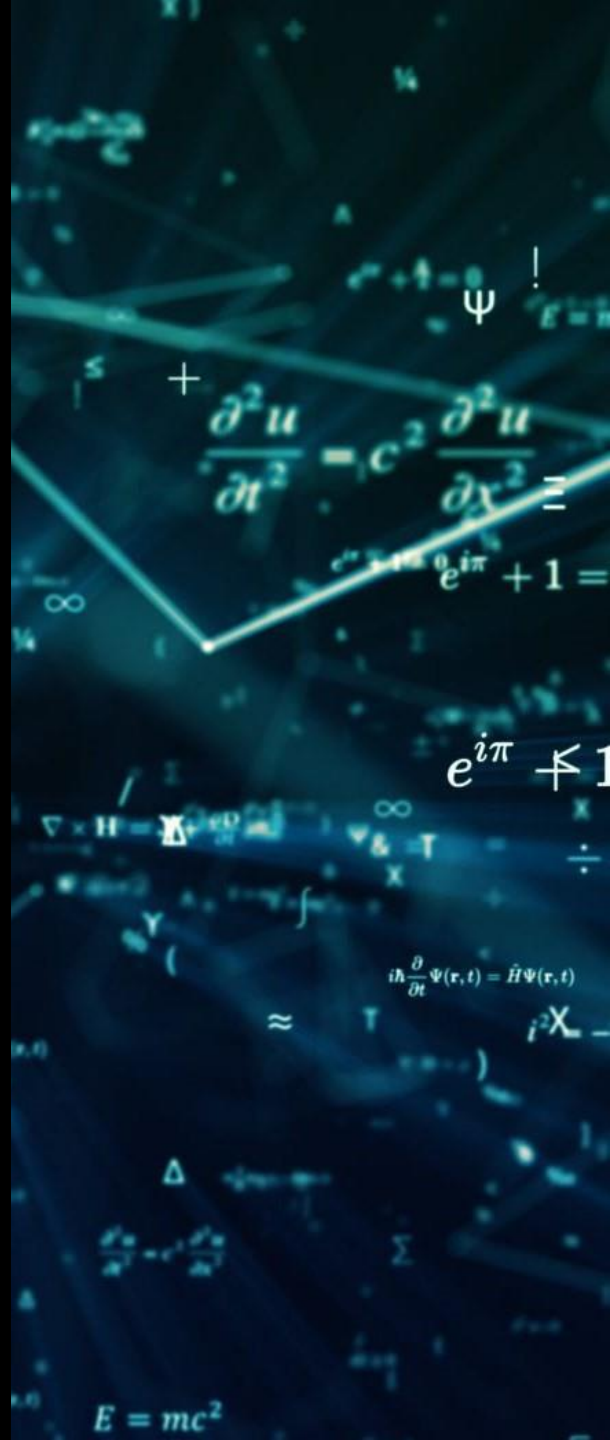
Artificial Intelligence Algorithms and Mathematics

CSCN 8000

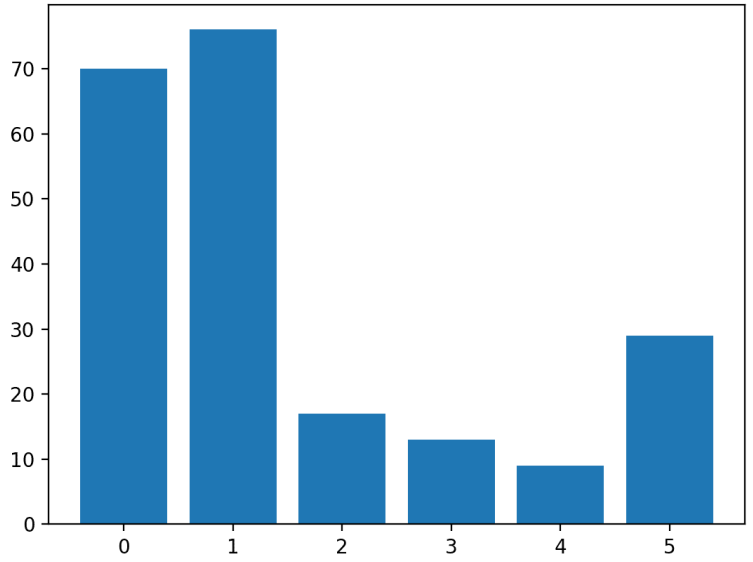


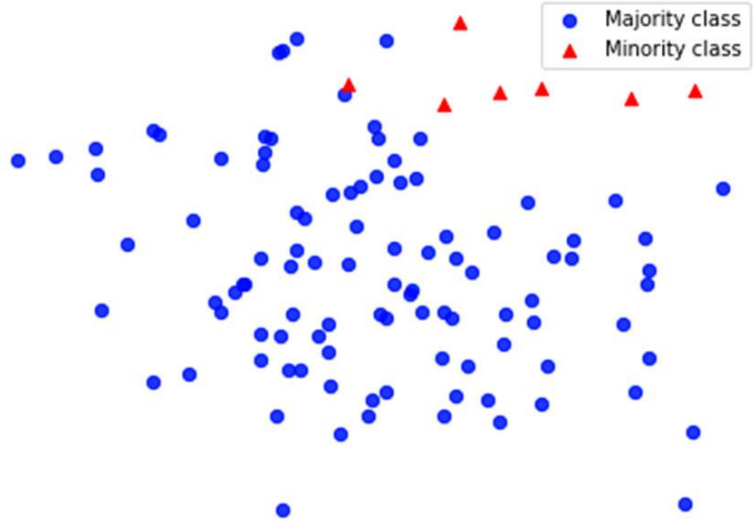
Classification

- Handling Class Imbalances
- Support Vector Machines

Handling Class Imbalances

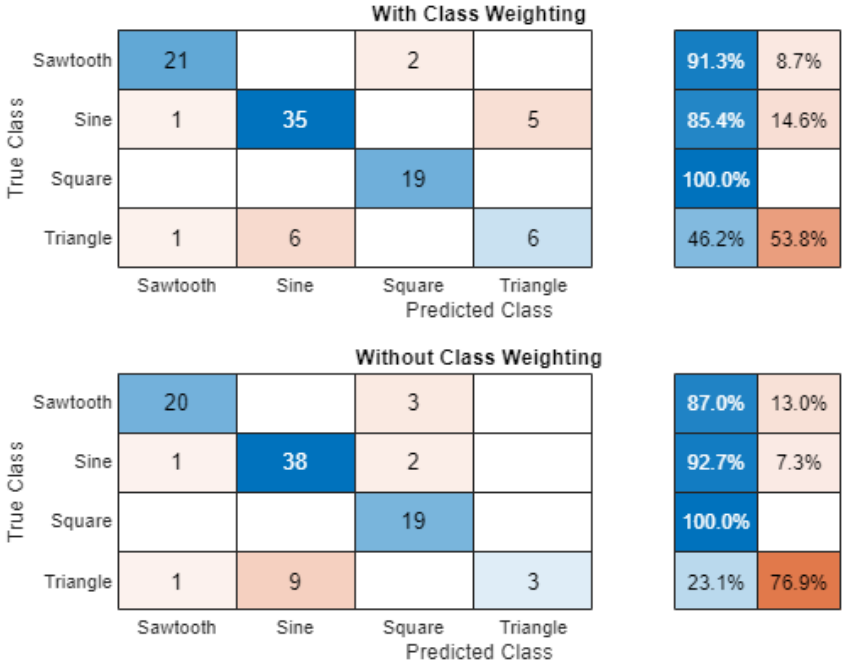
- In classification problems, classes can be highly imbalanced where some classes have very high frequency compared to others.
- This can result in poor performance on the minority class.





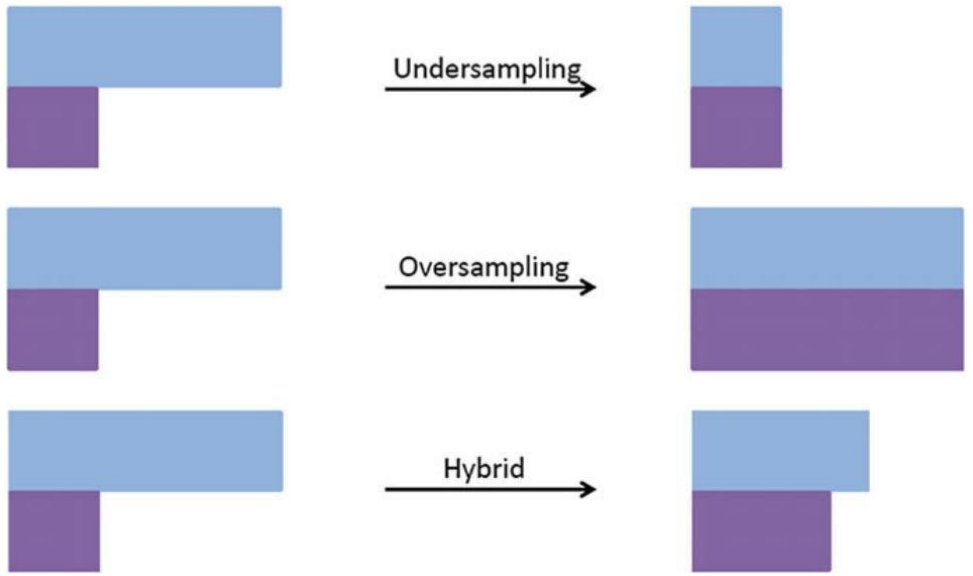
Class Weights

- One solution to handle class imbalances is ***Class Weights***
- Class weights basically assign higher weights to the samples from the minority class and lower weights to the samples from the majority class during the training process.
- This gives the algorithm more emphasis on correctly classifying the minority class, effectively balancing the impact of each class on the model's learning process.



