CS217: Artificial Intelligence and Machine Learning (associated lab: CS240)

Pushpak Bhattacharyya, Nihar Ranjan Sahoo
CSE Dept.,
IIT Bombay

Week8 of 3mar25, Prolog, Midsem discussion, SVM primal, dual, Kernel Trick

Main points covered: week7 of 17feb25

Inferencing in Predicate Calculus

- Forward chaining
  - Given P, $P \rightarrow Q$, to infer Q
  - P, match $L.H.S$ of
  - Assert Q from $R.H.S$
- Backward chaining
  - Q, Match $R.H.S$ of $P \rightarrow Q$
  - assert P
  - Check if P exists
- Resolution – Refutation
  - Negate goal
  - Convert all pieces of knowledge into clausal form (disjunction of literals)
  - See if contradiction indicated by null clause $\square$ can be derived

Wh-Questions and Knowledge

what

where

who

when

which

Factoid / Declarative

how procedural

why Reasoning

Knowledge Representation of Complex Sentence

- *"In every city there is a thief who is beaten by every policeman in the city"*

$$\forall x[\mathrm{city}(x) \rightarrow \{\exists y((\mathrm{thief}(y) \wedge \mathrm{lives\_in}(y, x)) \wedge \forall z(\mathrm{policeman}(z, x) \rightarrow \mathrm{beaten\_by}(z, y)))\}]$$

Interpretation in Logic

- Logical expressions or formulae are "FORMS" (placeholders) for whom <u>contents</u> are created through interpretation.
- Example:

$$\exists F\left[\{F(a)=b\}\wedge\forall x\{P(x)\rightarrow\left(F(x)=g\left(x,F\left(h(x)\right)\right)\right)\}\right]$$

- This is a Second Order Predicate Calculus formula.
- Quantification on 'F' which is a function.

Examples

- Interpretation:1

  $D=N$ (natural numbers)

  $a = 0$ and $b = 1$

  $x \in N$

  $P(x)$ stands for $x > 0$

  $g(m,n)$ stands for $(m \times n)$

  $h(x)$ stands for $(x - 1)$
- Above interpretation defines **Factorial**

Examples (contd.)

- Interpretation:2

  D={strings)

  $a = b = \lambda$

  $P(x)$ stands for "x is a non empty string"

  $g(m, n)$ stands for "append head of m to n"

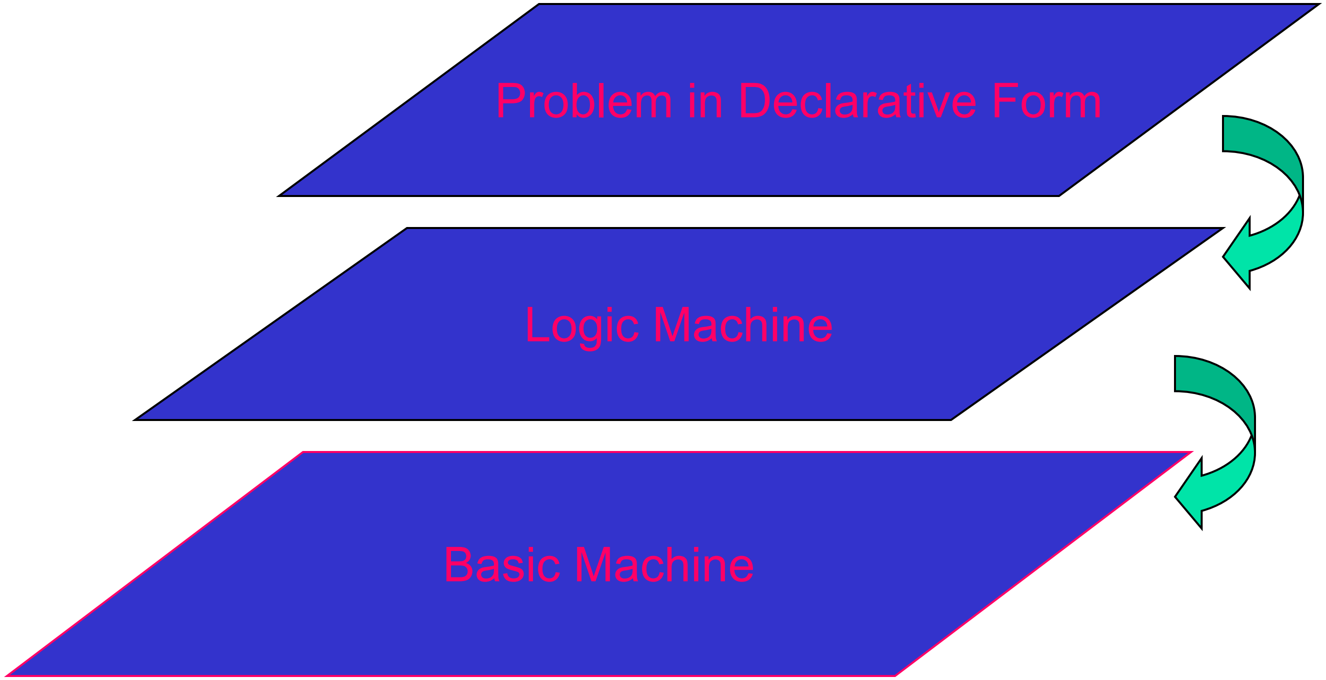
  $h(x)$ stands for $tail(x)$
- Above interpretation defines "reversing a string"

End main points

Prolog

Introduction

- PROgramming in LOGic
- Emphasis on *what* rather than *how*



A Typical Prolog program

Compute_length ([],0).

Compute_length ([Head|Tail], Length):-

Compute_length (Tail,Tail_length),

Length is Tail_length+1.

<u>High level explanation:</u>

The length of a list is 1 plus the length of the tail of the list, obtained by removing the first element of the list.

This is a declarative description of the computation.

Fundamentals

(absolute basics for writing Prolog Programs)

Facts

- *John likes Mary*
  - *like(john,mary)*
- Names of relationship and objects must begin with a lower-case letter.
- Relationship is written *first* (typically the *predicate* of the sentence).
- *Objects* are written separated by commas and are enclosed by a pair of round brackets.
- The full stop character '.' must come at the end of a fact.

More facts

Predicate	**Interpretation**
valuable(gold)	Gold is valuable.
owns(john,gold)	John owns gold.
father(john,mary)	John is the father of Mary
gives (john,book,mary)	John gives the book to Mary

Questions

- *Questions* based on facts
- Answered by *matching*

Two facts *match* if their predicates are same (spelt the same way) and the arguments each are same.

- If matched, prolog answers *yes*, else *no*.
- *No* does not mean falsity.

Prolog does *theorem proving*

- When a question is asked, prolog tries to match *transitively.*
- When no match is found, answer is *no.*
- This means *not provable* from the given facts.

Variables

- Always begin with a capital letter
  - *?- likes (john,X).*
  - *?- likes (john, Something).*
- But *not*
  - *?- likes (john,something)*

Example of usage of variable

Facts:

likes(john,flowers).
likes(john,mary).
likes(paul,mary).

