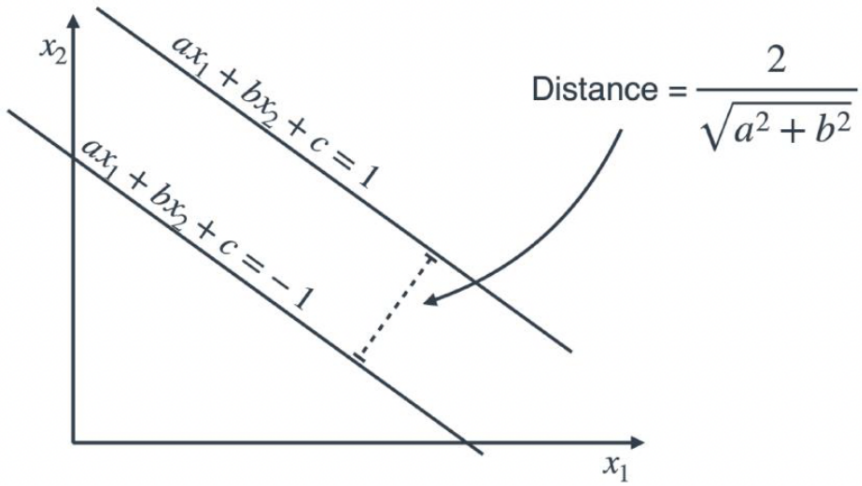
- The larger the weight vector, the smaller the (margin) Bad classifier.
- The smaller the weight vector, the larger the (margin) Good classifier.

Distance Loss Function (Large Margin)

What is the distance between two parallel straight lines?

More general form:

$$\frac{2}{\|w\|} = \frac{2}{\sqrt{w_1^2 + w_2^2 + w_3^2 + \cdots + w_n^2}}$$

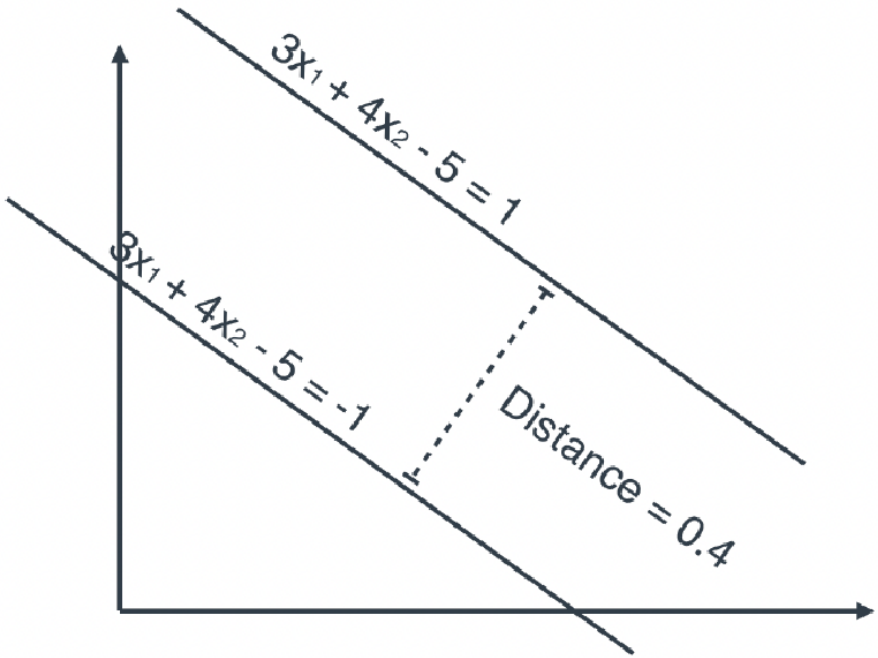


Note:

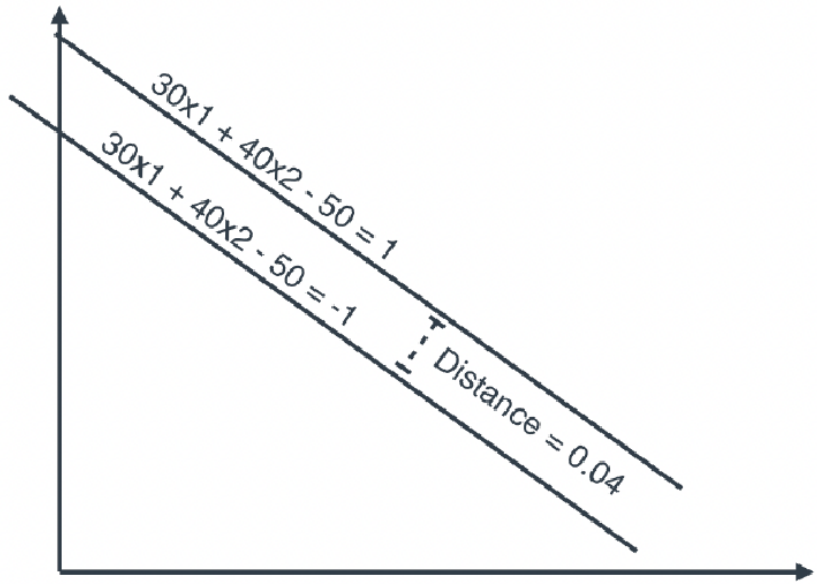
- The larger the weight vector, the smaller the (margin) Bad classifier.
- The smaller the weight vector, the larger the (margin) Good classifier.

There for maximizing the margin is equivalent to minimizing the weight vector!