Over/Under Sampling

- Oversampling and undersampling are approaches that aim to balance the representation of different classes in the training dataset, which can lead to better performance of the model, especially for the minority class.
- Oversampling:
  - Increase the number of instances of the minority class by generating synthetic samples or duplicating existing ones.
  - Examples: Random Oversampling / SMOTE / ADASYN
- Undersampling:
  - Reduce the number of instances of the majority class to achieve a better class balance.
  - Examples: Random/ Tomek Links/ ENNs.



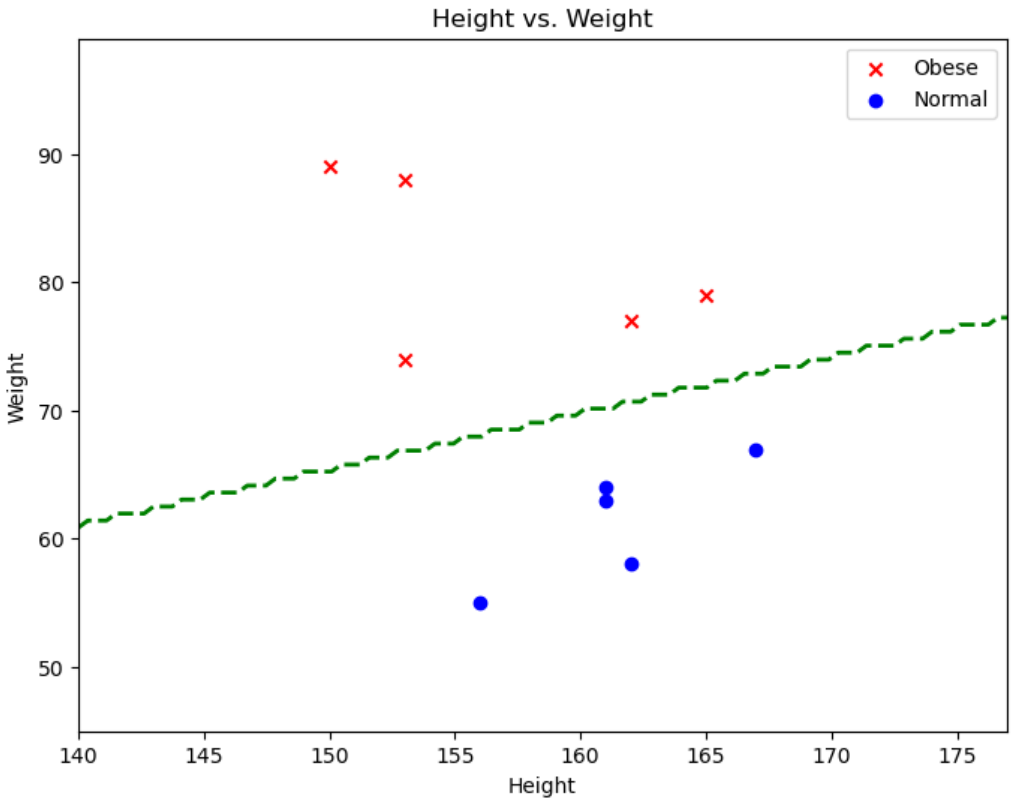
Recall: Decision Boundary Definition

- The decision boundary in a binary classification problem is a hypersurface that separates the feature space into regions corresponding to different classes.
- The decision boundary only exists as a line or hyperplane if the classes are ***linearly separable***
- The equation of the linear decision boundary is as follows:

$$\hat{y} = \vec{w}\,\vec{x} + b$$

  - $\vec{x}$ represents all input features.
  - $\hat{y}$ represents estimated class a point belongs to.
  - y represents the label of the point $\{0,1\}$

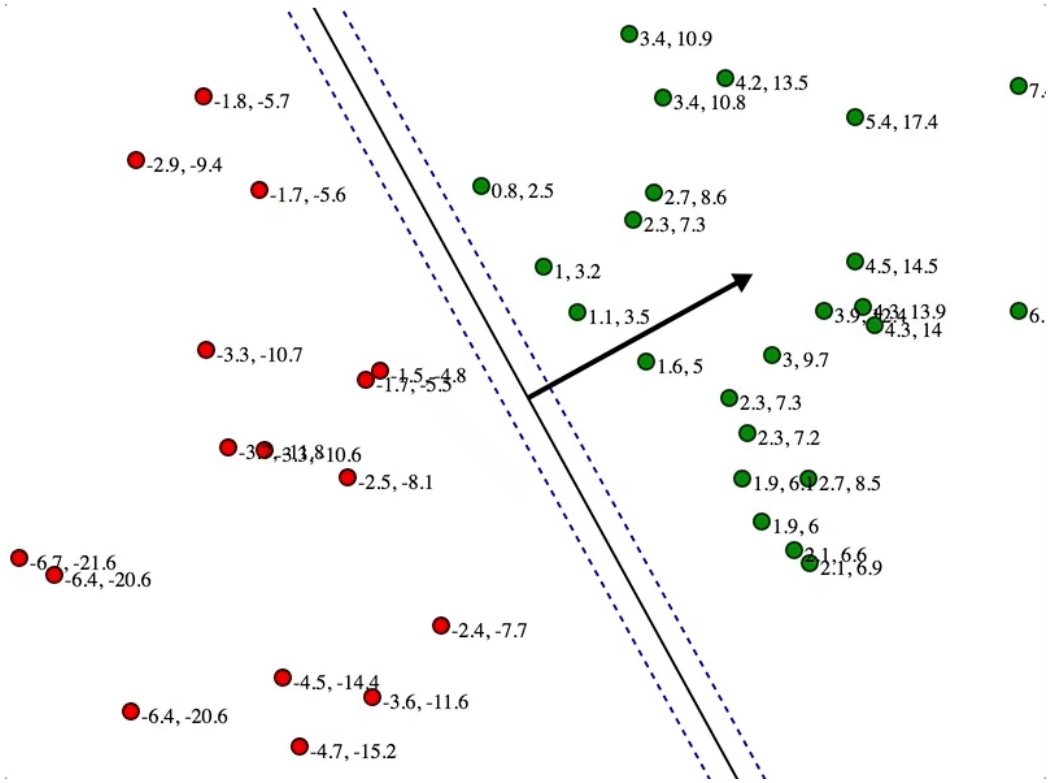
- $\vec{w}, b$ are learnable parameters to be estimated



$$\hat{y} = \vec{w}\,\vec{x} + b$$

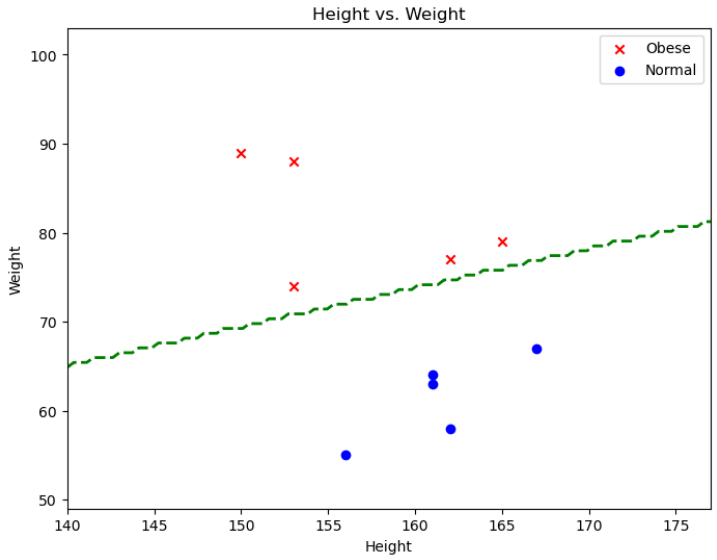
Support Vector Machines (SVM)

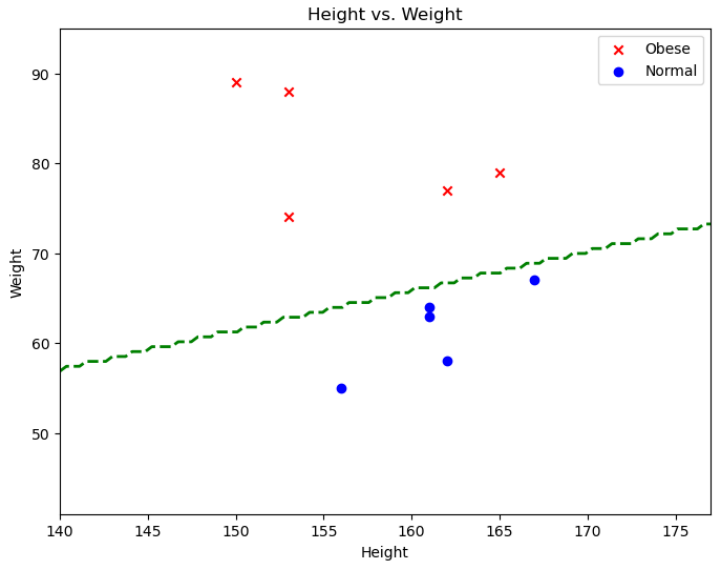
- Support Vector Machines (SVMs) are a powerful class of supervised machine learning algorithms used for both classification and regression tasks. They work by finding the optimal hyperplane that best separates different classes in a dataset.
- In order to reach the optimal hyperplane, several assumptions are made by the SVM algorithm. We will go through each assumption until reaching the final formulation of the SVM.

