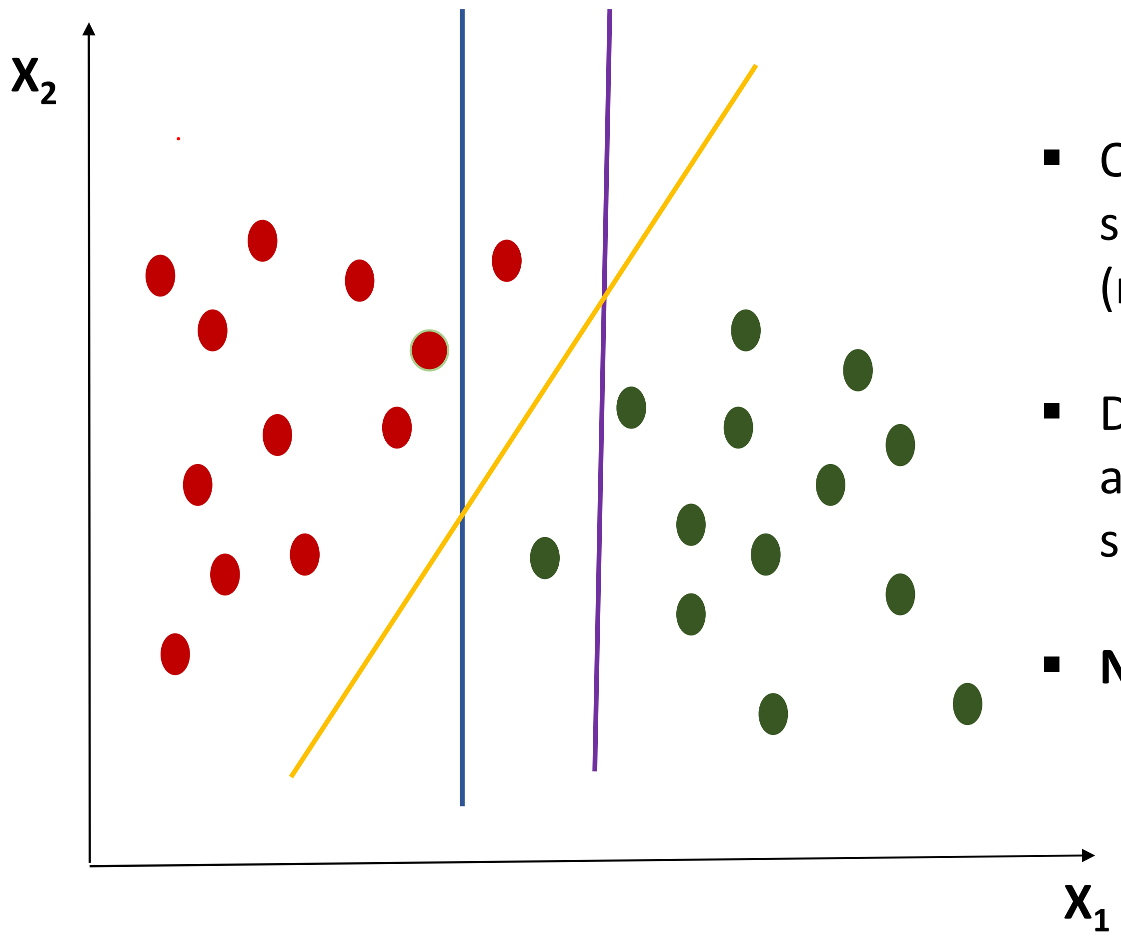
Support Vector Machine

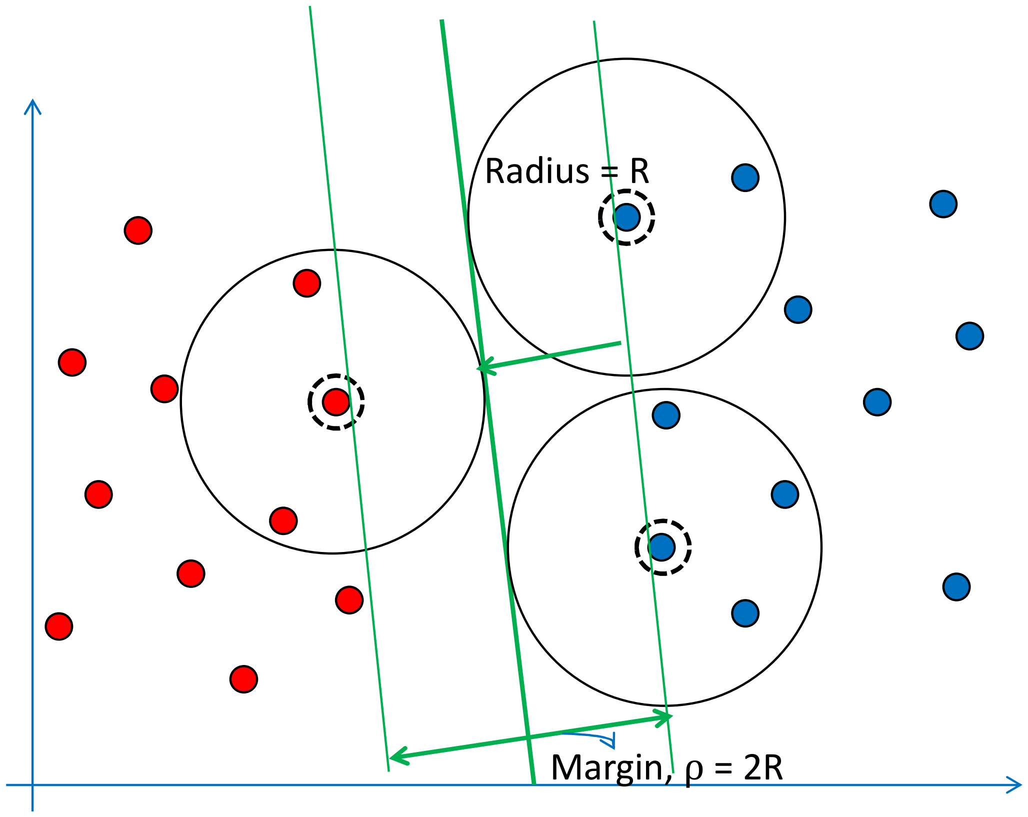
Not very parameters to fiddle with

Margin Classifier



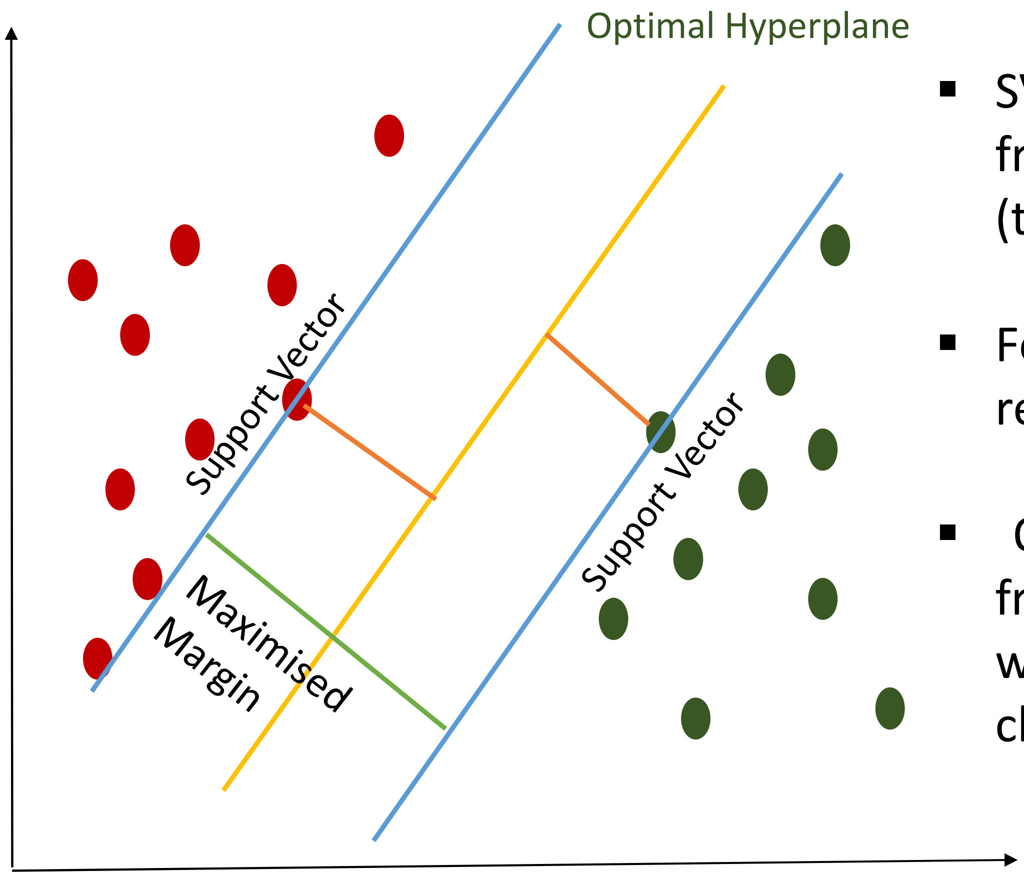
- Our task is to find an ideal line that separates this dataset in two classes (red and green)
- Design an algorithm that takes the data as an input and outputs a line that separates those classes if possible
- **Not a big task, right?**

Formalizing the Margin



Samples that support the Decision boundary are called ***Support Vectors***

How SVM finds the best line!



- SVM finds the minimum distance of the frontier from the closest support vector (this can belong to any class).
- For instance, blue frontier is closest to red/green circles
- Once we have these distances for all the frontiers, we simply choose the frontier with the maximum distance (from the closest support vector)

SVM learns the best hyperplane by maximising the margin(the one that lies in the middle of the separation between the data). Therefore, it is called the maximum margin classifier.

