CS 461: Machine Learning Principles

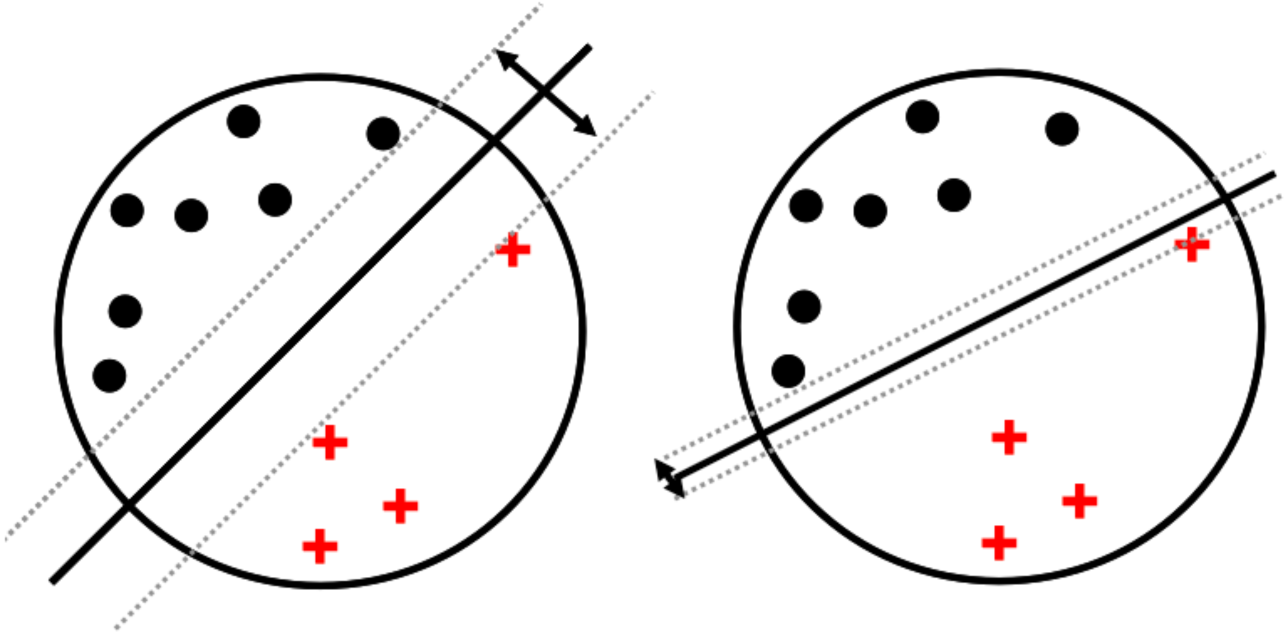
Class 13: Oct. 17

SVM (SMO Algorithm)

& Decision Tree

Instructor: Diana Kim

Computing a Large Margin Classifier



There are many hyperplanes to sperate the training data points.
But a maximum margin classifier on training set is desirable.

- high confident separation
- robust to perturbation of data (generalization)

SVM Problem [Hard Margin SVM]

$$w*, b* = \arg\max_{w,b} \frac{\Delta}{||w||}$$

$$\text{subject to} \quad t_n(w^t x_n + b) \geq \Delta \quad \forall n$$

$$w*, b* = \arg\min_{w,b} \frac{1}{2}||w||^2$$

$$\text{subject to} \quad t_n(w^t x_n + b) \geq 1 \quad \forall n$$

All data points are separated by the minimum distance of $\Delta/||W||$ hyperplane , we want to maximize the minimum distance.

SVM Primal and Dual Problem

- Primal

$$w*, b* = \arg\min_{w,b} \frac{1}{2}||w||^2$$

$$\text{subject to} \quad t_n(w^t x_n + b) \geq 1 \quad \forall n$$

- Dual

$$\lambda*_{n=1}^{N} = \arg\max_{\lambda*} \sum_{n=1}^{N} \lambda *_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n^* \cdot t_n \lambda_m^* \cdot t_m \cdot \kappa(x_n, x_m)$$

$$\text{subject to} \quad \lambda_n^* \geq 0 \quad \forall n$$

$$\sum_{n=1}^{N} \lambda_n^* \cdot t_n = 0$$

Advantage of Dual Formulation

- Dual

$$\lambda*_{n=1}^{N} = \arg\max_{\lambda*} \sum_{n=1}^{N} \lambda *_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n^* \cdot t_n \lambda_m^* \cdot t_m \cdot \kappa(x_n, x_m)$$

$$\text{subject to} \quad \lambda_n^* \geq 0 \quad \forall n$$

$$\sum_{n=1}^{N} \lambda_n^* \cdot t_n = 0$$

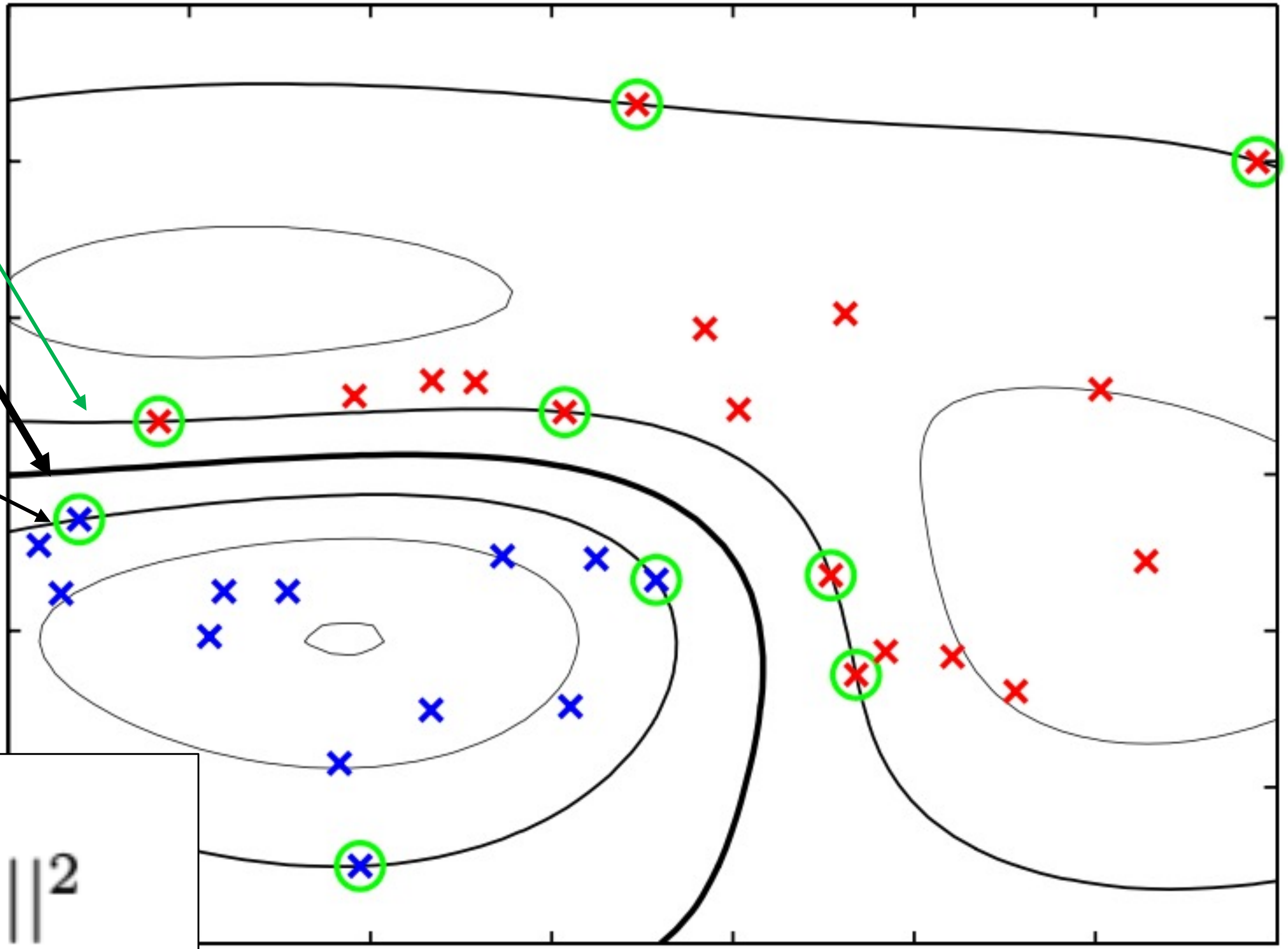
- By using kernel trick, maximum margin classifier can be applied efficiently to feature spaces whose dimensionality exceeds the number of data points. (M>>N)
- By using kernel trick, we can make training data linearly separable.