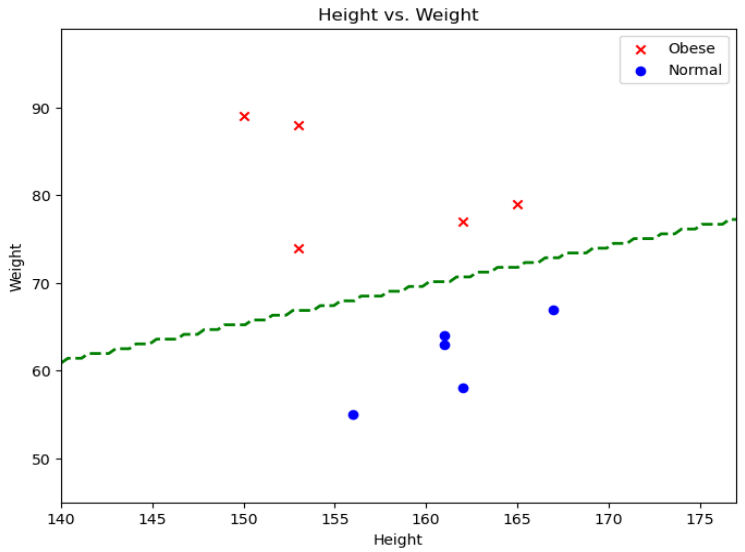


Unbiased Decision Boundary

- Assume we have 10 different people with different weight and height values. 5 of those people are labeled as "Obese" and 5 are labeled as "Normal". The plot of those people is shown below. Which figure represents an unbiased decision boundary in the middle of the classes?

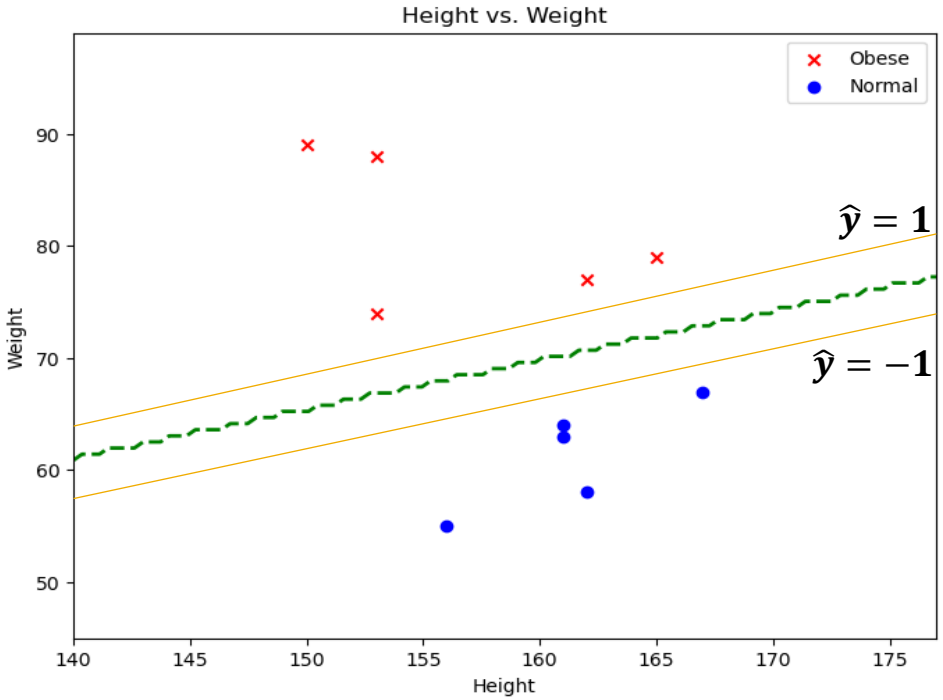






Unbiased Decision Boundary

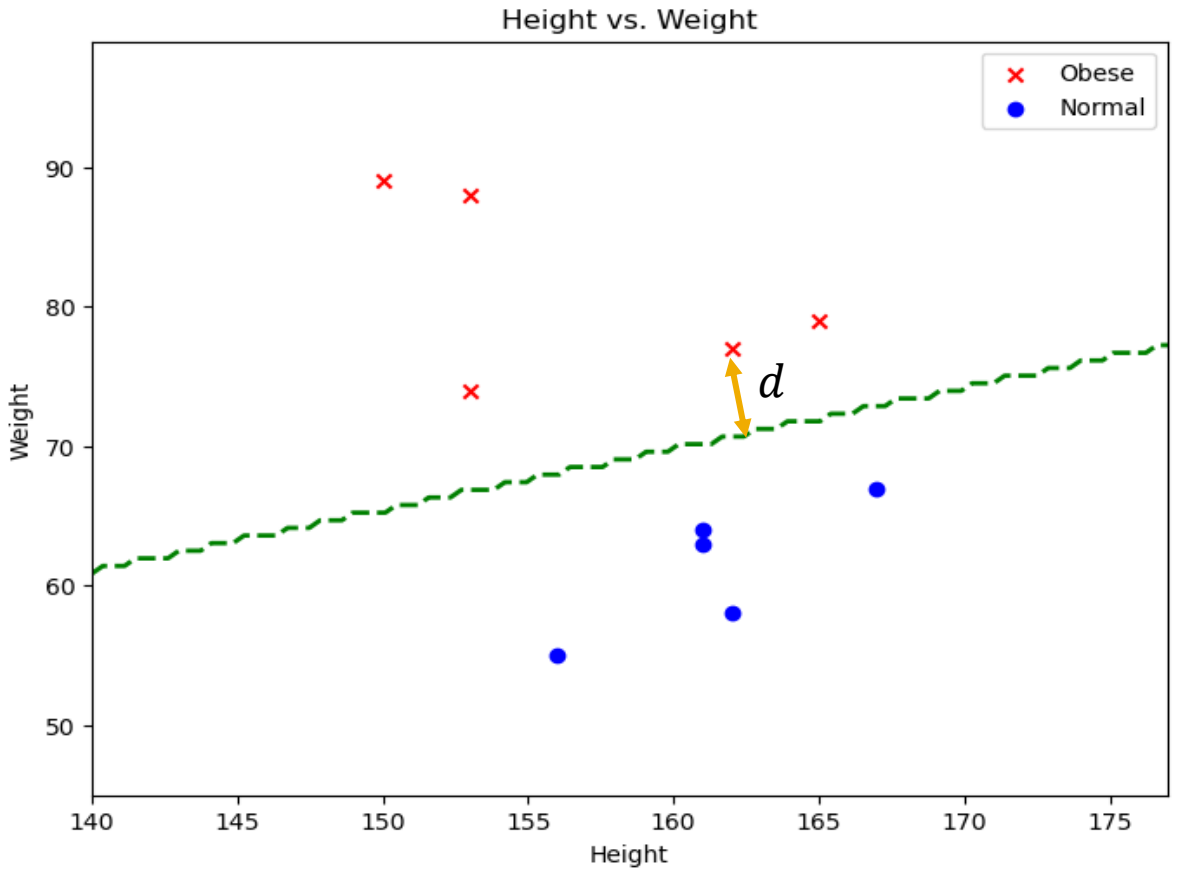
- Our first target is that we want an unbiased decision boundary lying directly in the middle of the two classes.
- To achieve that, we will first assume that all the points in each class have $\hat{y}$ equal to at least ± 1. → ***Ass. 1***
- We will also assume that each point in the classes have labels $y \in \{1, -1\}$ → ***Ass. 2***



Distance between point and boundary

- For any decision boundary ($\vec{\boldsymbol{w}}\,\vec{\boldsymbol{x}} + \boldsymbol{b}$), we represent the distance d_between the boundary and a point $\vec{\boldsymbol{x}}$, as follows:

