Solution 1

Step 1

We are given the prediction rule is defined as:

$$(2x_1 - x_2 - 6)$$

To find the decision boundary we set the prediction rule equal to zero:

$$2x_1 - x_2 - 6 = 0 \qquad \text{or} \qquad x_2 = 2x_1 - 6$$

We can rearrange this to express x_2 in terms of x_1:

$$x_2 = 2x_1 - 6$$

Step 2

Find the point (x_1, x_2) where the decision boundary intersects the x_1 axis (i.e. $x_2 = 0$):

$$x_2 = 2x_1 - 6 \;\rightarrow\; 0 = 2x_1 - 6 \;\rightarrow\; 2x_1 = 6 \;\rightarrow\; x_1 = 3$$

Hence, the decision boundary intersects the x_1 axis at $(3, 0)$

Step 3

Find the point (x_1, x_2) where the decision boundary intersects the x_2 axis (i.e. $x_1 = 0$):

$$x_2 = 2x_1 - 6 \;\rightarrow\; x_2 = 2(0) - 6 \;\rightarrow\; 2x_2 = -6$$

Hence, the decision boundary intersects the x_1 axis at $(0, 6)$

Step 4

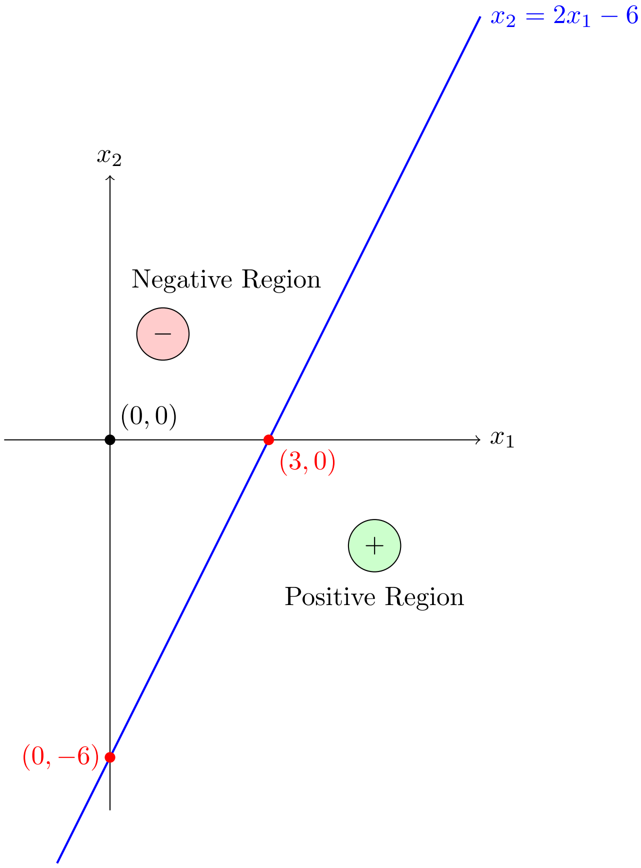
Test prediction rule at the point $(0, 0)$ to determine the classification above the decision boundary.

$$2x_1 - x_2 - 6 \;\rightarrow\; 2(0) - 0 - 6 \;\rightarrow\; -6$$

It follows that, $2x_1 - x_2 - 6 < 0$ at $(0, 0)$ and this area above the decision boundary will be classified as negative.

Step 5

Visualize the decision boundary:



$\therefore$ The decision boundary is the line $x_2 = 2x_1 - 6$, which intersects the x_1 axis at $(3, 0)$ and the x_2 axis at $(0, -6)$. The region below this line is classified as positive, and the region above it is classified as negative.

Solution 2 (a)

$\therefore$ The statement: *The data set is linearly separable.* is **definitely true**. The Perceptron algorithm will convergethe data is linearly separable and we are given: *it converges after making k updates.*

Solution 2 (b)

$\therefore$ The statement: *If the process were repeated with a different random permutation, it would again converge* is **definitely true**. Since the Perceptron algorithm will converge regardless of order.

Solution 2 (c)

$\therefore$ The statement: *If the process were repeated with a different random permutation, it would again converge after making k updates* is **possibly false**. The number of updates required for convergence can change depending on the order of data.

Solution 2 (d)

$\therefore$ The statement: *k is at most n* is **possibly false**. The number of updates k can exceed the number of data points n.

