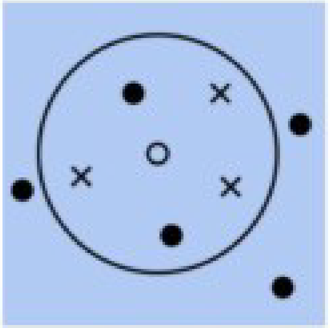
Support Vector Machines I

COMPCSI 762
Instructor: Thomas Lacombe
Based on slides from Meng-Fen Chiang

WEEK 9

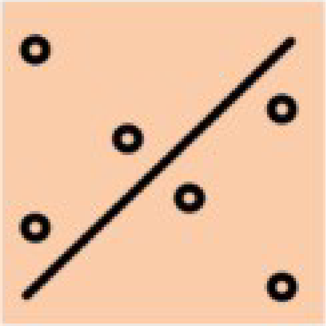
RECAP: Machine Learning Systems

- Instance-based Learning
  - Compare new data points to known data points
  - Non-parametric approaches
  - Memory-based approaches
  - Prediction can be expensive



use the entire dataset as a model (e.g., k-NN)

- Model-based Learning
  - Detect a pattern in the training data
  - Build a predictive model
  - Prediction is extremely fast



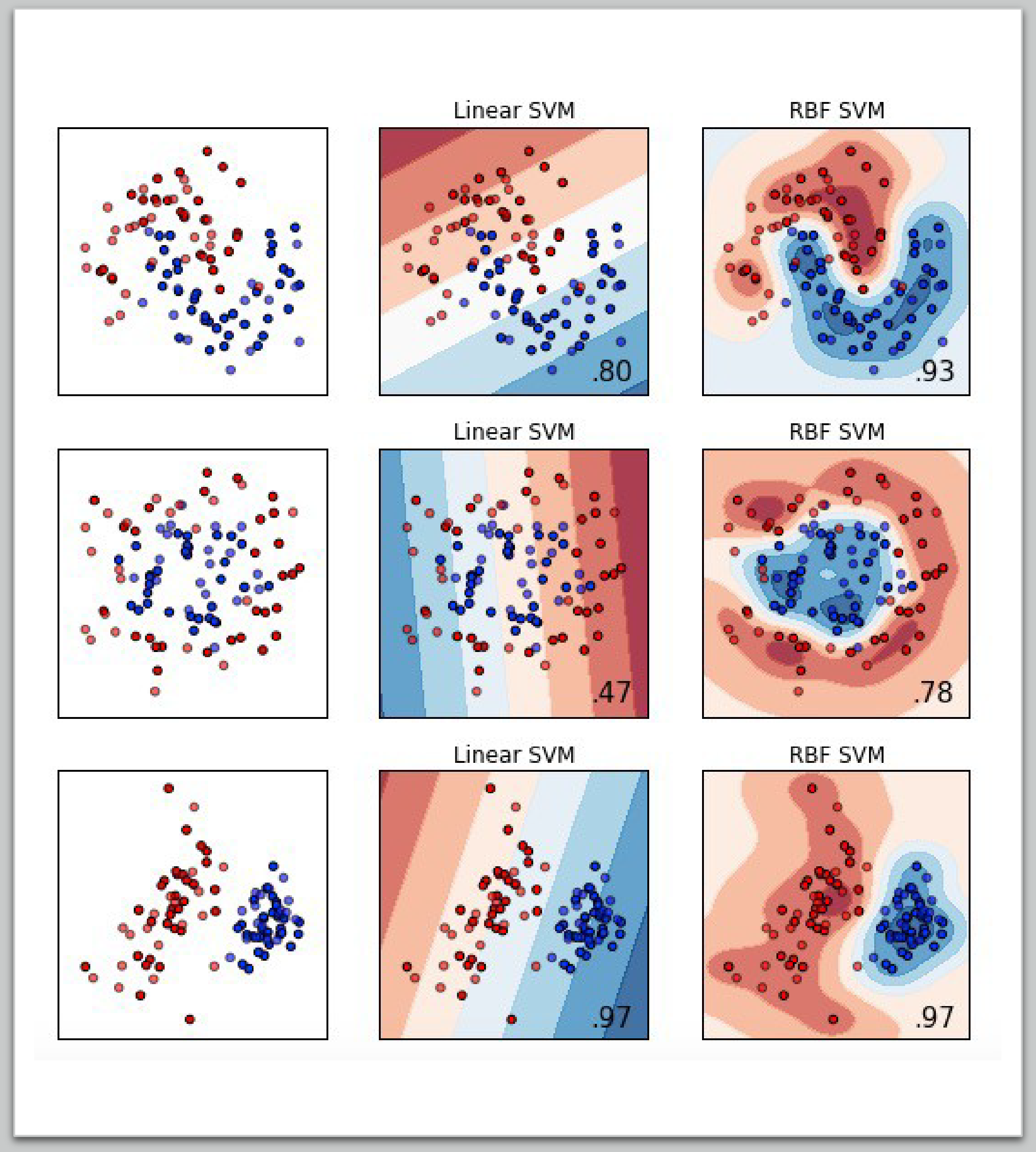
use the training data to create a model that has parameters learned from the training datasets (e.g., SVM)

OUTLINE

- Data Characteristics
  - Linearly Separable Data
  - Non-Linearly separable Data

- SVM (9.1,9.2,9.3)
  - Linearly Separable: Hard-margin SVMs
  - Non-Linearly Separable: Soft-margin SVMs
  - Non-Linearly Separable: Kernelized SVMs