Question:

?- likes(john,X)

Answer:

X=flowers and wait
;
mary
;
no

Conjunctions

- Use ‘,’ and pronounce it as *and*.
- Example
  - Facts:
    - likes(mary,food).
    - likes(mary,tea).
    - likes(john,tea).
    - likes(john,mary)
- ?-
    - likes(mary,X),likes(john,X).
    - Meaning *is anything liked by Mary also liked by John?*

Backtracking *(an inherent property of prolog programming)*

likes(mary,X),likes(john,X)

likes(mary,food)
likes(mary,tea)
likes(john,tea)
likes(john,mary)

1. First goal succeeds. *X=food*
2. Satisfy *likes(john,food)*

Backtracking *(continued)*

Returning to a marked place and trying to resatisfy is called *Backtracking*

likes(mary,X),likes(john,X)

likes(mary,food)
likes(mary,tea)
likes(john,tea)
likes(john,mary)

1. Second goal fails
2. Return to marked place
   and try to resatisfy the first goal

Backtracking *(continued)*

likes(mary,X),likes(john,X)

likes(mary,food)
likes(mary,tea)
likes(john,tea)
likes(john,mary)

1. First goal succeeds again, *X=tea*
2. Attempt to satisfy the *likes(john,tea)*

Backtracking *(continued)*

likes(mary,X),likes(john,X)

likes(mary,food)
likes(mary,tea)
likes(john,tea)
likes(john,mary)

1. Second goal also suceeds
2. Prolog notifies success and waits for a reply

Rules

- Statements about *objects* and their *relationships*
- Expess
  - *If-then conditions*
    - *I use an umbrella if there is a rain*
    - *use(i, umbrella) :- occur(rain).*
  - *Generalizations*
    - *All men are mortal*
    - *mortal(X) :- man(X).*
  - *Definitions*
    - *An animal is a bird if it has feathers*
    - *bird(X) :- animal(X), has_feather(X).*

Syntax

- **<head> :- <body>**
- Read **‘:-’ as ‘if’.**
- E.G.
  - *likes(john,X) :- likes(X,cricket).*
  - *“John likes X if X likes cricket”.*
  - *i.e., “John likes anyone who likes cricket”.*
- Rules always end with ‘.’.

Another Example

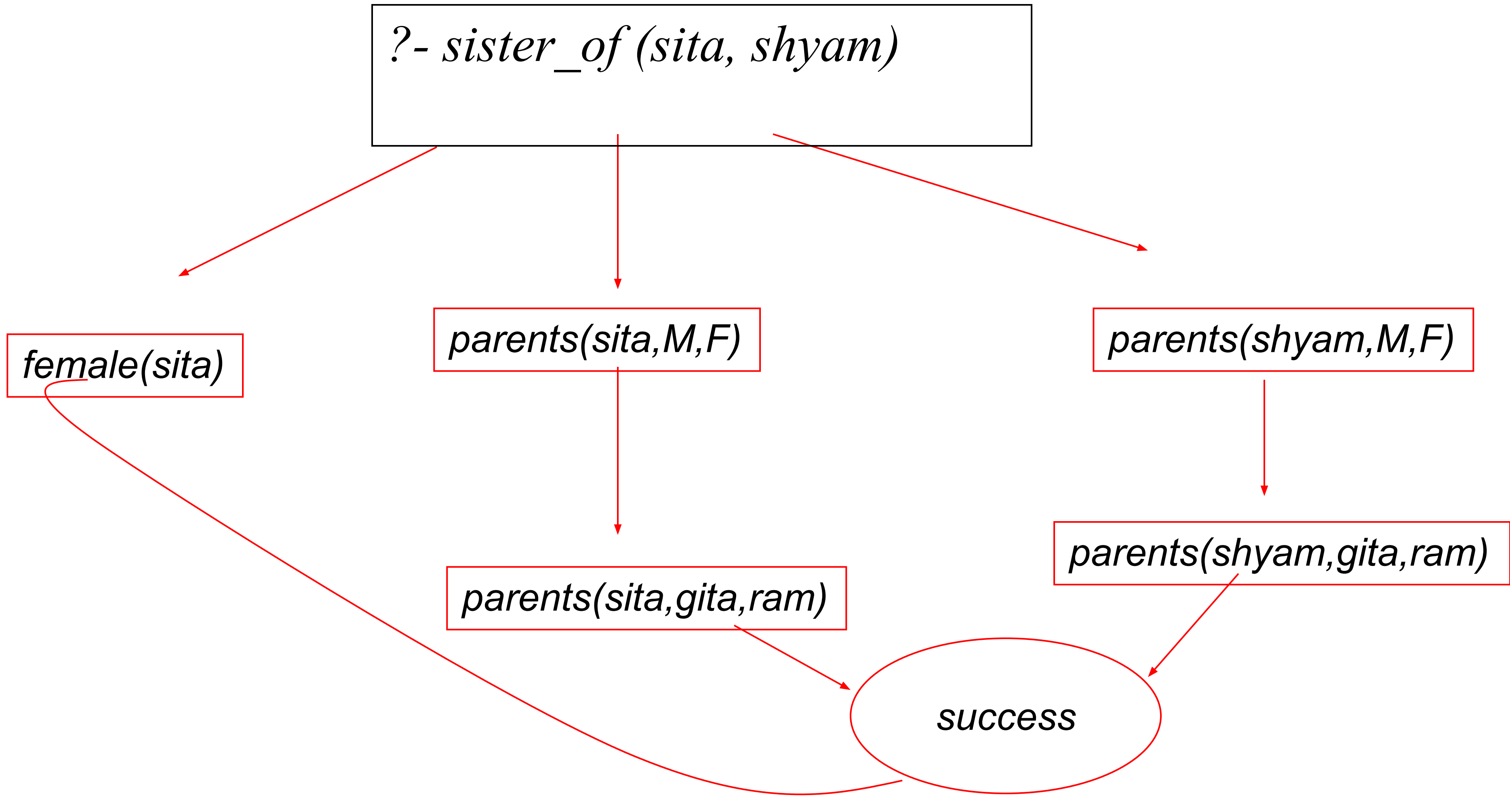
sister_of (X,Y):- female (X),
parents (X, M, F),
parents (Y, M, F).

