

EE5904  
ME5404  
Part II

Plan

Introduction  
Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

# Support Vector Machines and Reinforcement Learning

Dr. Peter C. Y. Chen

Associate Professor

Department of Mechanical Engineering

Faculty of Engineering

National University of Singapore

Email: mpechenp@nus.edu.sg

[http://guppy.mpe.nus.edu.sg/peter\\_chen/](http://guppy.mpe.nus.edu.sg/peter_chen/)

These lecture slides were prepared using LaTex and the Beamer class

©Peter C. Y. Chen, 2007-2018

## ● Schedule

<b>Plan</b>	Support Vector Machines	15/3 22/3 29/3	Introduction. Margins. Primal problem Dual problems. Soft margin Kernels. <a href="#">Demo for SVM project</a>
	Reinforcement Learning	05/4 12/4 19/4	Introduction. Markov processes Dynamic programming <i>Q</i> -Learning. <a href="#">Demo for RL project</a>

## ● Assessment

- Projects for Part II (20% CA) – **Reports due on 27/04/2018**
- Closed-book exam; student can bring 1 A4-size data sheet

## ● Material and Reference

- This set of lecture slides
- Supplementary notes
- Haykin, Neural Networks, Prentice-Hall, 2ed, 1999
- Sutton and Barto, Reinforcement Learning, MIT Press, 1998

EE5904  
ME5404  
Part II

## Support vector machines



Vladimir N. Vapnik

U. of London

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

## Reinforcement learning



Richard Bellman

1920-1984



Richard Sutton

U. of Alberta



Andrew Barto

U. of Massachusetts



Chris Watkins

U. of London

EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

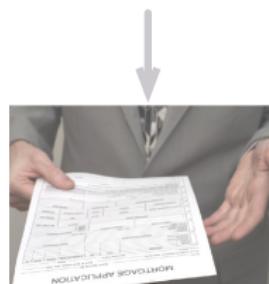
Q-learning

Explore/exploit

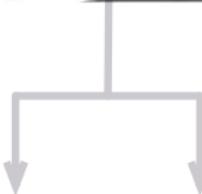
Implementation



Application



Evaluation



Decision

✓ Approve ✗ Reject

- age
- income
- education
- marital status
- debt
- etc

## Mortgage loan application: Use historical data with decisions made by human

EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

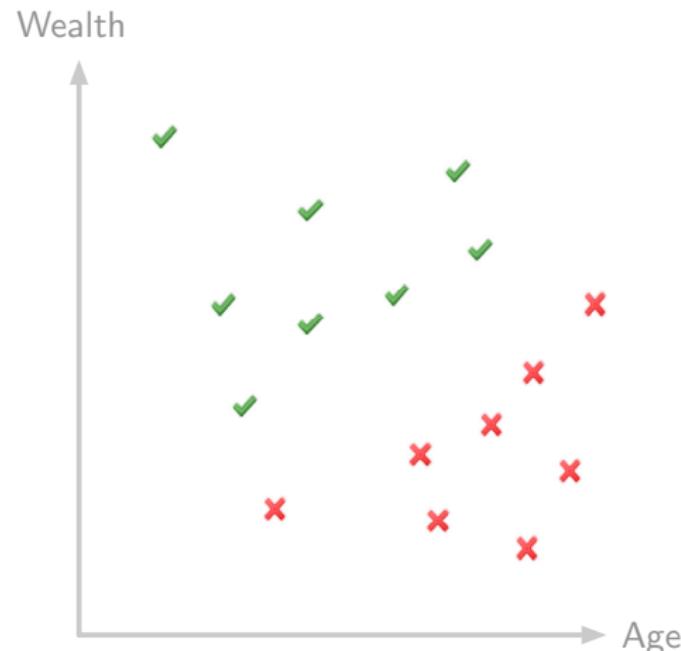
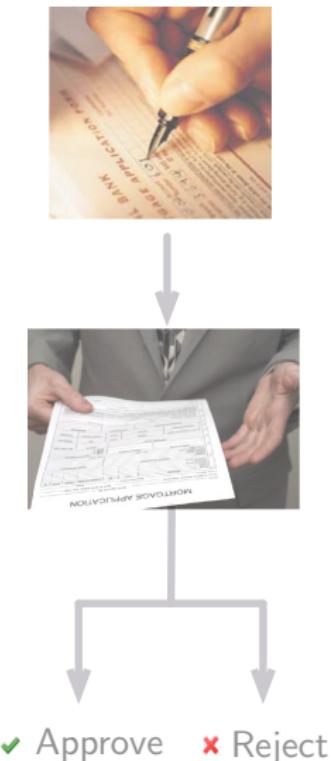
Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation



EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

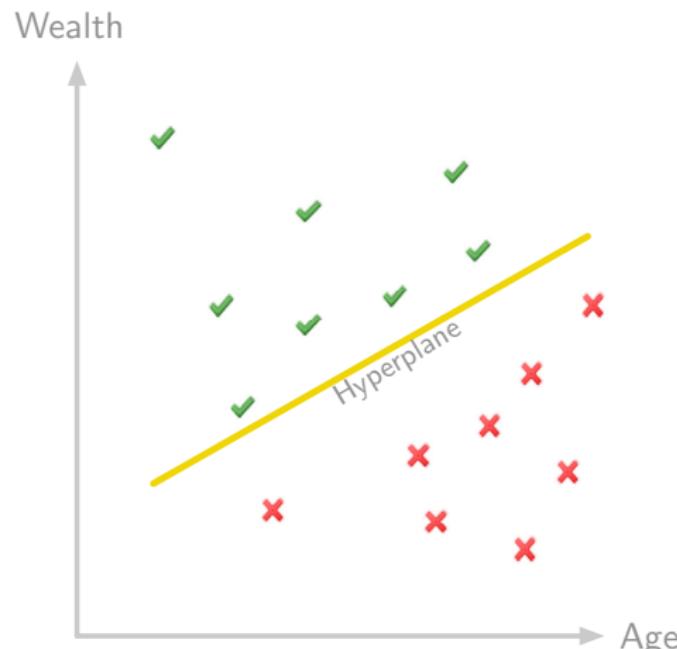
Q-learning

Explore/exploit

Implementation



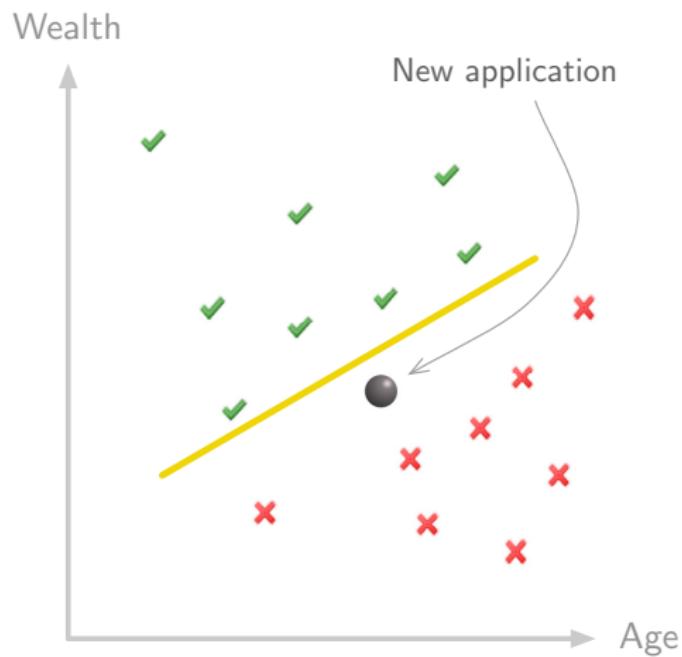
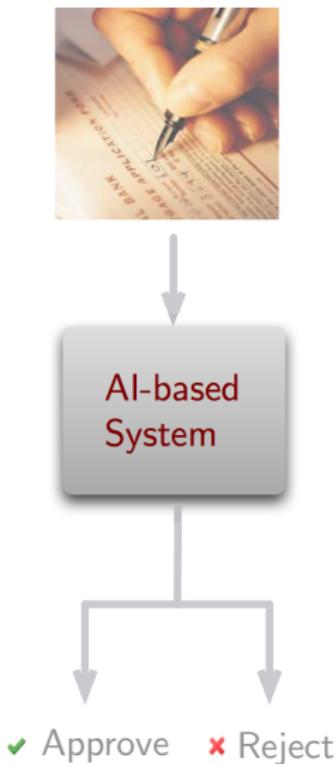
✓ Approve    ✗ Reject



