

Lecture 16: 19 March, 2026

Madhavan Mukund

<https://www.cmi.ac.in/~madhavan>

Data Mining and Machine Learning
January–April 2026

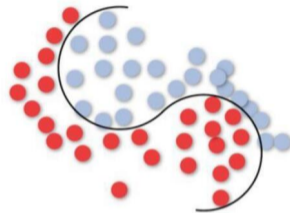
A geometric view of supervised learning

- Think of data as points in space
- Find a separating curve (surface)

Images - $N \times M$ grid
of coloured points

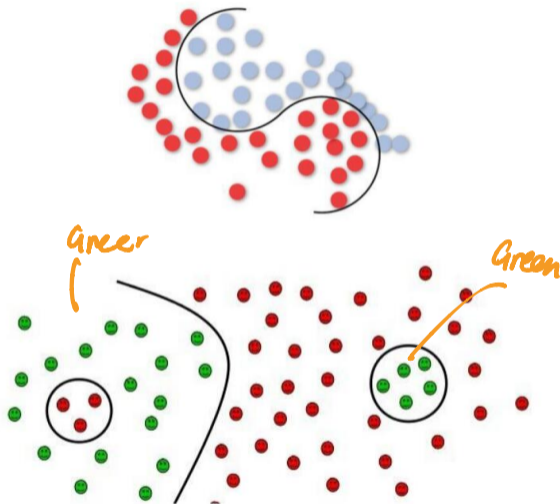
A geometric view of supervised learning

- Think of data as points in space
- Find a separating curve (surface)
- Separable case
 - Each class is a connected region
 - A single curve can separate them



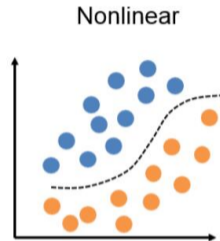
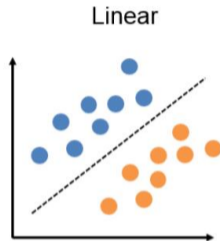
A geometric view of supervised learning

- Think of data as points in space
- Find a separating curve (surface)
- Separable case
 - Each class is a connected region
 - A single curve can separate them
- More complex scenario
 - Classes form multiple connected regions
 - Need multiple separators



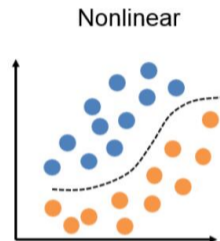
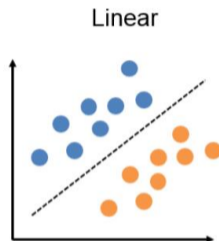
Linear separators

- Simplest case — linearly separable data



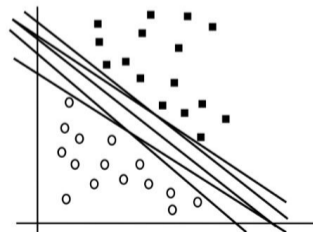
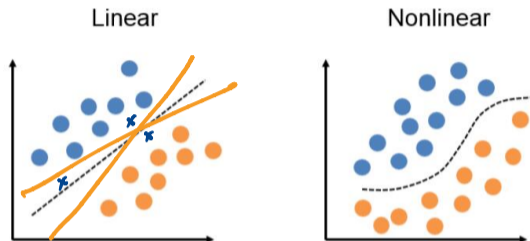
Linear separators

- Simplest case — linearly separable data
- Dual of linear regression
 - Find a line that passes close to a set of points
 - Find a line that separates the two sets of points



Linear separators

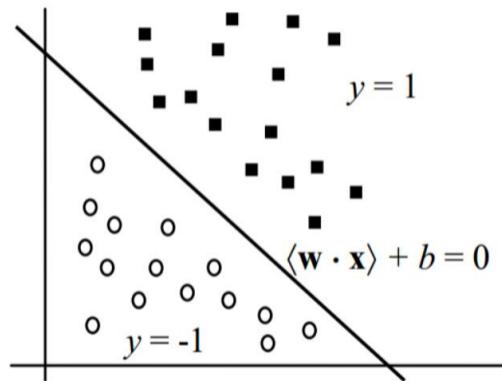
- Simplest case — linearly separable data
- Dual of linear regression
 - Find a line that passes close to a set of points
 - Find a line that separates the two sets of points
- Many lines are possible
 - How do we find the best one?
 - What is a good notion of “cost” to optimize?



Linear separators

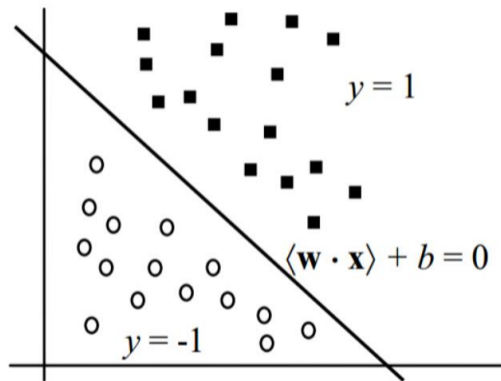
- Each input x has n attributes

$$\langle x_1, x_2, \dots, x_n \rangle$$



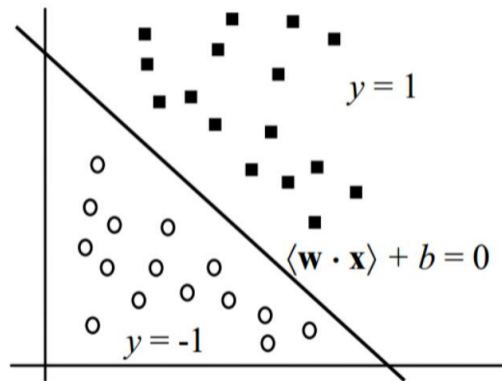
Linear separators

- Each input x has n attributes
 $\langle x_1, x_2, \dots, x_n \rangle$
- Linear separator has the form
 $w_1x_1 + w_2x_2 + \dots + w_nx_n + b$