The Outcomes of Gaussian Kernel SVM

(1) **Decision hyperplane**

(2) Support Vectors (Samples)

(3) Margins



Gaussian Kernel

$$\kappa(x, x') = \exp \frac{-||x - x'||^2}{2\sigma^2}$$

From the dual function, we compute $\lambda*$ (n = 1,.. N).
Then we can build a maximum margin classifier.
A subset of the training data points are used to build the classifier.

$$y(x) = \sum_{n=1}^{N} \lambda *_n t_n \kappa(x_n, x) + b$$

$$\text{The second and third KKT} = \begin{cases} \lambda*_n = 0 & \text{if} \quad w*^t x_n - 1 > 0 \\ \boxed{\lambda*_n > 0} & \text{if} \quad \boxed{w*^t x_n - 1 = 0 \quad \text{Support Vectors } x_n!} \end{cases}$$

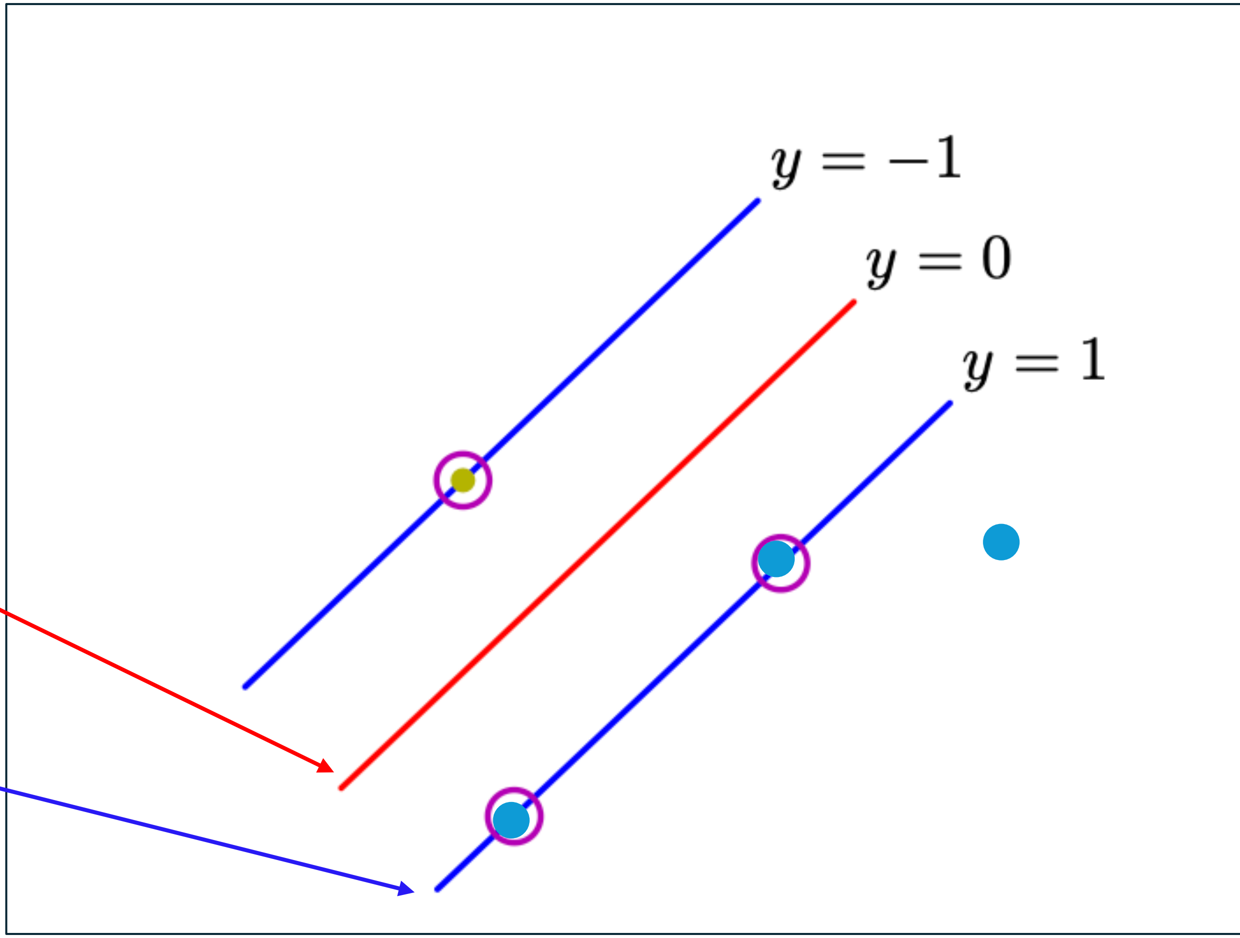
The Outcomes of Linear Kernel SVM

(1) Decision hyperplane

(2) Support Vectors (Samples)

(3) Margins

From Bishop Figure 7.1

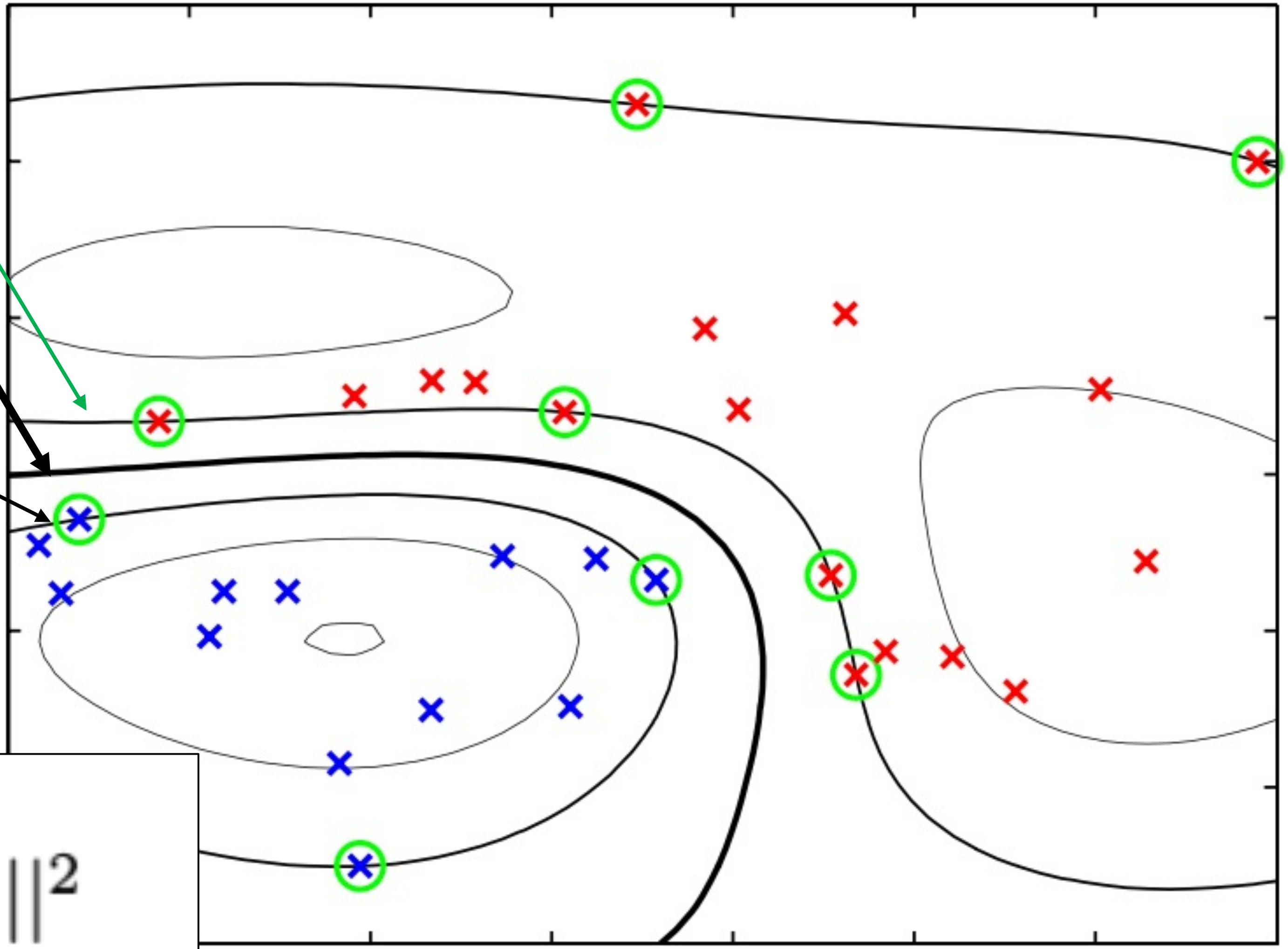


The Outcomes of Gaussian Kernel SVM

(1) **Decision hyperplane**

(2) Support Vectors (Samples)