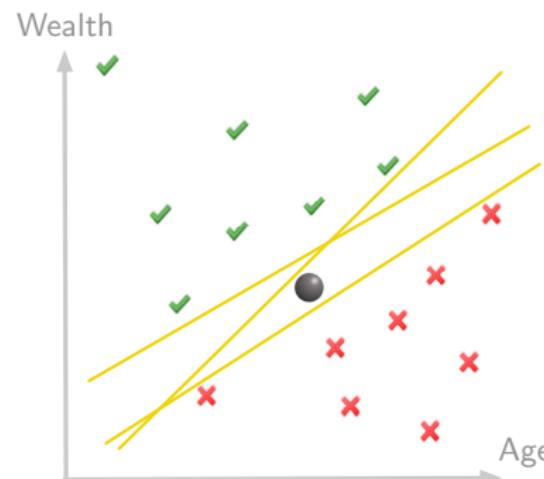
## Mortgage loan application: Trained system (i.e., weights found and fixed) generalizes

EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application  
  
**RL**  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation



- Many ways to separate the data
- Is there an “optimal” way ?



The theory of **support vector machines** provides a systematic method for separating the data “optimally”

## Plan

## Introduction

## Outcomes

## Math

## SVM

## Data

## Classification

## Margins

## Primal

## Lagrange

## Dual

## Soft margin

## Kernel

## Application

## RL

## Markov

## Q-function

## Bellman

## Optimality

## Value iteration

## Convergence

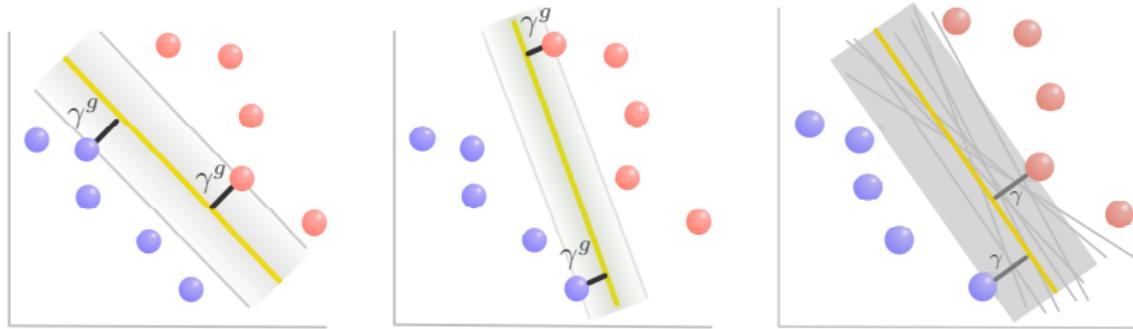
## Optimal policy

## Q-learning

## Explore/exploit

## Implementation

- Many hyperplanes correctly classify given data set  $S$
- $S$  has a geometric margin  $\gamma^g$  with respect to hyperplane



- The **optimal hyperplane** for a given  $S$  is one that gives **maximum  $\gamma$**  over all possible hyperplanes
- A SVM implements the optimal hyperplane for a given  $S$

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

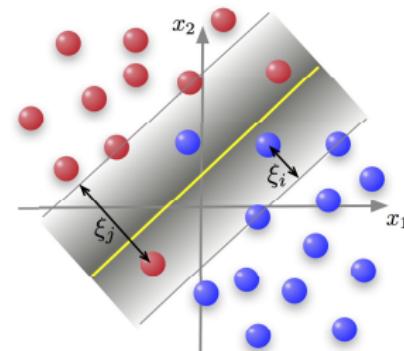
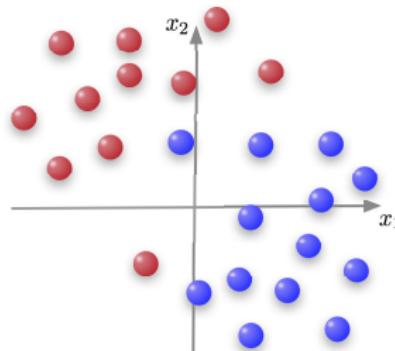
Optimal policy