Linear separators

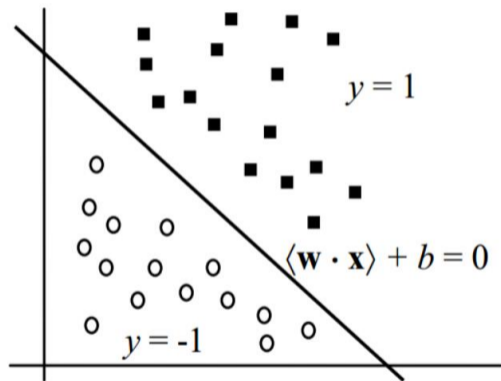
- Each input x has n attributes
 $\langle x_1, x_2, \dots, x_n \rangle$
- Linear separator has the form
 $w_1x_1 + w_2x_2 + \dots + w_nx_n + b$
- Classification criterion
 - $w_1x_1 + w_2x_2 + \dots + w_nx_n + b > 0$,
classify yes, $+1$
 - $w_1x_1 + w_2x_2 + \dots + w_nx_n + b < 0$,
classify no, -1



Linear separators

- Dot product $w \cdot x$

$$\langle w_1, w_2, \dots, w_n \rangle \cdot \langle x_1, x_2, \dots, x_n \rangle = w_1x_1 + w_2x_2 + \dots + w_nx_n$$



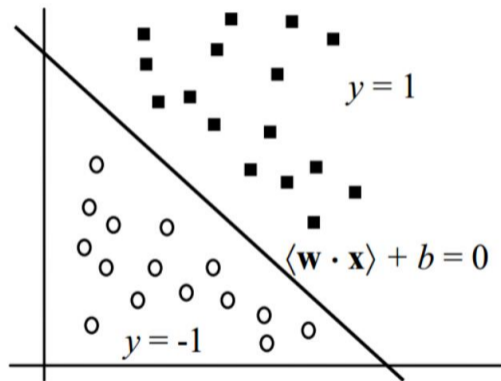
Linear separators

- Dot product $w \cdot x$

$$\langle w_1, w_2, \dots, w_n \rangle \cdot \langle x_1, x_2, \dots, x_n \rangle = w_1x_1 + w_2x_2 + \dots + w_nx_n$$

- Collapsed form

$$w \cdot x + b > 0, w \cdot x + b < 0$$



Linear separators

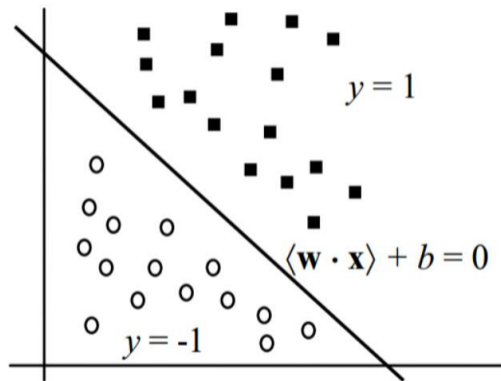
- Dot product $w \cdot x$

$$\langle w_1, w_2, \dots, w_n \rangle \cdot \langle x_1, x_2, \dots, x_n \rangle = w_1x_1 + w_2x_2 + \dots + w_nx_n$$

- Collapsed form

$$w \cdot x + b > 0, w \cdot x + b < 0$$

- Rename bias b as w_0 , create fictitious $x_0 = 1$



Linear separators

- Dot product $w \cdot x$

$$\langle w_1, w_2, \dots, w_n \rangle \cdot \langle x_1, x_2, \dots, x_n \rangle = w_1x_1 + w_2x_2 + \dots + w_nx_n$$

- Collapsed form

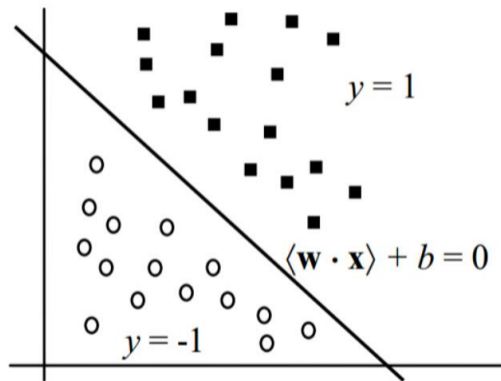
$$w \cdot x + b > 0, w \cdot x + b < 0$$

- Rename bias b as w_0 , create fictitious $x_0 = 1$

- Classification criteria become

$$w \cdot x > 0, w \cdot x < 0$$

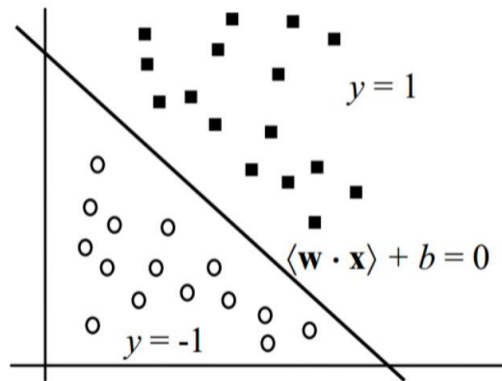
↑
Find w



Perceptron algorithm

(Frank Rosenblatt, 1958)

- Each training input is (x_i, y_i) , where $x_i = \langle x_{i_1}, x_{i_2}, \dots, x_{i_n} \rangle$ and $y_i = +1$ or -1
- Need to find $w = \langle w_0, w_1, \dots, w_n \rangle$
 - Recall $x_{i_0} = 1$, always



Perceptron algorithm

(Frank Rosenblatt, 1958)

- Each training input is (x_i, y_i) , where $x_i = \langle x_{i_1}, x_{i_2}, \dots, x_{i_n} \rangle$ and $y_i = +1$ or -1
- Need to find $w = \langle w_0, w_1, \dots, w_n \rangle$
 - Recall $x_{i_0} = 1$, always

Initialize $w = \langle 0, 0, \dots, 0 \rangle$

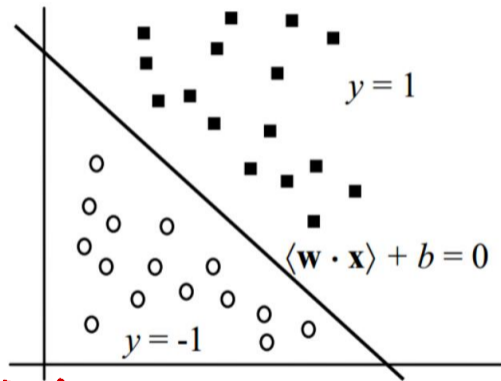
While there exists x_i, y_i such that

$y_i = +1$ and $w \cdot x_i < 0$, or

$y_i = -1$ and $w \cdot x_i > 0$

Update w to $w + x_i y_i$

Assume linearly separable



Correct

Wrong $w \cdot x_i - y_i > 0$

Wrong $w \cdot x_i - y_i < 0$

Perceptron algorithm

(Frank Rosenblatt, 1958)