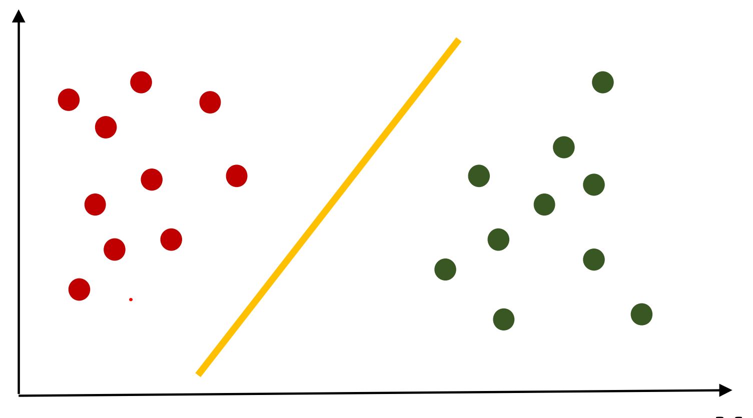
SVM vs MLP

- Input: set of (input, output) training pair samples; call the input sample features $x_1, x_2, \dots, x_n$ and the output result y.
- Output: set of weights w (or w_i), one for each feature, whose linear combination predicts the value of y. (Similar for both SVM and MLP)
- Important difference: We use the optimization of maximizing the margin (‘width’) to reduce the number of weights that are nonzero to just a few that correspond to the important features that ‘matter’ in deciding the separating line (hyperplane)…these nonzero weights correspond to the support vector

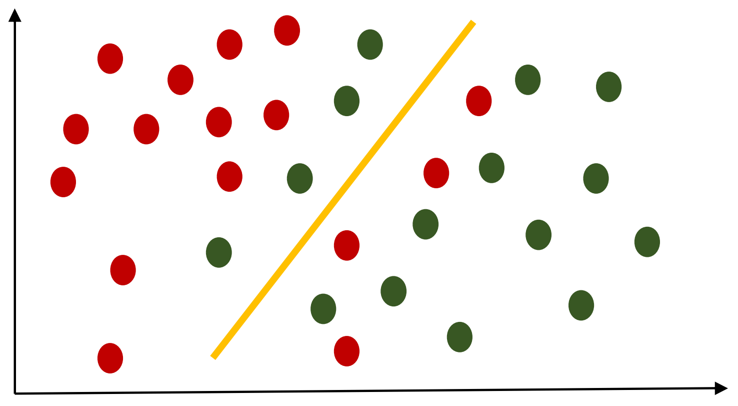
Types of SVM

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line

- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data

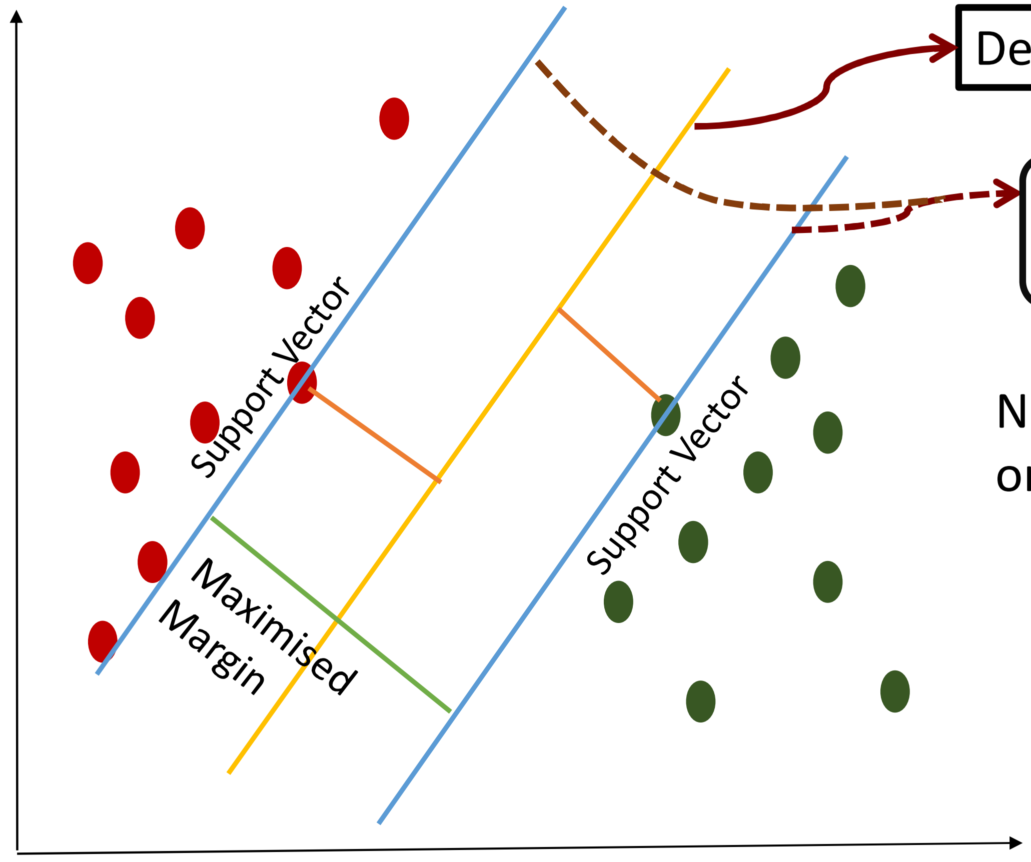


Linear SVM



Non-linear SVM

Formalizing the Margin



Dec. Boundary: $\mathbf{w}^T\mathbf{x} + \mathbf{b} = 0$

The parallel hyperplanes will be: $\mathbf{w}^T\mathbf{x} + \mathbf{b} = \pm\varepsilon$