Distance Loss Function (Large Margin)



Error function = 25
Good classifier



Error function = 2500
Bad classifier

Outline

Overall Loss Function

The overall loss function is the sum of the classification loss and the distance loss:

$$L(y, y') = \underbrace{L_{\text{classification}}(y, y')}_{\text{Hinge Loss}} + \underbrace{\frac{1}{2}\|w\|^2}_{\text{Regularization} \equiv \text{Large Margin}}$$

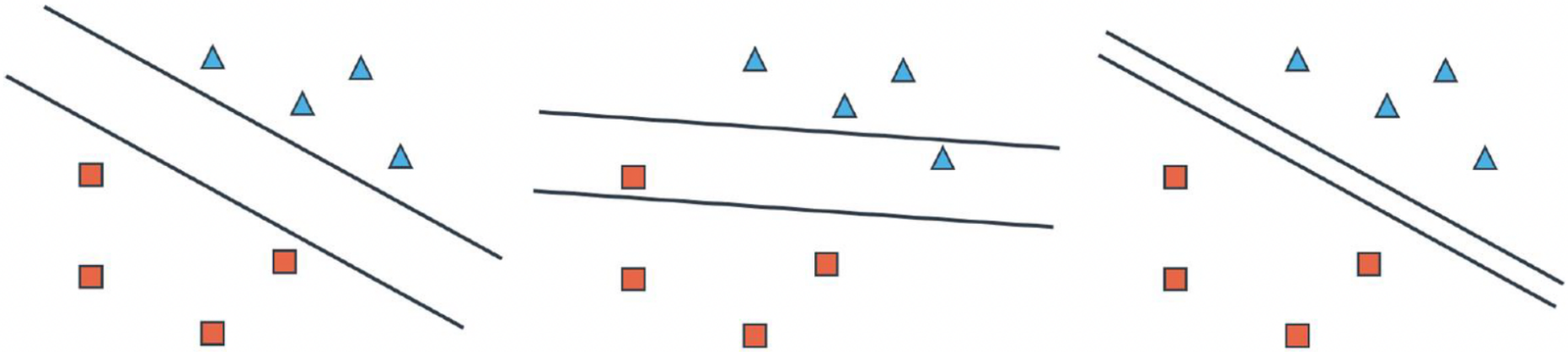
Overall Loss Function

The overall loss function is the sum of the classification loss and the distance loss:

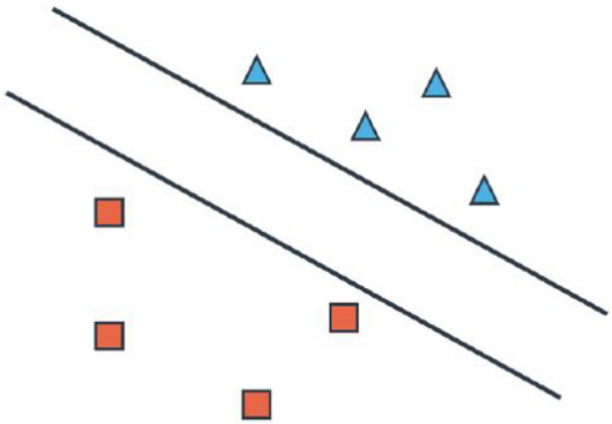
$$L(y, y') = \underbrace{L_{\text{classification}}(y, y')}_{\text{Hinge Loss}} + \underbrace{\frac{1}{2}\|w\|^2}_{\text{Regularization} \equiv \text{Large Margin}}$$

$$L(y, y') = \underbrace{max(0, 1 - y \cdot y')}_{\text{Hinge Loss}} + \underbrace{\frac{1}{2}\|w\|^2}_{\text{Regularization} \equiv \text{Large Margin}}$$

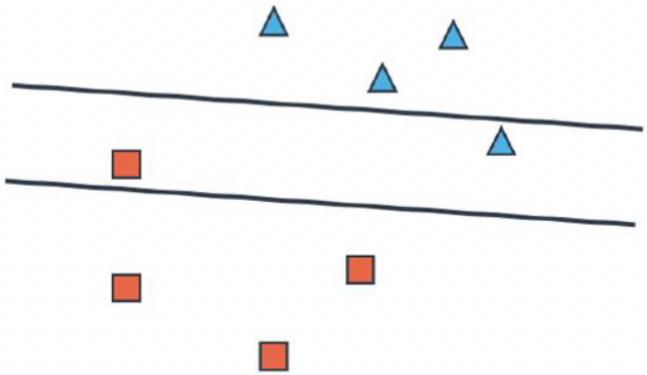
Overall Loss Function



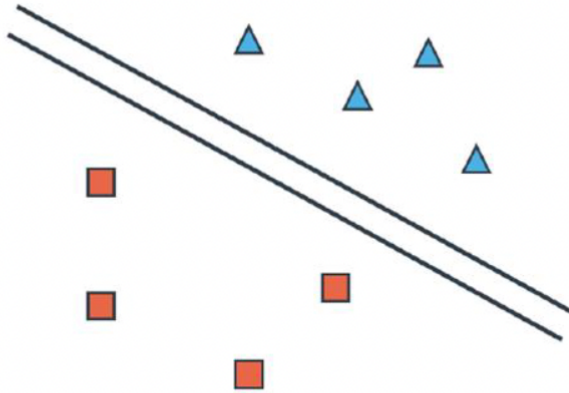
Overall Loss Function



Good Classifier



Bad classifier
(makes errors)



Bad classifier
(lines are too close together)

What is more important, classification loss or distance loss?

Overall Loss Function (C Parameter)

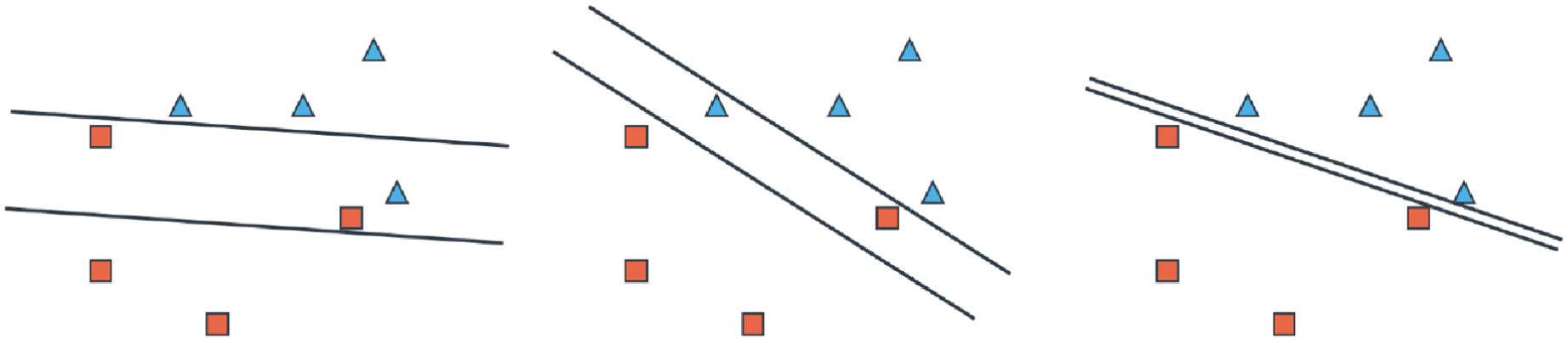
The parameter C is used to balance the classification loss and the margin loss.