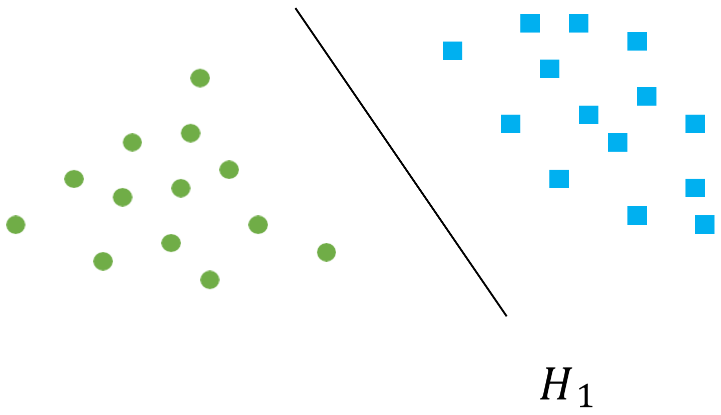
- Summary



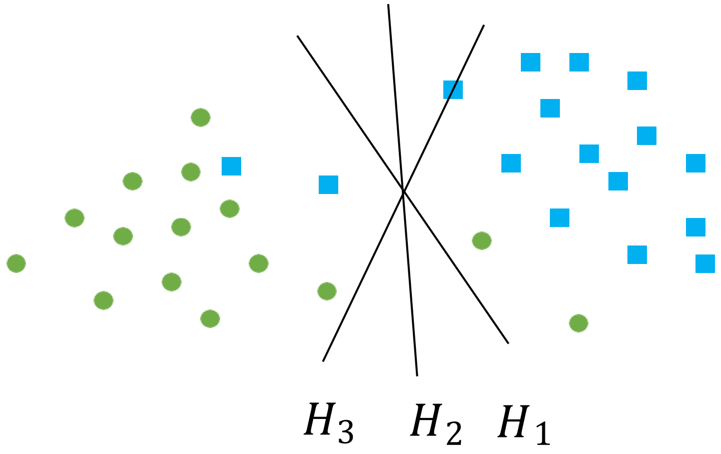
Data Characteristics

- A classification method for both linear and nonlinear data

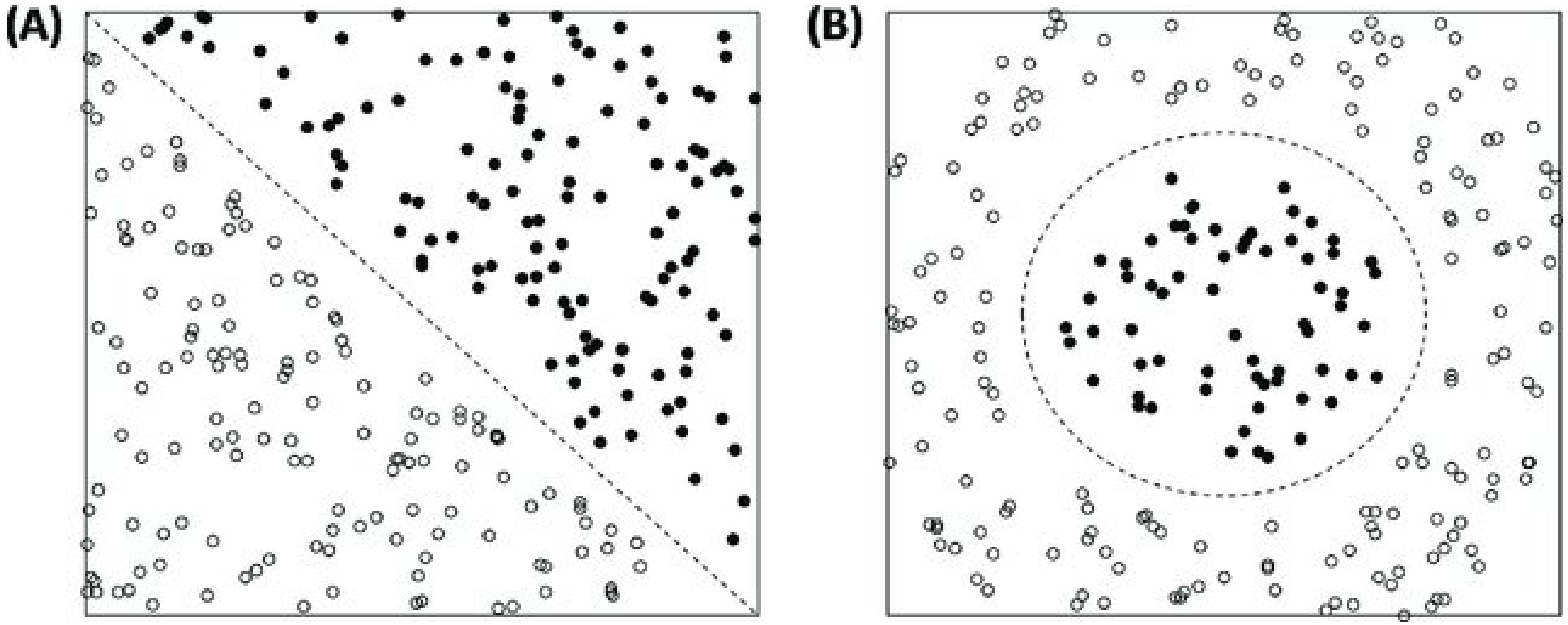
Linearly Separable Data



Non-Linearly Separable Data



Data Characteristics



Linearly Separable Data

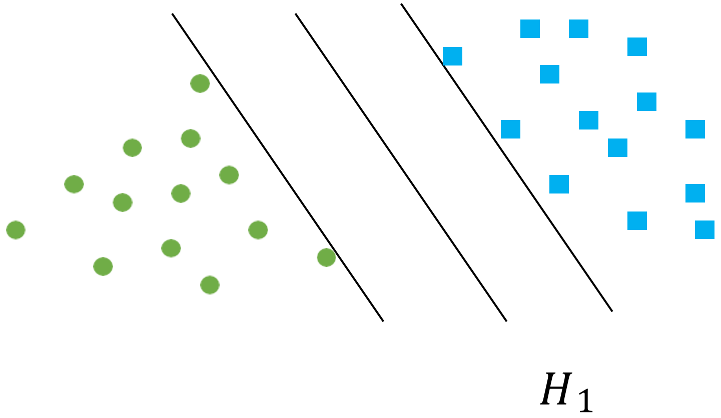
Non-Linearly Separable Data

Image source: [Neural networks differentiate between Middle and Later Stone Age lithic assemblages in eastern Africa, 2020]

Types of Support Vector Machines (SVMs)

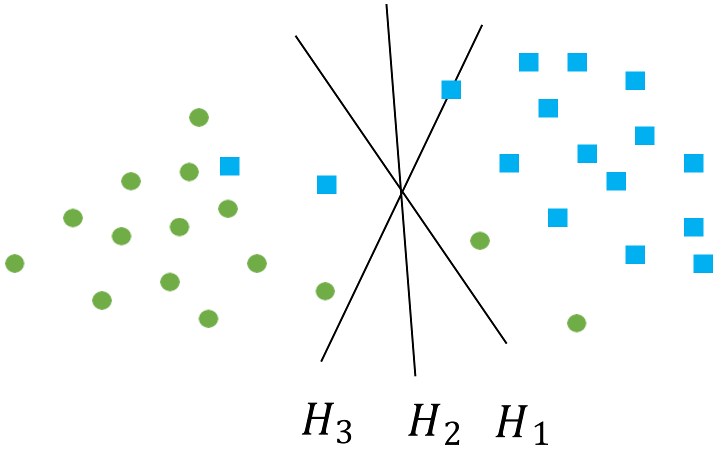
- SVM selects the maximum margin linear classifier

Linear SVMs: Hard-margin SVMs



- SVM selects the maximum margin linear classifier with partial misclassifications allowed

Linear SVMs: Soft-margin SVMs
Non-Linear SVMs: Kernelized SVMs

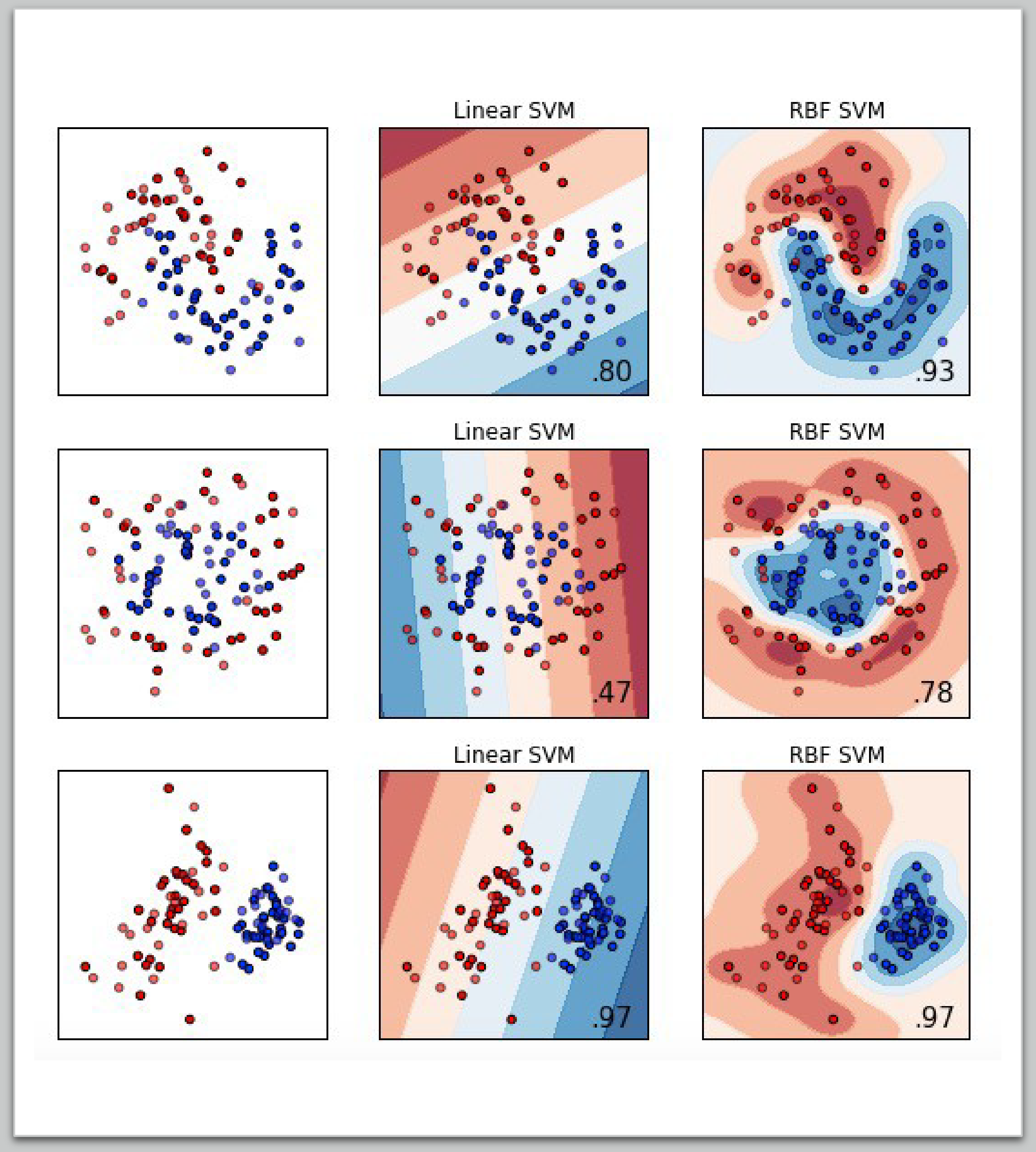


OUTLINE

- Data Characteristics
  - Linearly Separable Data
  - Non-Linearly separable Data