(3) Margins



Gaussian Kernel

$$\kappa(x, x') = \exp \frac{-||x - x'||^2}{2\sigma^2}$$

About Gaussian Kernel

Gaussian Kernel

$$\kappa(x, x') = \exp \frac{-||x - x'||^2}{2\sigma^2}$$

- Gaussian kernel embeds the infinite dimensional feature space. However, SVM with Gaussian is not sensitive to # data points because it the maximum margin boundary can be defined by two + / – data samples. (performance/complexity)

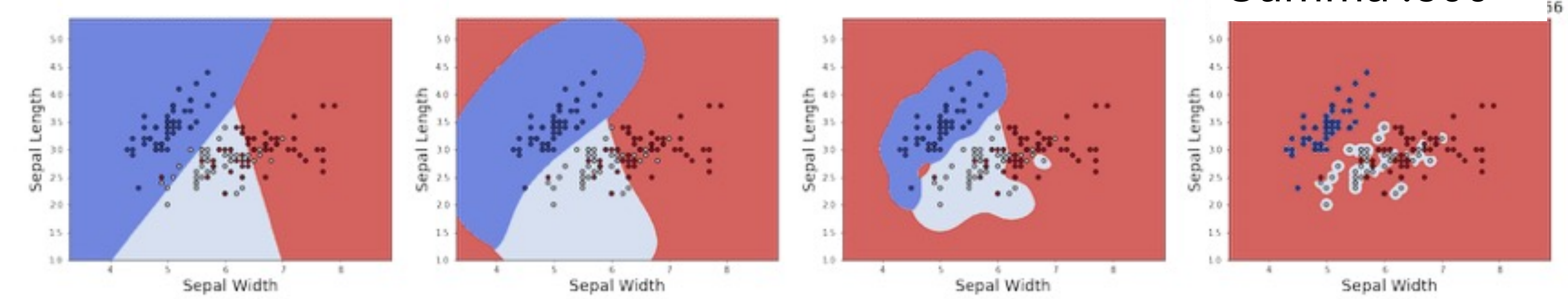
- The model complexity depends on σ.

The Effect of Gamma on the number of Support vectors & decision Boundary

$$\gamma = \frac{1}{\sigma^2}$$

From https://www.kaggle.com/code/gorkemgunay/understanding-parameters-of-svm

Gamma : 0.01 $\quad$ $Gamma$:1 $\quad$ Gamma :10 $\quad$ Gamma :500

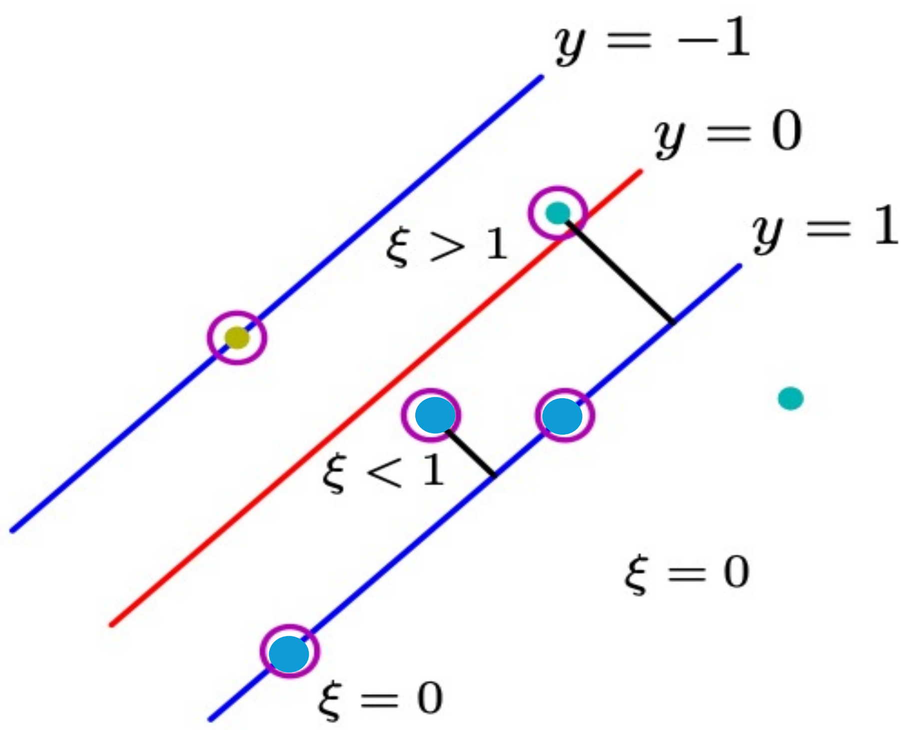


- Small gamma: some representative samples become support vectors.
- Large gamma: individual samples become support vectors

What happens for SVM if data samples are not linearly separable?

For each data points,
the constraint to define the margin is relaxed by the slack variable $\xi_n \geq 0$

From Bishop Figure 7.3



- hard margin

$$t_n(w^t x_n + b) \geq 1 \quad \forall n$$

- soft margin

$$t_n(w^t x_n + b) \geq \boxed{1 - \xi n} \quad \text{and} \quad \xi n \geq 0 \quad \forall n$$

+ margin can be reduced; even it can be a negative value.

- Hard Margin Case (Exact-Separable)

$$w*, b* = \arg\min_{w,b} \frac{1}{2}||w||^2$$

$$\text{subject to} \quad t_n(w^t x_n + b) \geq 1 \quad \forall n$$

- Soft Margin Case (Non-Separable)

$$w*, b* = \arg\min_{w,b} C \cdot \sum_{n=1}^{N} \xi_n + \frac{1}{2}||w||^2$$

$$\text{subject to} \quad t_n(w^t x_n + b) \geq 1 - \xi_n \quad \forall n$$

$$\text{subject to} \quad \xi_n \geq 0 \quad \forall n$$

Soft Margin SVM Primal and Dual Problem

- Primal

$$w*, b* = \arg\min_{w,b} C \cdot \sum_{n=1}^{N} \xi_n + \frac{1}{2}||w||^2$$

$$\text{subject to} \quad t_n(w^t x_n + b) \geq 1 - \xi_n \quad \forall n$$

$$\text{subject to} \quad \xi_n \geq 0 \quad \forall n$$

- Dual

$$\lambda*_{n=1}^{N} = \arg\max_{\lambda*} \sum_{n=1}^{N} \lambda *_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n^* \cdot t_n \lambda_m^* \cdot t_m \cdot \kappa(x_n, x_m)$$

$$\text{subject to} \quad 0 \leq \lambda_n^* \leq C$$

$$\sum_{n=1}^{N} \lambda_n^* \cdot t_n = 0$$

KKT conditions

$$\nabla_w L(w, b, \lambda_{n=1}^N, \xi_{n=1}^N, \mu_{n=1}^M) = \vec{w} - \sum_{n=1}^{N} (\lambda_n \cdot t_n \cdot \vec{x_n})$$

$$\nabla_b L(w, b, \lambda_{n=1}^N, \xi_{n=1}^N, \mu_{n=1}^M) = \sum_{n=1}^{N} \lambda_n \cdot t_n$$

$$\nabla_{\xi_n} L(w, b, \lambda_{n=1}^N, \xi_{n=1}^N, \mu_{n=1}^M) = C - \lambda_n - \mu_n$$

$$\lambda_n \cdot \{t_n(w^t x_n) + b - 1 + \xi_n\} = 0$$

$$\mu_n \xi n = 0$$

Soft Margin SVM Primal and Dual Problem

- Primal

$$w*, b* = \arg\min_{w,b} C \cdot \sum_{n=1}^{N} \xi_n + \frac{1}{2}||w||^2$$

$$\text{subject to} \quad t_n(w^t x_n + b) \geq 1 - \xi_n \quad \forall n$$

$$\text{subject to} \quad \xi_n \geq 0 \quad \forall n$$

- Dual

$$\lambda*_{n=1}^{N} = \arg\max_{\lambda*} \sum_{n=1}^{N} \lambda*_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \lambda_n^* \cdot t_n \lambda_m^* \cdot t_m \cdot \kappa(x_n, x_m)$$

$$\text{subject to} \quad 0 \leq \lambda_n^* \leq C$$

$$\sum_{n=1}^{N} \lambda_n^* \cdot t_n = 0$$

- $\lambda*_n = 0$
- Support vectors: $0 < \lambda*_n < C$
- Support vectors: $\lambda*_n = C$

All data samples must satisfy the KKT condition.