Q-learning

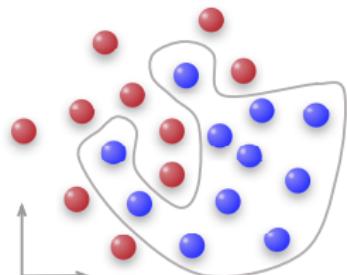
Explore/exploit

Implementation

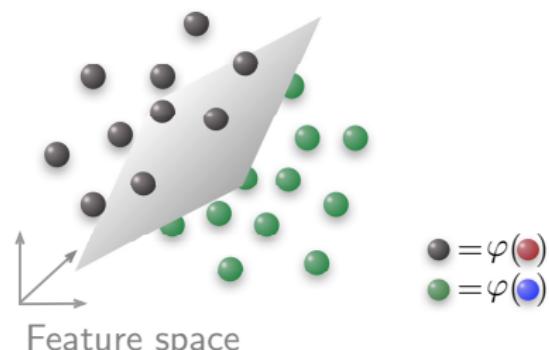
## 1. Find optimal hyperplane to minimize classification error



## 2. Transform data into higher dimension space for separation



Input space



## Advantage of SVM: Comparison of MLP and SVM solutions to XOR problem

EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

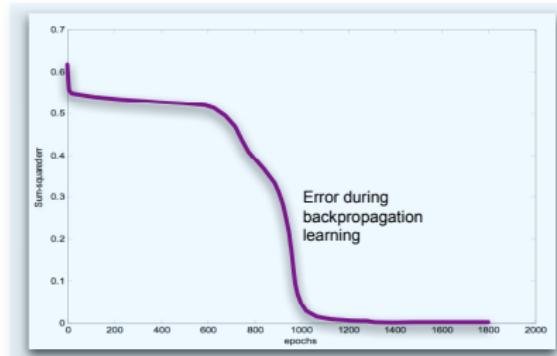
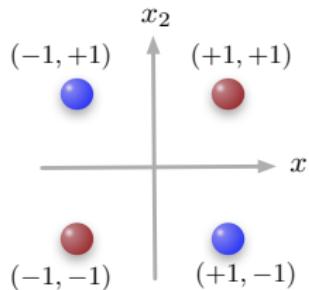
Convergence

Optimal policy

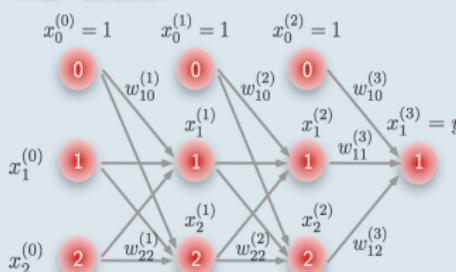
Q-learning

Explore/exploit

Implementation



## MLP solution



SUM of squared error

$$\text{bias} = \begin{bmatrix} -1.87 & 5.17 & 5.10 \\ -4.66 & 3.06 & 3.04 \end{bmatrix}$$

$$\mathbf{w}^{(1)} = \begin{bmatrix} \text{bias} & 1.06 & 3.69 & -5.29 \\ 2.53 & -4.67 & 3.54 \end{bmatrix}$$

$$\mathbf{w}^{(2)} = \begin{bmatrix} \text{bias} & -0.36 & 6.49 & -6.5147 \end{bmatrix}$$