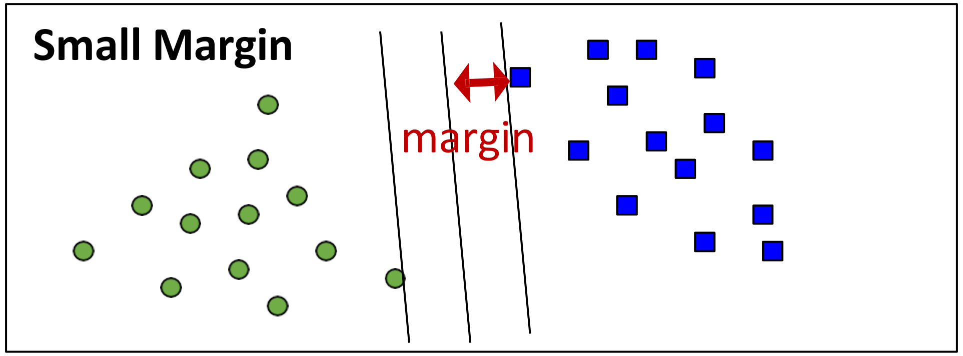
- SVM
  - Linearly Separable Data: Hard-margin SVMs (9.1)
  - Non-Linearly Separable Data: Soft-margin SVMs (9.2)
  - Non-Linearly Separable Data: Kernelized SVMs (9.3)

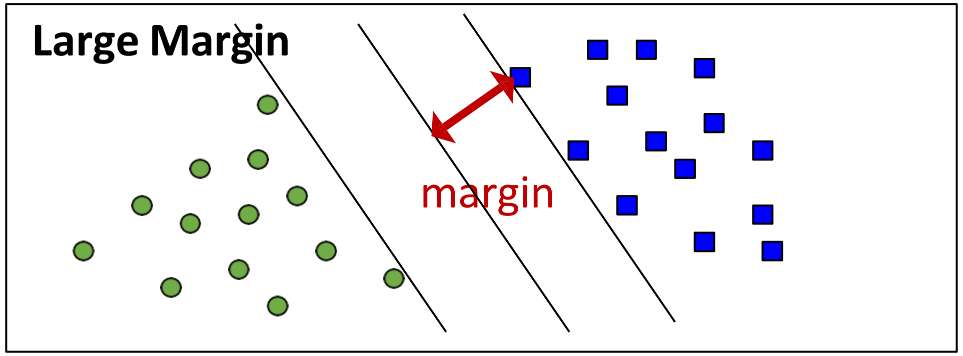
- Summary



Problem Definition: Margin Maximization

- Given a set of linearly separatable training data $S = \big((x_1, y_1), \dots, (x_n, y_n)\big), y_i \in \{+1, -1\}$
- We want to find a linear decision boundary (hyperplane) to separate the 2 classes.
- There is an infinite number of lines (hyperplanes) separating the two classes!

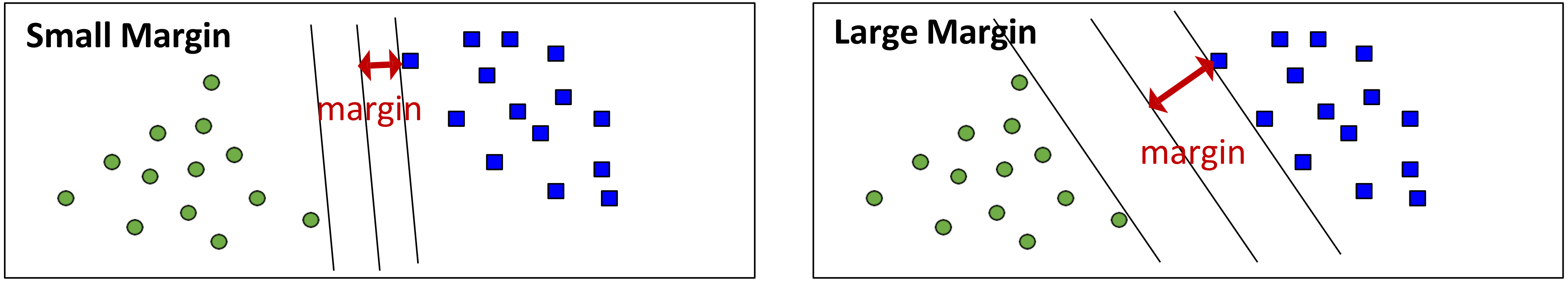




- Goal: The hard-margin SVM algorithm aims to find a linear classifier that **maximizes** (γ) the margin on S.

Why Margin Maximization?

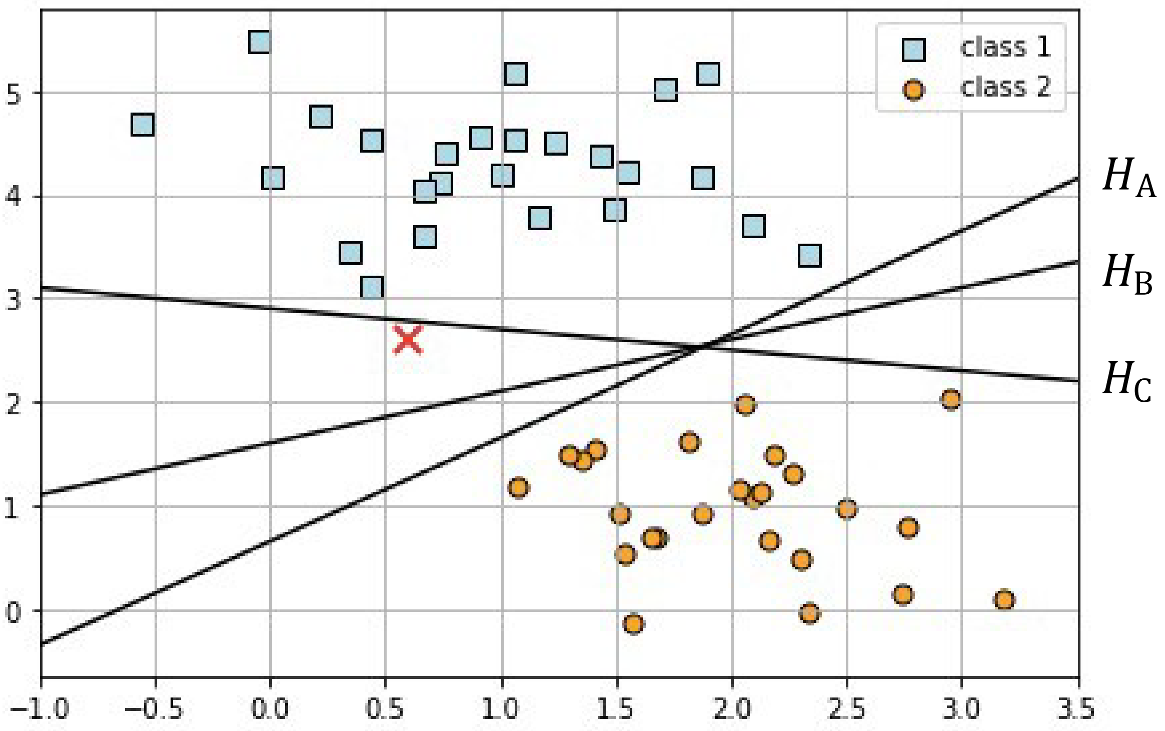
- Any linear classifier that separates S correctly will have margin $\gamma > 0$
- We want to find the best one (the one that minimizes classification error on unseen data)
- Assumption: the hyperplane with the largest margin will generalise best on unseen data
- SVM searches for the hyperplane with the largest margin, i.e., Maximum Marginal Hyperplane (MMH)