$$L(y, y') = C \cdot L_{\text{classification}} + L_{\text{margin}}$$

Overall Loss Function (C Parameter)

The parameter C is used to balance the classification loss and the margin loss.

$$L(y, y') = C \cdot L_{\text{classification}} + L_{\text{margin}}$$

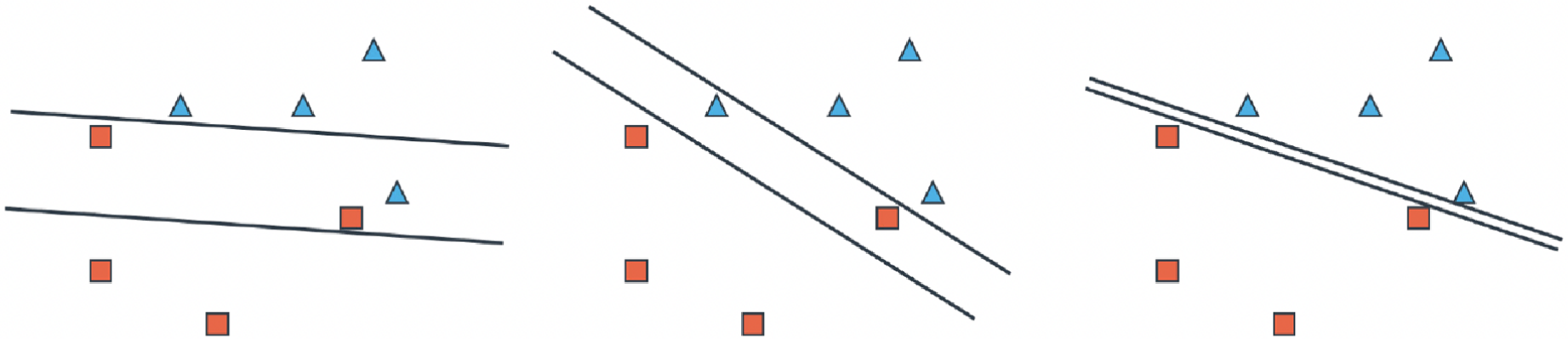


C = 0.01 C = 1 C = 100

Overall Loss Function (C Parameter)

The parameter C is used to balance the classification loss and the margin loss.

$$L(y, y') = C \cdot L_{\text{classification}} + L_{\text{margin}}$$



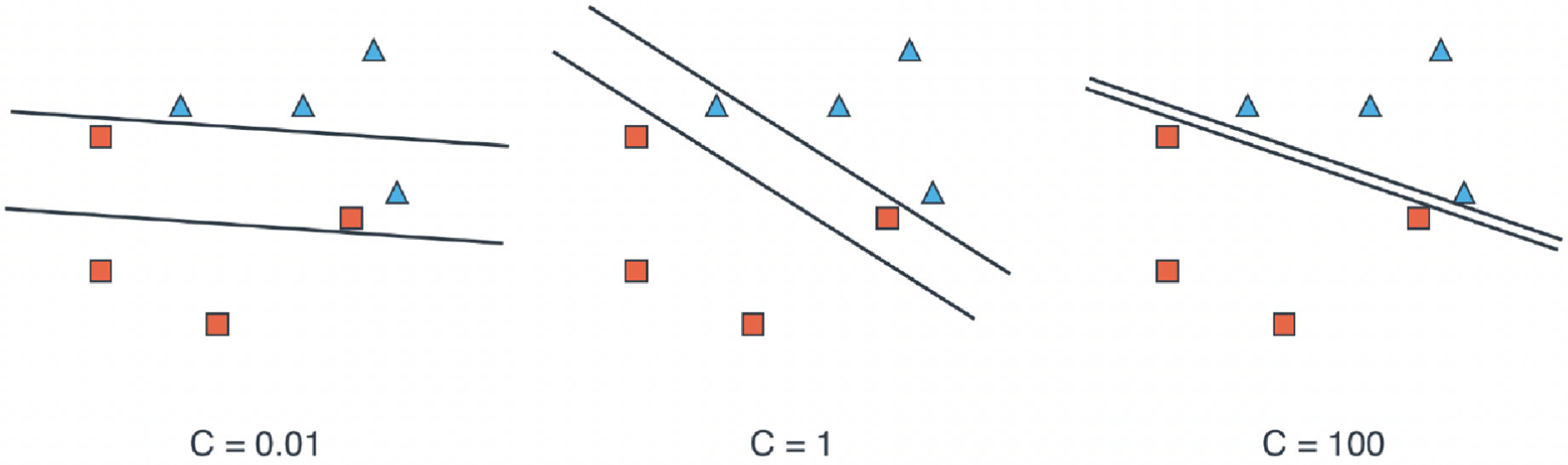
C = 0.01 C = 1 C = 100

How C parameter affects overfitting?

Overall Loss Function (C Parameter)

The parameter C is used to balance the classification loss and the margin loss.

$$L(y, y') = C \cdot L_{\text{classification}} + L_{\text{margin}}$$



How C parameter affects overfitting?

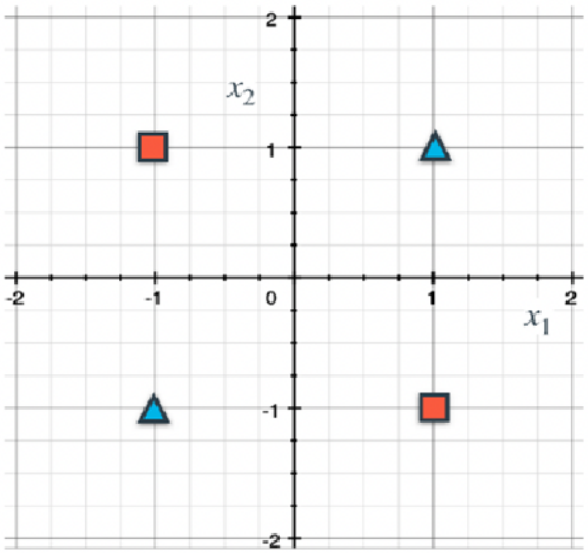
- Small C: More regularization (lower weight vector) ⇒ Less overfitting.
- Large C: Less regularization (higher weight vector) ⇒ More overfitting.