## SVM solution

$$x_1 \rightarrow \text{sgn} [-x_1 x_2] \rightarrow y$$

**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

**MLP****SVM**

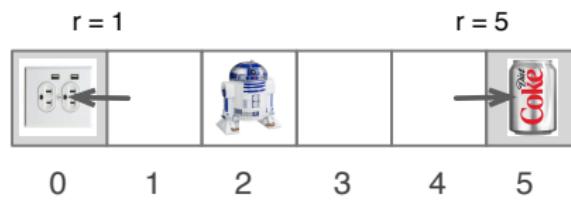
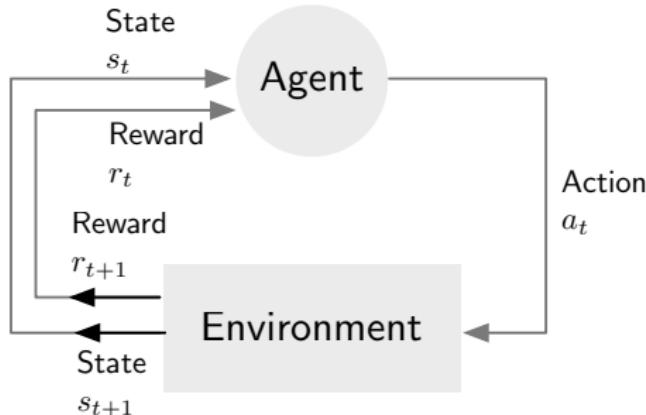
A solution among many

**Optimal** solution

Complicated structure

**Simpler** structureA “blackbox”  
(Convergence not guaranteed)**Tractable** solution process  
(Intuitive visualization  
for 2D problems)

## Agent learns to maximize reward when completing task



EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

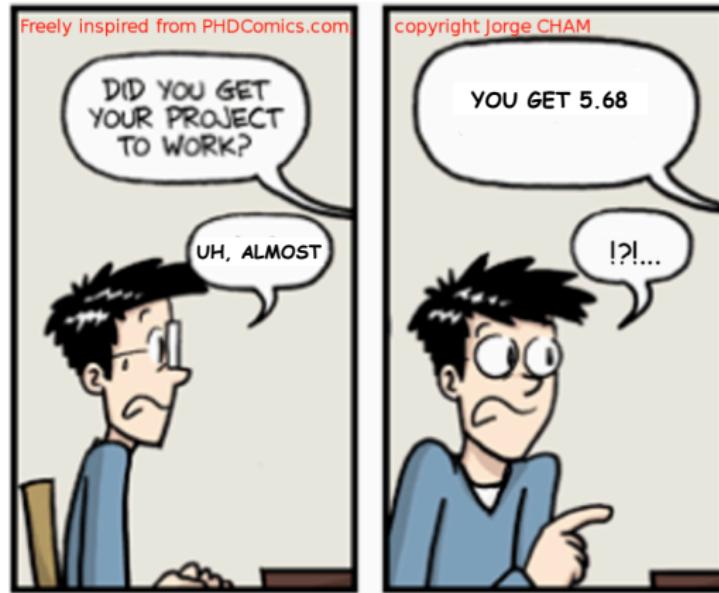
Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation



Adapted from slides by Olivier Sigaud

- Is 5.68 good or bad, how good or how bad?
- The score does not tell you how to get the project to work
- You have to learn how yourself by maximizing your scores

EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

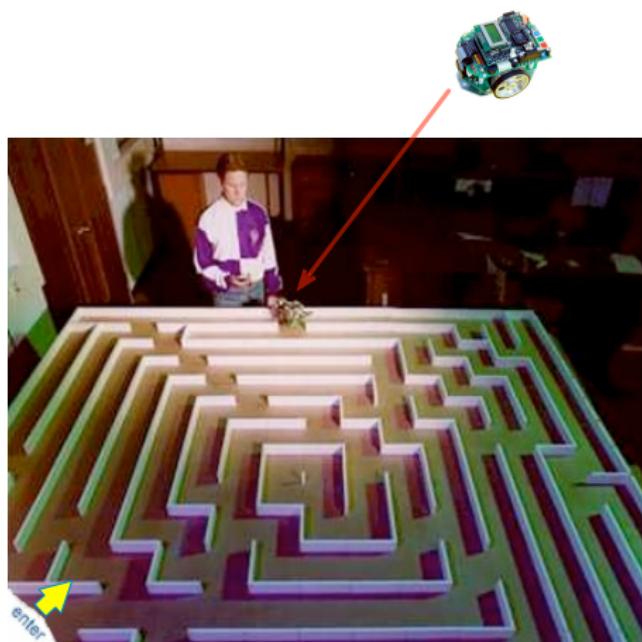
Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation



- Robot has no knowledge about maze before learning starts
- Robot explores maze and receives reward if it reaches goal
- Robot learns the “optimal” route to reach goal