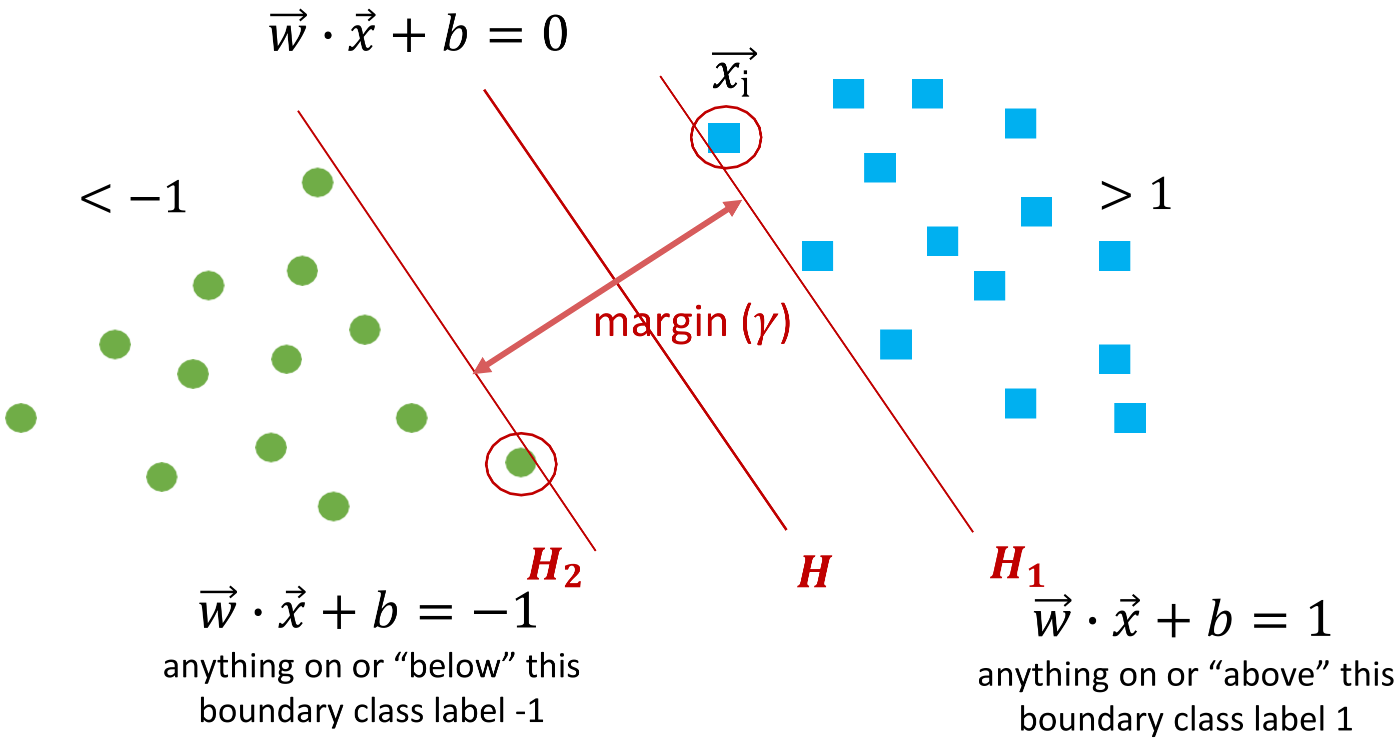
Example: Generalization Performance

We want a classifier which:

- Works well on training data
- Works well on the unseen Data



Margin Maximization Hyperplane (MMH)



$\vec{w}\cdot\vec{x}+b=0$

$\vec{x_{\mathrm{i}}}$

<-1

>1

margin (γ)

$\boldsymbol{H_2}$

$\boldsymbol{H}$

$\boldsymbol{H_1}$

$\vec{w}\cdot\vec{x}+b=-1$

anything on or "below" this boundary class label -1

$\vec{w}\cdot\vec{x}+b=1$

anything on or "above" this boundary class label 1

Decision rule: $f(\vec{x})=\begin{cases}+1, & \vec{w}\cdot\vec{x}+b\geq+1\\ -1, & \vec{w}\cdot\vec{x}+b\leq-1\end{cases}$

- Distance of closet data $\vec{x_{\mathrm{i}}}$ from the hyperplane H $\dfrac{|\vec{w}\cdot\vec{x_i}+b|}{\|\vec{w}\|}$
- $\dfrac{|\vec{w}\cdot\vec{x_i}+b|}{\|\vec{w}\|}=\dfrac{y_i(\vec{w}\cdot\vec{x_i}+b)}{\|\vec{w}\|}=\dfrac{1}{\|\vec{w}\|}$
- Margin: $\gamma=\dfrac{2}{\|\vec{w}\|}$
- Maximize γ is equivalent to minimize $\|\vec{w}\|$

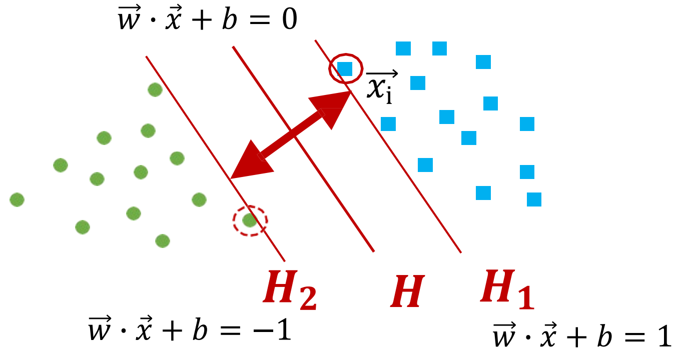
$$\min_{w,b}\frac{\|\vec{w}\|}{2}$$

s.t. $y_{\mathrm{i}}(\vec{w}\cdot\vec{x_i}+b)\geq 1$

(discrimination boundary is respected)

Margin Maximization Hyperplane (MMH)

- A separating hyperplane (H) can be formally defined as $\vec{w} \cdot \vec{x} + b = 0$
  - $\vec{w} = \{w_1, w_2, \dots, w_{\mathrm{n}}\}$ is a weight vector and $\boldsymbol{b}$ a scalar (bias)

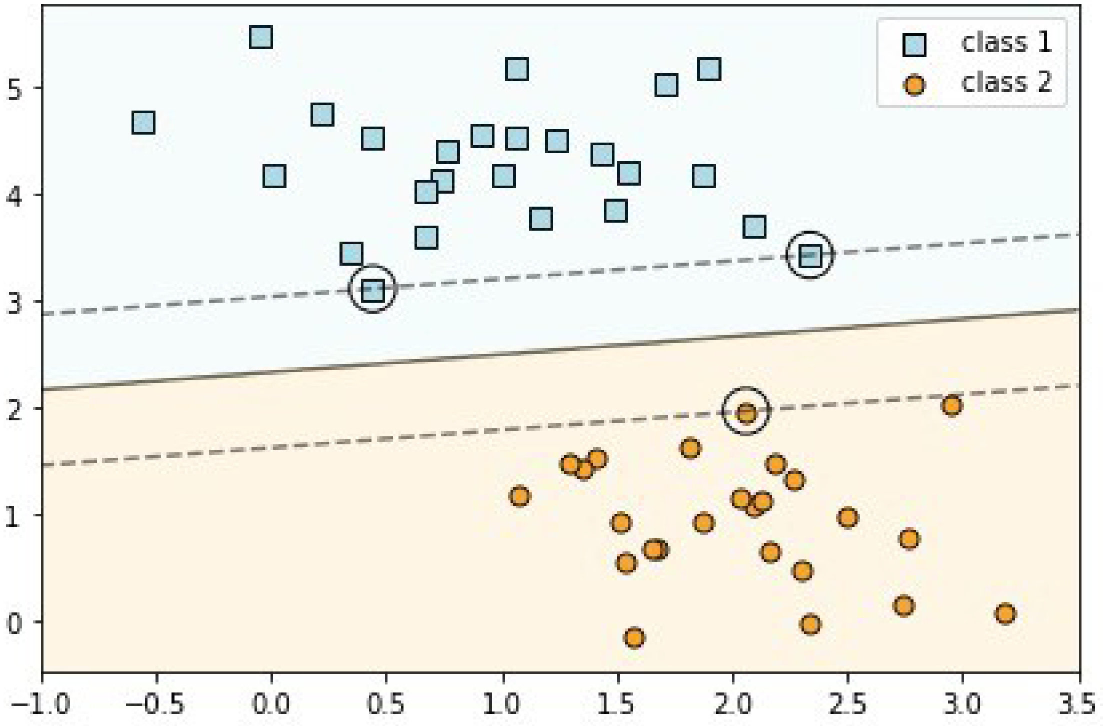

$\vec{w} \cdot \vec{x} + b = 0$
$\vec{x_\mathrm{i}}$
H_2
H
H_1
$\vec{w} \cdot \vec{x} + b = -1$
$\vec{w} \cdot \vec{x} + b = 1$