Solution 3

Step 1

A point (x, y) is misclassified when:

$$y(w \cdot x + b) \leq 0$$

The Perceptron algorithm tells us to update w and b when a point is misclassified as the following:

$$w = w + yx \text{ and } b = b + y$$

Step 2

We are given the following:

- Perceptron algorithm performs $p + q$ updates before converging
- p updates on data points with label $y_i = -1$
- q updates on data points with label $y_i = +1$

Step 3

Let the initial bias be $b = 0$. Each time a misclassified point is encountered, the bias is updated by adding the label y_i.

- For each of the p negative examples ($y_i = -1$), the bias decreases by 1: total change is $-p$
- For each of the q positive examples ($y_i = 1$), the bias increases by 1: total change is q

$\therefore$ The final value of the parameter b is $q - p$.

Solution 4 (a)

Step 1

Given:

- SVM classifier in $\mathbb{R}^2$
- Weight vector $w = (3, 4)$
- Bias term $b = -12$

The prediction rule would then be defined as:

$$(3x_1 + 4x_2 - 12)$$

Step 2

Find the point (x_1, x_2) where the decision boundary intersects the x_1 axis (i.e. $x_2 = 0$):

$$3x_1 + 4(0) - 12 = 0 \quad \rightarrow \quad 3x_1 = 12 \quad \rightarrow \quad x_1 = 4$$

Hence, the decision boundary intersects the x_1 axis at $(4, 0)$

Step 3

Find the point (x_1, x_2) where the decision boundary intersects the x_2 axis (i.e. $x_1 = 0$):

$$3(0) + 4x_2 - 12 = 0 \quad \rightarrow \quad 4x_2 = 12 \quad \rightarrow \quad x_2 = 3$$

Hence, the decision boundary intersects the x_2 axis at $(0, 3)$

Step 4

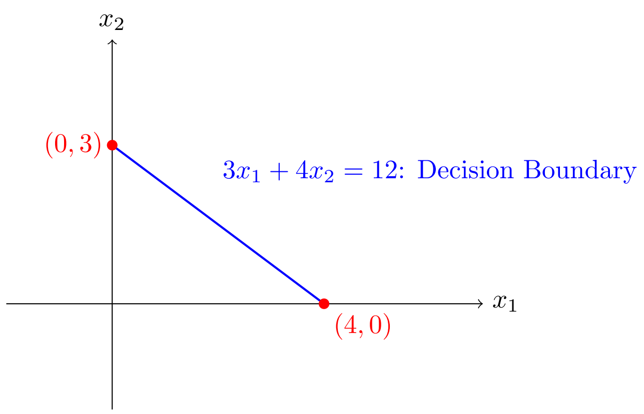
Test prediction rule at the point $(0, 0)$ to determine the classification below the decision boundary.

$$3(0) + 4(0) - 12 = -12$$

It follows that $3x_1 + 4x_2 - 12 < 0$ at $(0, 0)$, and so this region below the decision boundary will be classified as negative.

Step 5

Visualize the decision boundary:



Solution 4 (b)

Step 1

The margin boundaries are defined as:

$$w \cdot x + b = 1 \quad \text{(positive margin boundary)} \tag{1}$$
$$w \cdot x + b = -1 \quad \text{(negative margin boundary)} \tag{2}$$

Remember the prediction rule would then is defined as:

$$(3x_1 + 4x_2 - 12)$$

Step 2

Solve for right hand boundary $((3x_1 + 4x_2 - 13))$:
Find the point (x_1, x_2) where the right hand boundary intersects the x_1 axis (i.e. $x_2 = 0$):

$$3x_1 + 4(0) - 13 = 0 \;\; \rightarrow \;\; 3x_1 = 13 \;\; \rightarrow \;\; x_1 = \frac{13}{3}$$

Hence, the right hand boundary intersects the x_1 axis at $\left(\frac{13}{3}, 0\right)$.

Now, find the point (x_1, x_2) where the right hand boundary intersects the x_2 axis (i.e. $x_1 = 0$):

$$3(0) + 4x_2 - 13 = 0 \;\; \rightarrow \;\; 4x_2 = 13 \;\; \rightarrow \;\; x_2 = \frac{13}{4}$$

Hence, the right hand boundary intersects the x_2 axis at $\left(0, \frac{13}{4}\right)$.