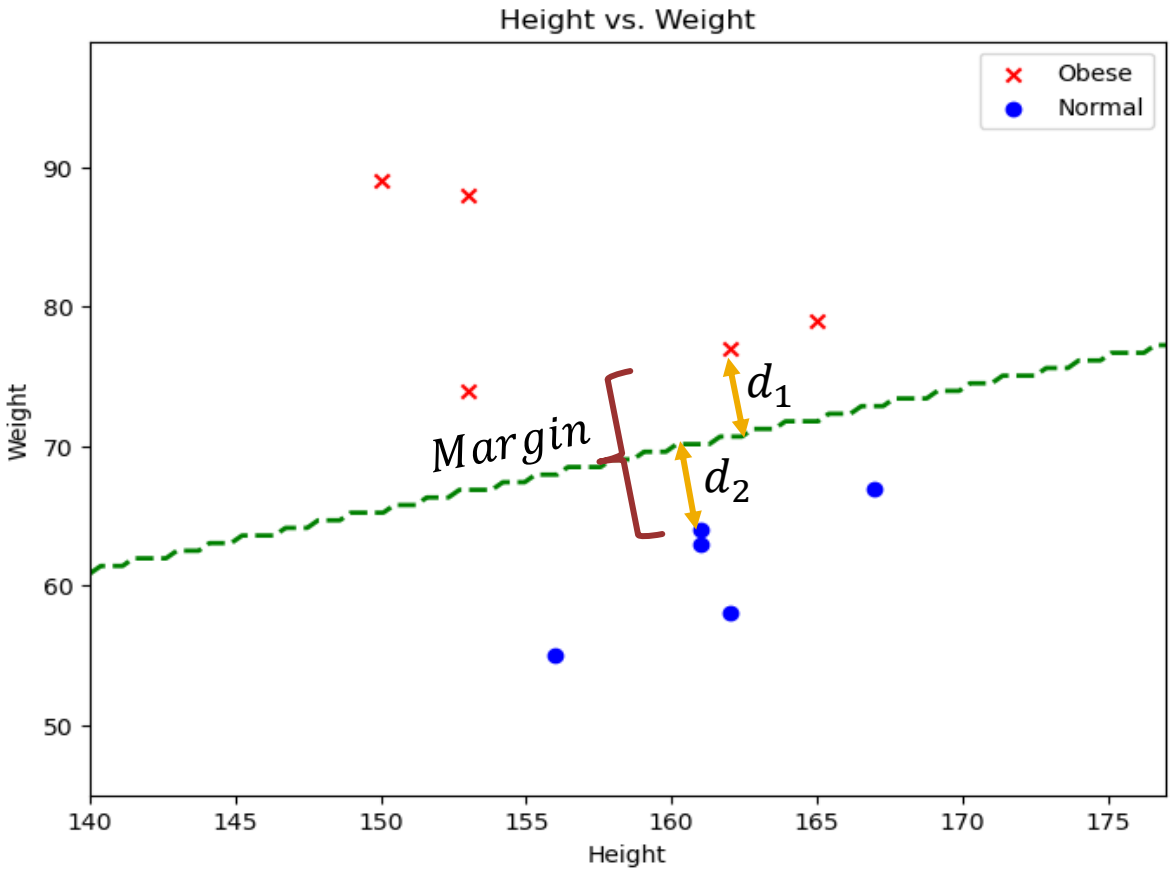
$$d = \frac{(\vec{\boldsymbol{w}}\,\vec{\boldsymbol{x}} + \boldsymbol{b})}{\|\vec{\boldsymbol{w}}\|} = \frac{\hat{\boldsymbol{y}}}{\|\vec{\boldsymbol{w}}\|}$$



Margin

- The margin represents the total distance between the decision boundary and the *closest* points in each class, where $Margin = d_1 + d_2$.

$$Margin = \left|\frac{\hat{\boldsymbol{y}}_1}{\|\vec{\boldsymbol{w}}\|}\right| + \left|\frac{\hat{\boldsymbol{y}}_2}{\|\vec{\boldsymbol{w}}\|}\right|$$

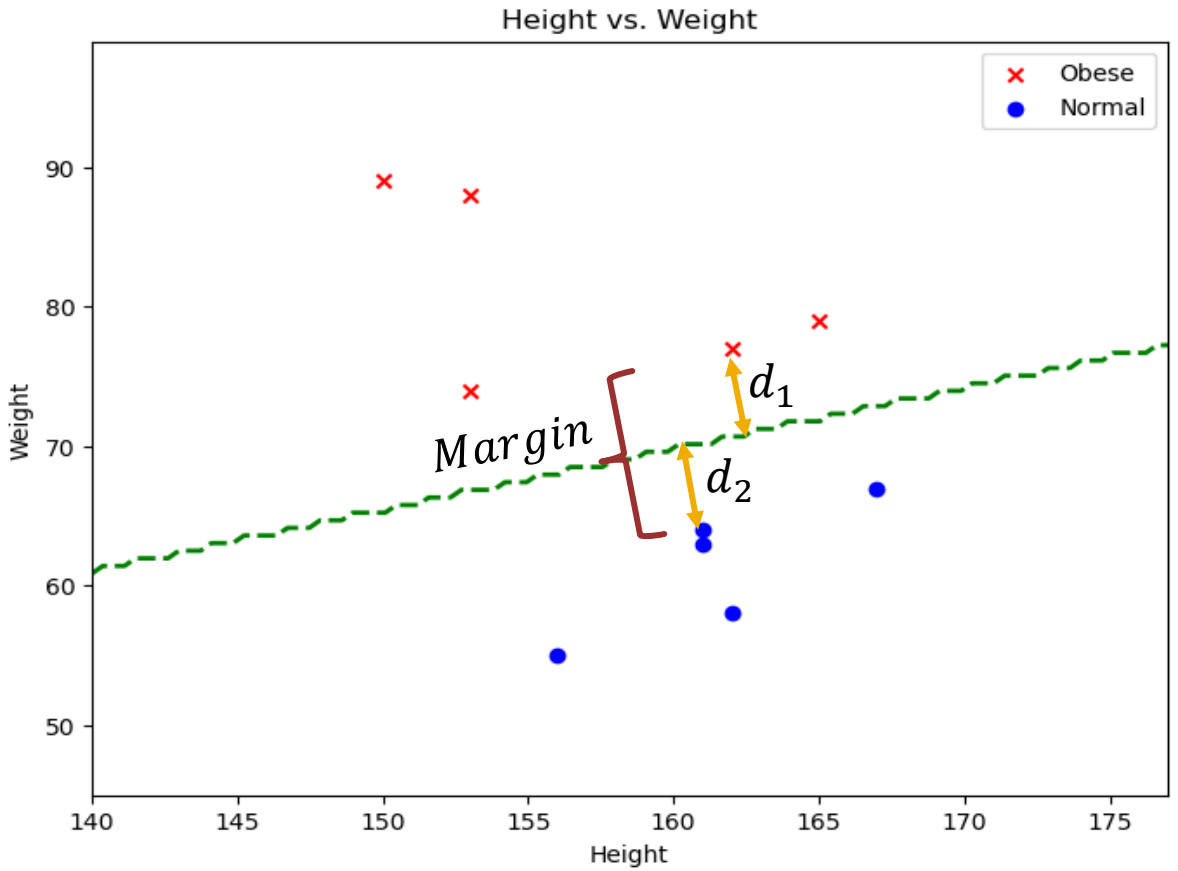


Margin

- The margin represents the total distance between the decision boundary and the *closest* points in each class, where $Margin = d_1 + d_2$.

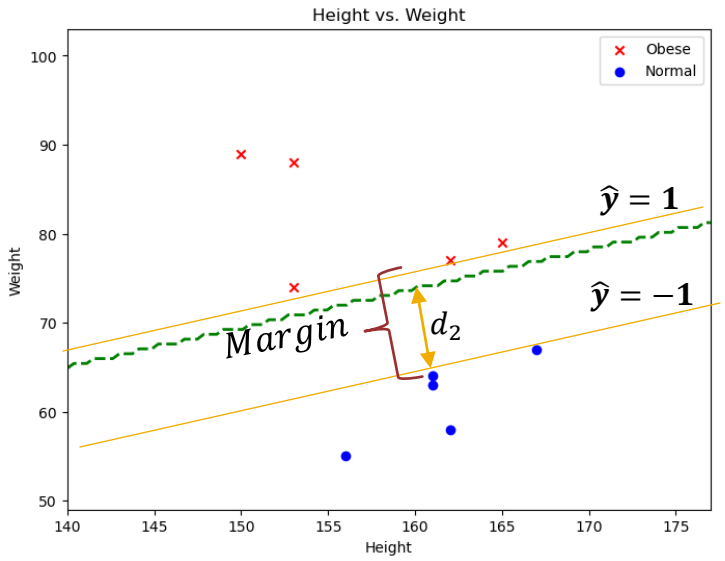
$$Margin = \left|\frac{\hat{y}_1}{\|\vec{w}\|}\right| + \left|\frac{\hat{y}_2}{\|\vec{w}\|}\right|$$

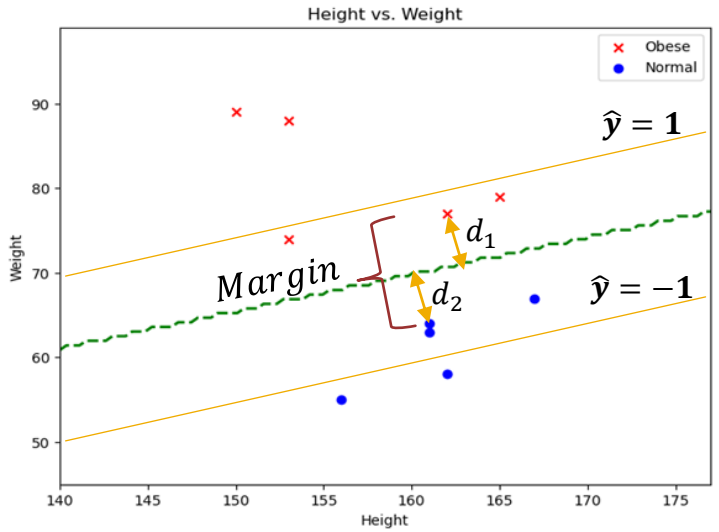
- Our target is to find the ***maximum*** possible margin by maximizing both d_1 $\boldsymbol{AND}$ d_2.