- Each training input is (x_i, y_i) , where $x_i = \langle x_{i_1}, x_{i_2}, \dots, x_{i_n} \rangle$ and $y_i = +1$ or -1
- Need to find $w = \langle w_0, w_1, \dots, w_n \rangle$
 - Recall $x_{i_0} = 1$, always

Initialize $w = \langle 0, 0, \dots, 0 \rangle$

While there exists x_i, y_i such that

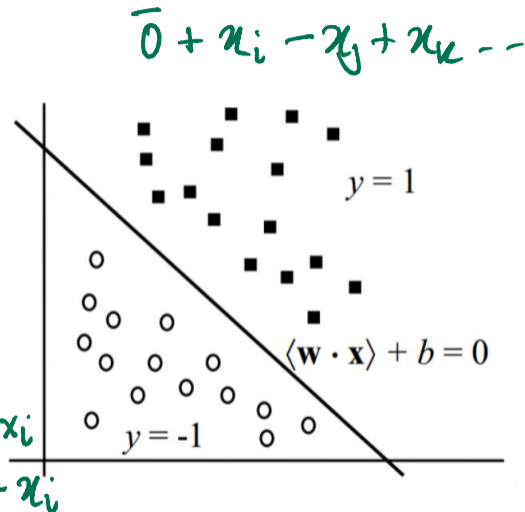
IF $y_i = +1$ and $w \cdot x_i < 0$, or

IF $y_i = -1$ and $w \cdot x_i > 0$

Update w to $w + x_i y_i$

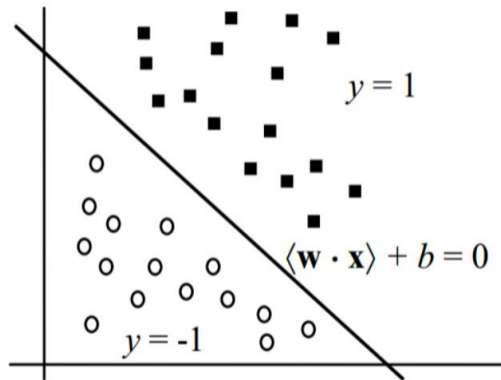
$w \leftarrow w + x_i$

$w \leftarrow w - x_i$



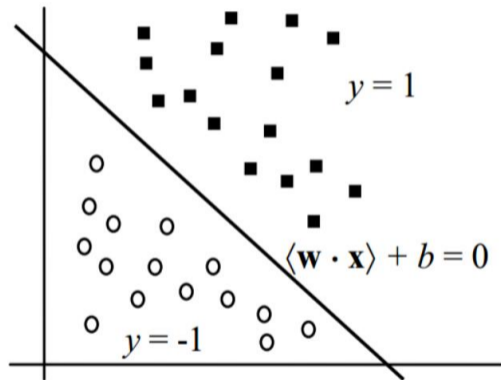
Perceptron algorithm ...

- Keep updating w as long as some training data item is misclassified
- Update is an offset by misclassified input



Perceptron algorithm ...

- Keep updating w as long as some training data item is misclassified
- Update is an offset by misclassified input
- Need not stabilize, potentially an infinite loop

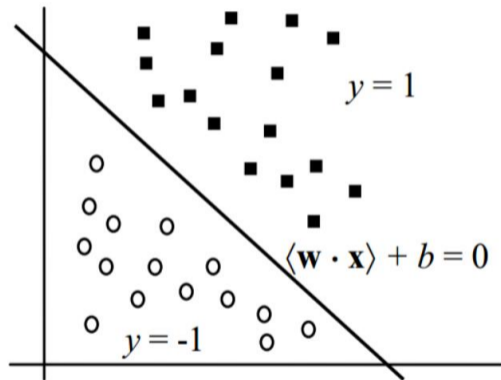


Perceptron algorithm ...

- Keep updating w as long as some training data item is misclassified
- Update is an offset by misclassified input
- Need not stabilize, potentially an infinite loop

Theorem

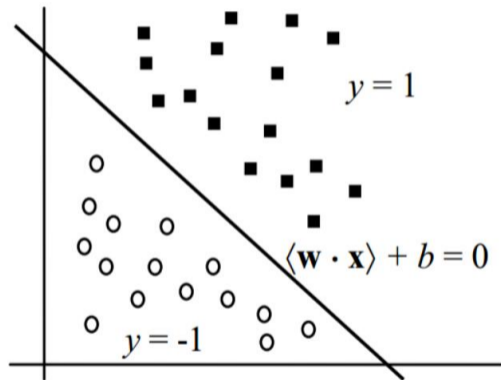
If the points are linearly separable, the Perceptron algorithm always terminates with a valid separator



Perceptron algorithm ...

Theorem

If the points are linearly separable, the Perceptron algorithm always terminates with a valid separator

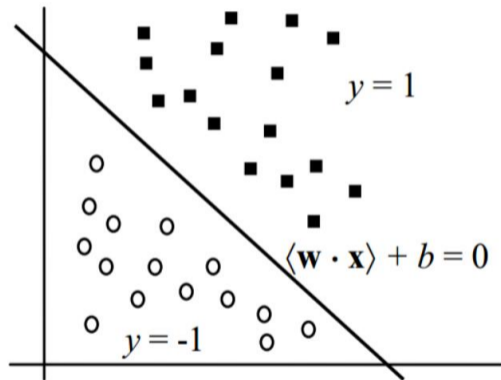


Perceptron algorithm ...

Theorem

If the points are linearly separable, the Perceptron algorithm always terminates with a valid separator

- Termination time depends on two factors

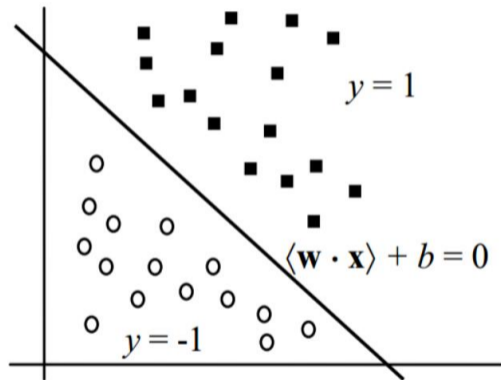


Perceptron algorithm ...