- $\lambda *_n = 0$
- Support vectors: $0 < \lambda *_n < C$
- Support vectors: $\lambda *_n = C$

"correctly labeled with a room to spare"

$$y_n(w^t x_n + b) \geq 0$$

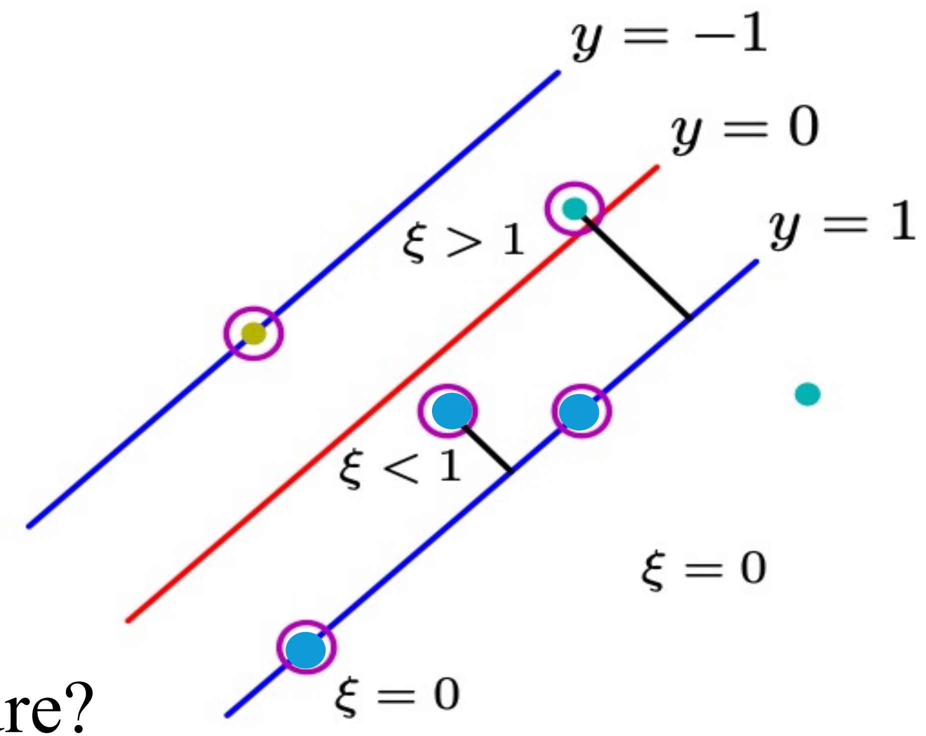
$$y_n(w^t x_n + b) = 1$$ "unbound"

$$y_n(w^t x_n + b) \leq 0$$

"incorrectly labeled or line within the margin"

From Bishop Figure 7.3

- $\lambda*_n = 0$
- Support vectors: $0 < \lambda*_n < C$
- Support vectors: $\lambda*_n = C$

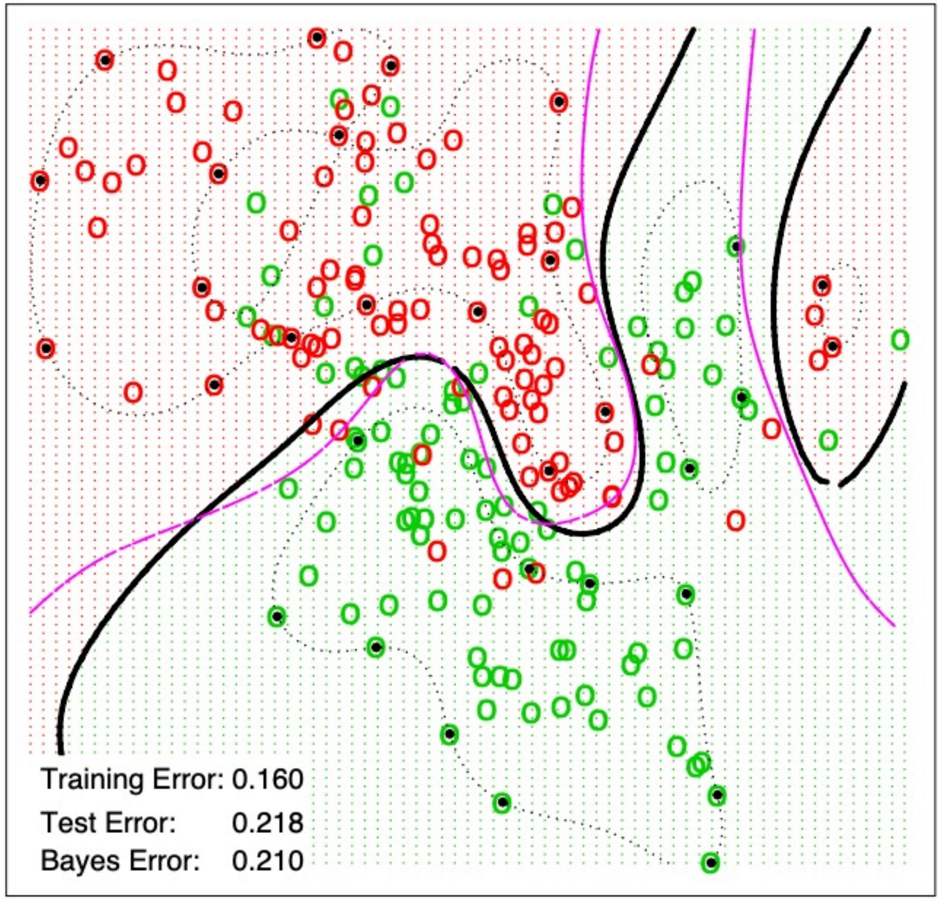


Q: How many support vectors are in this figure?

Q: Soft margin SVM can handle non-separable data.
Can soft margin algorithm function as regularization for SVM?

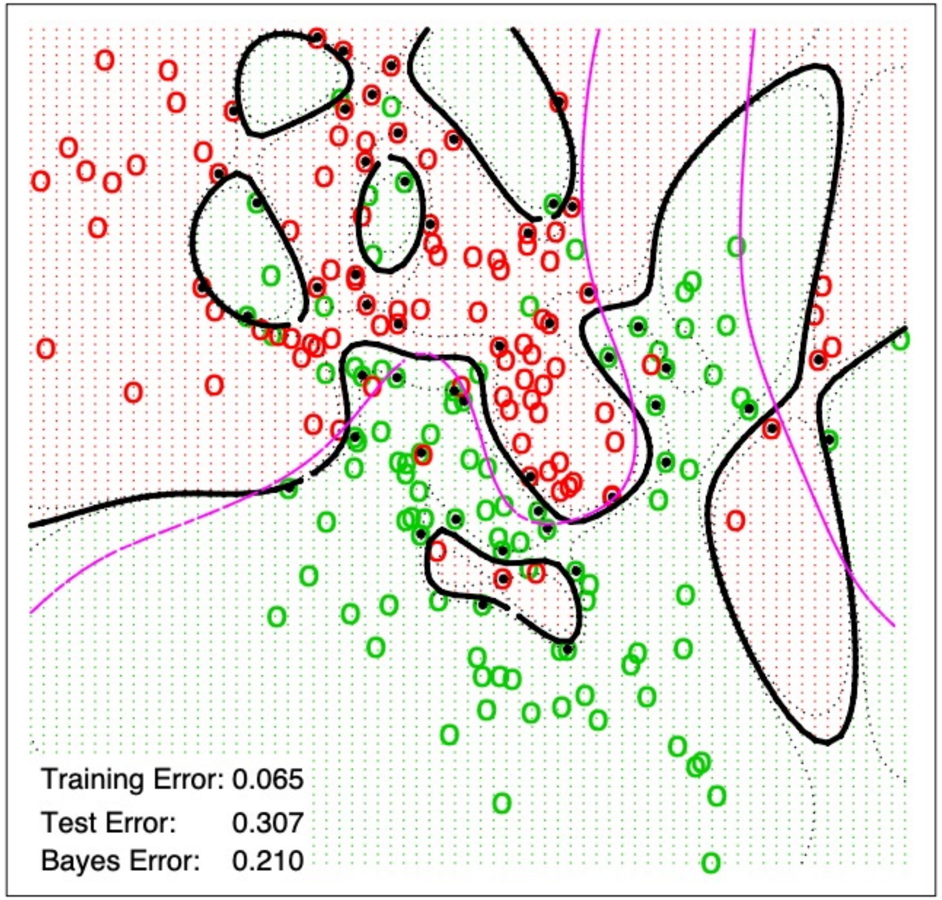
From the paper: https://jmlr.csail.mit.edu/papers/volume5/hastie04a/hastie04a.pdf