Outline

SVM Kernels

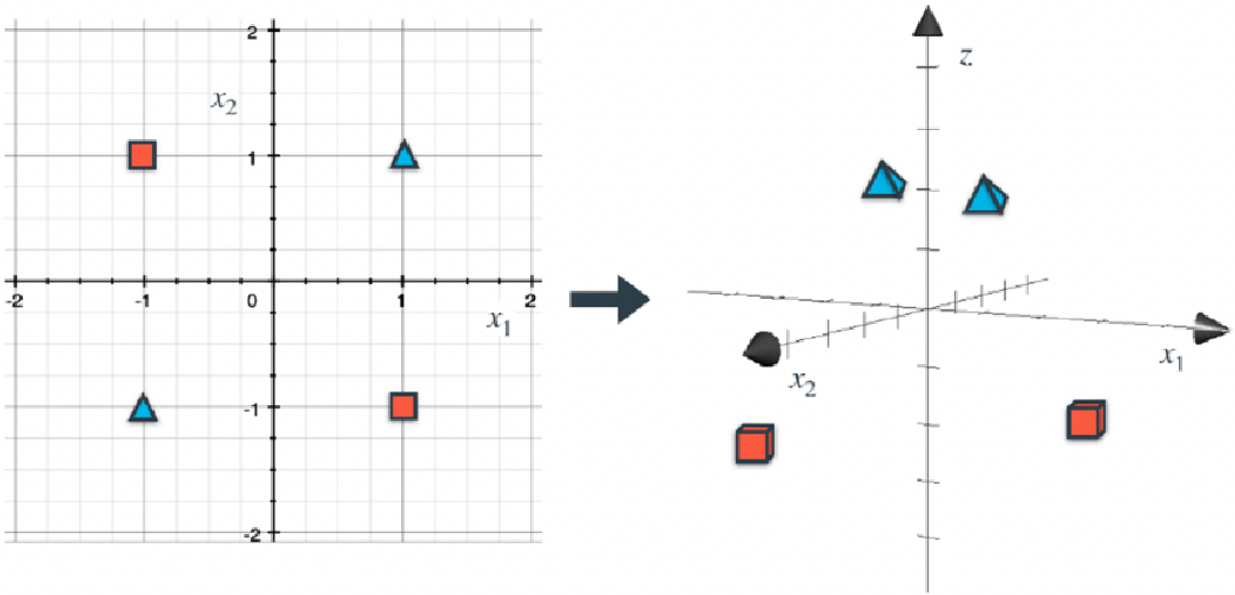
- SVM can only classify linearly separable data.
- SVM Kernels are used to transform the data into a higher dimensional space where it becomes linearly separable.
- Common Kernels:
  - Polynomial Kernel
  - RBF Kernel
  - Sigmoid Kernel