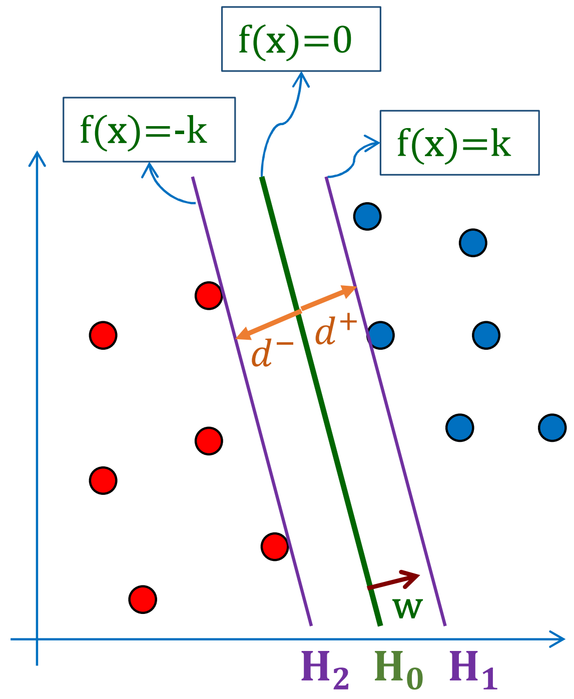
Note: The value of $\mathbf{w}^T\mathbf{x}_i + \mathbf{b}$ is dependent on the scale of $\mathbf{w}$

Formulation

- Let $f(x)=\mathbf{w}^T\mathbf{x}+b$.
- We want to maximize k such that:
  - $\mathbf{w}^T\mathbf{x}_i+b \geq k$ for $y_i=1$
  - $\mathbf{w}^T\mathbf{x}_i+b \leq -k$ for $y_i=-1$

- Value of $f(\mathbf{x})$ dependents on $\|\mathbf{w}\|$:
  1. Keep $\|w\|=1$, and maximize $f(x)$, or
  2. Let $|f(x)|\geq 1$, and minimize $\|w\|$.

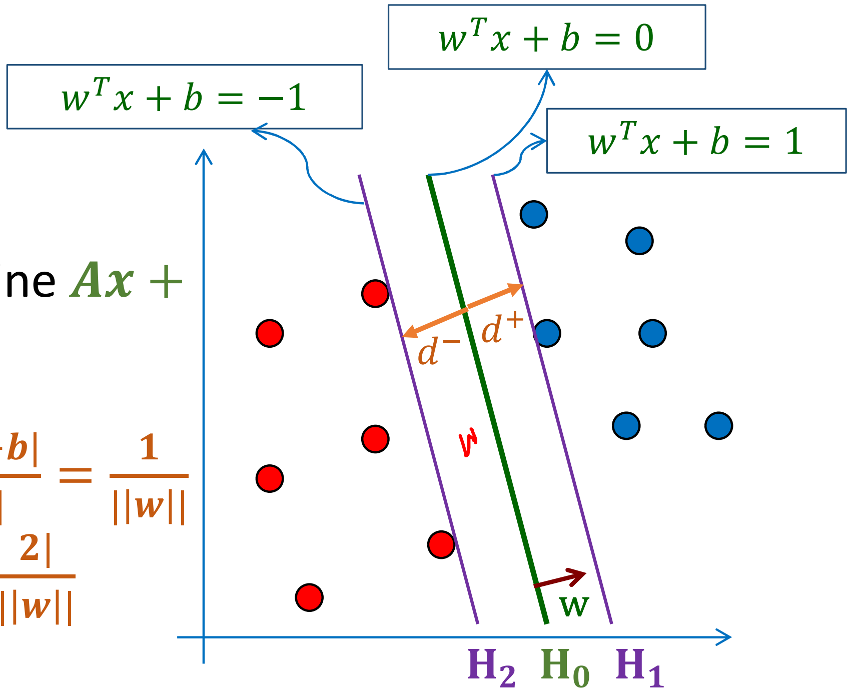
- We use approach (2) and formulate the problem as:
  - Minimize: $\frac{1}{2}\mathbf{w}^T\mathbf{w}$, and
  - $y_i(w^Tx_i+b) \geq 1$, for $i=1..N$



d^+ = the shortest distance to the closest positive point
d^- = the shortest distance to the closest negative point
The margin of the separating hyperplane is $d^+ + d^-$