- For 2-D it can be written as: $w_1 \cdot x_{i,1} + w_2 \cdot x_{i,2} + b = 0$
- The hyperplanes defining the sides of the margin:
  - $\boldsymbol{H_1}: w_1 \cdot x_{i,1} + w_2 \cdot x_{i,2} + b \geq 1$, for $\mathrm{y_i} = +1$, and
  - $\boldsymbol{H}_2: w_1 \cdot x_{i,1} + w_2 \cdot x_{i,2} + b \leq 1$, for $y_i = -1$
- Any training tuples that fall on margins $\boldsymbol{H_1}$ or $\boldsymbol{H_2}$ (i.e., the hyperplanes defining the margin) are support vectors

Example: Support Vectors

Three Support Vectors:

1. [0.44359863 3.11530945]
2. [2.33812285 3.43116792]
3. [2.06156753 1.96918596]



Margin Maximization Hyperplane (MMH)

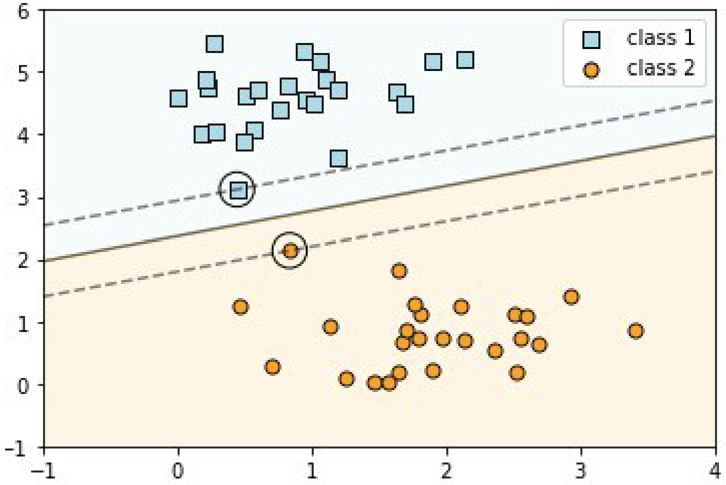
Linear model: $f(\vec{x}) = \begin{cases} +1, & \vec{w} \cdot \vec{x} + b \geq +1 \\ -1, & \vec{w} \cdot \vec{x} + b \leq -1 \end{cases}$

1. **Training Stage**: Learning the model is equivalent to determining the values of $\vec{w}$ and b
   - How to find $\vec{w}$ and b from training data S?
2. **Testing Stage**: Once $\vec{w}$ and b are found, given a test data $(\vec{x})$, use $f(\cdot)$ to determine the class label

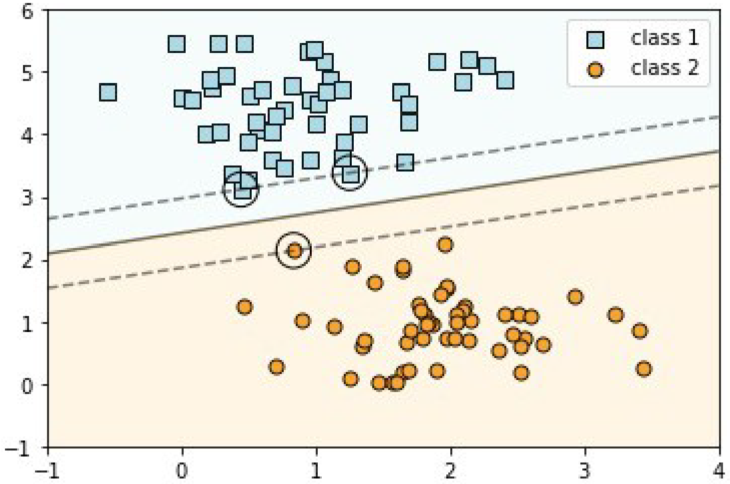
- Decision boundary depends only on support vectors
  - If we have data set with same support vectors, decision boundary will not change

Example: Support Vector Matters

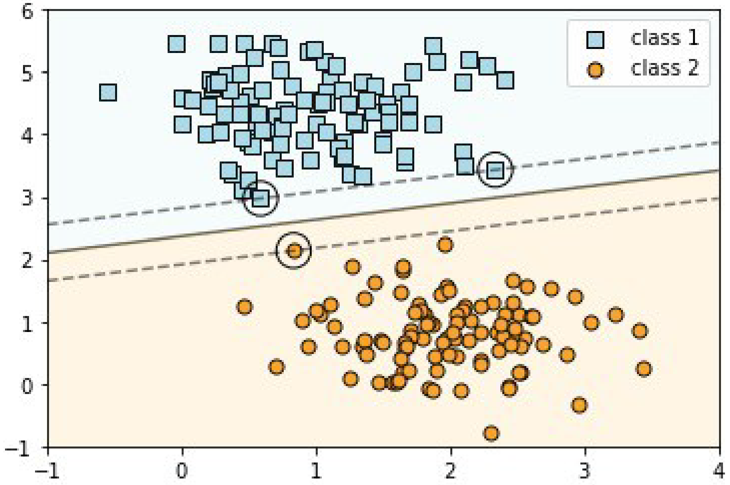
- Only the positions of the support vectors matter to decision boundary
- Other points further from the margin which are on the correct side do not modify the decision boundaries



N=50



N=100



N=200

Training Stage

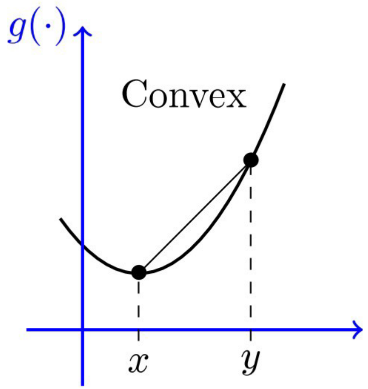
- Objective is to maximize: $\gamma = \frac{2}{\|\vec{w}\|}$
    - Equivalently, the objective is to minimize : $\min_{w,b} \frac{\|\vec{w}\|}{2}$
    - Subject to the following constraints:

$$y_{\mathrm{i}} = \begin{cases} +1, & \vec{w} \cdot \vec{x_i} + b \geq +1 \\ -1, & \vec{w} \cdot \vec{x_i} + b \leq -1 \end{cases}$$

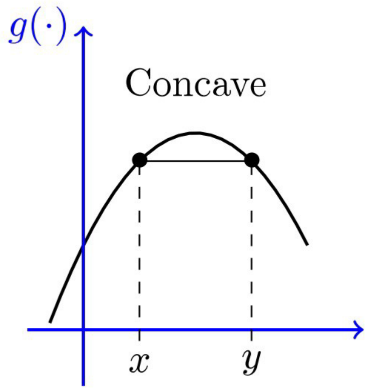
- Or $y_{\mathrm{i}}(\vec{w} \cdot \vec{x_i} + b) \geq 1, \qquad i = 1, 2, \ldots, n$

→ m inequality contrains

minimization



maximization



This becomes a **constrained (convex) quadratic optimization** problem:

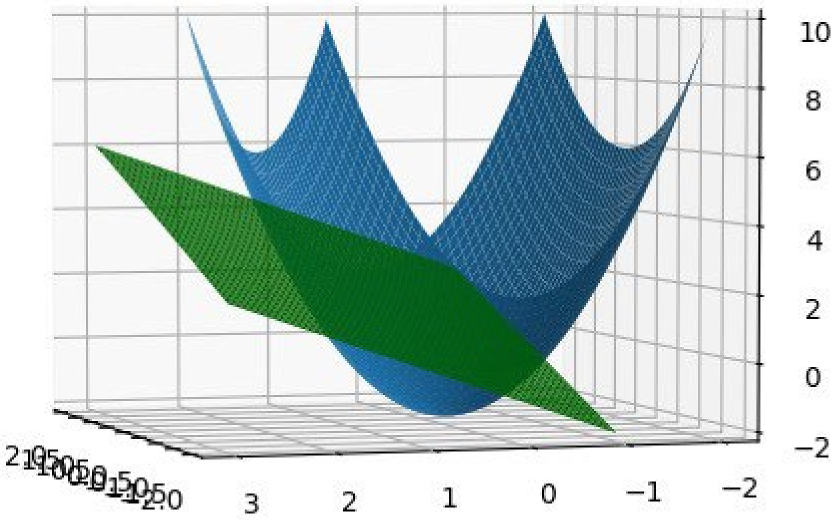
- Quadratic objective function with linear constraints → Quadratic Programming (QP)

Quadratic Programming (QP)

- QP is a well-studied solution algorithm

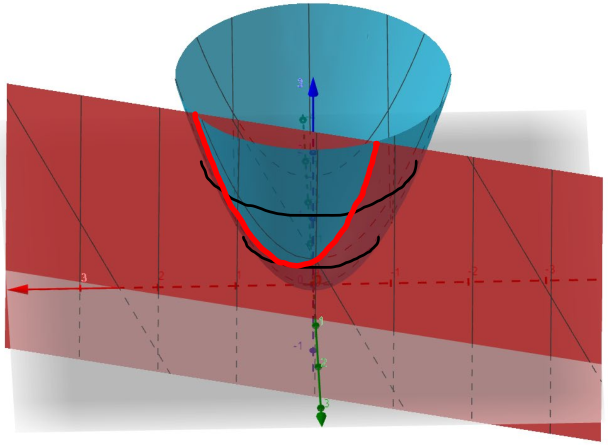
- Lagrange Multipliers and Constrained Optimization
  - Finding the **local minima** and **maxima** of a differentiable function subject to equality or inequality constraints

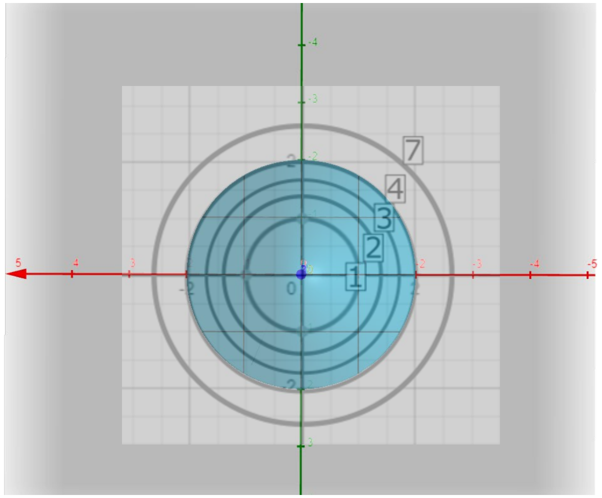
  - The point at which the function and constraint touch each other is the solution to the optimization problem

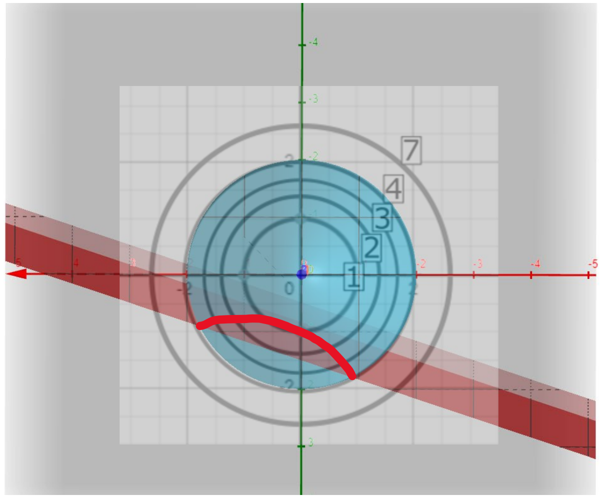


Lagrange Multipliers and Constrained Optimization

- Optimization function: $f = {x_1}^2 + {x_2}^2$
- Subject to the constraint: $g = 2\,x + 6\,y = c$, with $c = 5$





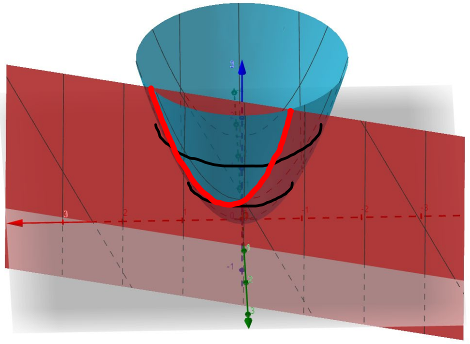


- At the minimum of f s.t. the constraint: $\nabla f = \alpha \nabla g$
- Finding the minimum is then equivalent to solving: $\nabla f - \alpha \nabla g = 0$

Lagrange Multipliers and Constrained Optimization

- Optimization function: $f = {x_1}^2 + y^2$
- Subject to the constraint: $g = 2\,x + 6\,y\ - 5$
- Lagrange function (Lagrangian multiplier α):
  - $\mathcal{L}(x,y,\lambda) = f(x,y) - \alpha g(x,y)$
    $= {x_1}^2 + y^2 - \alpha(2\,x + 6\,y\ - 5)$
  - Solution for the constrained problem is obtained by solving for the points where the partial derivatives of $\mathcal{L}$ are zero:

    - $\frac{d\mathcal{L}}{dx} = 2x - 2\alpha = 0$
    - $\frac{d\mathcal{L}}{dy} = 2y - 6\alpha = 0$
    - $\frac{d\mathcal{L}}{d\alpha} = -2x - 2y + 5 = 0$



Training Stage: Duality

- SVM primal problem form:

$$\min_{w,b} \frac{||\vec{w}||}{2} \qquad \text{s.t. } y_{\mathrm{i}}(\vec{w} \cdot \vec{x_i} + b) \geq 1, \quad i = 1, 2, \ldots, n$$

- We can convert it to the dual problem of SVM by introducing Lagrange multipliers (α_{i})

$$\mathcal{L}(\vec{w}, b, \alpha) = \frac{||\vec{w}||}{2} - \sum_{i=1}^{n} \alpha_i [y_{\mathrm{i}}(\vec{w} \cdot \vec{x_i} + b) - 1], \quad \alpha_i > 0$$

  - Solving the primal problem is equivalent to solving the dual problem:

$$\min_{w,b} \frac{||\vec{w}||}{2} \equiv \max_{\alpha} \min_{w,b} \mathcal{L}(\vec{w}, b, \alpha)$$

Training Stage: Duality

- SVM primal problem form:

$$\min_{w,b} \frac{\|\vec{w}\|}{2} \qquad \text{s.t. } y_{\mathrm{i}}(\vec{w} \cdot \vec{x_i} + b) \geq 1, \quad i = 1, 2, \ldots, n$$

- We can convert it to the dual problem of SVM by introducing Lagrange multipliers (α_{i})
    - Set the derivatives of SVM Lagrangian function w.r.t. $\vec{w}$ and b to be zero:

$$\mathcal{L}(\vec{w}, b, \alpha) = \frac{\|\vec{w}\|}{2} - \sum_{i=1}^{n} \alpha_i [y_{\mathrm{i}}(\vec{w} \cdot \vec{x_i} + b) - 1], \quad \alpha_i > 0$$

- $\frac{d\mathcal{L}}{d\vec{w}} = 0 \Rightarrow \vec{w} = \sum_{i=1}^{n} \alpha_i y_{\mathrm{i}} \vec{x_i}$
- $\frac{d\mathcal{L}}{db} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_{\mathrm{i}} = 0$