Step 3

Solve for left hand boundary $((3x_1 + 4x_2 - 11))$:
Find the point (x_1, x_2) where the left hand boundary intersects the x_1 axis (i.e. $x_2 = 0$):

$$3x_1 + 4(0) - 11 = 0 \;\; \rightarrow \;\; 3x_1 = 11 \;\; \rightarrow \;\; x_1 = \frac{11}{3}$$

Hence, the left hand boundary intersects the x_1 axis at $\left(\frac{11}{3}, 0\right)$.

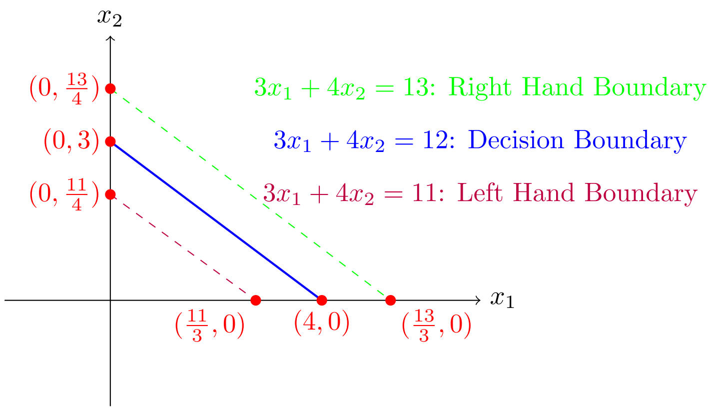
Now, find the point (x_1, x_2) where the left hand boundary intersects the x_2 axis (i.e. $x_1 = 0$):

$$3(0) + 4x_2 - 11 = 0 \;\; \rightarrow \;\; 4x_2 = 11 \;\; \rightarrow \;\; x_2 = \frac{11}{4}$$

Hence, the left hand boundary intersects the x_2 axis at $\left(0, \frac{11}{4}\right)$.

Step 4

Visualize the left hand and right hand boundaries:


x_2
x_1
$(0, \frac{13}{4})$
$(0, 3)$
$(0, \frac{11}{4})$
$(\frac{11}{3}, 0)$
$(4, 0)$
$(\frac{13}{3}, 0)$
$3x_1 + 4x_2 = 13$: Right Hand Boundary
$3x_1 + 4x_2 = 12$: Decision Boundary
$3x_1 + 4x_2 = 11$: Left Hand Boundary

Solution 4 (c)

Step 1

The margin of an SVM classifier is defined below where $||w||$, the Euclidean norm of the weight vector:

$$\text{Margin} = \frac{2}{||w||} = \frac{2}{\sqrt{w_1^2 + w_2^2}}$$

Step 2

Calculate the margin:

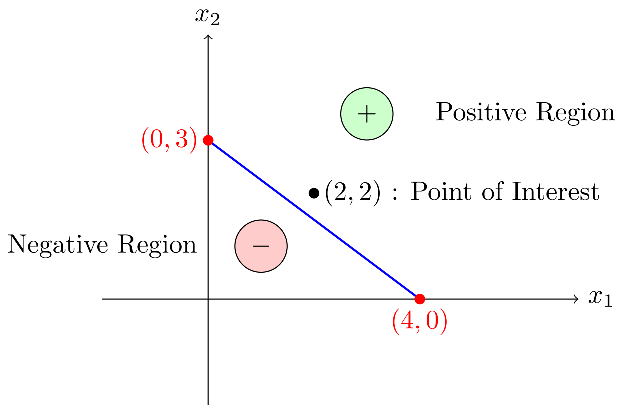
$$\begin{aligned}\text{Margin} &= \frac{2}{||w||} \\ &= \frac{2}{\sqrt{3^2 + 4^2}} \\ &= \frac{2}{\sqrt{25}} \\ &= \frac{2}{5} \\ &= 0.4\end{aligned}$$

$\therefore$ The margin of this SVM classifier is $\frac{2}{5}$ or 0.4 units.

Solution 4 (d)

Step 1

Use visulization from **Solution 4 (a)** and plot the point $(2, 2)$:



$\therefore$ The point $(2, 2)$ would be classified as positive.