Maximizing the Margin

- We want a classifier (linear separator) with as big a margin as possible.
- Recall the distance from a point (x_0, y_0) to a line $\boldsymbol{Ax + By + c = 0}$ is $\frac{|\boldsymbol{Ax_0 + By_0 + c}|}{\sqrt{\boldsymbol{A^2 + B^2}}}$,
- The distance between H_0 and H_1 is then $\frac{|\boldsymbol{w^T x + b}|}{||\boldsymbol{w}||} = \frac{\boldsymbol{1}}{||\boldsymbol{w}||}$
- The total distance between H_1 and H_2 is thus $\frac{\boldsymbol{2|}}{||\boldsymbol{w}||}$



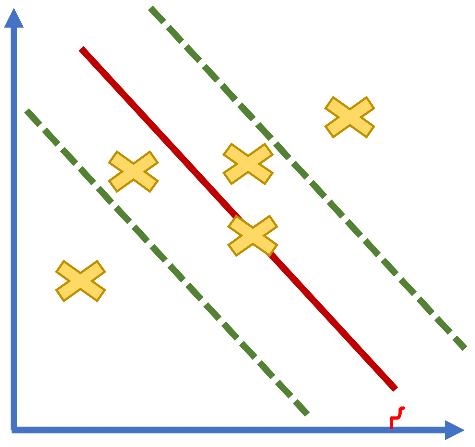
- In order to maximize the margin, we thus need to minimize $||\boldsymbol{w}||$. With the condition that there are no datapoints between H_1 and H_2 :
  - $w^T x + b \geq 1$ when $y_i = +1$
  - $w^T x + b \leq 1$ when $y_i = -1$

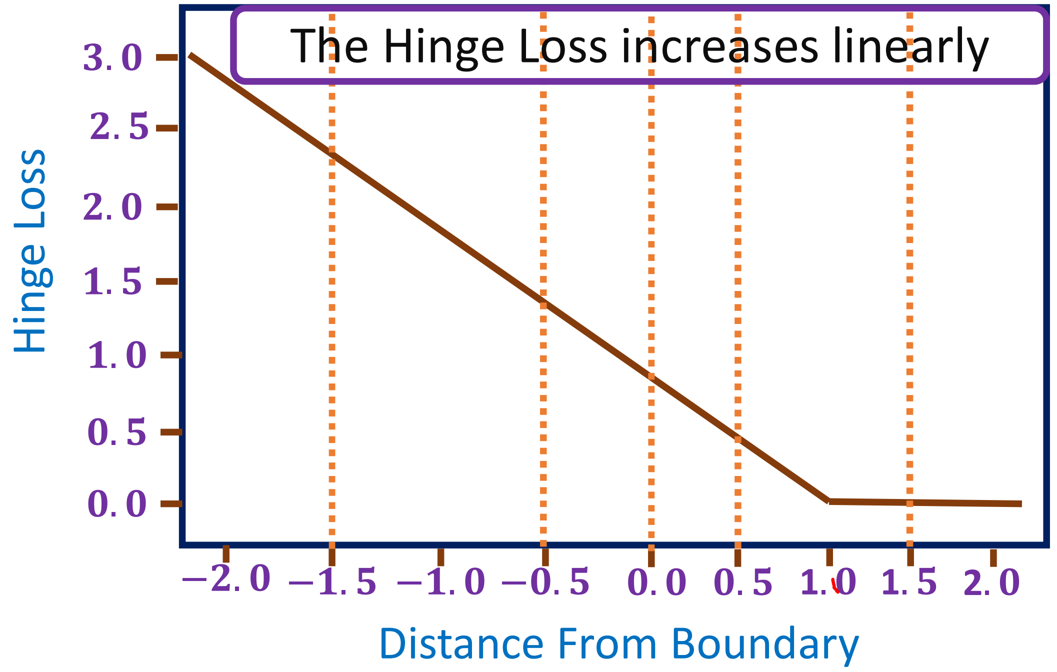
Can be combined into: $\boldsymbol{y_i\ (w^T x_i) \geq 1}$

Hinge Loss

- The hinge loss incorporates a margin or distance from the classification boundary into the cost calculation

- New observations even if classified correctly can incur a penalty if the margin from the decision boundary is not large enough





Hinge Loss Formula

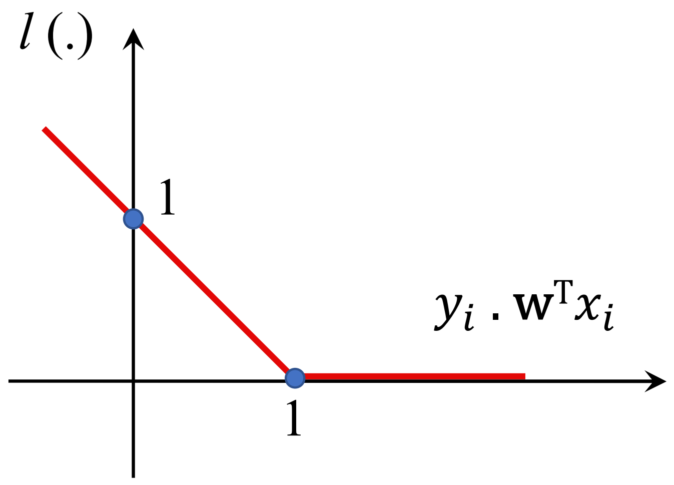
- Hinge Loss:

  $\frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} l(w;(x_i, y_i))$, (x_i s are augmented)

  Where,

  $$l(w;(x_i, y_i)) = max\{0, 1 - y_i\langle w, x_i\rangle\}$$

  - $\langle w, x_i\rangle$ denotes the inner product of vectors
  - Note: For correctly classified samples $y_i\langle w, x_i\rangle \geq 1$
  - λ is the relative importance of margin and training loss.