Radial Kernel: $C = 2, \gamma = 1$



For small C, Large Slack

Radial Kernel: $C = 10{,}000, \gamma = 1$



For large C, Small Slack

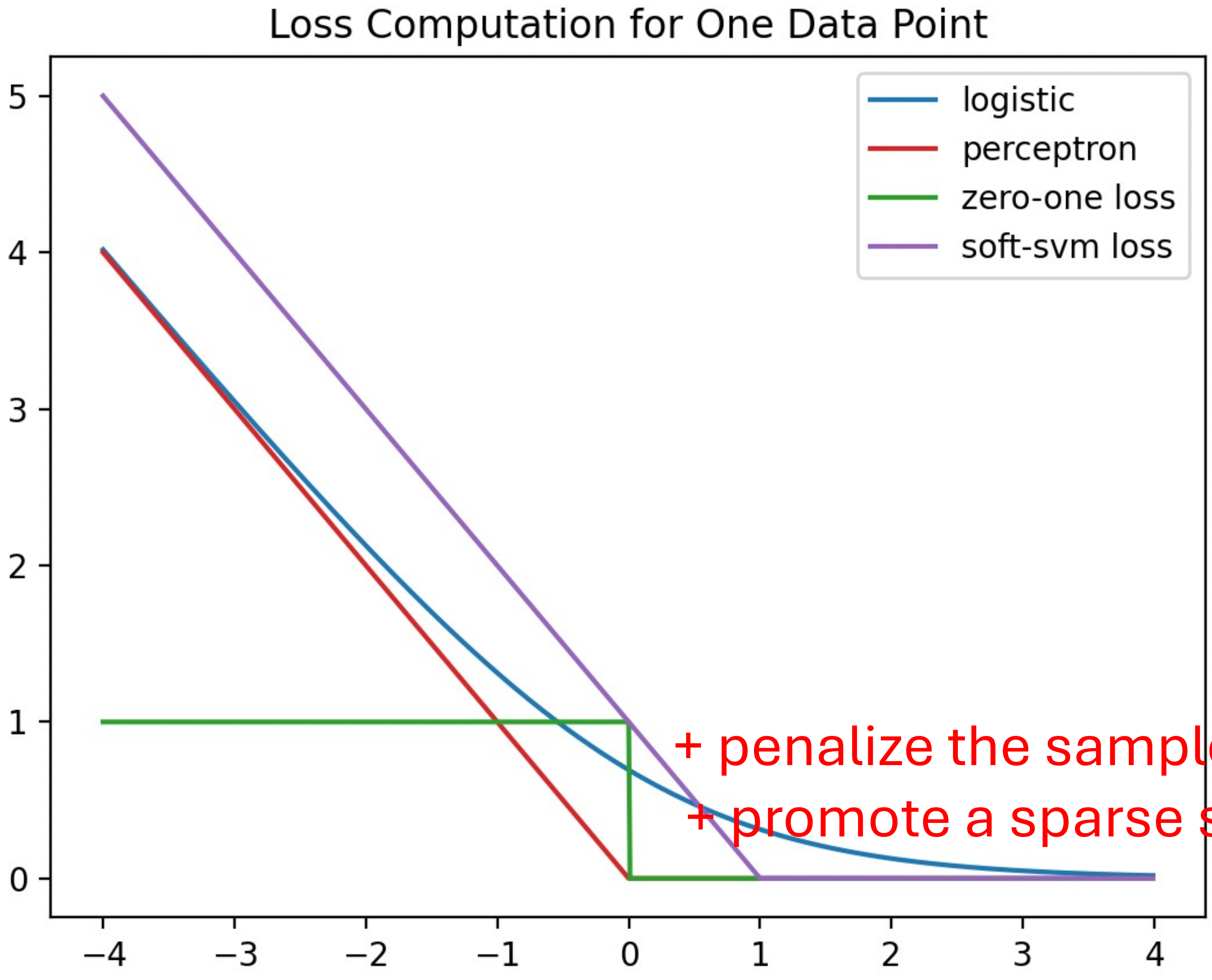
- Small C: Smooth boundaries for Soft-Margin so better generalization

We need to conduct cross validation to select proper σ^2 and C.

The Loss for One Data Point of soft Margin SVM

$$\text{Soft Margin SVM Loss }(x, y) = \begin{cases} \xi_n = 0 \quad \text{for} \quad yw^t x \geq 1 \\ \\ \xi_n = 1 - yw^t x \quad \text{for} \quad yw^t x < 1 \end{cases}$$

Loss Comparison for One Data Point (Objective Function)



+ penalize the samples within the margi

+ promote a sparse solution

Solving Minimal Optimization (SMO)

- Suppose given $\mathcal{D} : \{(x, y) : x \in \mathbb{R}^M \ and \ t \in \{-1, 1\}\}$ and $\kappa(x, x')$.
  A soft SVM dual function is defined below.
  How could we find the optimal $\lambda_1, \lambda_2 \ \ldots, \lambda_n$?

$$\lambda*_{n=1}^{N} = \arg\max_{\lambda*} \sum_{n=1}^{N} \lambda*_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n^* \cdot t_n \lambda_m^* \cdot t_m \cdot \kappa(x_n, x_m)$$

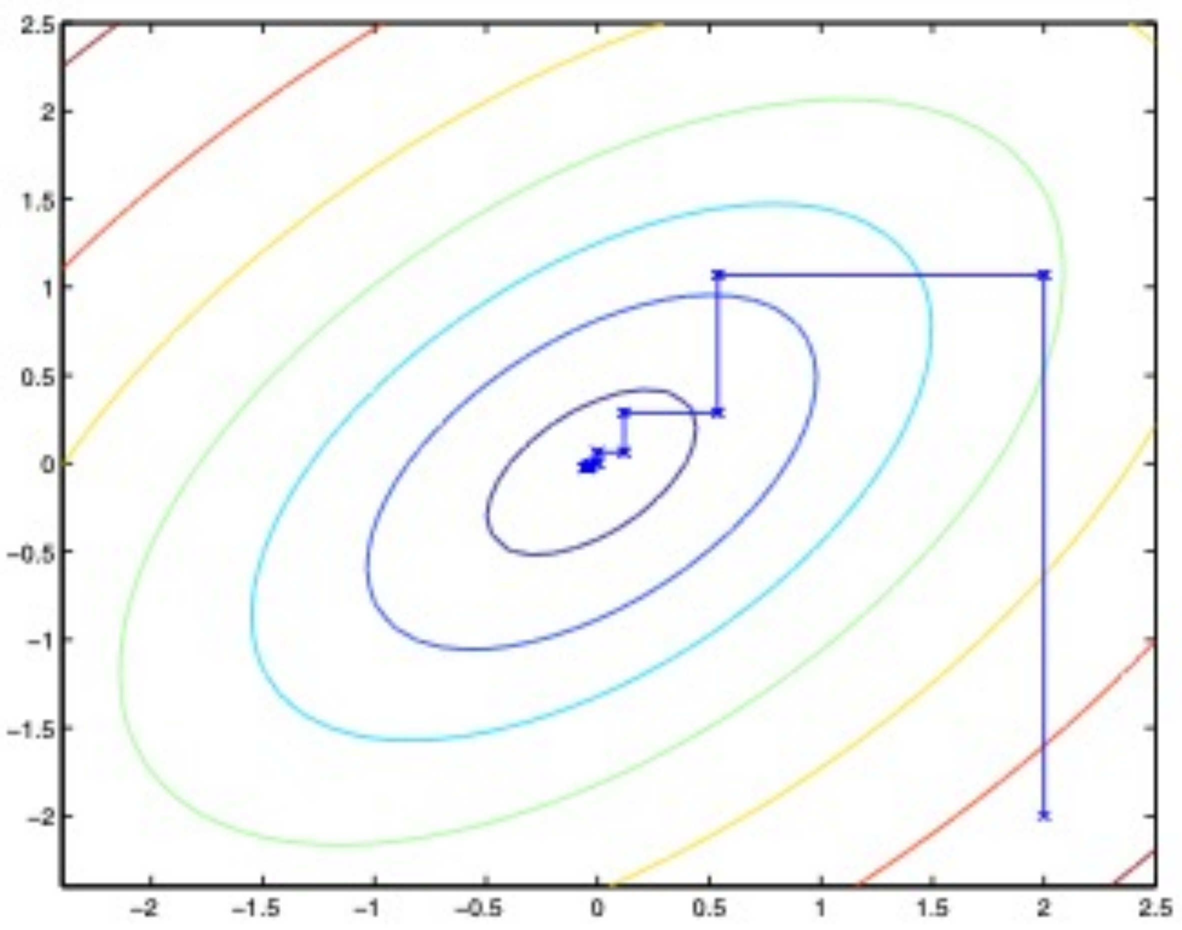
$$\text{subject to} \quad 0 \leq \lambda_n^* \leq C$$

$$\sum_{n=1}^{N} \lambda_n^* \cdot t_n = 0$$

Q: Can we solve this problem by using gradient descent?