X is a sister of Y is
X is a female and
X and Y have same parents

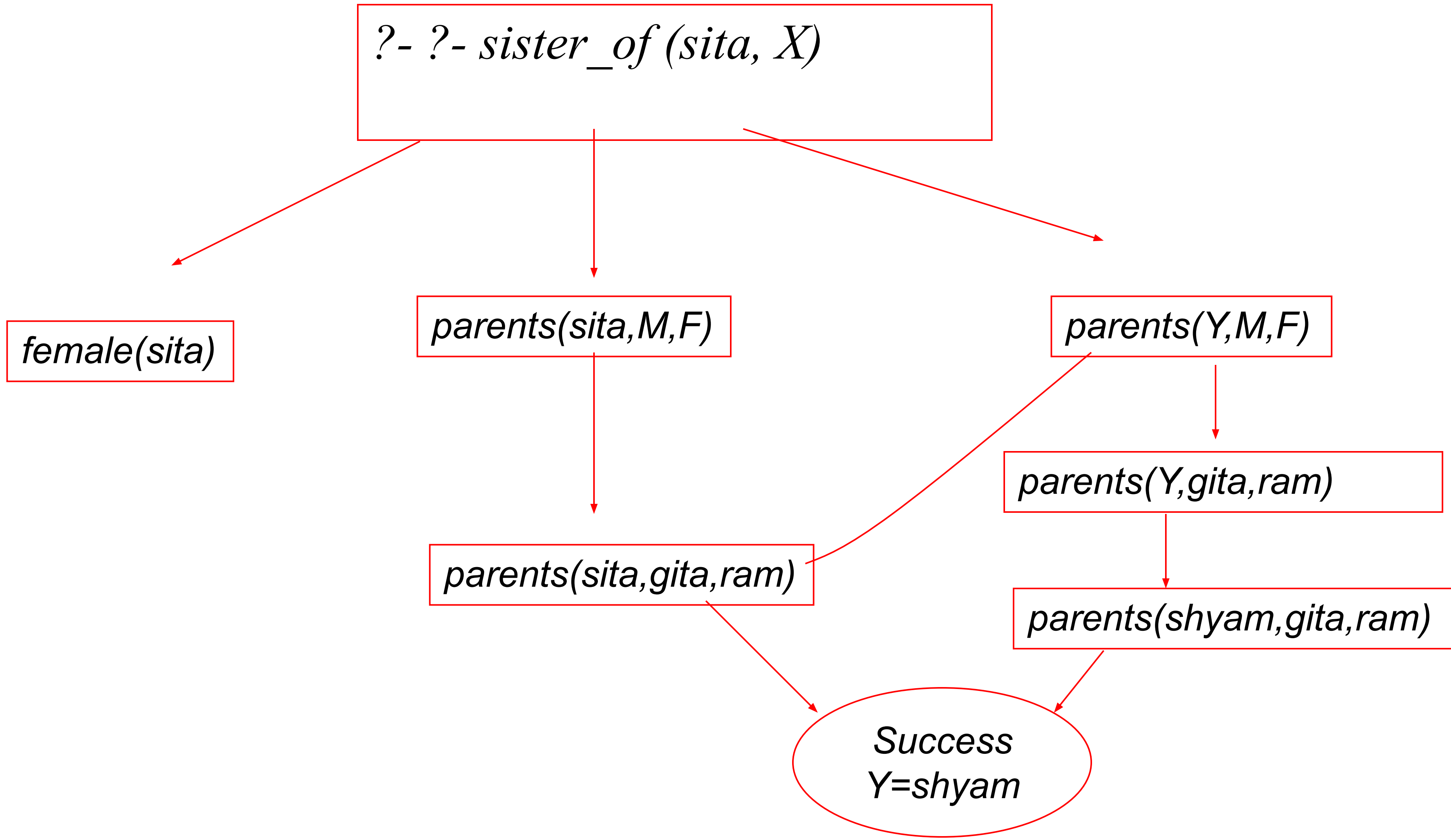
Question Answering in presence of *rules*

- Facts
  - male (ram).
  - male (shyam).
  - female (sita).
  - female (gita).
  - parents (shyam, gita, ram).
  - parents (sita, gita, ram).

Question Answering: Y/N type: *is sita the sister of shyam?*



Question Answering: wh-type: *whose sister is sita?*



Rules

- Statements about *objects* and their *relationships*
- Express
  - *If-then conditions*
    - *I use an umbrella if there is a rain*
    - *use(i, umbrella) :- occur(rain).*
  - *Generalizations*
    - *All men are mortal*
    - *mortal(X) :- man(X).*
  - *Definitions*
    - *An animal is a bird if it has feathers*
    - *bird(X) :- animal(X), has_feather(X).*

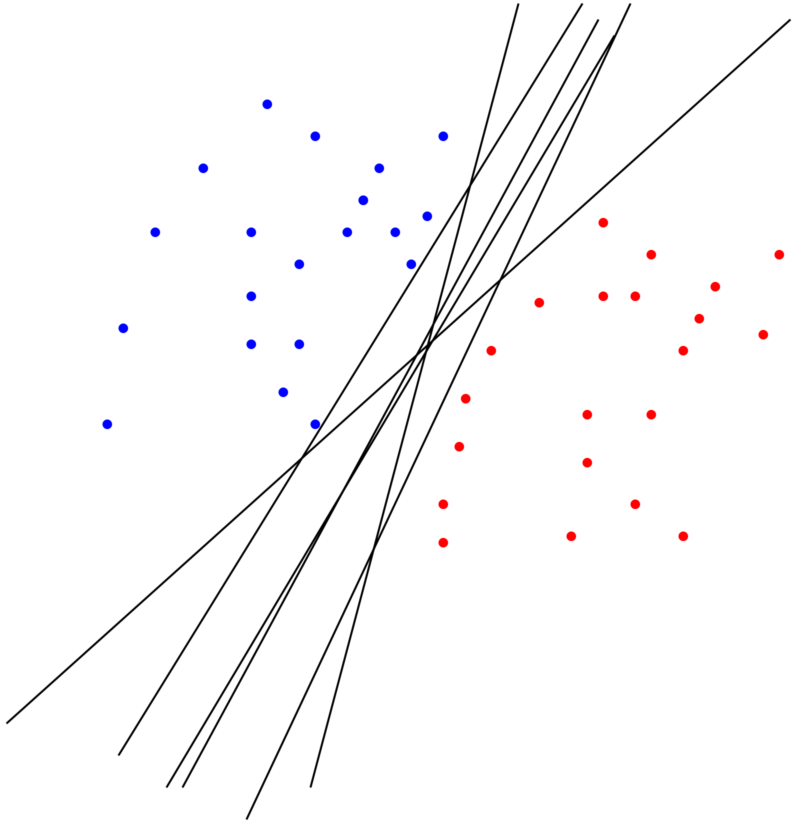
Support Vector Machine (SVM)

Introduction

- Support Vector Machine (SVM): Learns a linear separator for separating instances belonging to two different classes
  - In case of 1 dimensional instances, the separator is a point
  - In case of 2 dimensional instances, the separator is a line
  - In case of 3 dimensional instances, the separator is a plane
  - In case of instances in more than 3 dimensional space, the separator is a hyperplane

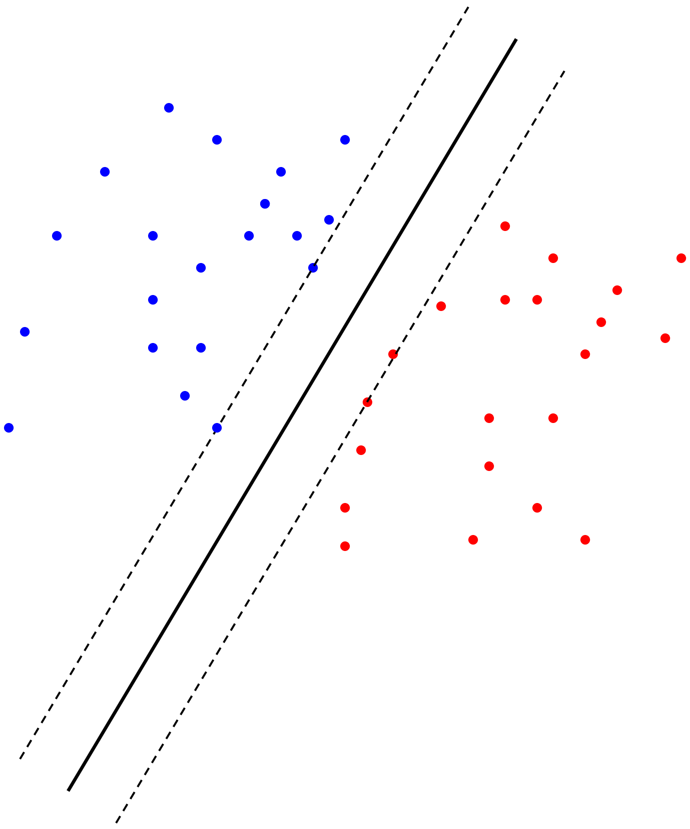
- Given a set of linearly separable instances, there exist infinite number of linear separators which can separate the instances into 2 classes
  - SVM chooses that linear separator which has the maximum "margin"
  - Intuition: A linear separator with the maximum margin will generalize better for new unseen instances in test data