Stochastic GD to Maximize Margin

- Loss for a random sample $(\boldsymbol{x}_i, y_i)$ at iteration $\boldsymbol{t}$:

$$J_m(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}_t\|^2 + l(\mathbf{w}_t; (x_i, y_i)),$$

Where,

$$l(\mathbf{w}_t; (x_i, y_i)) = max\{0, 1 - y_i\langle \mathbf{w}_t, x_i\rangle\}$$

$$\nabla J_m = \lambda \mathbf{w}_t - \mathbb{I}(y_i\langle \mathbf{w}_t, x_i\rangle < 1) y_i x_i$$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla J_m,$$

where $\eta_t = \frac{1}{\lambda t}$

The Pegasos Algorithm

Reference: "Pegasos: Primal Estimated sub-Gradient Solver for SVM", Mathematical Programming, Series B, 127(1):3-30, 2011. Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, Andrew Cotter,

- INPUT: Training data: **S**, λ, Max Iterations: **T**
- INITIALIZE: $\mathbf{w}_1$
- FOR $t = 1, 2, 3 \ \dots \ \mathrm{T}$
  - Choose $i = \{1, 2, \dots \ |\mathrm{S}|\}$ at random and pick sample (x_i, y_i)
  - Set $\eta_t = \frac{1}{\lambda t}$
  - IF $(y_i \langle \mathrm{w}_t, x_i \rangle < 1)$:
    - Update: $\mathrm{w}_{t+1} \leftarrow \mathrm{w}_t - \eta_t \lambda \mathrm{w}_t + \eta_t y_i x_i$
  - ELSE
    - Update: $\mathrm{w}_{t+1} \leftarrow \mathrm{w}_t - \eta_t \lambda \mathrm{w}_t$
- OUTPUT: $\mathbf{w}_{T+1}$

$$\nabla J_m = \lambda \mathrm{w}_t - \mathbb{I}(y_i \langle \mathrm{w}_t, x_i \rangle > 1) y_i x_i$$

SVM Formulation

- Need to maximize relative margin (not absolute)
  - Minimize $|\boldsymbol{w}|$ for fixed margin or maximize margin for $|w| = 1$.
  - Training samples should be outside margin.
- i.e., Minimize $\mathbf{J(w)} = \frac{1}{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}$, subject to: $\forall_{i},\ \boldsymbol{y_i}(\mathbf{w}^{\boldsymbol{T}}\boldsymbol{x_i} + \boldsymbol{b}) > \mathbf{1}$
- This is a constrained quadratic optimization problem
  - Can be solved by the Lagrangian multipler method
    1. Use the method of Lagrange Multipliers (α) to modify $\mathrm{J}(\mathbf{w})$ to $\mathrm{Q}(\boldsymbol{\alpha})$
    2. Use QP Solver to get α;
    3. Use α to find $\mathbf{w}$.

Learning a Linear SVM

- Convert the constrained minimization to an unconstrained optimization problem - represent constraints as penalty terms:

$$\min_{\boldsymbol{w},b} \frac{1}{2}||\boldsymbol{w}||^2 + \text{penalty\_term}$$

- For data $\{(x_i, y_i)\}_{i=1}^{N}$, use the following penalty

$$\max_{\alpha_i \geq 0} \alpha_i[1 - y_i(\boldsymbol{w}^T x_i + b)] = \begin{cases} 0 & \text{if } \; y_i(\boldsymbol{w}^T x_i + b) \geq 1 \\ \infty & \text{otherwise} \end{cases}$$

- Rewrite the minimization problem:

$$\min_{\boldsymbol{w},b} \frac{1}{2}||\boldsymbol{w}||^2 + \sum_{i=1}^{N} \max_{\alpha_i \geq 0} \alpha_i[1 - y_i(\boldsymbol{w}^T x_i + b)]$$

- Where $\boldsymbol{\alpha_i}$ are the Lagrange Multipliers

$$= \min_{\boldsymbol{w},b} \max_{\alpha_i \geq 0} \left\{ \frac{1}{2}||\boldsymbol{w}||^2 + \sum_{i=1}^{N} \alpha_i[1 - y_i(\boldsymbol{w}^T x_i + b)] \right\}$$