Coordinate Ascent Algorithm: One Coordinate at One Time.

https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf



At each step, the algorithm finds a minimum along the axis, we selected.

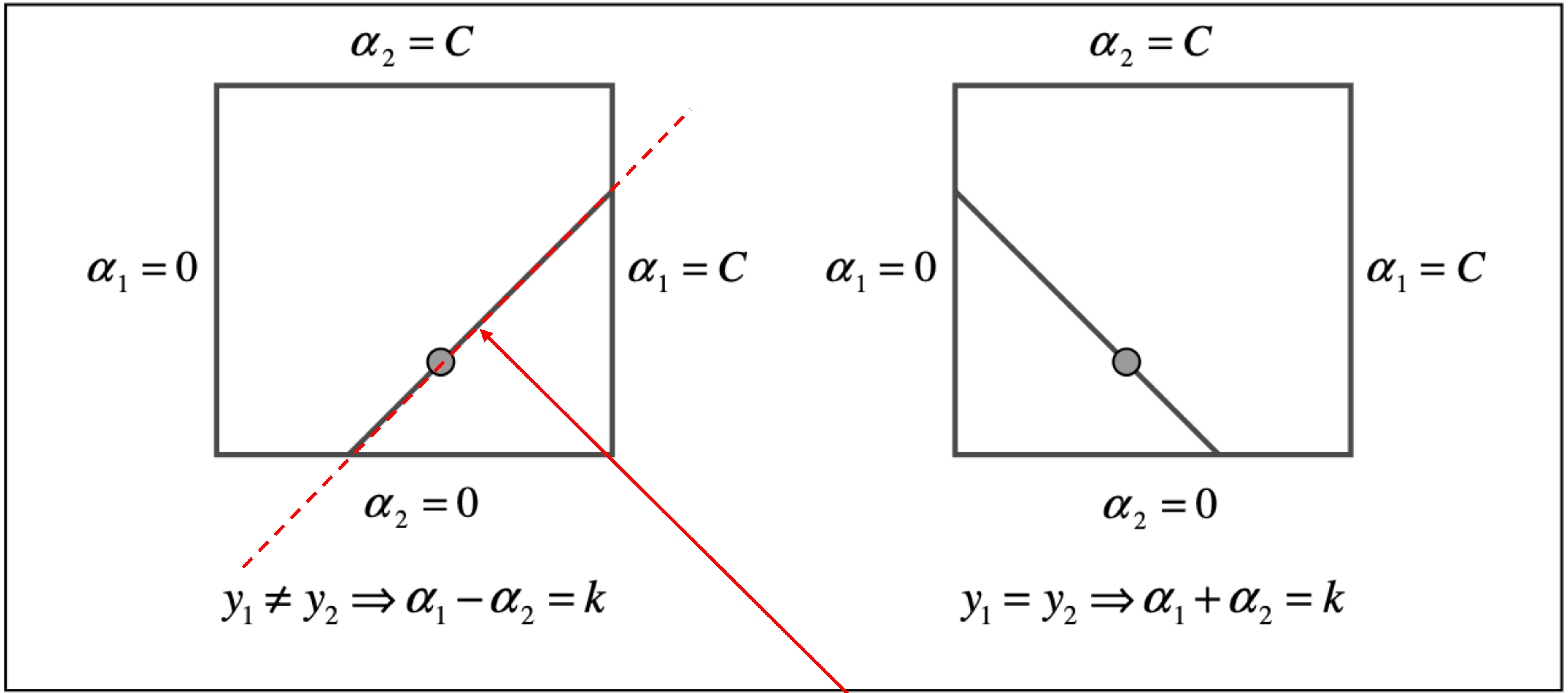
- Suppose given $\mathcal{D} : \{(x, y) : x \in \mathbb{R}^M \; and \; t \in \{-1, 1\}\}$ and $\kappa(x, x')$ we defined a soft margin SVM dual function below.
  How could we find the optimal $\lambda_1, \lambda_2 \; \dots, \lambda_n$?

$$\lambda *_{n=1}^{N} = \arg\max_{\lambda *} \sum_{n=1}^{N} \lambda *_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n^* \cdot t_n \lambda_m^* \cdot t_m \cdot \kappa(x_n, x_m)$$

$$\boxed{\begin{aligned} \text{subject to} \quad & 0 \le \lambda_n^* \le C \\ & \sum_{n=1}^{N} \lambda_n^* \cdot t_n = 0 \end{aligned}}$$

Let apply the concept of coordinate ascent to solve dual problem, but we have <u>the two </u>constraints.

In stead of updating one coordinate,
we will update a pair of two coordinates: (α_1, α_2) but others will be fixed.



$$y_1\alpha_1 + y_2\alpha_2 = -\sum_{i=3}^{N} y_i\alpha_i$$

$$y_1\alpha_1 + y_2\alpha_2 = k$$

$$L = \max(0, \alpha_2 - \alpha_1), \qquad H = \min(C, C + \alpha_2 - \alpha_1)$$

- we are going to find a minimum along the line.

Finding a minimum $\alpha_{2(new)}$ along the line : $y_1\alpha_1 - y_2\alpha_2 = k$

$$\alpha_{2new} = \alpha_{2old} + \frac{y_2(E_1 - E_2)}{\eta}$$

$$\eta = \kappa(x_1, x_1) + \kappa(x_2, x_2) - 2\kappa(x_1, x_2)$$

$$E_1 = (\sum_{n=1}^{N} \lambda_n^* t_n \kappa(x_n, x_1) + b) - y_1$$

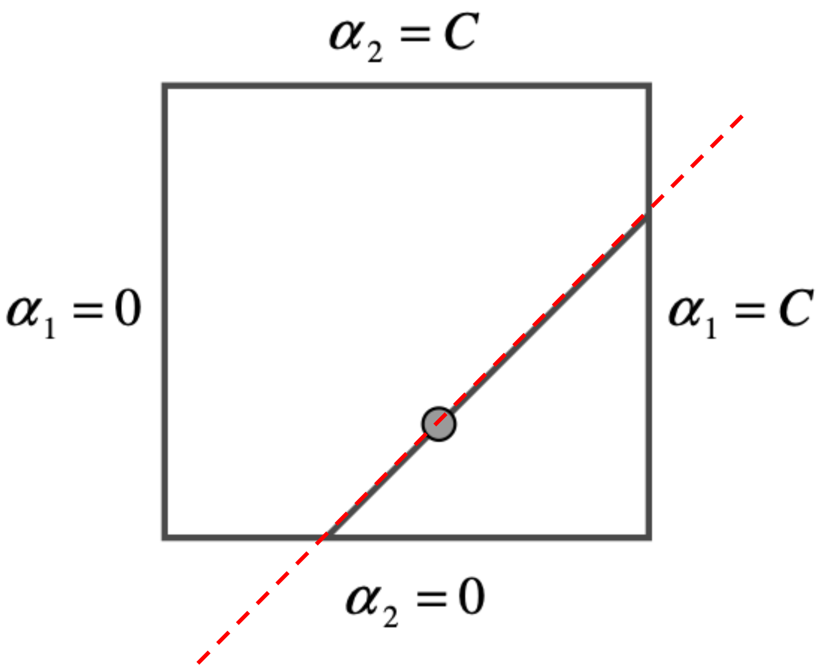
$$E_2 = (\sum_{n=1}^{N} \lambda_n^* t_n \kappa(x_n, x_2) + b) - y_2$$

This is derived by $\frac{\mathrm{d}J'(\alpha_2)}{\mathrm{d}\alpha_2} = 0$
The new alpha point is the minimum solution along the line.