Theorem

If the points are linearly separable, the Perceptron algorithm always terminates with a valid separator

- Termination time depends on two factors
 - Width of the band separating the positive and negative points
 - Narrow band takes longer to converge

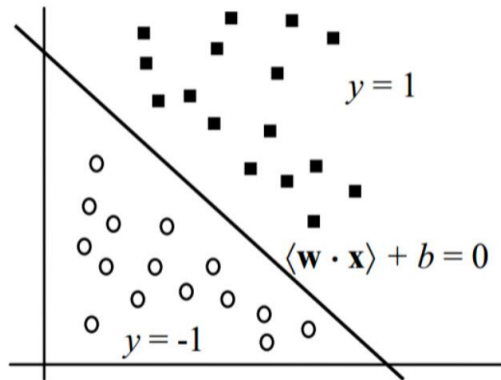


Perceptron algorithm ...

Theorem

If the points are linearly separable, the Perceptron algorithm always terminates with a valid separator

- Termination time depends on two factors
 - Width of the band separating the positive and negative points
 - Narrow band takes longer to converge
 - Magnitude of the x values
 - Larger spread of points takes longer to converge



Perceptron Algorithm — Proof

Theorem

If there is w^* satisfying $(w^* \cdot x_i)y_i \geq 1$ for all i , then the Perceptron Algorithm finds a solution w with $(w \cdot x_i)y_i > 0$ for all i in at most $r^2 |w^*|^2$ updates, where

$$r = \max_i |x_i|.$$

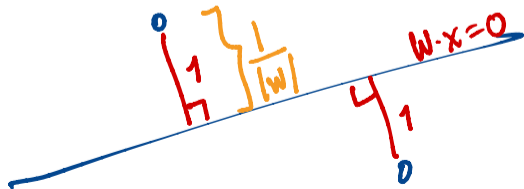
some solution

"best" solution

$$Width = \frac{1}{|w^*|}$$

Scale so that

$$w \cdot x > 1$$
$$w \cdot x \leq -1$$



Perceptron Algorithm — Proof

Theorem

If there is w^* satisfying $(w^* \cdot x_i)y_i \geq 1$ for all i , then the Perceptron Algorithm finds a solution w with $(w \cdot x_i)y_i > 0$ for all i in at most $r^2|w^*|^2$ updates, where $r = \max_i |x_i|$.

- Assume w^* exists. Keep track of two quantities: $w^T w^*$, $|w|^2$.

$w_0 \rightarrow w_1 \rightarrow$
 $w_2 \rightarrow \dots$

current guess hypothetical best

Perceptron Algorithm — Proof

Theorem

If there is w^* satisfying $(w^* \cdot x_i)y_i \geq 1$ for all i , then the Perceptron Algorithm finds a solution w with $(w \cdot x_i)y_i > 0$ for all i in at most $r^2|w^*|^2$ updates, where $r = \max_i |x_i|$.

- Assume w^* exists. Keep track of two quantities: $\underline{w^T w^*}$, $|w|^2$.
- Each update increases $w^T w^*$ by at least 1.

$$\underline{(w + x_i y_i)^T w^*} = \underline{w^T w^*} + \underline{x_i^T y_i w^*} \geq w^T w^* + 1$$

"wrong" answer

Perceptron Algorithm — Proof

Theorem

If there is w^* satisfying $(w^* \cdot x_i)y_i \geq 1$ for all i , then the Perceptron Algorithm finds a solution w with $(w \cdot x_i)y_i > 0$ for all i in at most $r^2|w^*|^2$ updates, where $r = \max_i |x_i|$.

- Assume w^* exists. Keep track of two quantities: $w^T w^*$, $|w|^2$.

- Each update increases $w^T w^*$ by at least 1.

$$(w + x_i y_i)^T w^* = w^T w^* + \underbrace{x_i^T y_i w^*}_{\geq 1} \geq w^T w^* + 1$$

- Each update increases $|w|^2$ by at most r^2

$$(w + x_i y_i)^T (w + x_i y_i) = |w|^2 + \underbrace{2x_i^T y_i w}_{\leq 0} + \underbrace{|x_i y_i|^2}_{\leq r^2} \leq |w|^2 + \underbrace{|x_i|^2}_{\leq r^2} \leq |w|^2 + r^2$$

- Note that we update only when $x_i^T y_i w < 0$

Perceptron Algorithm — Proof (cont'd)

- Assume Perceptron Algorithm makes m updates

Perceptron Algorithm — Proof (cont'd)

- Assume Perceptron Algorithm makes m updates

- Then, $w^\top w^* \geq m$, $|w|^2 \leq mr^2$
Fact 1 Fact 2

Perceptron Algorithm — Proof (cont'd)

- Assume Perceptron Algorithm makes m updates
- Then, $w^\top w^* \geq m$, $|w|^2 \leq mr^2$
- $m \leq |w||w^*|$, because $a \cdot b = |a||b| \cos \theta$

Perceptron Algorithm — Proof (cont'd)

- Assume Perceptron Algorithm makes m updates
- Then, $w^\top w^* \geq m$, $|w|^2 \leq mr^2$
- $$m \leq |w||w^*|$$
$$m/|w^*| \leq |w|$$

Perceptron Algorithm — Proof (cont'd)

- Assume Perceptron Algorithm makes m updates

- Then, $w^\top w^* \geq m$, $|w|^2 \leq mr^2$

$$|w| \leq r\sqrt{m}$$

- $m \leq |w||w^*|$

$$m/|w^*| \leq |w|$$

$$m/|w^*| \leq r\sqrt{m}, \text{ because } |w|^2 \leq mr^2$$

Perceptron Algorithm — Proof (cont'd)

- Assume Perceptron Algorithm makes m updates

- Then, $w^\top w^* \geq m$, $|w|^2 \leq mr^2$

- $m \leq |w||w^*|$

$$m/|w^*| \leq |w|$$

$$m/|w^*| \leq r\sqrt{m}$$

$$\sqrt{m} \leq r|w^*|$$

Perceptron Algorithm — Proof (cont'd)

- Assume Perceptron Algorithm makes m updates

- Then, $w^\top w^* \geq m$, $|w|^2 \leq mr^2$

- $m \leq |w||w^*|$

$$m/|w^*| \leq |w|$$

$$m/|w^*| \leq r\sqrt{m}$$

$$\sqrt{m} \leq r|w^*|$$

$$m \leq r^2|w^*|^2$$

Perceptron Algorithm — Proof (cont'd)