SVM: Linear Separator



- An example of two dimensional instances
- Two classes:
  - Positive and Negative
- Linearly separable data
- Infinite number of linear separators are possible

SVM: Linear Separator



- **Goal**: To find the linear separator which provides the maximum margin of separation between two classes
- Support vectors: Instances which lie on the margin boundaries

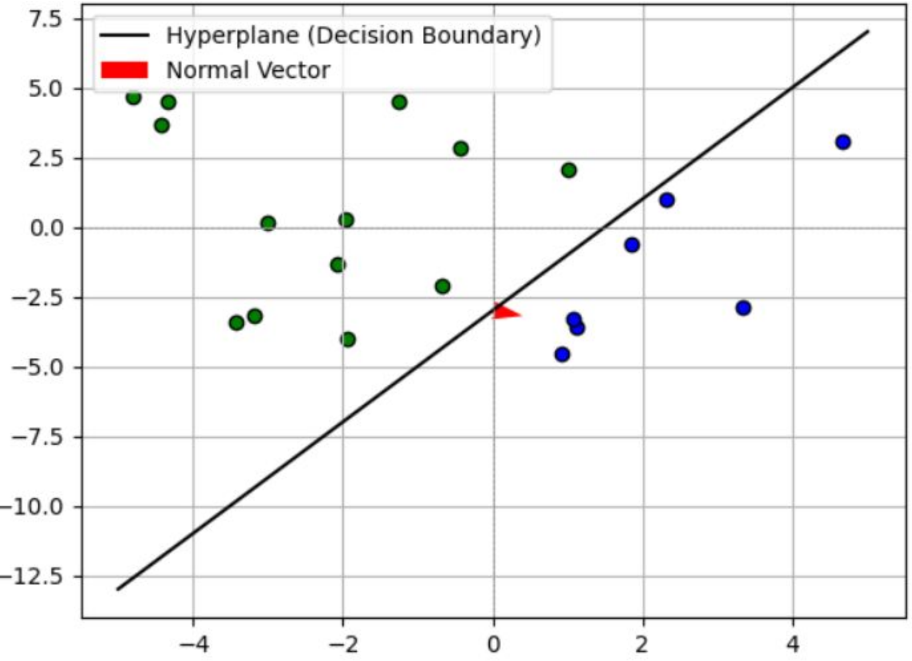
- Any other possible way to find hyperplanes?
- What are the issues?

Representation of a hyperplane

The general form of a hyperplane in an n-dimensional space is given by:

$$w_1x_1 + w_2x_2 + \cdots + w_nx_n + b = 0$$

- Can be expressed in a vector form as: $w^Tx + b = 0$ or $w \cdot x + b = 0$



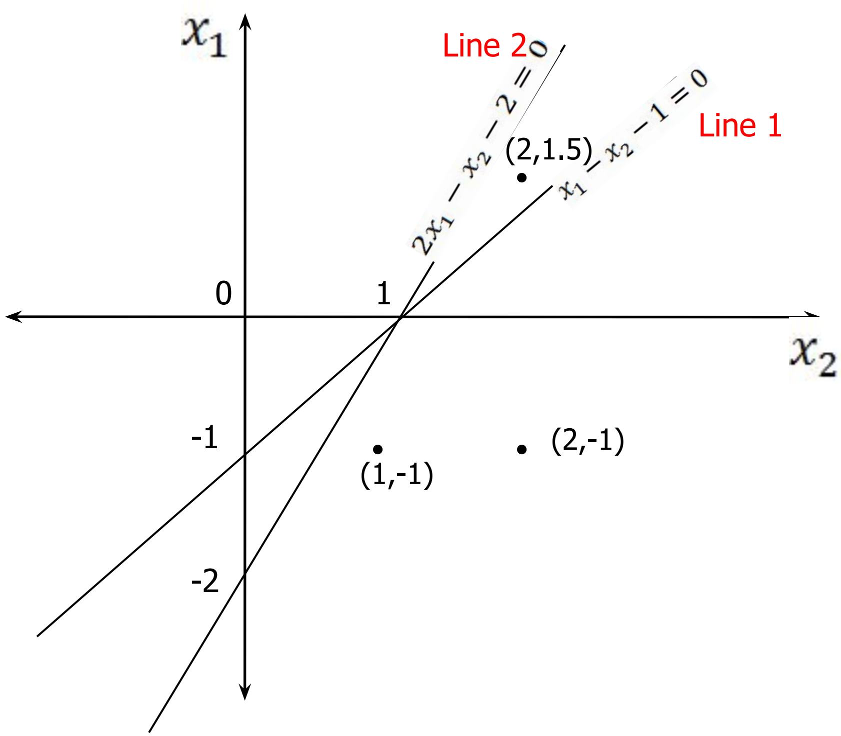
- The vector ***w*** is perpendicular to the hyperplane and ***b*** is the bias term (controls the offset of the hyperplane from the origin). $w \perp \text{hyperplane}$
- $w \cdot x + b > 0$ → Points on one side of the hyperplane.

  $w \cdot x + b < 0$ → Points on the other side.
- If b=0, the hyperplane passes through the origin.
- If $b \neq 0$, the hyperplane is shifted away from the origin.

Representation of a hyperplane

The general form of a hyperplane in an n-dimensional space is given by: $w_1x_1 + w_2x_2 + \cdots + w_nx_n + b = 0$

- Can be expressed in a vector form as: $w^Tx + b = 0$ or $w \cdot x + b = 0$



- For a particular linear separator, any point $\mathbf{x}^i$ lying on the "positive" side will have positive value of