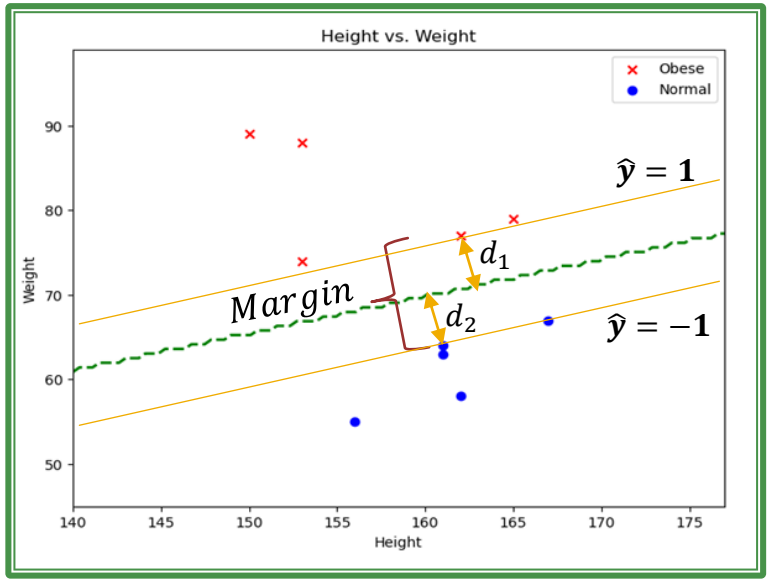


Margin

- Which of the following figures follows ***Ass. 1*** and achieves maximum possible margin by maximizing **both** d_1 ***AND*** d_2?





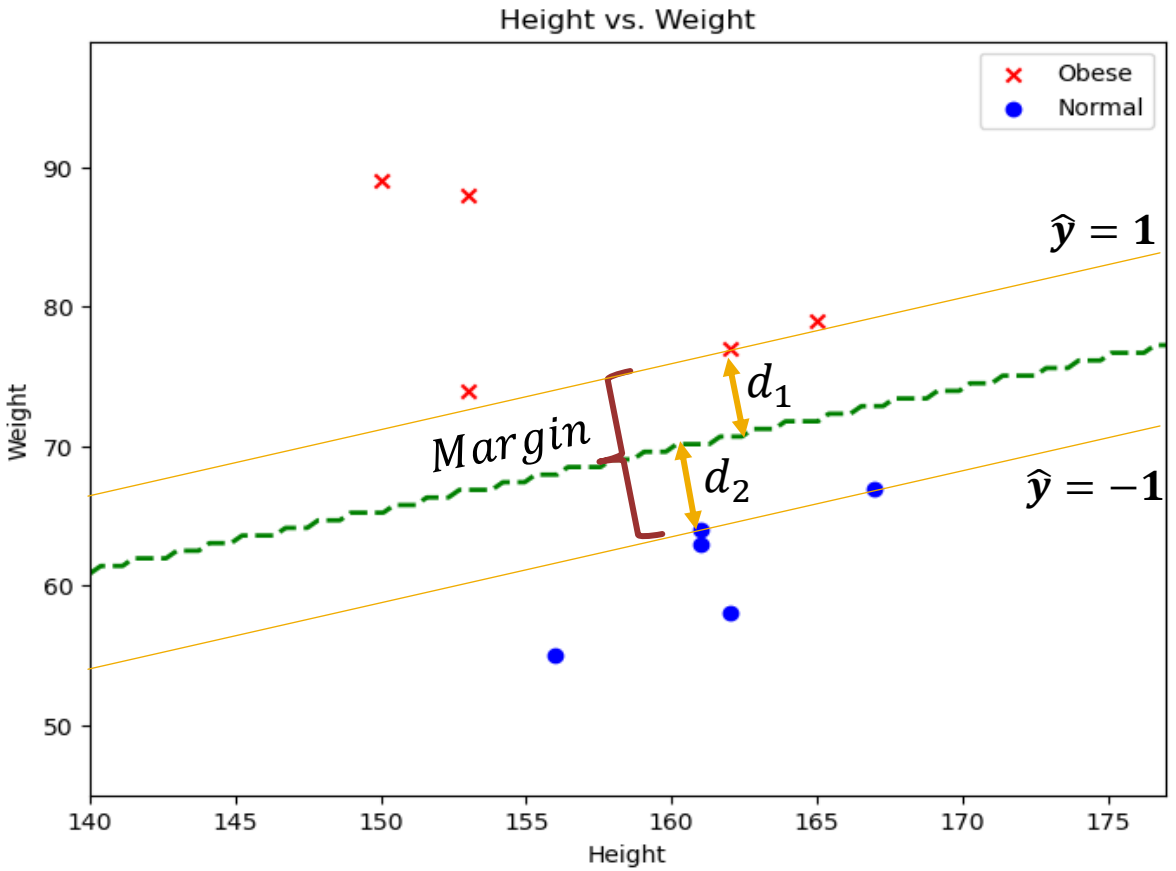


SVM Formulation

- To have the maximum possible values of **both** d_1 ***AND*** d_2 to maximize the margin while abiding by ***Ass. 1*** → Assume that the closest points to the boundary from each class have $\hat{y}$ exactly equal to $\pm\mathbf{1}$. (***Ass.3***)
- The margin now will be equal to:

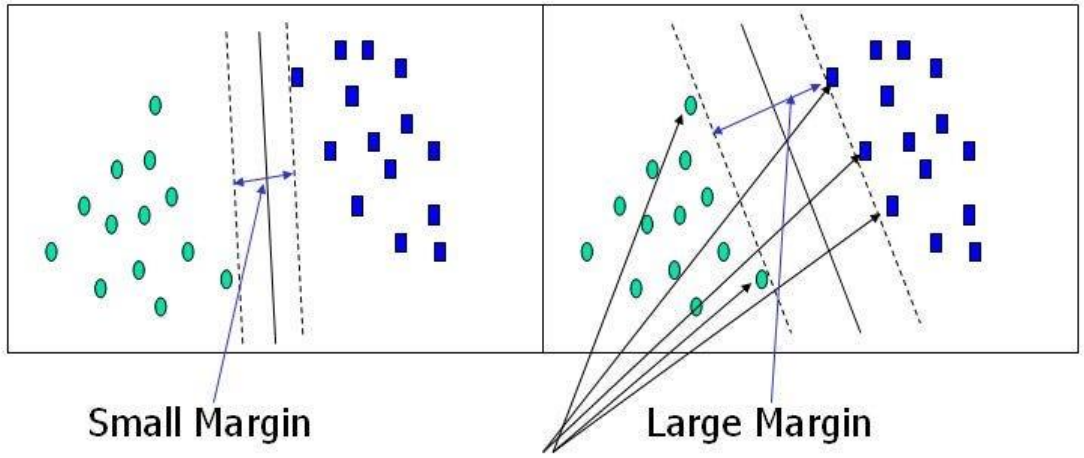
$$Margin = \left|\frac{\hat{y}_1}{\|\vec{w}\|}\right| + \left|\frac{\hat{y}_2}{\|\vec{w}\|}\right|$$

$$= \frac{1}{\|\vec{w}\|} + \frac{1}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|}$$



SVM Formulation

- Target of SVM:
  - Maximize the margin (to have the best centered decision boundary)
  - Assume that the closest points to the boundary from each class have $\hat{y}$ exactly equal to $\pm 1 \rightarrow$ ***Ass.3***
    - If we follow ***Ass. 3***, then ***Ass.1*** is already satisfied properly.
    - Ensures that all points are correctly classified and boundary is centered.