- Assume Perceptron Algorithm makes m updates

- Then, $w^\top w^* \geq m$, $|w|^2 \leq mr^2$

- $m \leq |w||w^*|$

$$m/|w^*| \leq |w|$$

$$m/|w^*| \leq r\sqrt{m}$$

$$\sqrt{m} \leq r|w^*|$$

$$m \leq r^2|w^*|^2$$

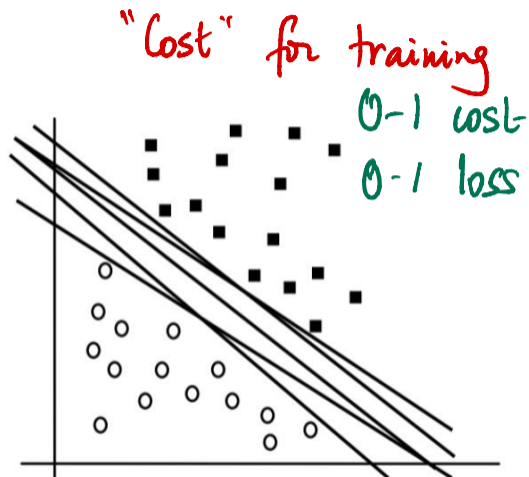
$$\vec{0} + x_i - x_j + x_i - x_k \dots$$

Finally $w \cdot x$
to classify x
 $(\sum n_i x_i) \cdot x$

- Note (for later) that final w is of the form $\sum_i n_i x_i$

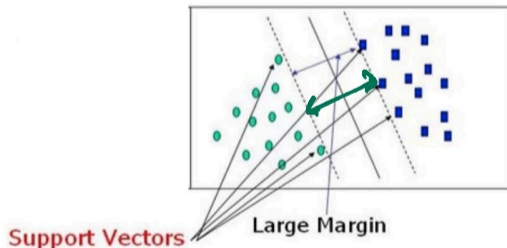
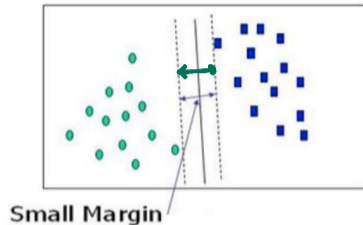
Linear separators

- Simplest case — linearly separable data
- Perceptron algorithm is a simple procedure to find a linear separator, if one exists
- Many lines are possible
 - Does the Perceptron algorithm find the best one?
 - What is a good notion of “cost” to optimize?



Margin

- Each separator defines a margin
 - Empty corridor separating the points
 - Separator is the centre line of the margin
- Wider margin makes for a more robust classifier
 - More gap between the classes
- Optimum classifier is one that maximizes the width of its margin
- Margin is defined by the training data points on the boundary
 - Support vectors



Finding a maximum margin classifier

- Recall our original linear classifier

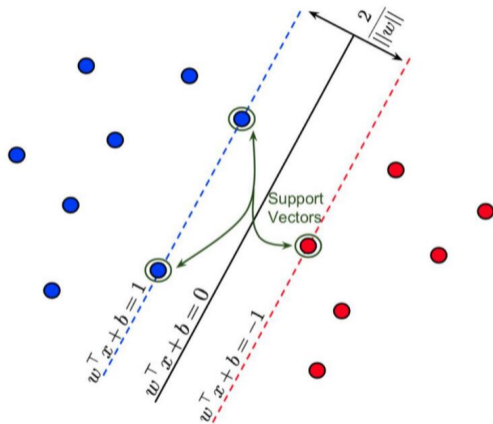
$w_1x_1 + w_2x_2 + \dots + w_nx_n + b > 0$, classify
yes, $+1$

$w_1x_1 + w_2x_2 + \dots + w_nx_n + b < 0$, classify
no, -1

- Scale margin so that separation is 1 on
either side

$w_1x_1 + w_2x_2 + \dots + w_nx_n + b > 1$, classify
yes, $+1$

$w_1x_1 + w_2x_2 + \dots + w_nx_n + b < -1$, classify
no, -1

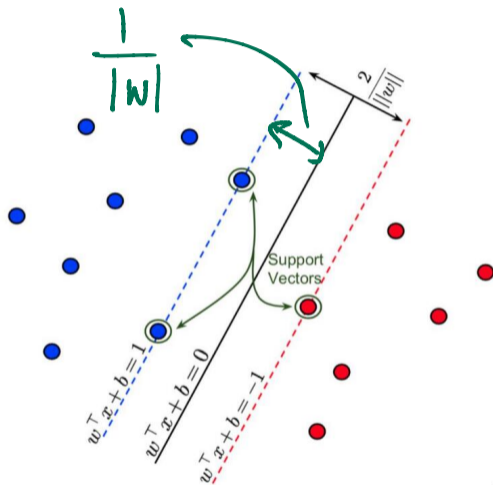


Finding a maximum margin classifier

- Scale margin so that separation is 1 on either side
 $w_1x_1 + w_2x_2 + \dots + w_nx_n + b > 1$, classify yes, +1
 $w_1x_1 + w_2x_2 + \dots + w_nx_n + b < -1$, classify no, -1

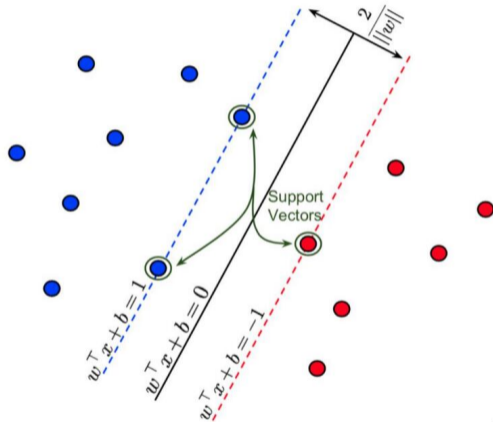
- Using Pythagoras's theorem, perpendicular distance to nearest support vector is $\frac{1}{|w|}$, where

$$|w| = \sqrt{w_1^2 + w_2^2 + \dots + w_n^2}$$



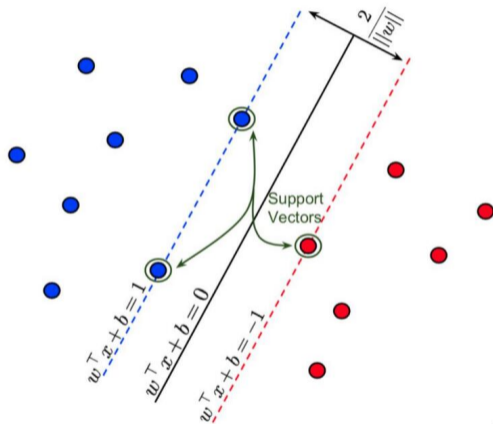
Optimization problem

- Want to maximize the overall margin $\frac{2}{\|w\|}$



Optimization problem

- Want to maximize the overall margin $\frac{2}{\|w\|}$
- Equivalently, minimize $\frac{\|w\|}{2}$