For derivations:

https://dsmilab.github.io/Yuh-Jye-Lee/assets/file/teaching/2017_machine_learning/SMO_algorithm.pdf

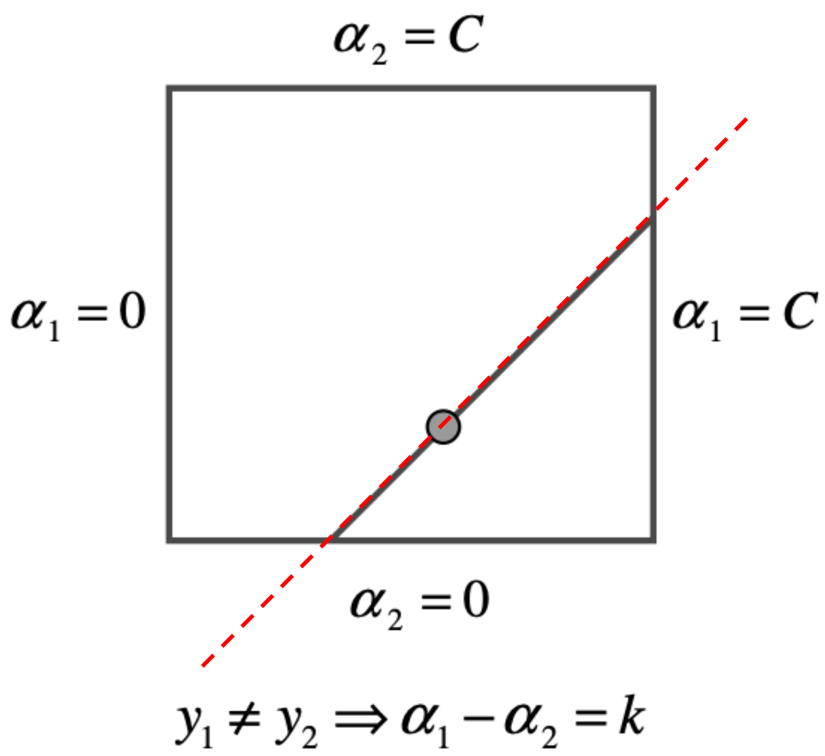
Clipping α_2 :
the minimum solution did not consider the bounds.
Hence, we need clipping



$y_1 \neq y_2 \Rightarrow \alpha_1 - \alpha_2 = k$

$$\alpha_2^{\text{new,clipped}} = \begin{cases} H & \text{if} \quad \alpha_2^{\text{new}} \geq H; \\ \alpha_2^{\text{new}} & \text{if} \quad L < \alpha_2^{\text{new}} < H; \\ \mathrm{L} & \text{if} \quad \alpha_2^{\text{new}} \leq L. \end{cases}$$

Once we find α_2, then we can update α_1. How?



SMO Algorithm

Repeat until KKT conditions are satisfied for all N training samples within certain tolerance (preset usually $10^{-3} \sim 10^{-2}$)

1. Pick two alphas (α_1 and α_2).

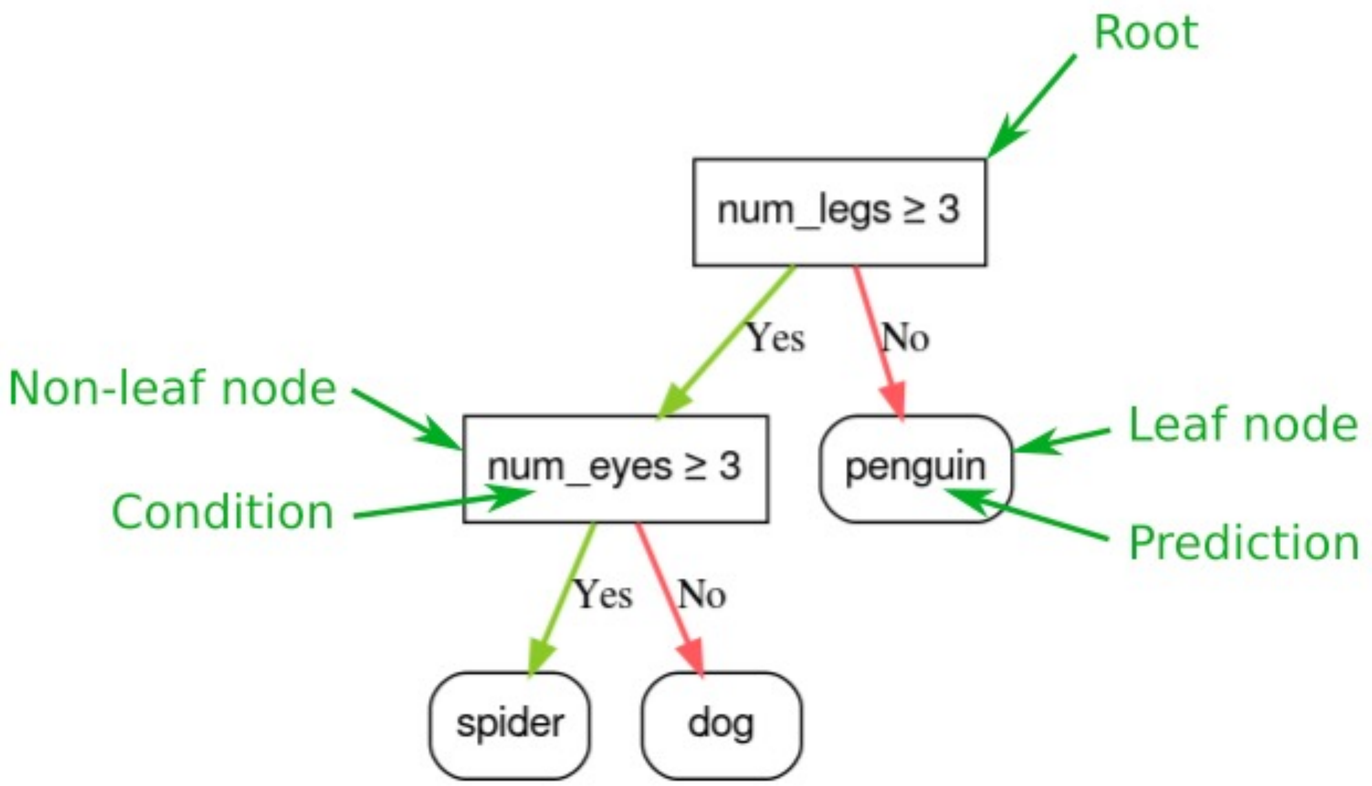
2. Define the range L and H for α_2.