Solution 5 (a)

Step 1

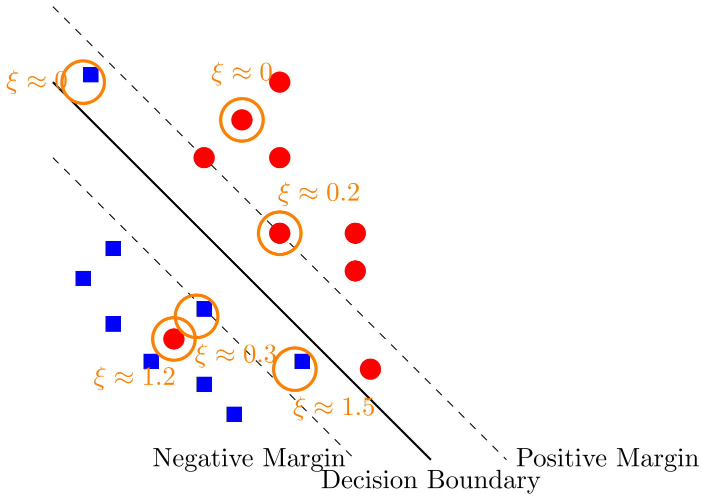
In a soft-margin SVM, the support vectors are the data points that:

- Lie exactly on the margin boundaries (slack variable $\xi_i = 0$)
- Lie between the margin boundary and the decision boundary (slack variable $0 < \xi_i < 1$)
- Lie on the wrong side of the margin boundary (slack variable $\xi_i \geq 1$)

The slack variable ξ_i represents the degree of misclassification for each point. It measures how far a point is from its correct margin boundary.

Step 2

Let's identify the support vectors in the given figure:



Step 3

Let's explain the identified support vectors:

1. **Support vectors with $\xi \approx 0$:**
   - The blue square at approximately $(0.4, 5.0)$ lies very close to the negative margin boundary.
   - The red circle at approximately $(2.5, 4.5)$ lies very close to the positive margin boundary.
   - These points have slack variables close to zero because they are almost exactly on their respective margin boundaries.
2. **Support vectors with $0 < \xi < 1$:**
   - The blue square at approximately $(1.9, 1.9)$ is between the negative margin and the decision boundary.
   - The red circle at approximately $(3, 3)$ is between the positive margin and the decision boundary.
   - These points have slack variables between 0 and 1 because they are inside the margin but on the correct side of the decision boundary.
3. **Support vectors with $\xi \geq 1$:**
   - The red circle at approximately $(1.6, 1.6)$ is on the wrong side of the decision boundary.
   - The blue square at approximately $(3.2, 1.2)$ is also on the wrong side of the decision boundary.
   - These points have slack variables greater than or equal to 1 because they are misclassified by the decision boundary.

$\therefore$ The support vectors are the points circled in orange in the figure above, with their approximate slack variable values indicated. Support vectors include points exactly on the margin boundaries ($\xi \approx 0$), points between the margin and decision boundary ($0 < \xi < 1$), and misclassified points ($\xi \geq 1$).

Solution 5 (b)

Step 1

Recall the optimization problem for soft-margin SVM:

$$\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i$$

subject to:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i$$

The parameter C controls the trade-off between maximizing the margin (minimizing $||w||^2$) and minimizing the classification error (minimizing $\sum \xi_i$).

Step 2

When C is increased:

- The penalty for misclassification and margin violations becomes higher.
- The optimization will prioritize reducing the slack variables ξ_i over maximizing the margin.
- This means the model will try harder to correctly classify all training points, potentially at the expense of a smaller margin.