Optimization problem

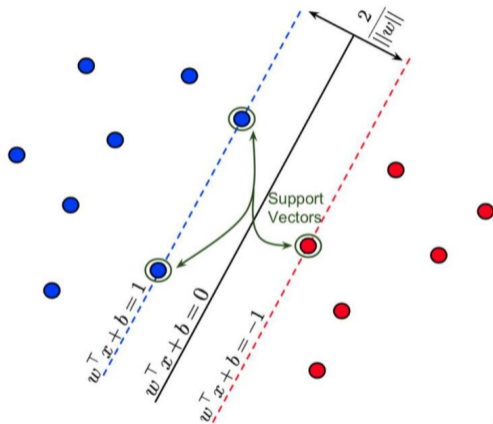
- Want to maximize the overall margin $\frac{2}{|w|}$

- Equivalently, minimize $\frac{|w|}{2}$

- Also, w should classify each (x_i, y_i) correctly

$$w_1x_1^i + w_2x_2^i + \cdots w_nx_n^i + b > 1, \\ \text{if } y_i = 1$$

$$w_1x_1^i + w_2x_2^i + \cdots w_nx_n^i + b < -1, \\ \text{if } y_i = -1$$



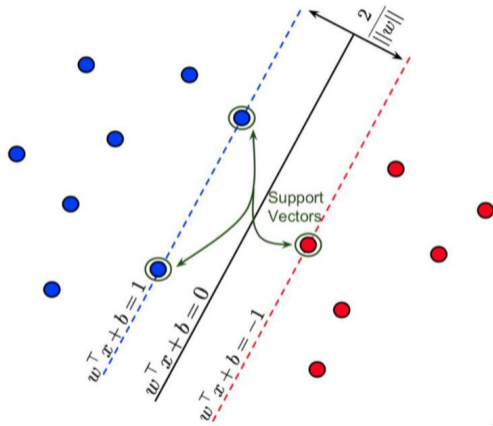
Optimization problem

$$\text{Minimize } \frac{|w|}{2}$$

Subject to

$$w_1 x_1^i + w_2 x_2^i + \dots + w_n x_n^i + b > 1, \text{ if } y_i = 1$$

$$w_1 x_1^i + w_2 x_2^i + \dots + w_n x_n^i + b < -1, \text{ if } y_i = -1$$



Optimization problem

$$\text{Minimize } \frac{|w|}{2}$$

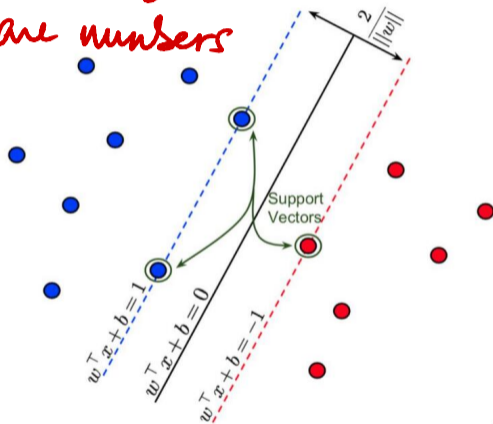
Subject to

$$w_1 x_1^i + w_2 x_2^i + \dots + w_n x_n^i + b > 1, \text{ if } y_i = 1$$

$$w_1 x_1^i + w_2 x_2^i + \dots + w_n x_n^i + b < -1, \text{ if } y_i = -1$$

- The constraints are linear

Unknowns are w_i
 x_i, y_i are numbers



Optimization problem

$$\text{Minimize } \frac{|w|}{2}$$

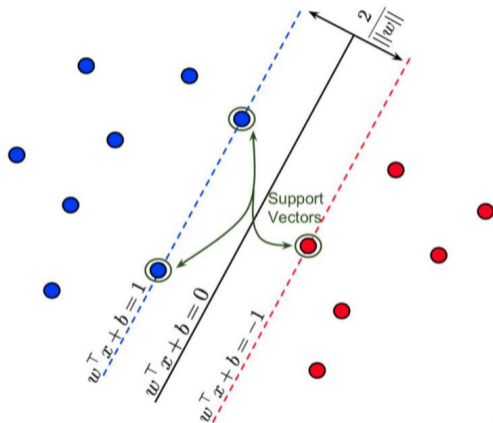
Subject to

$$w_1x_1^i + w_2x_2^i + \cdots + w_nx_n^i + b > 1, \text{ if } y_i = 1$$

$$w_1x_1^i + w_2x_2^i + \cdots + w_nx_n^i + b < -1, \text{ if } y_i = -1$$

- The constraints are linear
- The objective function is not linear

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



Optimization problem

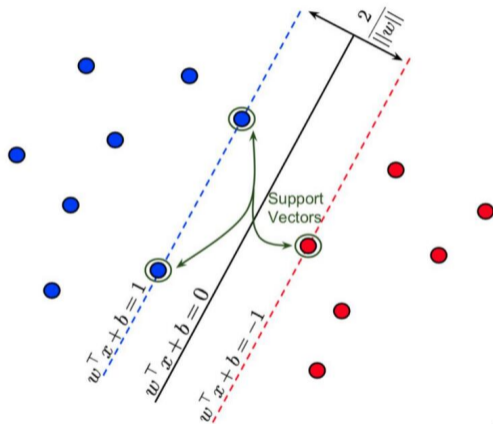
$$\text{Minimize } \frac{|w|}{2}$$

Subject to

$$w_1x_1^i + w_2x_2^i + \cdots + w_nx_n^i + b > 1, \text{ if } y_i = 1$$

$$w_1x_1^i + w_2x_2^i + \cdots + w_nx_n^i + b < -1, \text{ if } y_i = -1$$

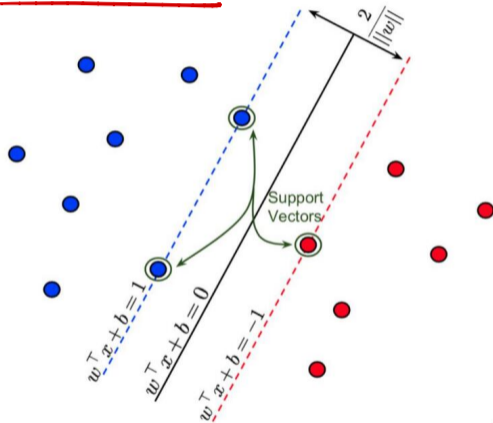
- The constraints are linear
- The objective function is not linear
 $|w| = \sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}$