SVM Formulation

- In mathematical form, the SVM target can rewritten as follows:

$$maximize\ \frac{\mathbf{2}}{\|\vec{\boldsymbol{w}}\|},\ \longrightarrow\ Margin$$

$$such\ that, y_i(\vec{w}\ \vec{x_i} + b) \geq 1\ for\ all\ points\ \longrightarrow\ \underline{\textbf{\textit{Ass. 3 and Ass. 1}}}$$

- It's preferred to do minimization rather than maximization, and to have differentiable terms:
  - Take reciprocal
  - Use $\|\vec{\boldsymbol{w}}\|^{\mathbf{2}}$ instead of $\|\vec{\boldsymbol{w}}\|$

$$minimize\ \frac{1}{2}\|\vec{\boldsymbol{w}}\|^{\mathbf{2}},$$
$$such\ that, y_i(\vec{\boldsymbol{w}}\ \vec{x_i} + \boldsymbol{b}) \geq 1\ for\ all\ points$$

$\underline{\textbf{\textit{Target Function}}}$

SVM Loss Function

$$minimize \frac{1}{2} \|\vec{\boldsymbol{w}}\|^2,$$
$$such\ that, y_i(\vec{\boldsymbol{w}}\,\vec{x_i} + \boldsymbol{b}) \geq 1$$

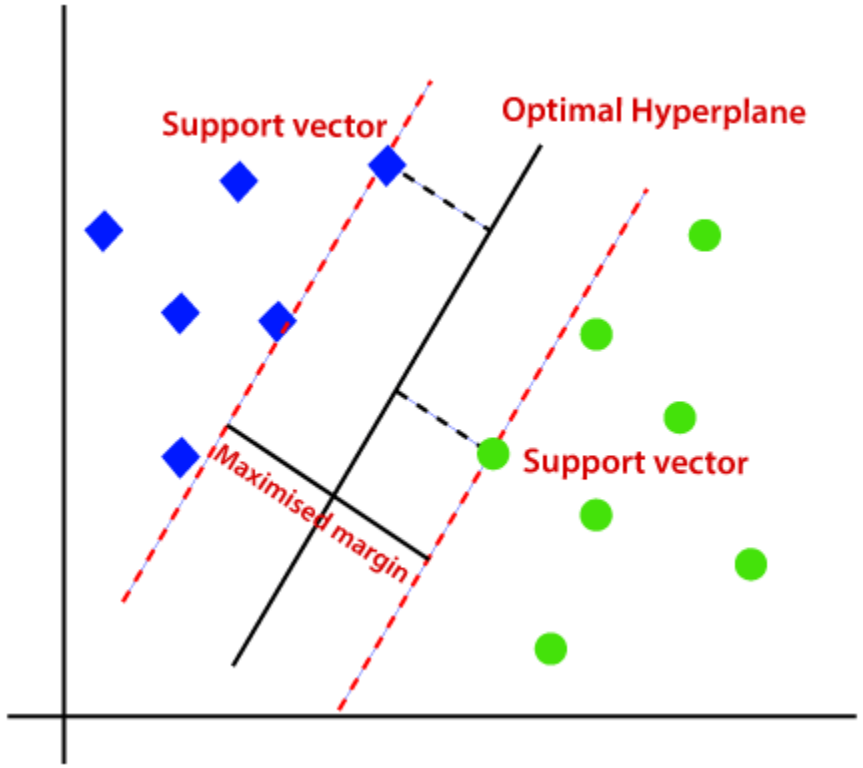
- To transform this condition-based target function to one loss (cost) function to be minimized, a concept called Lagrange Multipliers is used.

$$L(\vec{\boldsymbol{w}}, \boldsymbol{b}, \boldsymbol{\alpha_i}) = \frac{1}{2} \|\vec{\boldsymbol{w}}\|^2 - \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{\alpha_i}(y_i(\vec{\boldsymbol{w}}\,\vec{x_i} + \boldsymbol{b}) - \boldsymbol{1})$$

- This loss function can be minimized using Gradient Descent to reach an optimal decision boundary that follows the constraints and assumptions of SVM.

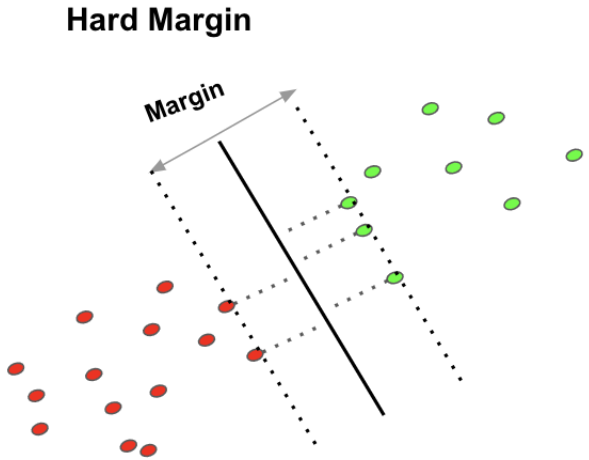
Support Vectors

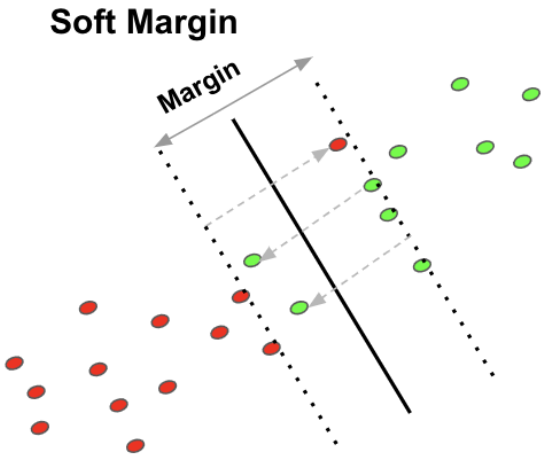
- The target function of the SVM algorithm is based on maximizing the margin which depends ***only*** on the closest points to the decision boundary.
- We call those closest points the **support vectors.**
- In other words, the decision boundary in an SVM algorithm is only affected by the support vectors, any other point in the dataset doesn't affect the final decision boundary.



Hard vs Soft Margin

- The formulation we defined so far assumes a **Hard Margin** which only works if the data is perfectly linearly separable.
  - In other words, maximizing the margin while classifying **all** the points correctly only works if perfect linear separation exists.
- In practice, datasets are not perfectly linearly separable and have some noise. To make the SVM more robust, we need to define a **Soft Margin** instead.
  - The soft margin allows some points to be misclassified by lying inside the margin or on the incorrect side of the decision boundary.





Soft Margin

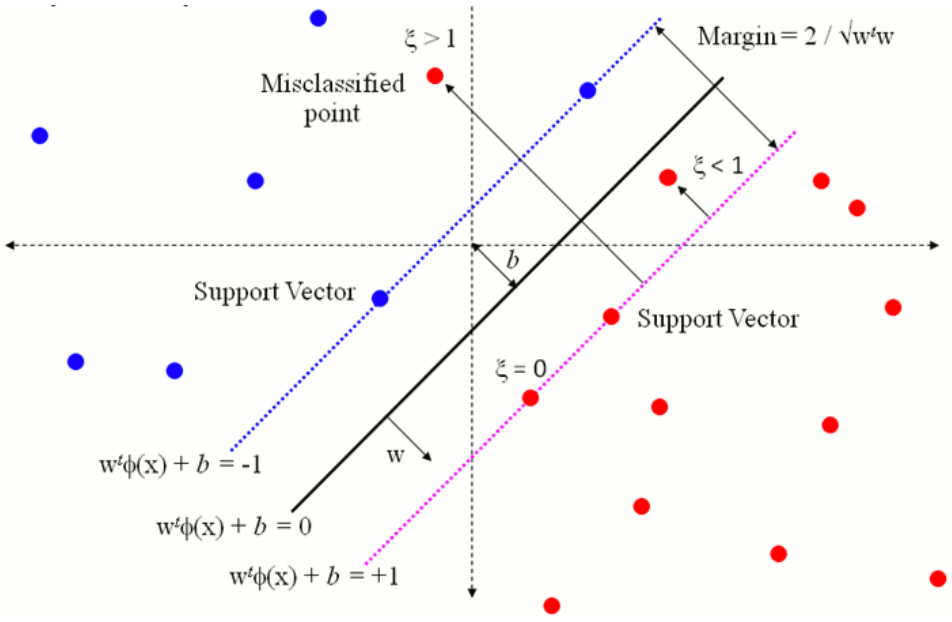
- To achieve a Soft Margin, we introduce a **slack variable** ξ_i for each point which represents the misclassification error of every point.

$$\xi_i = \begin{cases} 0\ , if\ x_i\ is\ beyond\ correct\ margin \\ 1 - y_i \widehat{\boldsymbol{y_i}}\ , if\ x_i\ opposite\ to\ margin \end{cases}$$

- In one equation:

$$\xi_i = \max(0, 1 - y_i \widehat{\boldsymbol{y_i}})$$

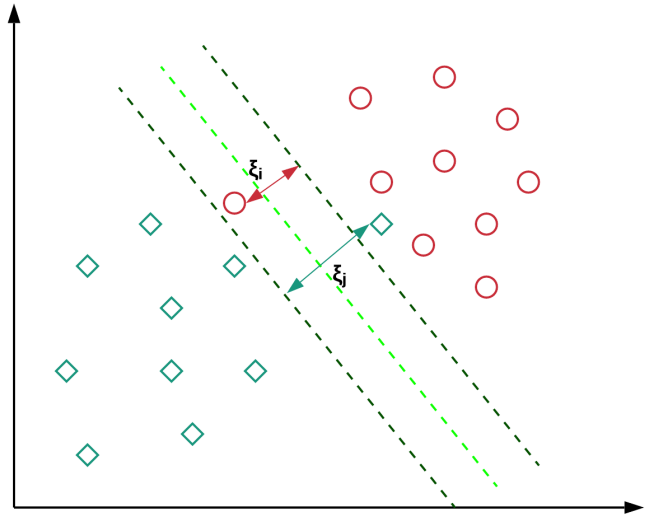
- Points that lie far from their correct margin in the opposite side have higher ξ_i indicating higher error.