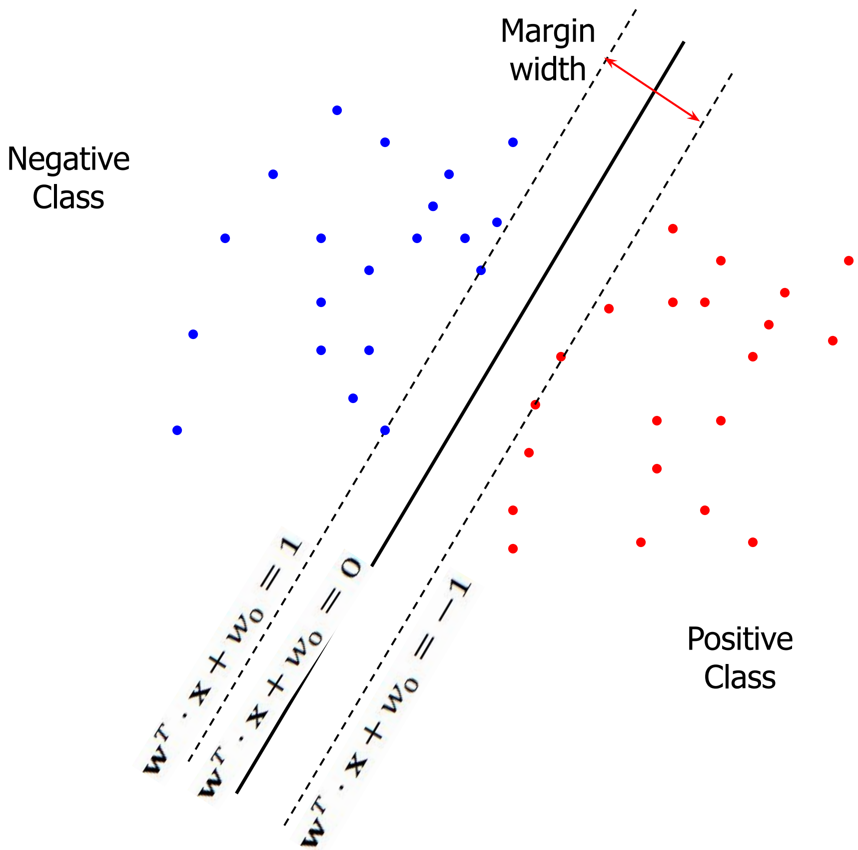
$$w \cdot x^i + b$$

- Also, any point $\mathbf{x}^i$ lying on the "negative" side will have negative value of

$$w \cdot x^i + b$$

- E.g., the point (2,1.5) is on positive side of Line 2 (0.5) and on negative side of Line 1 (-0.5)

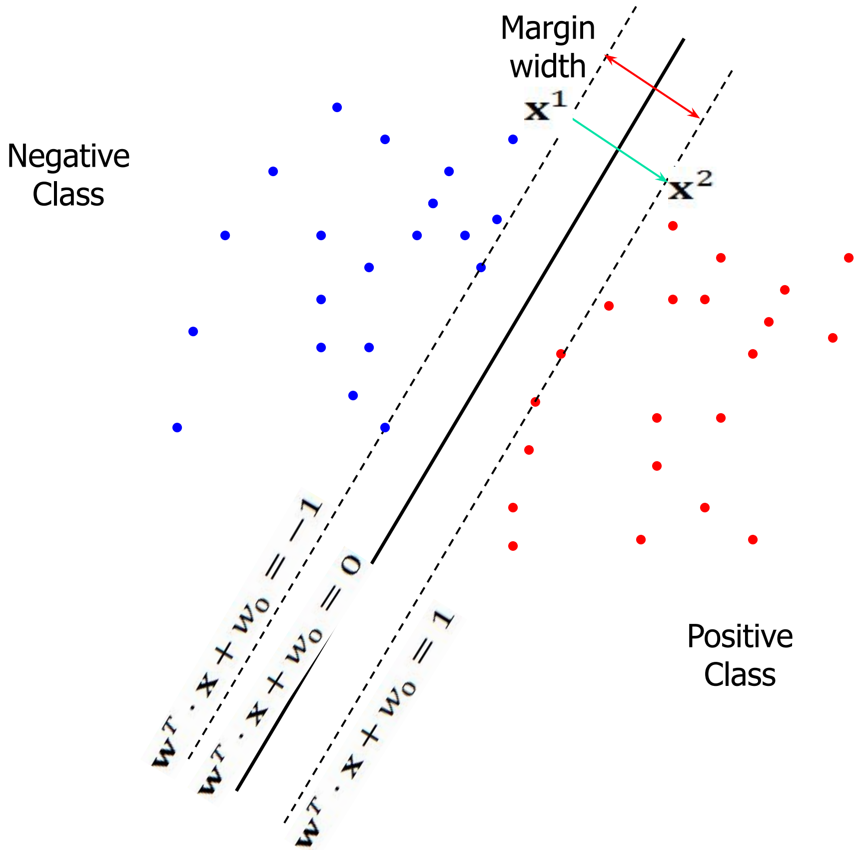
SVM: Training



- **Training instances**:
  $\{\langle \mathbf{x}^1, y^1 \rangle, \langle \mathbf{x}^2, y^2 \rangle, \cdots \langle \mathbf{x}^N, y^N \rangle\}$
- $\mathbf{x}^i$ is a point in n-dimensional space
- $y^i \in \{+1, -1\}$ its corresponding true class label
- **Goal**: To find optimal linear separator which maximizes the margin
- All positive class points (+1 label) must be on or beyond the +1 margin line.
- All negative class points (-1 label) must be on or beyond the -1 margin line.

- Since maximizing the margin is the core objective, adding an arbitrary scaling factor k is unnecessary because it cancels out in optimization.
- For +1 points: $w \cdot x^i + b$ >= 1
- For -1 points: $w \cdot x^i + b$ <= -1

Computing Margin Width



- Consider two points $\mathbf{x}^1$ and $\mathbf{x}^2$ such that they lie on the opposite margins and the vector $\mathbf{x}^2 - \mathbf{x}^1$ is perpendicular to the linear separator
- The vector $\mathbf{w}$ is also perpendicular to the linear separator
- margin = d = $\dfrac{w^T(x^2 - x^1)}{\|w\|}$
- Therefore, by definition,

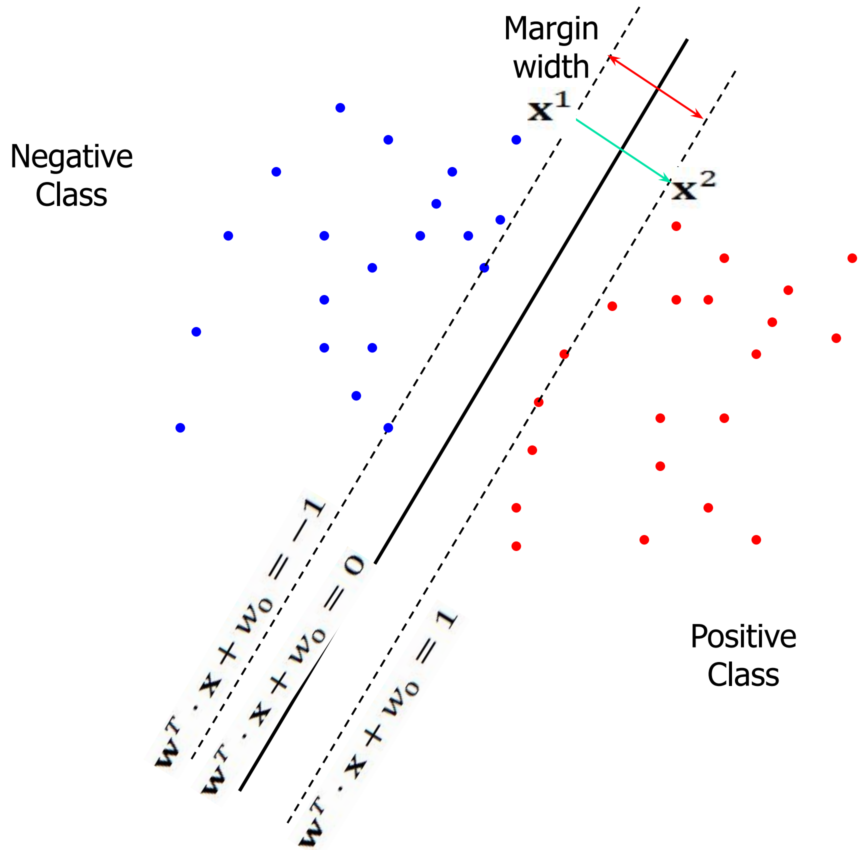
$$(\mathbf{x}^2 - \mathbf{x}^1) = \lambda \cdot \mathbf{w}$$