- This is a **quadratic optimization problem**, not linear programming



Solution to optimization problem

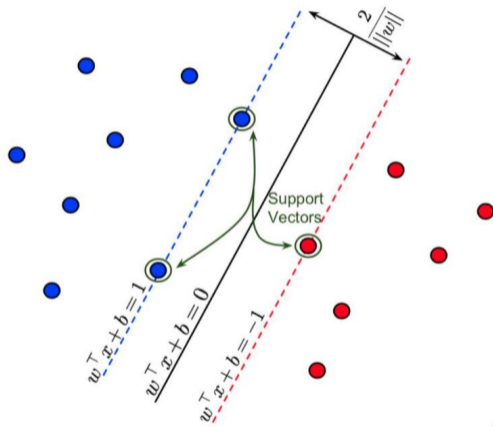
- Convex optimization theory
- Can be solved using computational techniques

KKT conditions



Solution to optimization problem

- Convex optimization theory
- Can be solved using computational techniques
- Solution expressed in terms of Lagrange multipliers $\alpha_1, \alpha_2, \dots, \alpha_N$, one multiplier per training input
- α_j is non-zero iff x_j is a support vector

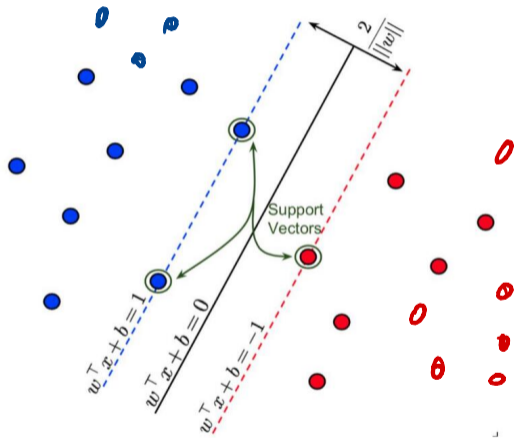


Solution to optimization problem

- Convex optimization theory
- Can be solved using computational techniques
- Solution expressed in terms of Lagrange multipliers $\alpha_1, \alpha_2, \dots, \alpha_N$, one multiplier per training input
- α_i is non-zero iff x_i is a support vector
- Final classifier for new input z

$$\text{sign} \left[\sum_{i \in \text{sv}} y_i \alpha_i (x_i \cdot z) + b \right]$$

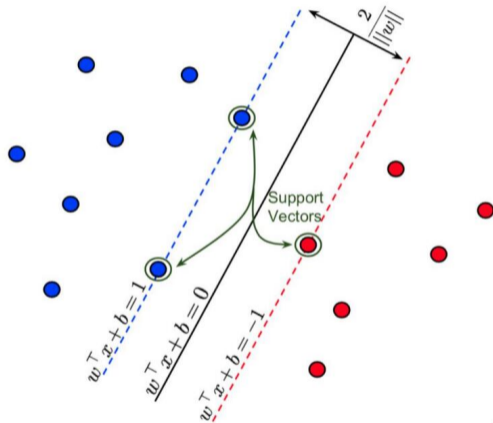
- sv is set of support vectors



Support Vector Machine (SVM)

$$\text{sign} \left[\sum_{i \in \text{sv}} y_i \alpha_i (x_i \cdot z) + b \right]$$

- Solution depends only on support vectors
 - If we add more training data away from support vectors, separator does not change

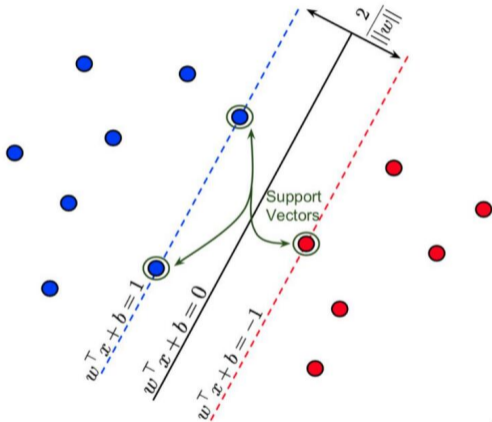


Support Vector Machine (SVM)

$$\text{sign} \left[\sum_{i \in \text{sv}} y_i \alpha_i (x_i \cdot z) + b \right]$$

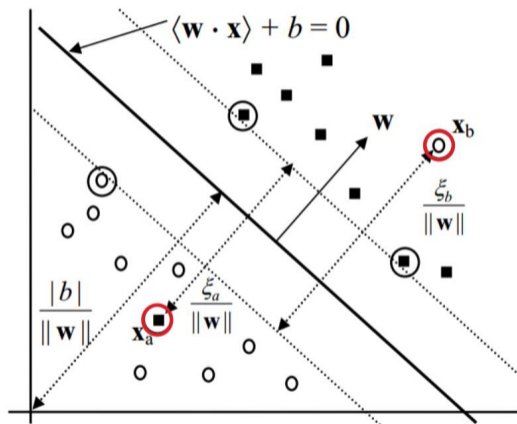
$$\sum n_i x_i$$

- Solution depends only on support vectors
 - If we add more training data away from support vectors, separator does not change
- Solution uses dot product of support vectors with new point
 - Will be used later, in the non-linear case



The non-linear case

- Some points may lie on the wrong side of the classifier
- How do we account for these?