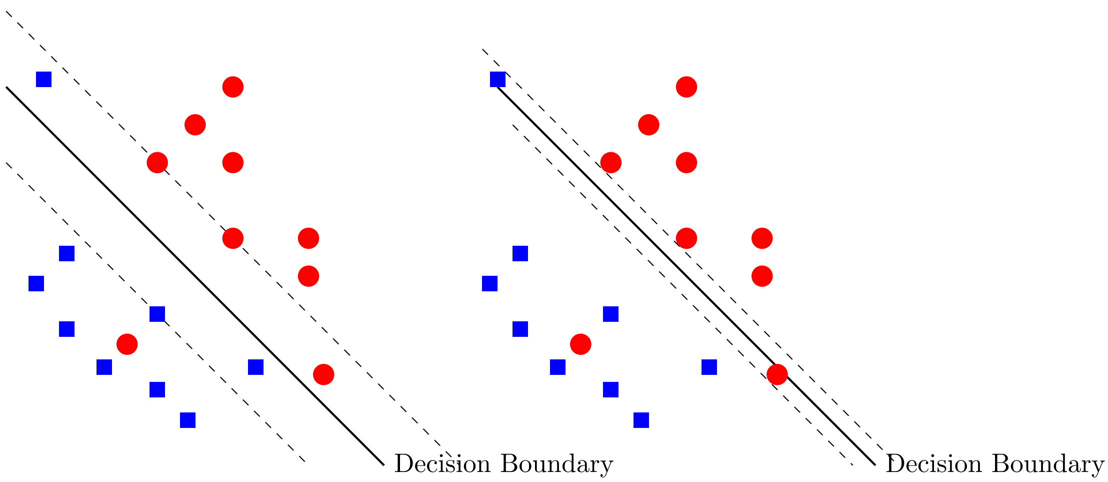
Step 3

Let's compare the effects of different values of C:

Small C	**Medium C**	**Large C**
Larger margin More misclassifications Underfitting risk	Balanced trade-off Some misclassifications Good generalization	Smaller margin Fewer misclassifications Overfitting risk

In our specific example, increasing C would likely:

- Make the decision boundary adjust to reduce the misclassifications (the red circle at $(1.6, 1.6)$ and the blue square at $(3.2, 1.2)$).
- Potentially reduce the margin between the dashed lines to better accommodate the training points.



Medium C (Original) Large C (Adjusted)

$\therefore$ If the factor C in the soft-margin SVM optimization problem were increased, we would expect the margin to **decrease**. This is because a larger C places more emphasis on correctly classifying all training points, even if it means having a smaller margin.

Solution 6 (a)

Step 1

Let's recall the dual form of the Perceptron algorithm:
In the dual form, we represent the weight vector w as a linear combination of the training examples:

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

where α_i counts how many times example i has been misclassified during training.

Step 2

In the standard Perceptron algorithm, when a point (x_i, y_i) is misclassified, we update:

$$w \leftarrow w + y_i x_i$$

In the dual form, this corresponds to incrementing α_i by 1:

$$\alpha_i \leftarrow \alpha_i + 1$$

Initially, all α_i values are set to 0. Each time a point is misclassified, its corresponding α_i is incremented by 1.

Step 3

Since the algorithm performs k updates in total, and each update increments exactly one α_i by 1, we know that:

- Each α_i represents the number of times example i was misclassified
- Each α_i must be a non-negative integer
- Some examples might never be misclassified, so their α_i remains 0
- Some examples might be misclassified multiple times, so their α_i could be greater than 1

$\therefore$ The statement "Each α_i is either 0 or 1" is **possibly false**. While some α_i values may be 0 (never misclassified) or 1 (misclassified once), it's possible for an example to be misclassified multiple times during training, resulting in $\alpha_i > 1$.

Solution 6 (b)

Step 1

We know that the Perceptron algorithm makes a total of k updates, and each update corresponds to incrementing exactly one α_i by 1.

Step 2

Initially, all α_i values are set to 0. After k updates, each incrementing one α_i by 1, the sum of all α_i values must be equal to k:

$$\sum_{i=1}^{n} \alpha_i = k$$

This is because each update contributes exactly 1 to the sum of α_i values, and there are k updates in total.

$\therefore$ The statement "$\sum_i \alpha_i = k$" is **necessarily true**. Since each of the k updates increments exactly one α_i by 1, and all α_i values start at 0, the sum of all α_i values must equal k.

Solution 6 (c)

Step 1

We need to determine the maximum number of nonzero coordinates in the vector α.

Step 2

A coordinate α_i is nonzero if and only if the corresponding example (x_i, y_i) has been misclassified at least once during training.
In the worst case, each of the k updates could be applied to a different example, resulting in k different examples having their α_i incremented to 1.
However, it's also possible that some examples are misclassified multiple times, which would result in fewer than k nonzero coordinates in α.

Step 3

Let's consider a simple example to illustrate:
Suppose we have 5 training examples and the algorithm makes $k = 3$ updates:

- If the updates are applied to examples 1, 2, and 3, then $\alpha = (1, 1, 1, 0, 0)$ with 3 nonzero coordinates.
- If the updates are applied to examples 1, 1, and 2, then $\alpha = (2, 1, 0, 0, 0)$ with 2 nonzero coordinates.