From Linear to Non-linear Boundaries



Not separable
by a line

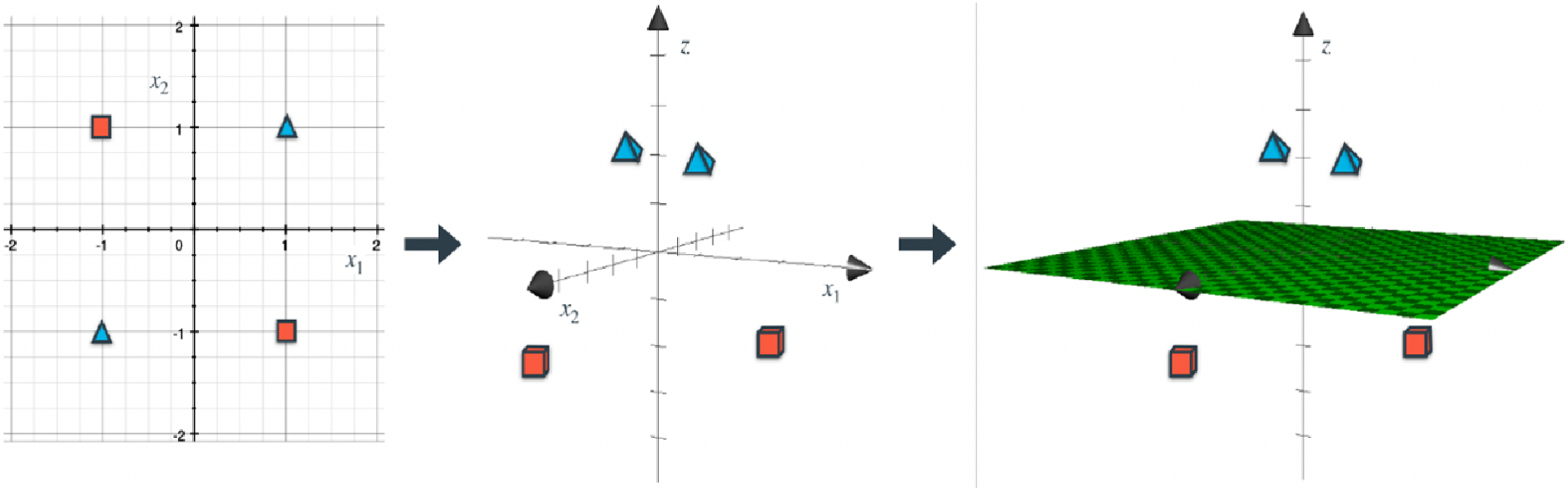
From Linear to Non-linear Boundaries



Not separable
by a line

Bring triangles up
Bring squares down

From Linear to Non-linear Boundaries



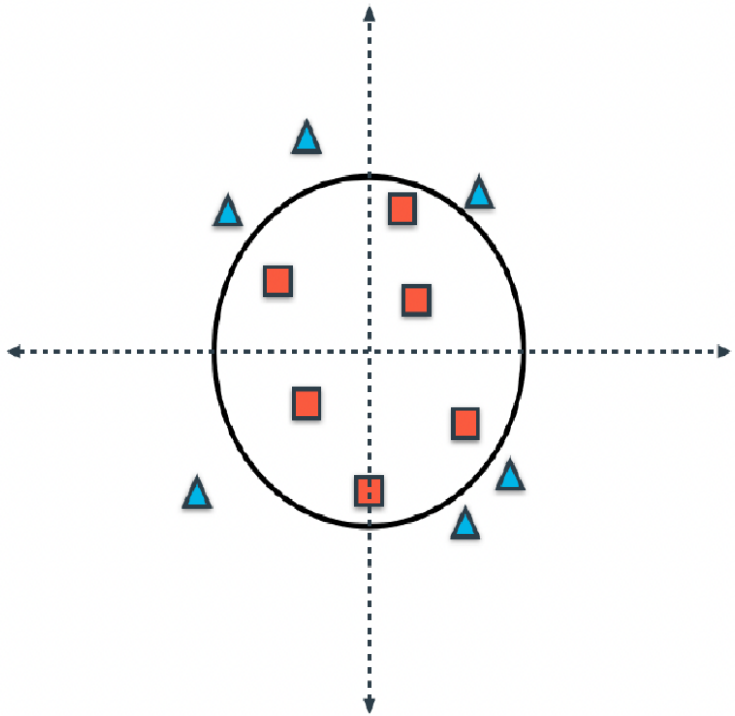
Not separable by a line

Bring triangles up
Bring squares down

Now separable by a plane

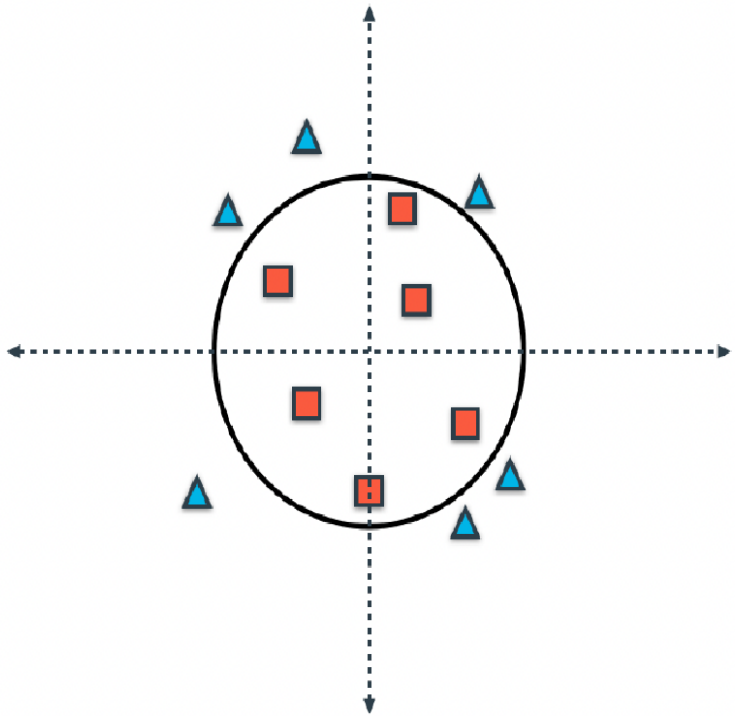
From Linear to Non-linear Boundaries

x_1	x_2	y
0.3	0.3	0
0.2	0.8	0
-0.6	0.4	0
0.6	-0.4	0
-0.4	-0.3	0
0	-0.8	0
-0.4	1.2	1
0.9	-0.7	1
-1.1	-0.8	1
0.7	0.9	1
-0.9	0.8	1
0.6	-1	1



From Linear to Non-linear Boundaries

x_1	x_2	y
0.3	0.3	0
0.2	0.8	0
-0.6	0.4	0
0.6	-0.4	0
-0.4	-0.3	0
0	-0.8	0
-0.4	1.2	1
0.9	-0.7	1
-1.1	-0.8	1
0.7	0.9	1
-0.9	0.8	1
0.6	-1	1



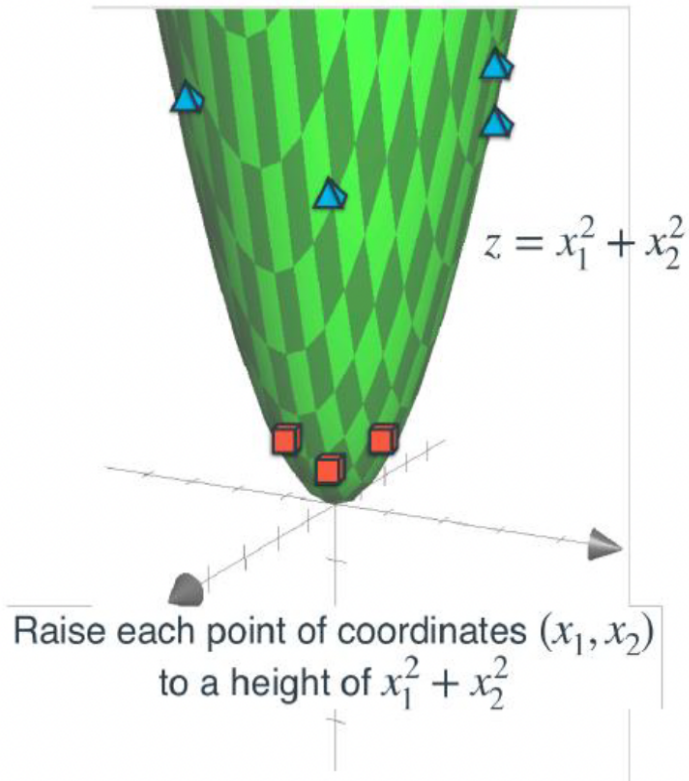
We need a circle with this formula: $x_1^2 + x_2^2 = 1$

From Linear to Non-linear Boundaries

x_1	x_2	$x_1^2 + x_2^2$	y
0.3	0.3	0.18	0
0.2	0.8	0.68	0
-0.6	0.4	0.52	0
0.6	-0.4	0.52	0
-0.4	-0.3	0.25	0
0	-0.8	0.64	0
-0.4	1.2	1.6	1
0.9	-0.7	1.3	1
-1.1	-0.8	1.85	1
0.7	0.9	1.3	1
-0.9	0.8	1.45	1
0.6	-1	1.36	1

From Linear to Non-linear Boundaries

x_1	x_2	$x_1^2 + x_2^2$	y
0.3	0.3	0.18	0
0.2	0.8	0.68	0
-0.6	0.4	0.52	0
0.6	-0.4	0.52	0
-0.4	-0.3	0.25	0
0	-0.8	0.64	0
-0.4	1.2	1.6	1
0.9	-0.7	1.3	1
-1.1	-0.8	1.85	1
0.7	0.9	1.3	1
-0.9	0.8	1.45	1
0.6	-1	1.36	1