Soft Margin: Target Function

- Hard Margin Target Function:
  - Maximizes the margin only
  - Constrains all points to strictly lie in the correct margin area.

$$minimize \frac{1}{2} \|\vec{\boldsymbol{w}}\|^2,$$
$$such\ that, y_i(\vec{\boldsymbol{w}}\,\vec{x_i} + \boldsymbol{b}) \geq 1$$



- Soft Margin Target Function:
  - Maximizes the margin
  - All points are not constrained to strictly lie in the correct margin area.
  - Minimizes the total misclassification error of all points after multiplying by a constant scalar value C.

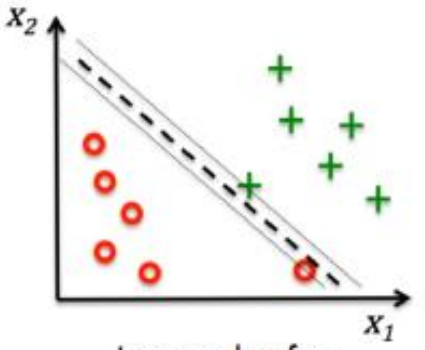
$$minimize \frac{1}{2} \|\vec{\boldsymbol{w}}\|^2 + C \sum_{i=1}^{N} \xi_i,$$
$$such\ that, y_i(\vec{\boldsymbol{w}}\,\vec{x_i} + \boldsymbol{b}) \geq 1 - \xi_i$$

Soft Margin: C Hyperparameter

- Soft Margin Target Function:

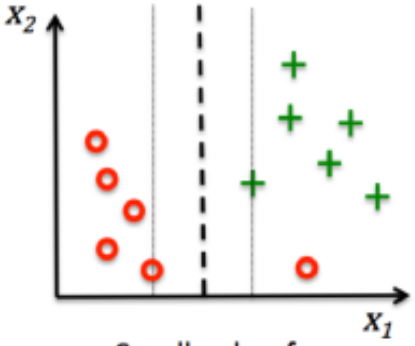
$$minimize\ \frac{1}{2}\|\vec{\boldsymbol{w}}\|^{\mathbf{2}} + C\sum_{i=1}^{N}\xi_i\,,$$

$$such\ that, y_i(\vec{\boldsymbol{w}}\,\vec{x_i} + \boldsymbol{b}) \geq 1 - \xi_i$$



Large value for parameter *C*

- The C value is a manually defined hyperparameter and is the most important parameter in the SVM algorithm.
- It affects how *strict* is the SVM algorithm in allowing misclassification errors. In other words, it controls the trade-off between having a maximized margin and minimum misclassified points.
  - Very Large C value → Prioritize minimizing errors → Very narrow margin.
  - Very Low C value → Prioritize maximizing the margin → More misclassified points



Small value for parameter *C*

Soft Margin: Loss Minimization

$$L(\vec{\boldsymbol{w}}, \boldsymbol{b}, \boldsymbol{\alpha_i}) = \frac{1}{2}\|\vec{\boldsymbol{w}}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}\boldsymbol{\alpha_i}(y_i(\vec{\boldsymbol{w}}\,\vec{x_i} + \boldsymbol{b}) - \mathbf{1} - \xi_i)$$

- As done in Logistic Regression, get the values of $\vec{\boldsymbol{w}}, \boldsymbol{b}$ at which $L(\vec{\boldsymbol{w}}, \boldsymbol{b})$ is minimum → Get the values of $\vec{\boldsymbol{w}}, \boldsymbol{b}$ at which $\frac{d(L)}{d\vec{\boldsymbol{w}}} = 0$, and $\frac{d(L)}{d\boldsymbol{b}} = 0$.
- After getting the derivatives, we substitute back in the loss function to get the final objective function (called the dual formulation) that could be solved using Gradient Descent.

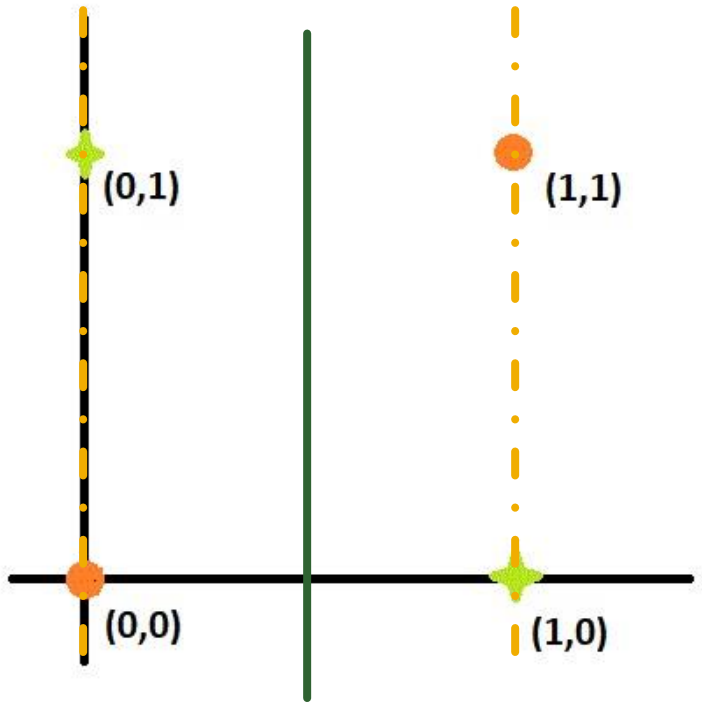
$$\max \; -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n}\alpha_i$$

$$\textbf{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0, \;\; 0 \le \alpha_i \le C$$

Dual Formulation

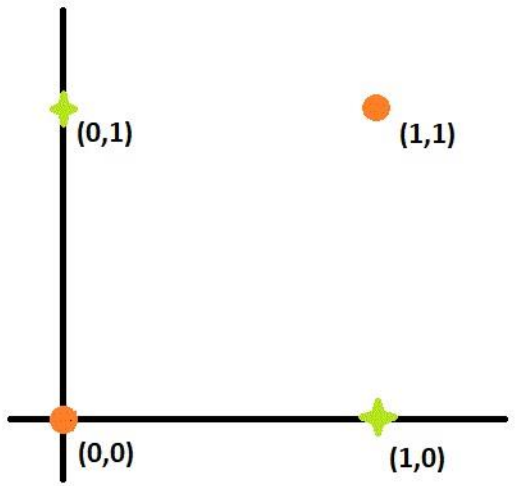
Non-Linearly Separable

- Recall the XOR problem below. Can the Soft Margin SVM algorithm correctly solve the problem?
- The Soft Margin SVM only gives us the ability to be robust to noise in the dataset by allowing some points to be misclassified.
- It doesn't result in a non-linear decision boundary which is needed to solve the XOR problem.

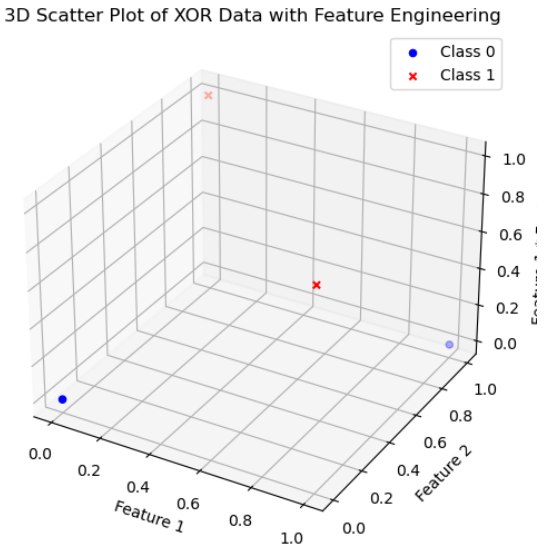


Non-Linearly Separable

- Non-linearly separable datasets could actually be linearly separable in a **higher** dimensional space.
- In the XOR problem, we could add a third dimension representing the operation $x\ XOR\ y$. The problem then becomes linearly separable in the higher dimension ($3D$).