$$w^T x^1 + b = -1$$

$$w^T x^2 + b = 1$$

$$\mathbf{w}^T \cdot (\mathbf{x}^2 - \mathbf{x}^1) = 2$$

Computing Margin Width



- Substituting $(\mathbf{x}^2 - \mathbf{x}^1) = \lambda \cdot \mathbf{w}$

$$\mathbf{w}^T \cdot (\mathbf{x}^2 - \mathbf{x}^1) = 2$$

$$\lambda \mathbf{w}^T \cdot \mathbf{w} = 2 \Rightarrow \lambda = \frac{2}{\mathbf{w}^T \cdot \mathbf{w}}$$

- Margin width:

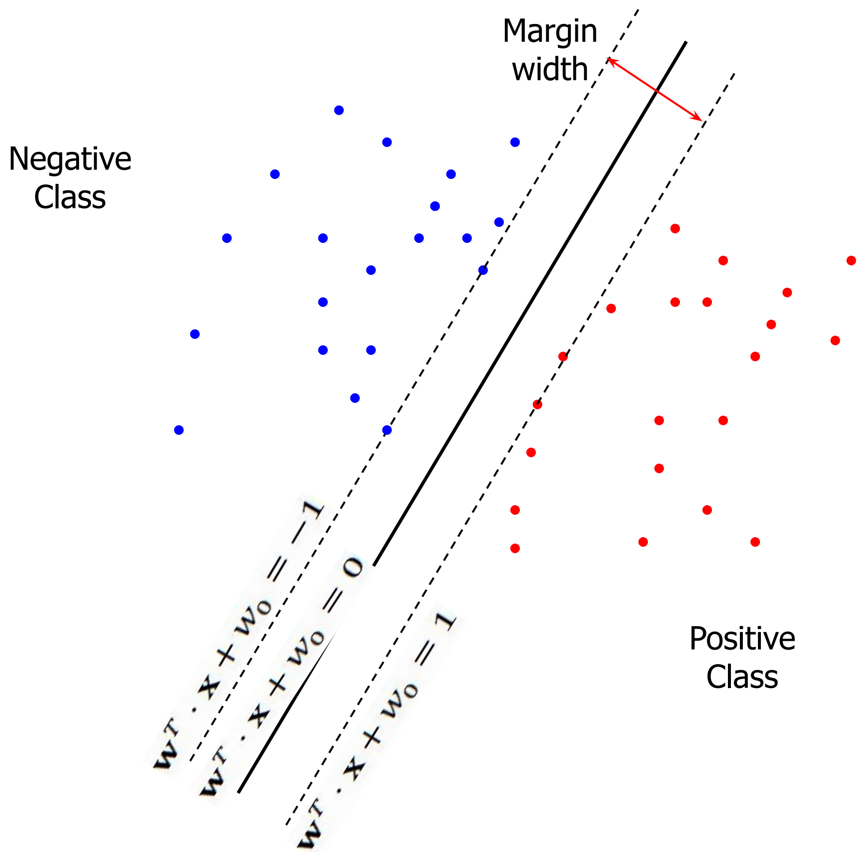
$$\|\mathbf{x}^2 - \mathbf{x}^1\| = \sqrt{(\mathbf{x}^2 - \mathbf{x}^1)^T \cdot (\mathbf{x}^2 - \mathbf{x}^1)}$$

$$\|\mathbf{x}^2 - \mathbf{x}^1\|^2 = \lambda^2 (\mathbf{w}^T \cdot \mathbf{w})$$

$$\|\mathbf{x}^2 - \mathbf{x}^1\|^2 = \frac{4}{(\mathbf{w}^T \cdot \mathbf{w})^2} (\mathbf{w}^T \cdot \mathbf{w})$$

$$\|\mathbf{x}^2 - \mathbf{x}^1\|^2 = \frac{4}{\mathbf{w}^T \cdot \mathbf{w}} \propto \frac{1}{\mathbf{w}^T \cdot \mathbf{w}}$$

SVM: Optimization Problem



- Objective:
  - Maximize the margin

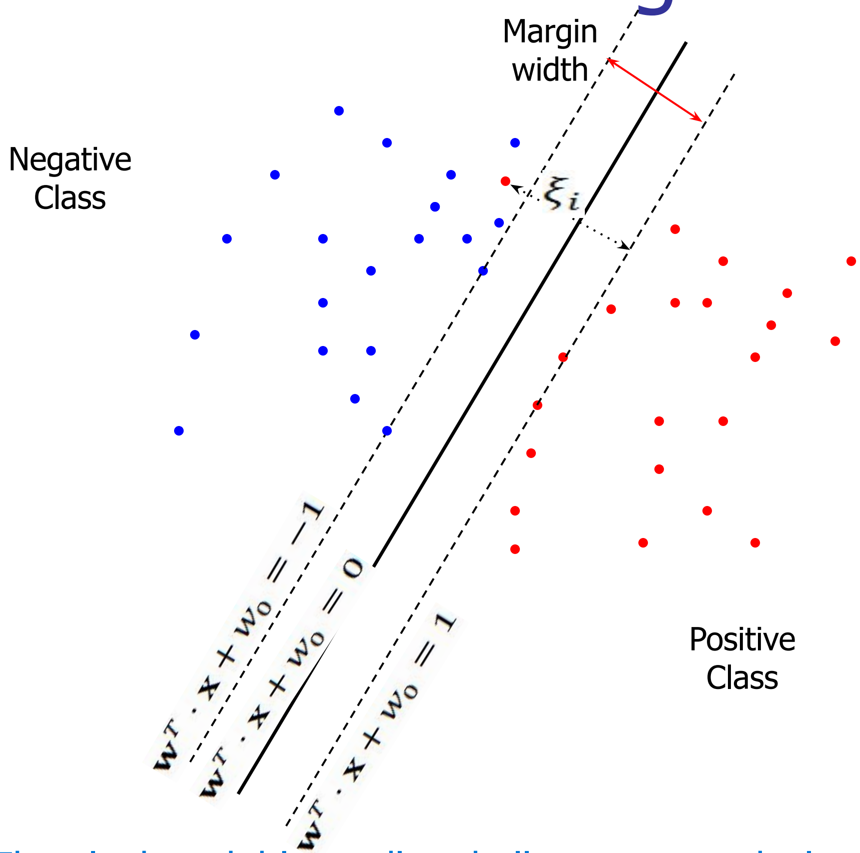
$$\min_{w,b} \left(\frac{1}{2} w^T w \right)$$

- Subject to the following constraints:
  - Every training instance should lie on the appropriate (positive / negative) side of the linear separator

$$y^i (w^T x^i + b) \geq 1, \quad \forall 1 \leq i \leq N$$

$$-y^i (w^T x^i + b) + 1 \leq 0, \quad \forall 1 \leq i \leq N$$

SVM: Soft-margin Formulation



- Objective:
  - Maximize the margin and minimize the training error

$$\min_{w,b,\xi} \left(\frac{1}{2} w^T w\right) + C \sum_{i=1}^{N} \xi_i$$

- Subject to the following constraints:
  - Introducing slack variables so that the constraint is satisfied for training instances lying on incorrect side

$$y^i (w^T x^i + b) \geq 1 - \xi_i, \quad \forall 1 \leq i \leq N$$

$$-\xi_i \leq 0, \quad \forall 1 \leq i \leq N$$

The slack variable epsilon_i allows some misclassifications or margin violations.
The parameter C controls the trade-off between maximizing the margin and minimizing the classification error.