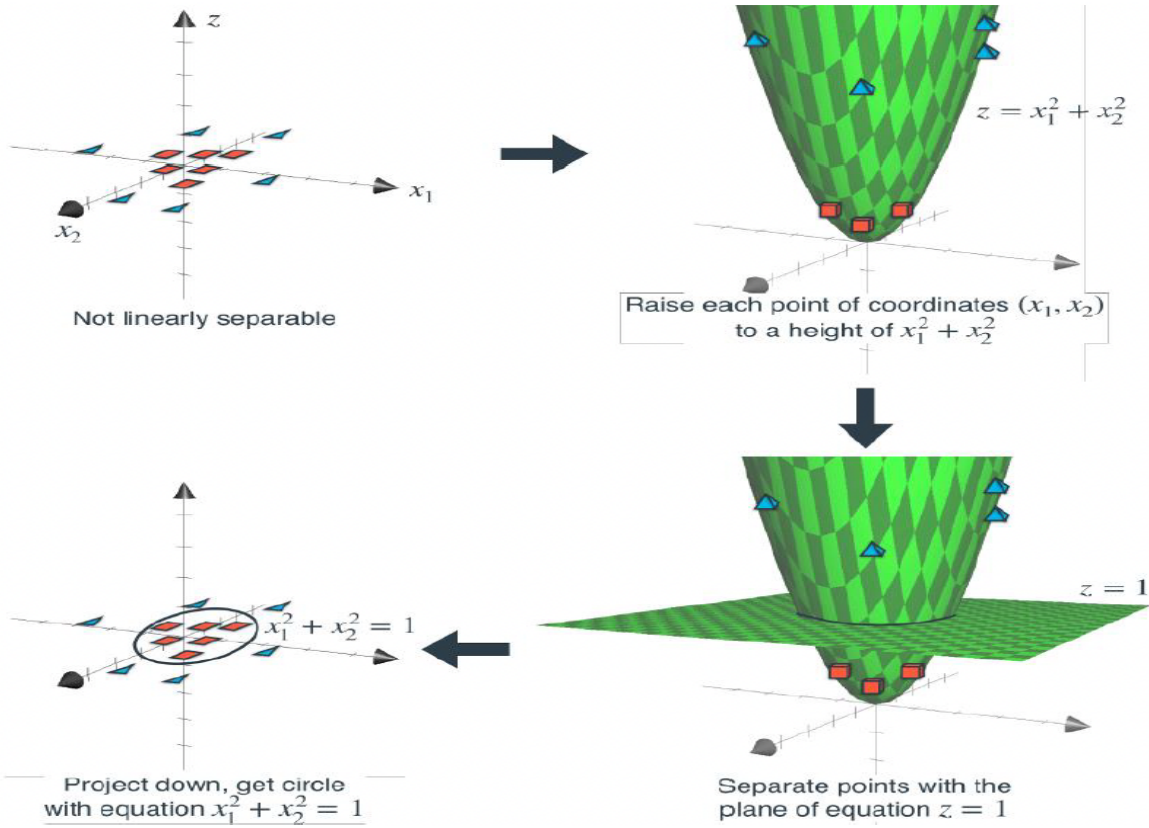
From Linear to Non-linear Boundaries

We may not have the luxury to look at a plot and eyeball an expression that will help us out.

We consider all the possible monomials of degree 2:

- These are the following three monomials:

$$x_1^2, \quad x_2^2, \quad x_1 x_2$$



Why do we need Kernels? Why not feature cross?

Suppose we have 3D input data (x_1, x_2, x_3):

- Feature cross for 2-degree:

Why do we need Kernels? Why not feature cross?

Suppose we have 3D input data (x_1, x_2, x_3):

- Feature cross for 2-degree:
  - Add 6 features: $x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3$

Why do we need Kernels? Why not feature cross?

Suppose we have 3D input data (x_1, x_2, x_3):

- Feature cross for 2-degree:
  - Add 6 features: $x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3$
- Feature cross for 3-degree:

Why do we need Kernels? Why not feature cross?

Suppose we have 3D input data (x_1, x_2, x_3):

- Feature cross for 2-degree:
  - Add 6 features: $x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3$
- Feature cross for 3-degree:
  - Add 10 features: $x_1^3, x_2^3, x_3^3, x_1^2x_2, x_1^2x_3, x_2^2x_1, x_2^2x_3, x_3^2x_1, x_3^2x_2, x_1x_2x_3$

Why do we need Kernels? Why not feature cross?

Suppose we have 3D input data (x_1, x_2, x_3):

- Feature cross for 2-degree:
  - Add 6 features: $x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3$
- Feature cross for 3-degree:
  - Add 10 features: $x_1^3, x_2^3, x_3^3, x_1^2x_2, x_1^2x_3, x_2^2x_1, x_2^2x_3, x_3^2x_1, x_3^2x_2, x_1x_2x_3$
- **Feature cross grows exponentially with the degree! It needs alot of memory!**

Why do we need Kernels? Why not feature cross?

Suppose we have 3D input data (x_1, x_2, x_3):

- Feature cross for 2-degree:
  - Add 6 features: $x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3$
- Feature cross for 3-degree:
  - Add 10 features: $x_1^3, x_2^3, x_3^3, x_1^2x_2, x_1^2x_3, x_2^2x_1, x_2^2x_3, x_3^2x_1, x_3^2x_2, x_1x_2x_3$
- **Feature cross grows exponentially with the degree! It needs alot of memory!**
- Instead of finding higher degree features from the same sample, we can find higher degree features between different samples (using Kernels).

Polynomial Kernel

The Polynomial Kernel is defined as:

$$K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + c)^d$$

where $\mathbf{a}$ and $\mathbf{b}$ are the input vectors (data samples), c is the bias, and d is the degree of the polynomial.

Polynomial Kernel

The Polynomial Kernel is defined as:

$$K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + c)^d$$

where $\mathbf{a}$ and $\mathbf{b}$ are the input vectors (data samples), c is the bias, and d is the degree of the polynomial. The polynomial kernel is a non-linear transformation of the input data.

Polynomial Kernel

The Polynomial Kernel is defined as:

$$K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + c)^d$$

where $\mathbf{a}$ and $\mathbf{b}$ are the input vectors (data samples), c is the bias, and d is the degree of the polynomial. The polynomial kernel is a non-linear transformation of the input data.