Solution to Linear SVM

- Let: $$J(\boldsymbol{w}, b; \alpha) = \frac{1}{2}||\boldsymbol{w}||^2 + \sum_{i=1}^{N} \alpha_i[1 - y_i(\boldsymbol{w}^T x_i + b)]$$
- Swap the "max" and "min": This is a lower bound

$$\max_{\alpha_i \geq 0} \min_{\boldsymbol{w}, b} J(\boldsymbol{w}, b; \alpha) \leq \min_{\boldsymbol{w}, b} \max_{\alpha_i \geq 0} J(\boldsymbol{w}, b; \alpha)$$

- Equality holds in certain conditions
- Solving: $$\max_{\alpha_i \geq 0} \min_{\boldsymbol{w}, b} J(\boldsymbol{w}, b; \alpha) = \max_{\alpha_i \geq 0} \min_{w, b} \left(\frac{1}{2}||\boldsymbol{w}||^2 + \sum_{i=1}^{N} \alpha_i[1 - y_i(\boldsymbol{w}^T x_i + b)] \right)$$
- First minimize J() w.r.t w, b for fixed Lagrange multipliers:

$$\frac{\partial J(\boldsymbol{w}, b; \alpha)}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{N} \alpha_i y_i x_i = 0$$

$$\Longrightarrow \quad \boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$\frac{\partial J(\boldsymbol{w}, b; \alpha)}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0$$

Solution to Linear SVM

$$\max_{\alpha_i \geq 0} \min_{\boldsymbol{w},b} J(\boldsymbol{w}, b; \alpha) = \max_{\alpha_i \geq 0} \min_{w,b} \left(\frac{1}{2} ||\boldsymbol{w}||^2 + \sum_{i=1}^{N} \alpha_i [1 - y_i(\boldsymbol{w}^T x_i + b)] \right)$$

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i x_i, \text{and}, -\sum_{i=1}^{N} \alpha_i y_i = 0$$

- Substituting back we get final optimization:

$$Q(\alpha) = \max_{\alpha_i \geq 0} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j \left(x_i^T x_j \right) \right\}$$

- Train SVM by maximizing $Q(\alpha)$ subject to $\alpha_i \geq 0$; $\sum_{i=1}^{N} \alpha_i y_i = 0$

SVM Solution

1. Maximize:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i{}^{\mathrm{T}} \mathbf{x}_j$$

2. Use QP Solver to get α_i s.

3. $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$; (very few α_i s are non-zero: support vectors)

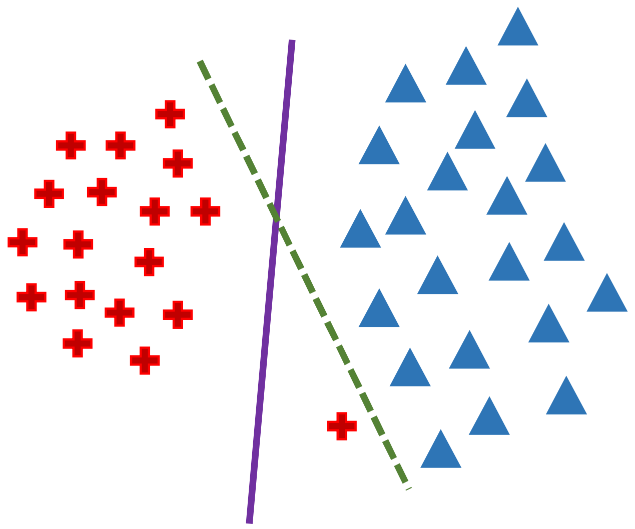
4. $\mathbf{b} = \mathbf{1} - \mathbf{w}^{\mathrm{T}} \mathbf{x}_{s+} = \mathbf{1} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i{}^{\mathrm{T}} \mathbf{x}_{s+}$

Note: Classification:

$$\mathbf{f}(\mathbf{x}_t) = \mathbf{w}^{\mathrm{T}} \mathbf{x}_t + \mathbf{b} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i{}^{\mathrm{T}} \mathbf{x}_t + b$$

Geometry of Data

- Maximizing the margin is a good idea as long as we assume that the classes are linear separable and that the data is noise free.
- If data is noisy, we might be sacrificing generalizability in order to minimize classification error with a very narrow margin
- With every decision boundary, there is a trade-off between maximizing margin and minimizing the error.



SVC – Soft Margin

- We want to balance maximizing the margin and minimizing the error, we modify the objective function that takes both into account:

$$\begin{cases} \min_{w,b} ||w||^2 + \lambda \,\mathrm{Error}(w,b) \\ \text{such that } y_i(w^T x_i + b) \geq 1, \; i = 1, \dots, N \end{cases}$$

where λ is an intensity parameter.

- How should we compute the error for a given decision boundary?
- The support vectors have distance $1/\|w\|$ to the decision boundary.
  - **(margin violation)** points that are on the correct side of the boundary but are inside the margin. They have distance $\xi/\|w\|$, where $0 < \xi < 1$.
  - **(misclassification)** points that are on the wrong side of the boundary. They have distance $\xi/\|\mathrm{w}\|$, where $\xi > 1$.
- Specifying a nonnegative quantity for ξ_i is equivalent to quantifying the error on the point x_i.