In general, the number of nonzero coordinates in α is at most k (when each update is applied to a different example) and at least 1 (when all updates are applied to the same example).

$\therefore$ The statement "α has at most k nonzero coordinates" is **necessarily true**. Since there are k updates in total, at most k different examples can be misclassified, resulting in at most k nonzero α_i values.

Solution 6 (d)

Step 1

We need to determine whether the convergence of the Perceptron algorithm implies that the training data is linearly separable.

Step 2

Recall the Perceptron Convergence Theorem: If the training data is linearly separable, then the Perceptron algorithm will converge in a finite number of updates.
The converse is also true: If the Perceptron algorithm converges, then the training data must be linearly separable. This is because:

- Convergence means the algorithm has found a weight vector w and bias b such that all training examples are correctly classified.
- This means there exists a hyperplane defined by $w \cdot x + b = 0$ that separates the positive and negative examples.
- By definition, this makes the data linearly separable.

Step 3

It's important to note that if the data is not linearly separable, the Perceptron algorithm will never converge - it will continue to make updates indefinitely as it cycles through the data.
Since we're told that the algorithm converges after k updates, this implies that the algorithm has found a separating hyperplane, which means the data must be linearly separable.

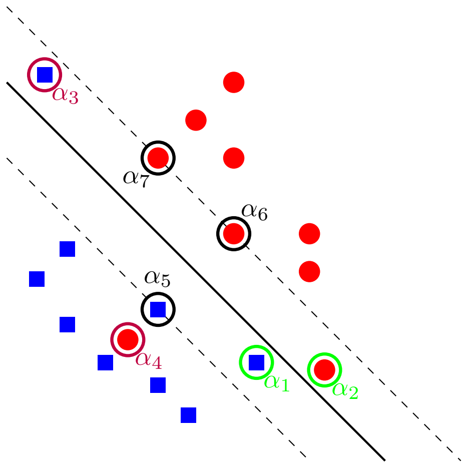
$\therefore$ The statement "The training data must be linearly separable" is **necessarily true**. The convergence of the Perceptron algorithm implies that it has found a hyperplane that correctly classifies all training examples, which by definition means the data is linearly separable.

Solution 5 (a)

Step 1

Support vectors(α_i)are the training points that lie exactly on the margin. However, we are told the decision boundary was obtained upon running *soft-margin SVM*. The soft-margin support vectors include the following points:

- points inside margin
- points that are misclassified
- points on the margin



Step 2

The slack variables for the points circled are as follows:

- points inside margin: α_1 and α_2 with slack variable values of approximately 0.5
- points that are misclassified: α_3 and α_4 with slack variable values of $\alpha_3 \approx 1.5$ and $\alpha_4 \approx 2.5$
- points on the margin: α_5, α_6, and α_7 with slack variable values of approximately 0

$\therefore$ the support vectors have been marked in **Step 1** and corresponding slack variable values have been indicated in **Step 2**.

Solution 5 (b)

Step 1

We know that increasing or decreasing the factor C or penalty weight has the following effects:

- decreasing C the margin will grow in size and will allow for more misclassified points
- increasing C the margin will shrink in size and fewer mistakes will be allowed

$\therefore$ increasing the factor C will shrink the margin and fewer mistakes will be allowed.

Solution 6 (a)

∴ The statement: *Each α_i is either 0 or 1.* is **possibly false**. A training point may be misclassified more than once, in which case its corresponding α_i would be incremented multiple times and exceed 1.

Solution 6 (b)

∴ The statement: $\sum_i \alpha_i = k$ is **definitely true**. The algorithm performs k total updates, and each update increases one of the α_i by exactly 1, so the sum of all α_i must equal k.

Solution 6 (c)

∴ The statement: *α has at most k nonzero coordinates.* is **definitely true**. Since only the points that were misclassified at least once are updated, and there are k updates in total, at most k different α_i can be nonzero.

Solution 6 (d)

∴ The statement: *The training data must be linearly separable.* is **definitely true**. The Perceptron algorithm is guaranteed to converge only when the data is linearly separable. Since it converged in k updates, the data must be separable.