Example (assume 2D input data) $\mathbf{a}$: (a_1, a_2), $\mathbf{b}$: (b_1, b_2), $c = 1, d = 2$:

- $K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + 1)^2 =$

Polynomial Kernel

The Polynomial Kernel is defined as:

$$K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + c)^d$$

where $\mathbf{a}$ and $\mathbf{b}$ are the input vectors (data samples), c is the bias, and d is the degree of the polynomial. The polynomial kernel is a non-linear transformation of the input data.

Example (assume 2D input data) $\mathbf{a}$: (a_1, a_2), $\mathbf{b}$: (b_1, b_2), $c = 1, d = 2$:

- $K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + 1)^2 =$
- $(a_1 b_1 + a_2 b_2 + 1)^2 =$

Polynomial Kernel

The Polynomial Kernel is defined as:

$$K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + c)^d$$

where $\mathbf{a}$ and $\mathbf{b}$ are the input vectors (data samples), c is the bias, and d is the degree of the polynomial. The polynomial kernel is a non-linear transformation of the input data.

Example (assume 2D input data) $\mathbf{a}$: (a_1, a_2), $\mathbf{b}$: (b_1, b_2), $c = 1, d = 2$:

- $K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + 1)^2 =$
- $(a_1 b_1 + a_2 b_2 + 1)^2 =$
- $(a_1 b_1)^2 + (a_2 b_2)^2 + 1 + 2a_1 b_1 + 2a_2 b_2 + 2a_1 b_1 a_2 b_2$

Polynomial Kernel

The Polynomial Kernel is defined as:

$$K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + c)^d$$

where $\mathbf{a}$ and $\mathbf{b}$ are the input vectors (data samples), c is the bias, and d is the degree of the polynomial. The polynomial kernel is a non-linear transformation of the input data.

Example (assume 2D input data) $\mathbf{a}$: (a_1, a_2), $\mathbf{b}$: (b_1, b_2), $c = 1, d = 2$:

- $K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + 1)^2 =$
- $(a_1 b_1 + a_2 b_2 + 1)^2 =$
- $(a_1 b_1)^2 + (a_2 b_2)^2 + 1 + 2a_1 b_1 + 2a_2 b_2 + 2a_1 b_1 a_2 b_2$

Polynomial Kernel

The Polynomial Kernel is defined as:

$$K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + c)^d$$

where $\mathbf{a}$ and $\mathbf{b}$ are the input vectors (data samples), c is the bias, and d is the degree of the polynomial. The polynomial kernel is a non-linear transformation of the input data.

Example (assume 2D input data) $\mathbf{a}$: (a_1, a_2), $\mathbf{b}$: (b_1, b_2), $c = 1, d = 2$:

- $K(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + 1)^2 =$
- $(a_1 b_1 + a_2 b_2 + 1)^2 =$
- $(a_1 b_1)^2 + (a_2 b_2)^2 + 1 + 2a_1 b_1 + 2a_2 b_2 + 2a_1 b_1 a_2 b_2$

Note: We don't need to know the explicit form of the transformed feature space. We can directly compute the kernel function on the input data on the fly.

Outline

Non-linear SVM Decision Function

The decision function for non-linear SVM is:

$$y' = f(x) = \text{sign}\left(\sum_{i=1}^{N} w_i y^{(i)} K(x^{(i)}, x) + b\right)$$

where $K(x^{(i)}, x)$ is the kernel function.

- y' is the predicted label.

Non-linear SVM Decision Function

The decision function for non-linear SVM is:

$$y' = f(x) = \text{sign}\left(\sum_{i=1}^{N} w_i y^{(i)} K(x^{(i)}, x) + b\right)$$

where $K(x^{(i)}, x)$ is the kernel function.

- y' is the predicted label.
- w_i are the weights we learn from the training data.

Non-linear SVM Decision Function

The decision function for non-linear SVM is:

$$y' = f(x) = \text{sign}\left(\sum_{i=1}^{N} w_i y^{(i)} K(x^{(i)}, x) + b\right)$$

where $K(x^{(i)}, x)$ is the kernel function.

- y' is the predicted label.
- w_i are the weights we learn from the training data.
- b is the bias.

Non-linear SVM Decision Function

The decision function for non-linear SVM is:

$$y' = f(x) = \text{sign}\left(\sum_{i=1}^{N} w_i y^{(i)} K(x^{(i)}, x) + b\right)$$

where $K(x^{(i)}, x)$ is the kernel function.

- y' is the predicted label.
- w_i are the weights we learn from the training data.
- b is the bias.
- We can think of the kernel as a **similarity measure** between the input data and all the training samples. The higher the similarity, the higher the dot product kernel value.

Non-linear SVM Decision Function

The decision function for non-linear SVM is:

$$y' = f(x) = \text{sign}\left(\sum_{i=1}^{N} w_i y^{(i)} K(x^{(i)}, x) + b\right)$$

where $K(x^{(i)}, x)$ is the kernel function.

- Accumulating more +ve similarity than -ve similarity will push the decision function towards +ve label.

Non-linear SVM Decision Function

The decision function for non-linear SVM is:

$$y' = f(x) = \text{sign}\left(\sum_{i=1}^{N} w_i y^{(i)} K(x^{(i)}, x) + b\right)$$

where $K(x^{(i)}, x)$ is the kernel function.

- Accumulating more +ve similarity than -ve similarity will push the decision function towards +ve label.
- Accumulating more -ve similarity than +ve similarity will push the decision function towards -ve label.

Non-linear SVM Decision Function

The decision function for non-linear SVM is:

$$y' = f(x) = \text{sign}\left(\sum_{i=1}^{N} w_i y^{(i)} K(x^{(i)}, x) + b\right)$$

where $K(x^{(i)}, x)$ is the kernel function.

- Accumulating more +ve similarity than -ve similarity will push the decision function towards +ve label.
- Accumulating more -ve similarity than +ve similarity will push the decision function towards -ve label.
- Cons: The weight vector length is proportional to the training samples.

Advantages of Kernels

- We don't need to know the explicit form of the transformed feature space.

Advantages of Kernels

- We don't need to know the explicit form of the transformed feature space.
- We can compute the kernel function on the input data on the fly.

Advantages of Kernels

- We don't need to know the explicit form of the transformed feature space.
- We can compute the kernel function on the input data on the fly.
- We can use different kernels for different data.