    - Substituting them in the Lagrangian function $\mathcal{L}$, we obtain the final dual optimization function:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_{\mathrm{i}} y_j \vec{x_i} \vec{x_j}$$

Training Stage: Solving the Dual Problem

- Dual Problem Optimization

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_{\mathrm{i}} y_j \vec{x_i}\,\vec{x_j}$$

$$\text{s.t. } \alpha_i \geq 0 \text{ and } \textstyle\sum_{i=1}^{n} \alpha_i y_{\mathrm{i}} = 0, \qquad i = 1, 2, \dots, n$$

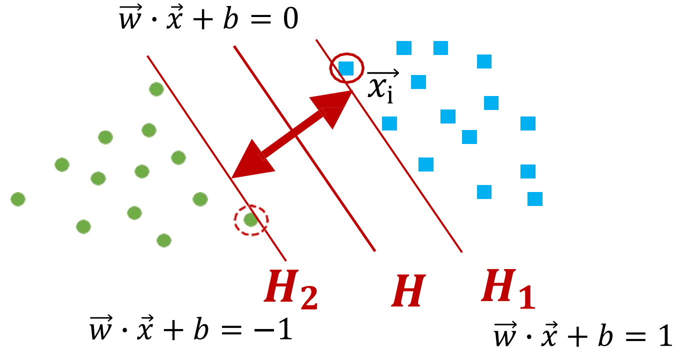
- This can be solved efficiently using numerical optimization
  - $\alpha_{\mathrm{i}} > 0$ for support vectors $\vec{x_{\mathrm{i}}}$ that lie on the margin $H_1$and H_2
  - $\alpha_{\mathrm{i}} = 0$ for other training points
- Thus, the solution for $\vec{w}$ corresponding to the maximal margin classifier can be written as a linear combination of just the support vectors: $\vec{w} = \sum_{x_i \in SV} \alpha_i y_{\mathrm{i}} \vec{x_i}$

Training Stage: Solving the Dual Problem

- For support vectors, we have $y_i - (\vec{w} \cdot \vec{x} + b) = 0$
  - Blue support vector ($y_{\mathrm{i}} = +1$)
  - Green support vector ($y_{\mathrm{i}} = -1$)

- Thus, the solution for b from any of the support vectors

$$b = \frac{1}{|SV|} \sum_{x_i \in SV} y_i - (\vec{w} \cdot \vec{x_i})$$



$\vec{w} \cdot \vec{x} + b = 0$
$\vec{x_{\mathrm{i}}}$
H_2
H
H_1
$\vec{w} \cdot \vec{x} + b = -1$
$\vec{w} \cdot \vec{x} + b = 1$

Testing Stage

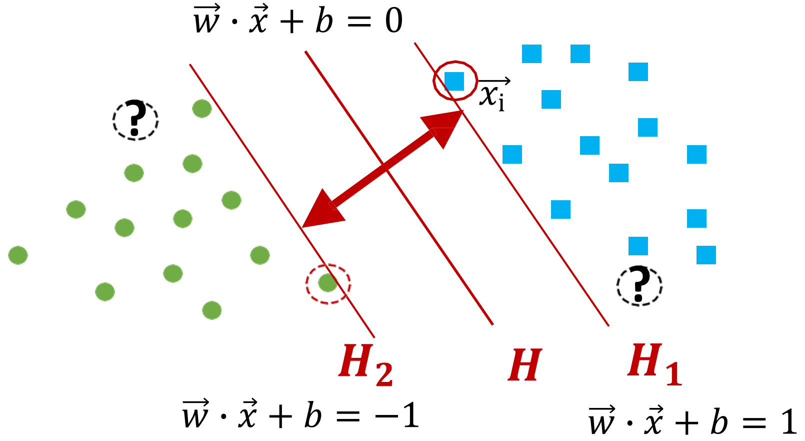
- Given a new data point $\vec{x}$, we use the learned SVM classifier ($\vec{w}$ and $\vec{b}$) to derive the class label as follow:

$$f(\vec{x}) = \begin{cases} +1, & \vec{w}\cdot\vec{x}+b \geq 0 \\ -1, & \vec{w}\cdot\vec{x}+b \leq 0 \end{cases}$$

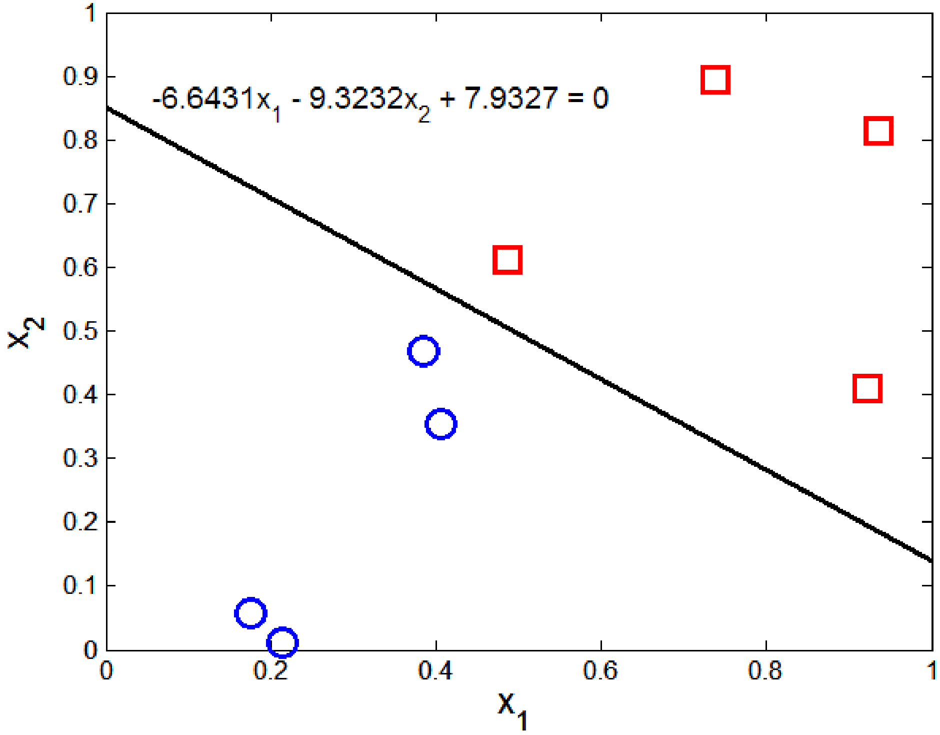
(Primal Form)

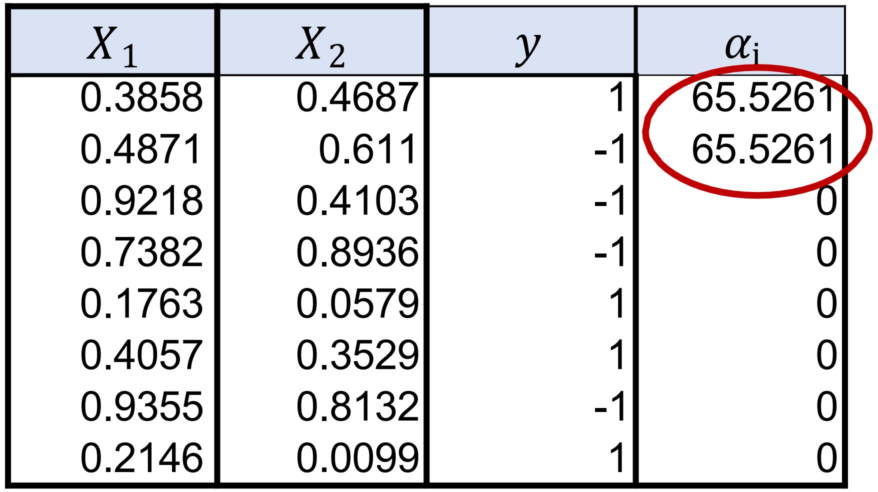
$$= \begin{cases} +1, & \sum_{x_i \in SV} \alpha_i y_\mathrm{i}(\vec{x_i}\cdot\vec{x})+b > 0 \\ -1, & \sum_{x_i \in SV} \alpha_i y_\mathrm{i}(\vec{x_i}\cdot\vec{x})+b < 0 \end{cases}$$

(Dual Form)


$\vec{w}\cdot\vec{x}+b=0$
$\vec{x_\mathrm{i}}$
?
?
H_2
H
H_1
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$

Example: Training Stage





X_1	X_2	y	α_i
0.3858	0.4687	1	65.5261
0.4871	0.611	-1	65.5261
0.9218	0.4103	-1	0
0.7382	0.8936	-1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	-1	0
0.2146	0.0099	1	0

Support Vectors

Source: https://www users.cs.umn.edu/~kumar001/dmbook/index.php

Example: Testing Stage

- Given a new data point $\vec{x} = [0.5,0.9]$, what is the class label of $\vec{x}$ using the trained SVM?