3. Compute $$\alpha_{2new} = \alpha_{2old} + \frac{y_2(E_1 - E_2)}{\eta}$$

4. Clipping by L and H.

5. Update α_1.

Decision Tree

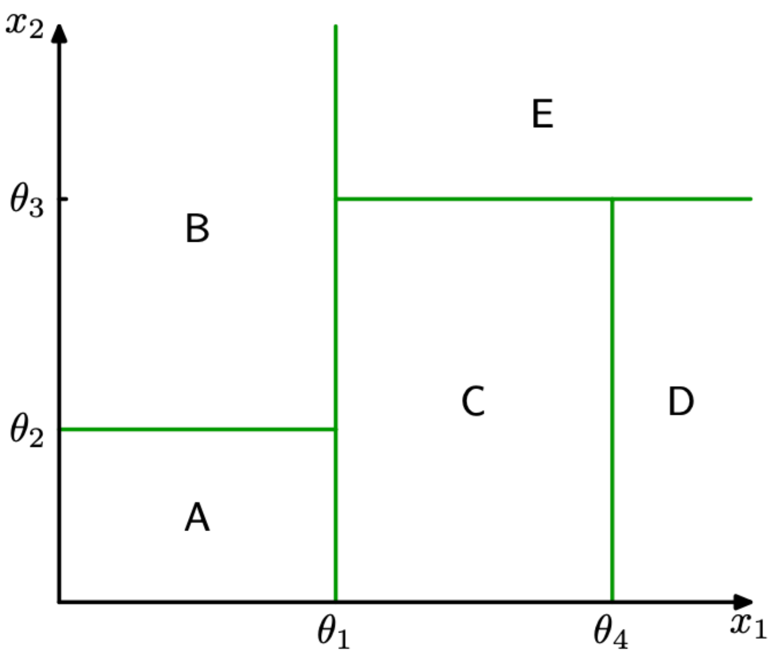


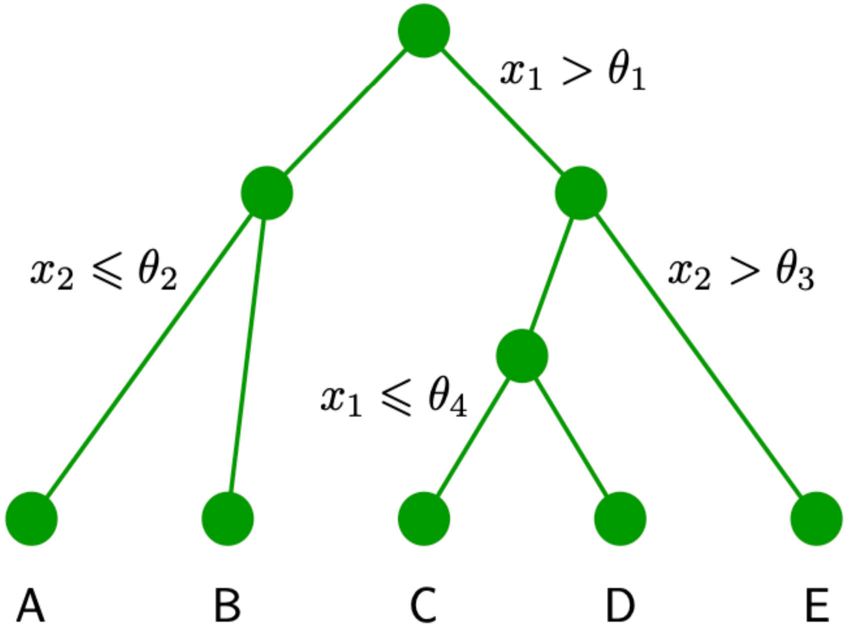
https://developers.google.com/machine-learning/decision-forests/decision-trees

From Bishop 14.5 and 14.6

Decision Tree

:partitioning input space into the regions whose edges are aligned with axes.





Suppose there exist a logical flow in our mind
as deciding to wait a restaurant. We want to estimate the tree structure.

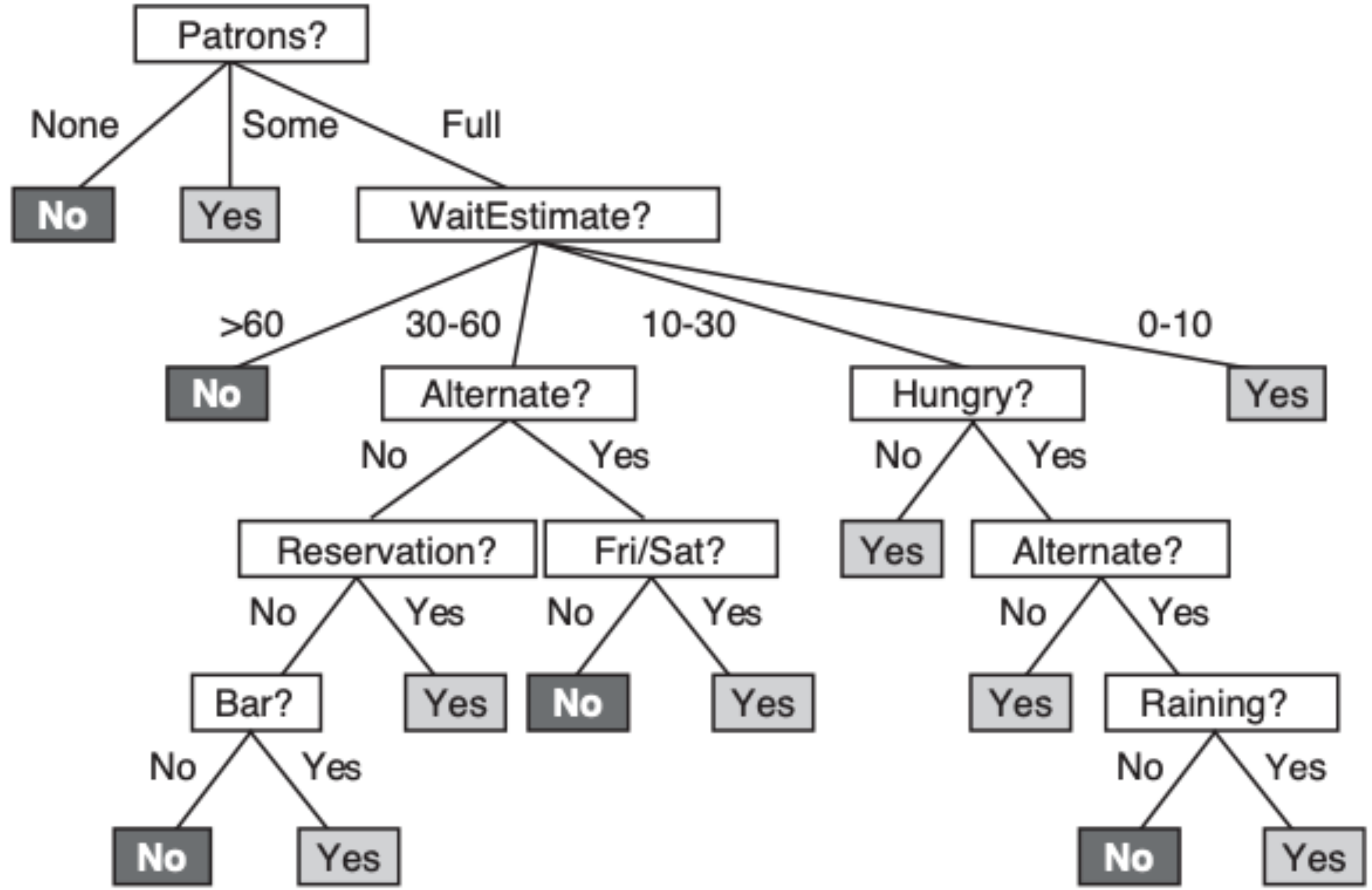


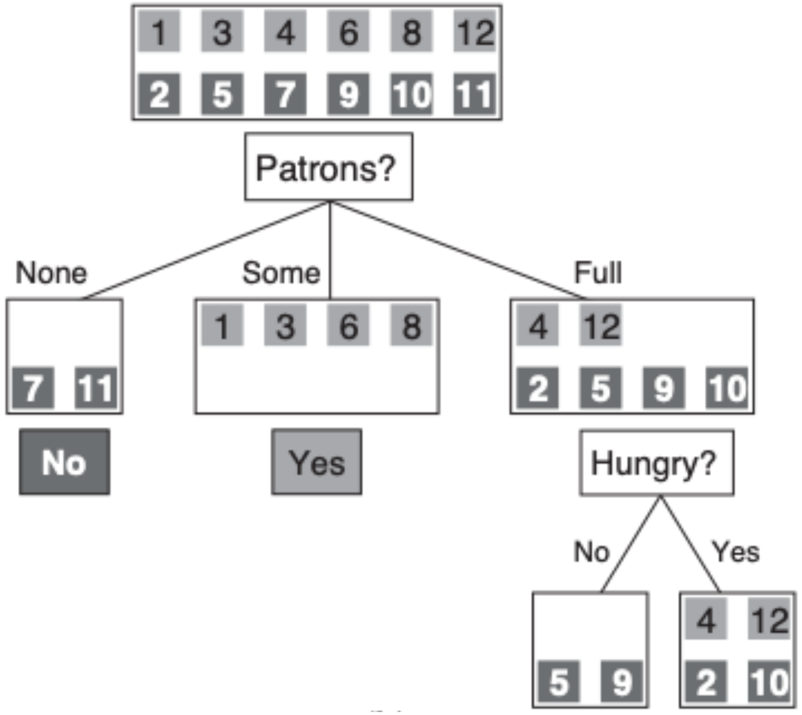
Figure 18.2 A decision tree for deciding whether to wait for a table.

Learning a decision tree.
Suppose we have categorical data representation.

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Figure 18.3 Examples for the restaurant domain.

A Simple Tree Algorithm



1. Choose the best feature to split

2. For each value that feature takes, sort training example to leaf nodes.

3. Stop if the leaf contain all training examples with same labels (nothing to divide)

4. Assign each leaf majority vote of labels of training examples.

5. Repeat 1,2,3,4 and create subtrees recursively until training error falls below some threshold.

Two Major Problems in Learning a Decision Tree

1. Choose the best feature to split (based on mutual information)

2. Decide threshold to make the data points categorical