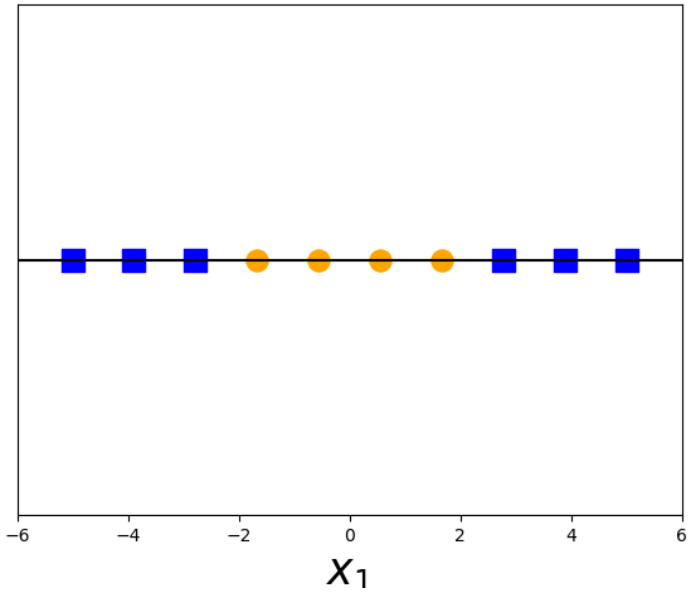
$[0, 0]$
$[0, 1]$
$[1, 0]$
$[1, 1]$

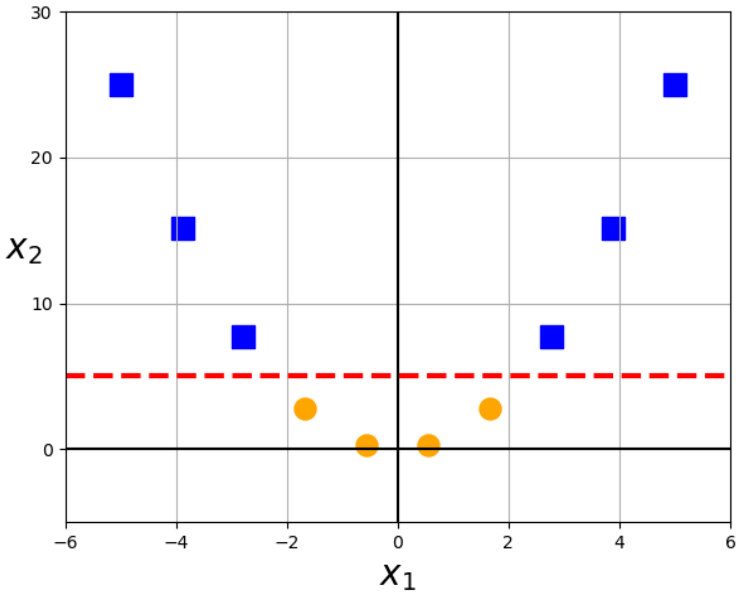


$[0, 0,0]$
$[0, 1,1]$
$[1, 0,1]$
$[1, 1,0]$

Another Example

- Find a non-linear transformation on the 1-D dataset below to transform it to a linearly separable version in a higher dimensional space.
- Solution: In $2D$ Space, $y = x^2$





Kernels

- A kernel function can be used to raise data to higher dimensional feature spaces.
- They are basically functions that take two vectors as input and returns the equivalent for the dot product of their ***transformed*** versions in higher dimensional spaces.

$$k(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}).\phi(\boldsymbol{y})$$

- Examples:
  - Polynomial Kernel:

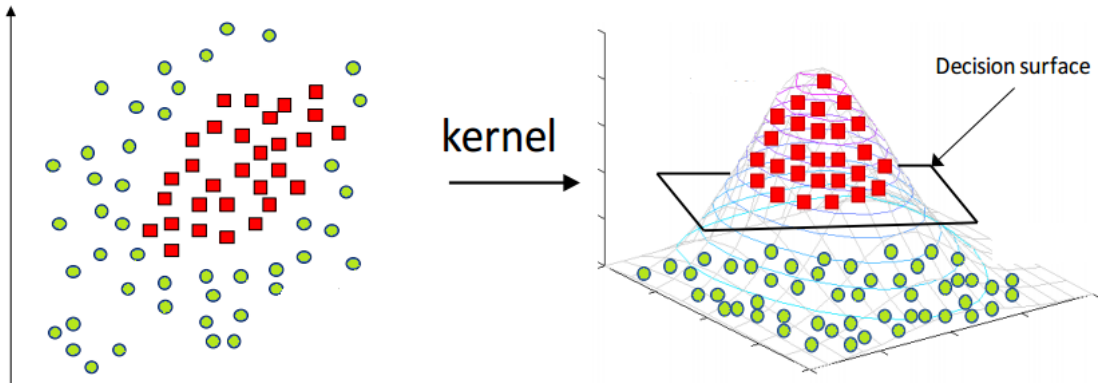
$$k(\boldsymbol{x}, \boldsymbol{y}) = \left[1 + \left(\mathbf{x}^{\mathrm{T}}.\mathbf{y}\right)\right]^{\mathrm{d}}$$

  - Gaussian Radial Basis Function:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-|\boldsymbol{x} - \boldsymbol{y}|^2/2\sigma^2)$$

  - Sigmoid Functions:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \tanh(c\mathbf{x}^{\mathrm{T}}.\mathbf{y} + h)$$



Kernel Trick

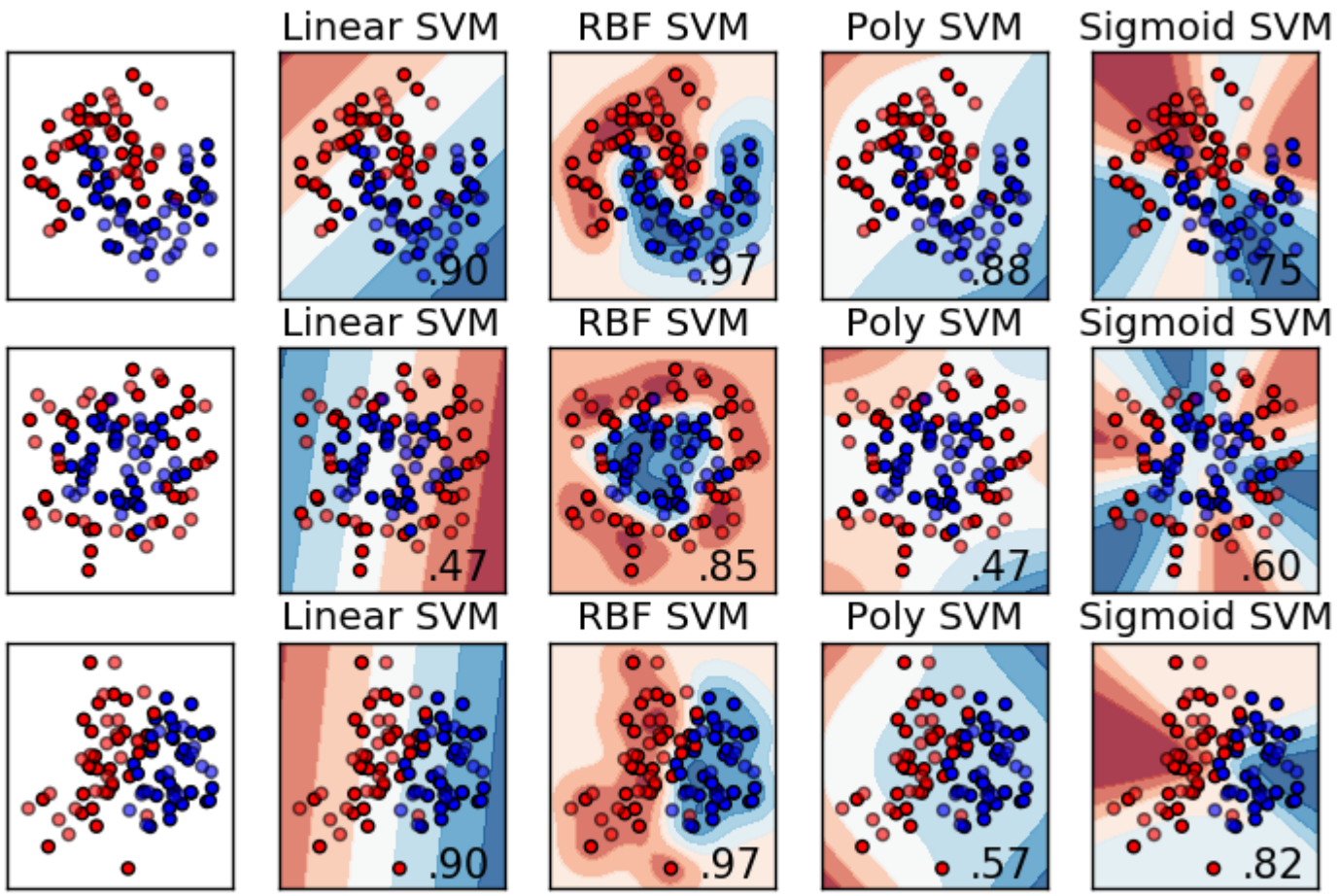
- Since kernels basically mimic the dot product operation in higher dimensionality spaces, how can they be used in SVM?
- Recall the dual formulation which is the core objective function to be optimized in SVMs:

$$\max \; -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n}\alpha_i$$

$$\textbf{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0, \;\; 0 \le \alpha_i \le C$$

- The dual formulation already contains a dot product between the points in the original feature space →$x_i^T . x_j$
- The Kernel Trick replaces this dot product with a kernel function $k(x_i, x_j)$. This allows us to solve the objective function (dual formulation) in **higher** dimensional feature space.
- If a dataset was non-linearly separable in the original dimensions, it could potentially be linearly separable in higher dimensionalities.

Kernel Trick



Extra Resources

- More information regarding deriving the dual formulation and Lagrangian multipliers:
  - https://www.adeveloperdiary.com/data-science/machine-learning/support-vector-machines-for-beginners-duality-problem/#lagrange-multiplier

Thank you!

- Any questions?

Disclaimer

Due to nature of the course, various materials have compiled from different open source resources with some moderation. I sincerely acknowledge their hard work and contribution

CONESTOGA

Thank You

Youssef Abdelkareem

yabdelkareem@conestogac.on.ca

**** Credits for Dr. Anasuya Bhima for assisting in the slides structure***