- **High** C => **Low** ξ_i
- **Low** C => **High** ξ_i **—>** This can allow for large outliers
- What happens if we remove $C \sum_{i=1}^{N} \xi_i$?

Optimization using Lagrange Multipliers

- One Lagrange multiplier is associated with each distinct constraint

$$
\begin{aligned}
&L(\alpha_1, \cdots, \alpha_N, \mu_1, \cdots, \mu_N) \\
&\qquad = \min_{\mathbf{w}, w_0, \xi} \left(\frac{1}{2}\mathbf{w}^T \cdot \mathbf{w}\right) + C \cdot \sum_{i=1}^{N} \xi_i \\
&\qquad + \sum_{i=1}^{N} \alpha_i \cdot \left(-y^i\left(\mathbf{w}^T \cdot \mathbf{x}^i + w_0\right) + 1 - \xi_i\right) \\
&\qquad + \sum_{i=1}^{N} \mu_i \cdot (-\xi_i)
\end{aligned}
$$

$$
s.t. \quad \alpha_i \geq 0, \mu_i \geq 0, \forall_{1 \leq i \leq N}
$$

Optimization using Lagrange Multipliers

- Differentiating w.r.t. $\mathbf{w}$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i y^i \mathbf{x}^i = 0 \qquad \Rightarrow \boxed{\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i y^i \mathbf{x}^i}$$

- Differentiating w.r.t. w_0

$$\frac{\partial L}{\partial w_0} = \sum_{i=1}^{N} -\alpha_i y^i = 0 \quad \Rightarrow \boxed{\sum_{i=1}^{N} \alpha_i y^i = 0}$$

- Differentiating w.r.t. ξ_i

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \quad \Rightarrow \boxed{\mu_i + \alpha_i = C}$$

Doubt in differentiation w.r.t. epsilon_i

- Substituting optimal values:

$$L(\alpha_1, \cdots, \alpha_N, \mu_1, \cdots, \mu_N)$$

$$= \left(\frac{1}{2}\mathbf{w}^{*T} \cdot \mathbf{w}^*\right) + C \cdot \sum_{i=1}^{N} \xi_i^*$$

$$+ \sum_{i=1}^{N} \alpha_i \cdot \left(-y^i\left(\mathbf{w}^{*T} \cdot \mathbf{x}^i + w_0^*\right) + 1 - \xi_i^*\right) + \sum_{i=1}^{N} \mu_i \cdot (-\xi_i^*)$$

$$= \frac{1}{2}\left(\sum_{i=1}^{N} \alpha_i y^i \mathbf{x}^i\right)^T \cdot \left(\sum_{j=1}^{N} \alpha_j y^j \mathbf{x}^j\right) + C \cdot \sum_{i=1}^{N} \xi_i^*$$

$$+ \sum_{i=1}^{N} -\alpha_i y^i \left(\sum_{j=1}^{N} \alpha_j y^j \mathbf{x}^j\right)^T \mathbf{x}^i - w_0^* \sum_{i=1}^{N} \alpha_i y^i$$

$$+ \sum_{i=1}^{N} \alpha_i (1 - \xi_i) + \sum_{i=1}^{N} \mu_i \cdot (-\xi_i^*)$$

Terms involving ξ_i cancel each other out because

$$\mu_i + \alpha_i = C$$

Sum in the red circle is zero

- Finally, we get:

$$L(\alpha_1, \cdots, \alpha_N) = \frac{-1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y^i y^j \left(\mathbf{x}^{i^T} \mathbf{x}^j\right) + \sum_{i=1}^{N} \alpha_i$$

- Dual optimization problem:
  - Objective function:

$$\max_{\alpha_1,\cdots,\alpha_N} \frac{-1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y^i y^j \left(\mathbf{x}^{i^T} \mathbf{x}^j\right) + \sum_{i=1}^{N} \alpha_i$$

  - Subject to the following constraints:

$$\alpha_i \geq 0, \mu_i \geq 0, \mu_i + \alpha_i = C, \forall_i \;\; and \;\; \sum_{i=1}^{N} \alpha_i y^i = 0$$

$$\Rightarrow \;\; \alpha_i \geq 0, \alpha_i \leq C, \forall_i \;\; and \;\; \sum_{i=1}^{N} \alpha_i y^i = 0$$

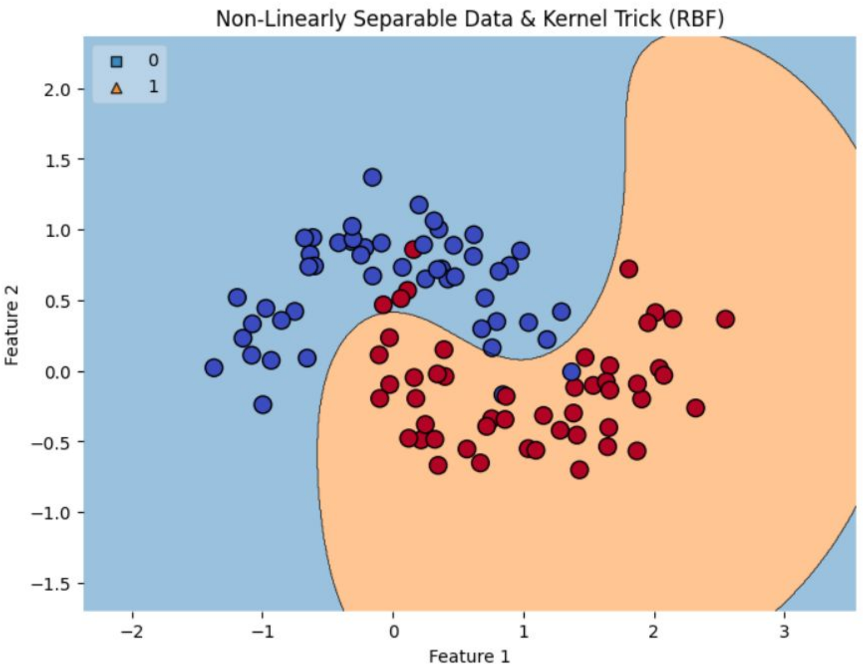
Any Quadratic Programming Solver can be used for solving this