Support Vectors

$X_{"}$	$X_{\#}$	y	$\alpha_{!}$
0.3858	0.4687	1	65.5261
0.4871	0.611	-1	65.5261
0.9218	0.4103	-1	0
0.7382	0.8936	-1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	-1	0
0.2146	0.0099	1	0

$$f(\vec{x}) = \begin{cases} +1, & \sum_{x_i \in SV} \alpha_i y_\mathrm{i}(\vec{x_i} \cdot \vec{x}) + b > 0 \\ -1, & \sum_{x_i \in SV} \alpha_i y_\mathrm{i}(\vec{x_i} \cdot \vec{x}) + b < 0 \end{cases}$$

$$sign\left(\sum_{x_i \in SV} \alpha_i y_\mathrm{i}(\vec{x_i} \cdot \vec{x}) + b\right)$$

$$= sign\left(65.5261 \cdot 1 \cdot \begin{bmatrix}0.3858\\0.4687\end{bmatrix}^\mathrm{T} [0.5,0.9] + 65.5261 \cdot (-1) \cdot \begin{bmatrix}0.4871\\0.611\end{bmatrix}^\mathrm{T} [0.5,0.9]\right)$$

Quiz: Linear SVM

- Question: Suppose we want to build a hard-margin SVM classifier for two-class classification in one dimension space ($d = 1, x_i \in \mathbb{R}$) contains three sample points:
  - point x_1 = 3 with label y_1 = 1
  - point x_2 = 1 with label y_2 = 1
  - point x_3 = -1 with label y_3 = -1

  What are the values of $\vec{w}$ and $\vec{b}$ given by our hard-margin SVM?

- Solve the optimization problem for w and b with the following constraints

$$\min_{w,b} \frac{w^2}{2} \quad \text{s.t.} \begin{cases} w * x_1 + b \geq 1 \\ w * x_2 + b \geq 1 \\ w * x_3 + b \leq -1 \end{cases}$$

A: w = 1, b = 1

B: w = 1, b = 0

C: w = 0, b = 1

D: w = ∞, b = 0

Advantages v.s. Disadvantages

Advantages

- SVMs depends on relatively few support vectors
  - SVMs are very compact models, and take up very little memory
  - SVMs work well with high-dimensional data, even with more dimensions than samples ($d > |S|$)
- Once the model is trained, the prediction phase is very fast

Disadvantages

- For large numbers of training samples, the computational cost can be prohibitive
- The results do not have a direct probabilistic interpretation

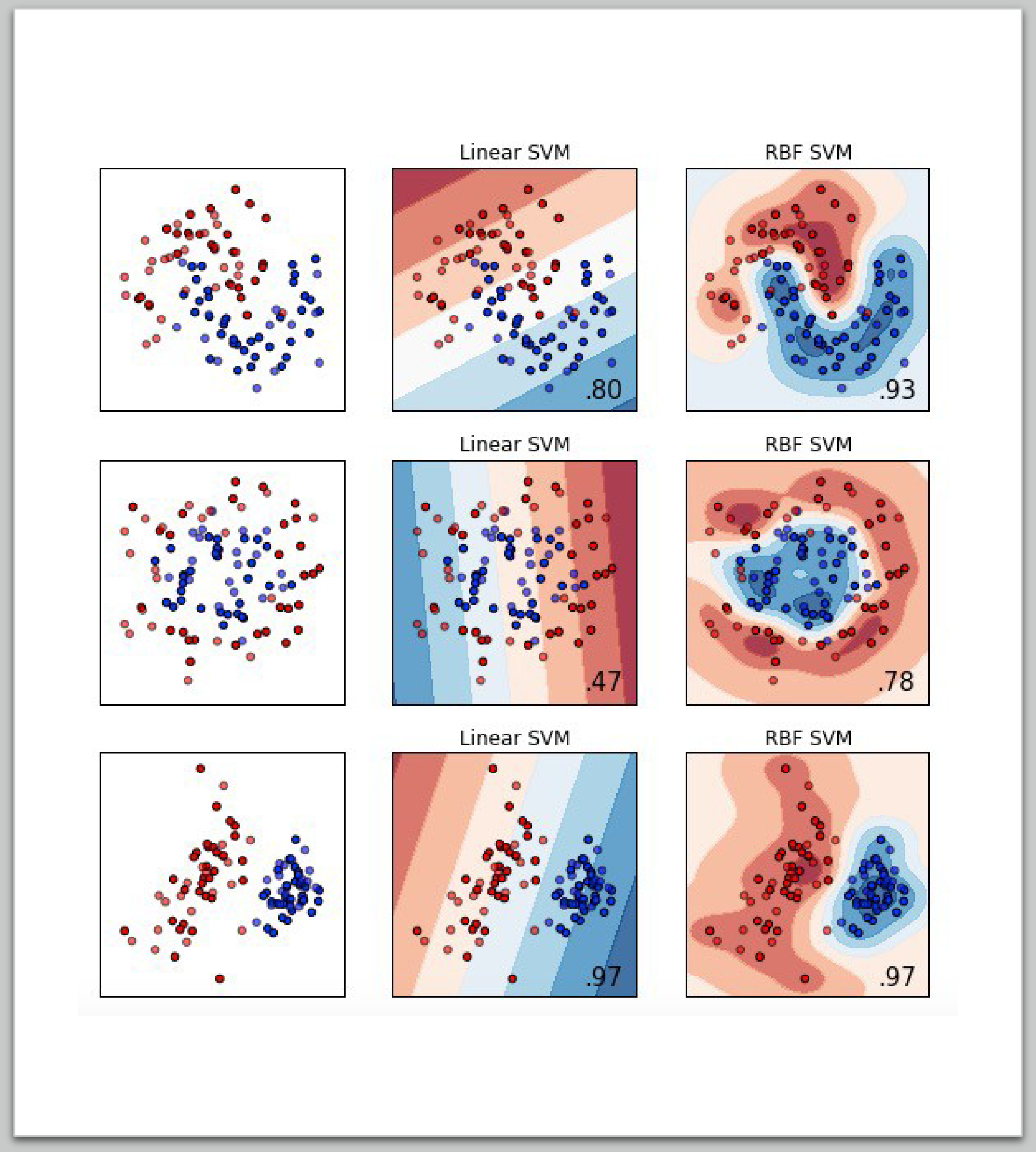
Jupyter Notebook

Hard-margin SVM Coding Example

SUMMARY

- Data Characteristics
  - Linearly Separable Data
  - Non-Linearly separable Data

- Linearly Separable Data: Hard-margin SVMs
  - Margin Maximization Hyperplane (MMH)
  - Primal Form Optimization
  - Duality Form Optimization
  - Training Phase
  - Testing Phase
  - Advantages v.s. Disadvantages



Resources

- SVM Website: http://www.kernel-machines.org/

- Representative Implementation
  - **LIBSVM**: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - **SVM-light**: simpler but performance is not better than LIBSVM, support only binary classification and only in C
  - **SVM-torch**: another recent implementation also written in C
  - **Scikit-Learn**: a set of supervised learning methods used for classification, regression and outliers detection. [link]

Resources (Contd.)

- Book Chapters: Christopher Bishop, “Pattern Recognition and Machine Learning” (PDF)
  - Sec 7.1.1-7.1.2
  - Sec 4.1.1
  - Sec 6.1, 6.2
  - Appendix E

- Literatures
  - C.J.C. Burges, Chris J.C. Burges "A Tutorial on Support Vector Machines for Pattern Recognition.” Data Mining and Knowledge Discovery, 1998