The non-linear case

- Some points may lie on the wrong side of the classifier
- How do we account for these?
- Add an error term to the classifier requirement
- Instead of

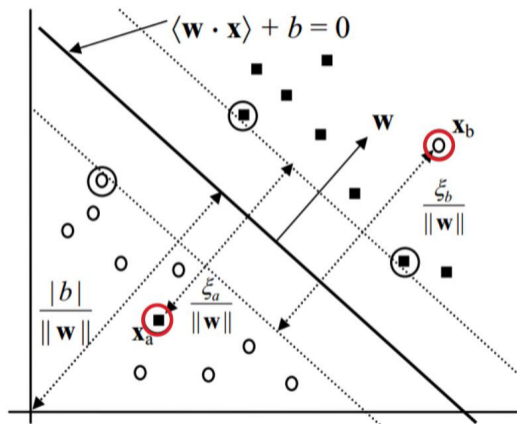
$$w \cdot x + b > 1, \text{ if } y_i = 1$$

$$w \cdot x + b < -1, \text{ if } y_i = -1$$

we have

$$w \cdot x + b > 1 - \xi_i, \text{ if } y_i = 1$$

$$w \cdot x + b < -1 + \xi_i, \text{ if } y_i = -1$$

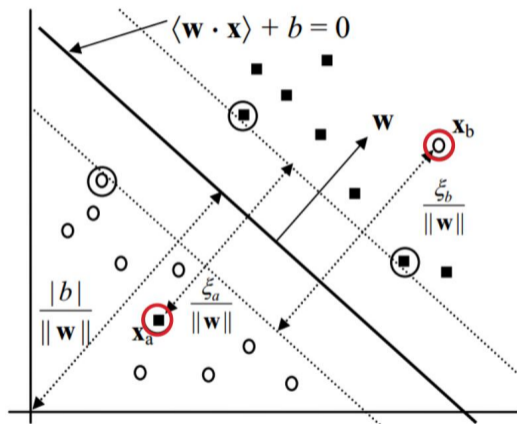


Soft margin classifier

$$w \cdot x + b > 1 - \xi_i, \text{ if } y_i = 1$$

$$w \cdot x + b < -1 + \xi_i, \text{ if } y_i = -1$$

- Error term always non-negative,
- If the point is correctly classified, error term is 0
- **Soft margin** — some points can drift across the boundary
- Need to account for the errors in the objective function
 - Minimize the need for non-zero error terms



Soft margin optimization

Minimize $\frac{|w|}{2} + \sum_{i=1}^N \xi_i^2$

wide margin

ξ_i^2 is arbitrary

Subject to

$\xi_i \geq 0$ *New*

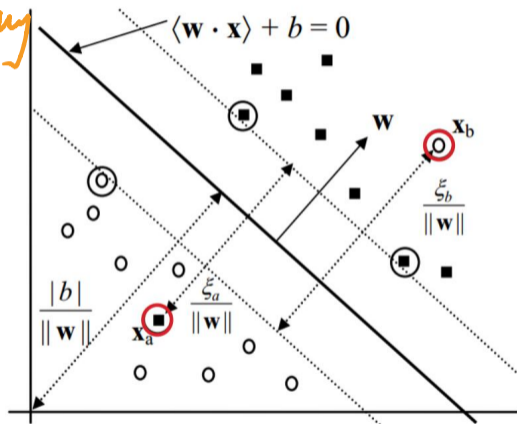
less errors

$w \cdot x_i + b > 1 - \xi_i$, if $y_i = 1$

$w \cdot x_i + b < -1 + \xi_i$, if $y_i = -1$

- Constraints include requirement that error terms are non-negative
- Again the objective function is quadratic

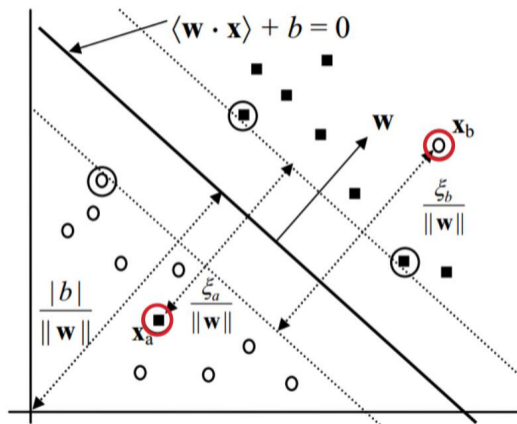
Unknown w_i, ξ_i



Soft margin optimization

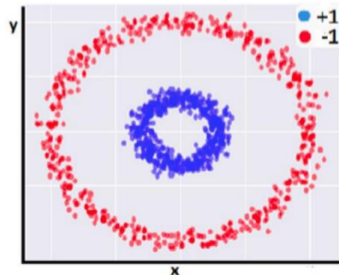
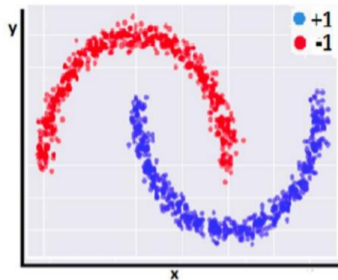
- Can again be solved using convex optimization theory
- Form of the solution turns out to be the same as the hard margin case
 - Expression in terms of Lagrange multipliers α_j
 - Only terms corresponding to support vectors are actively used

$$\text{sign} \left[\sum_{i \in \text{sv}} y_i \alpha_i (x_i \cdot z) + b \right]$$



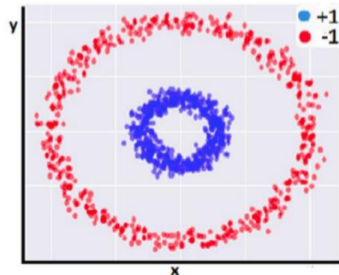
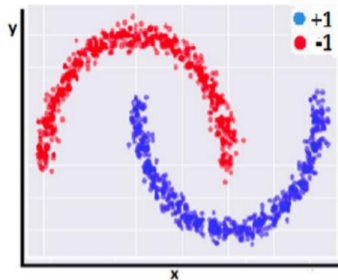
The non-linear case

- How do we deal with datasets where the separator is a complex shape?



The non-linear case

- How do we deal with datasets where the separator is a complex shape?
- Geometrically transform the data
 - Typically, add dimensions



The non-linear case

- How do we deal with datasets where the separator is a complex shape?
- Geometrically transform the data
 - Typically, add dimensions
- For instance, if we can “lift” one class, we can find a planar separator between levels

“Kernel” methods