Outline

Outline

Hinge Loss Function and Its Derivative

Hinge Loss Function

The hinge loss function for binary classification is defined as:

$$L(y, y') = \max(0, 1 - y \cdot y')$$

where $y \in \{-1, +1\}$ is the true label and y' is the predicted score.

The derivative of the hinge loss with respect to y' is:

$$\frac{\partial L}{\partial w_i} = \begin{cases} 0, & \text{if } (1 - y \cdot y' < 0) \equiv (y \cdot y' > 1) \\ -y \cdot \frac{dy'}{dw_i}, & \text{if } (1 - y \cdot y' > 0) \equiv (y \cdot y' < 1) \end{cases}$$

Or more concisely:

$$\frac{\partial L}{\partial w_i} = -y \cdot \frac{dy'}{dw_i} \qquad if\,(y \cdot y' < 1)$$

Hinge Loss Function and Its Derivative

$$\frac{\partial L}{\partial w_i} = -y \cdot \frac{dy'}{dw_i} \qquad if\,(y \cdot y' < 1)$$

Remember:

$$y' = \text{sign}\left(\sum_{i=1}^{N} w_i y^{(i)} K(x^{(i)}, x) + w_0\right)$$

Thus,

Hinge Loss Function and Its Derivative

$$\frac{\partial L}{\partial w_i} = -y \cdot \frac{dy'}{dw_i} \qquad if\,(y \cdot y' < 1)$$

Remember:

$$y' = \text{sign}\left(\sum_{i=1}^{N} w_i y^{(i)} K(x^{(i)}, x) + w_0\right)$$

Thus,

$$\frac{dy'}{dw_i} = \begin{cases} y^{(i)} K(x^{(i)}, x), & \text{for } i = 1, \ldots, N \\ 1, & \text{for } i = 0 \text{ (bias term)} \end{cases}$$

SVM Classifier [Training with SGD]

- Pick random weights $w_1, w_2, \ldots, w_N$ and a random bias w_0.
- Repeat **many times**:
  1. Pick a random data point $(x^{(i)}, y^{(i)})$ where $y^{(i)} \in \{-1, +1\}$.
  2. Compute Model Prediction:

$$y'^{(i)} = \sum_{j=1}^{N} w_j y^{(j)} K(x^{(j)}, x^{(i)}) + w_0$$

  3. **Update the weights and bias if margin is violated:**

If $(y^{(i)} \cdot y'^{(i)} < 1)$:

$$w_j = w_j + \eta y^{(i)} y^{(j)} K(x^{(j)}, x^{(i)}) - \eta \lambda w_j \quad \text{for } j = 1, \ldots, N$$

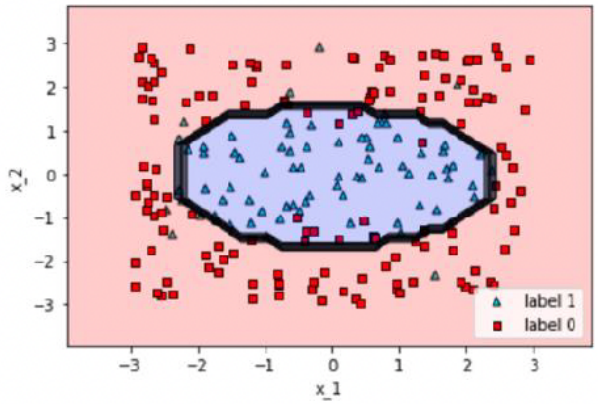
$$w_0 = w_0 + \eta y^{(i)}$$

where η is the learning rate.

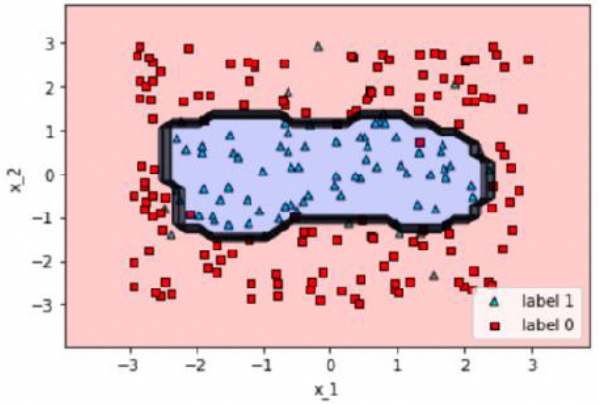
- Return the model you've obtained.

SVM Classifier [Results]

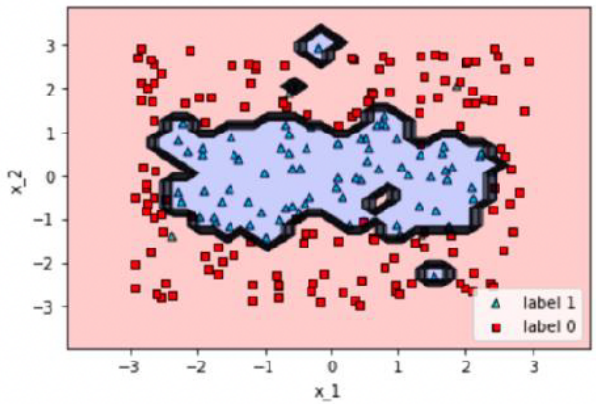
Gamma = 0.1
Accuracy: 0.8772727272727273



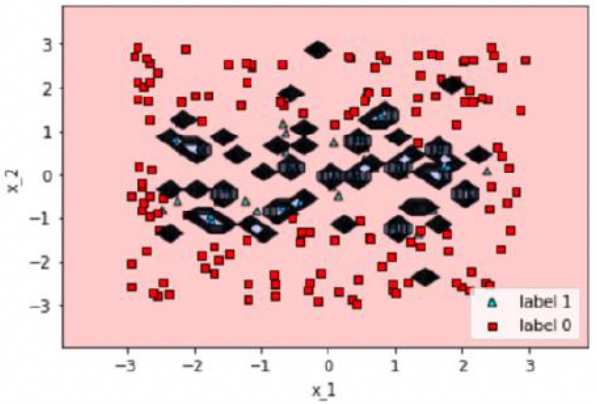
Gamma = 1
Accuracy: 0.9045454545454545



Gamma = 10
Accuracy: 0.9636363636363636



Gamma = 100
Accuracy: 0.990909090909091



Questions