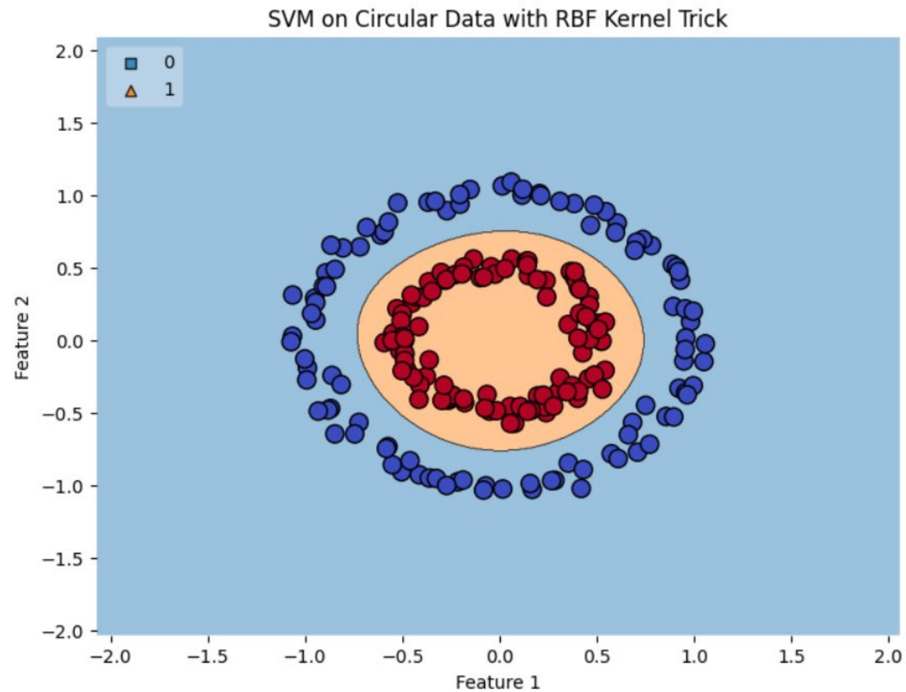
- Vary C and show its impact on decision boundary.

Using SVM for Predictions

- How to predict the class label for a new instance $\mathbf{x}$ given a trained SVM
- Primal Form: $\mathbf{w}^T \cdot \mathbf{x} + w_0$
  - Compute $\mathbf{x} = [2, 4]$; Positive value indicates the positive class and vice versa
  - E.g., $\mathbf{w} = [2, -1] \; and \; w_0 = -2$ be the learned parameters
  - For the new instance $\mathbf{w}^T \cdot \mathbf{x} + w_0 = -2$ and hence Negative class is predicted
- Dual Form:
  - Compute $\mathbf{w}^T \cdot \mathbf{x} + w_0 = \left(\sum_{i=1}^{N} \alpha_i y^i \mathbf{x}^i \right)^T \mathbf{x} = \sum_{i=1}^{N} \alpha_i y^i \mathbf{x}^{i^T} \mathbf{x}$
  - Positive value indicates the positive class and vice versa
  - Practically, most of the α_i values are zeros; non-zero only for support vectors

Kernel Trick for SVM





- SVM works well with linearly separable data.
- But, many real-world data are non-linearly separable in nature
- The kernel function implicitly maps data into a higher-dimensional space where it becomes linearly separable.
- Instead of explicitly transforming data into a higher-dimensional space, we use a kernel function to compute the dot product in that space efficiently.

Kernel Trick for SVM

Explicit Transformation: High Computational Cost

If we explicitly map data from a lower-dimensional space R^d to a higher-dimensional space R^D using a feature transformation $\phi(x)$, we need to compute the dot product in the new space.

Given n training samples, each with d features, transforming them explicitly to a higher dimension D takes:

$$O(nD)$$

Then, computing the dot product between two transformed vectors $\phi(x^i)$ and $\phi(x^j)$ in R^D requires:

$$O(D)$$

Training SVM typically involves solving a quadratic programming problem, which has a complexity of:

$$O(n^2D)\text{(in the worst case)}$$

due to the need to compute pairwise dot products in the high-dimensional space.
If D is very large (e.g., infinite in the case of the Radial Basis Function (RBF) kernel), the computation becomes impractical.

Kernel Trick for SVM

Kernel Function is the Saviour: Avoids Curse of Dimensionality, Efficient Computation

The kernel function $K(x^i,x^j)=\langle\phi(x^i),\phi(x^j)\rangle$ computes the dot product in high dimensional space in: O(d) because it only depends on the original d-dimensional data.

The SVM training complexity **remains**: $O(n^2d)$
instead of **$O(n^2D)$**, making it computationally feasible even when D is very large or infinite.

Kernel Type	Equation	Description
Linear Kernel	$K(x_i, x_j) = x_i \cdot x_j$	Used when data is already linearly separable.
Polynomial Kernel	$K(x_i, x_j) = (x_i \cdot x_j + c)^d$	Maps data into a higher-degree polynomial space.
Radial Basis Function (RBF) Kernel	$K(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2)$	Maps data to an infinite-dimensional space. Useful for complex boundaries.
Sigmoid Kernel	$K(x_i, x_j) = \tanh(\alpha x_i \cdot x_j + c)$	Similar to neural network activation functions.

Kernel Trick for SVM

Linear Kernel	$K(x_i, x_j) = x_i \cdot x_j$	Used when data is already linearly separable.

$$K(x^i,x^j)=x_i^\top x_j$$

- Directly computes the dot product in O(d).
- No need for explicit mapping.

Polynomial Kernel	$K(x_i, x_j) = (x_i \cdot x_j + c)^d$	Maps data into a higher-degree polynomial space.

$$K(x_i,x_j)=(x_i^\top x_j+c)^p$$

- Requires computing $x_i^\top x_j$ (which takes O(d)) and then raising it to power p (which takes O(1)).
- Total complexity: **O(d)**.

Kernel Trick for SVM

How the Kernel Trick Works in SVM?

1. Choose an appropriate kernel function based on the dataset.
2. Compute the kernel matrix $K(x_i, x_j)$, which represents inner products in higher-dimensional space.
3. Use this matrix in the SVM optimization problem without explicitly transforming the data.
4. The SVM classifier finds the optimal hyperplane in the transformed space.

Advantage of Dual over Primal

Handles High-Dimensional Data Efficiently (Kernel Trick)

- The **primal form** works directly in the original feature space, making it impractical when the number of features (D) is large or infinite (e.g., in kernelized SVM).
- The **dual form** allows the use of the **kernel trick**, enabling SVMs to operate in an **implicit high-dimensional space** without explicitly transforming data.

$$
\begin{aligned}
&L(\alpha_1, \cdots, \alpha_N, \mu_1, \cdots, \mu_N) \\
&\quad = \min_{\mathbf{w}, w_0, \xi} \left(\frac{1}{2}\mathbf{w}^T \cdot \mathbf{w}\right) + C \cdot \sum_{i=1}^{N} \xi_i \\
&\quad + \sum_{i=1}^{N} \alpha_i \cdot \left(-y^i\left(\mathbf{w}^T \cdot \mathbf{x}^i + w_0\right) + 1 - \xi_i\right) \\
&\quad + \sum_{i=1}^{N} \mu_i \cdot (-\xi_i)
\end{aligned}
$$

$$
\max_{\alpha_1, \cdots, \alpha_N} \frac{-1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^i y^j \left(\mathbf{x}^{i^T} \mathbf{x}^j\right) + \sum_{i=1}^{N} \alpha_i
$$

Primal

Dual