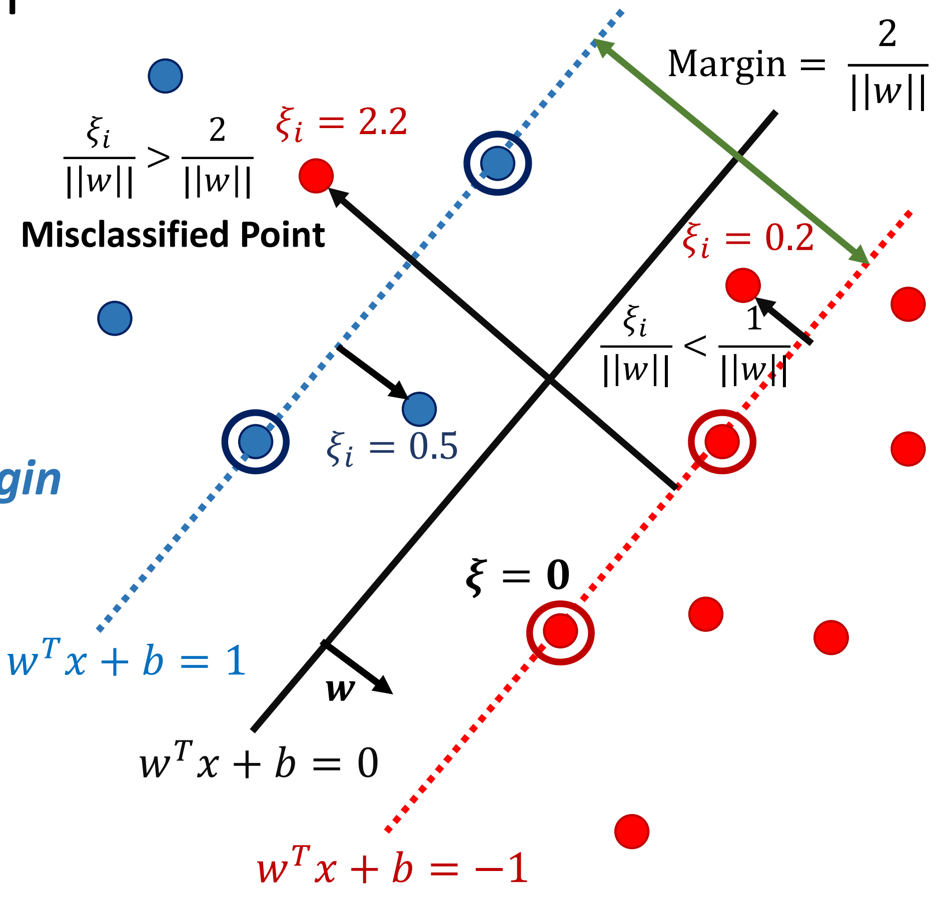
SVC: Soft Margin Illustration

- The error terms $\xi_N'^s$ are incorporated into our optimization problem by:

$$\begin{cases} \min_{w,b} ||w||^2 + \lambda \sum_{i=1}^{N} \xi_i \\ \text{such that } y_i(w^T x_i + b) \geq 1 - \xi_i,\ i = 1, \dots, N \end{cases}$$

- The solution to this problem is called ***soft margin support vector classification***

- small λ/C penalizes errors less and hence the classifier will have a large margin
- large λ/C penalizes errors more and hence the classifier will accept narrow margins to improve classification
- setting $\lambda = \infty$ produces the hard margin solution

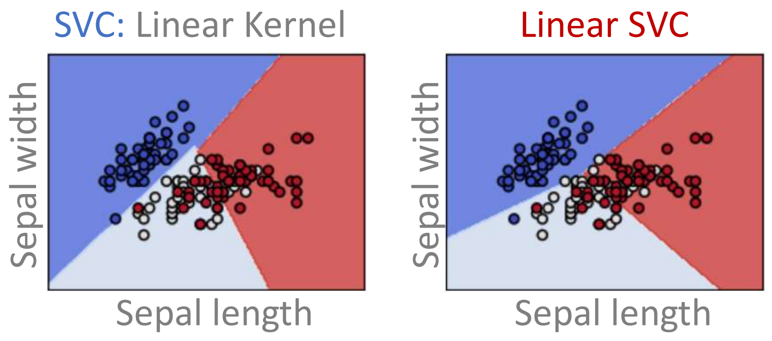


Scikit Learn Implementation

Two Popular implementations:

- libsvm, liblinear
- sklearn.svm.SVC, sklearn.svm.LinearSVC
- Multiclass: 1-vs-1 , 1-vs-rest

- Usage:



```
>>>  from sklearn import svm
>>>  X = [[0, 0], [1, 1]]
>>>  y = [0, 1]
>>>  clf = svm.SVC()
>>>  clf.fit(X, y)
>>>  clf.predict([[2., 2.]])
```

```
>>>  # get support vectors
>>>  clf.support_vectors_
>>>  # get support vector indices
>>>  clf.support_
```

https://scikit-learn.org/stable/modules/svm.html