# AlphaGo used RL to beat world No. 1 ranked player in 3-game match in 2017

EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

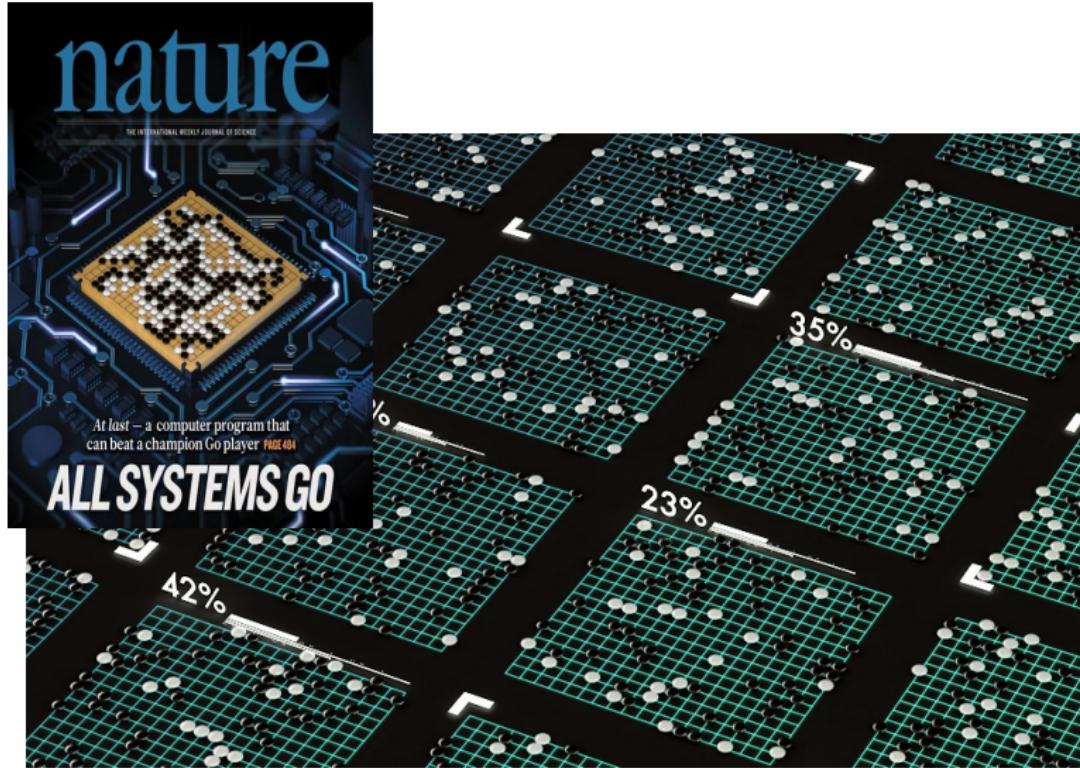
Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation



On successful completion of this part of module, students will be able to

1. Formulate pattern classification problems in the framework of support vector machines and apply optimization methods to find a solution
2. Develop computational solutions based on Markov decision theory for practical machine-learning problems involving optimization of outcomes

$\mathcal{R}$ : Set of real numbers

$\mathbf{w} \in \mathcal{R}^{n \times 1}$ :  $n$ -dimensional real column vector

$\mathbf{A} \in \mathcal{R}^{n \times n}$ :  $n \times n$  matrix

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

$$\text{sgn}[u] = \begin{cases} +1 & \text{if } u \geq 0 \\ -1 & \text{if } u < 0 \end{cases}$$

The signum function (also called sign function) is defined as

The gradient of a function  $f(\mathbf{w})$  is defined as

$$\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial f(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial f(\mathbf{w})}{\partial w_n} \end{bmatrix}$$

**Plan**  
**Introduction**

**Outcomes**

**Math**

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

The inner product of two vectors  $\mathbf{x} \in \mathcal{R}^{n \times 1}$  and  $\mathbf{y} \in \mathcal{R}^{n \times 1}$  is defined as

$$\langle \mathbf{x} \cdot \mathbf{y} \rangle = \sum_{i=1}^n x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n = \mathbf{x}^T \mathbf{y}$$

Two vectors  $\mathbf{u}$  and  $\mathbf{v}$  are said to be orthogonal if

$$\langle \mathbf{u} \cdot \mathbf{v} \rangle = \mathbf{u}^T \mathbf{v} = 0$$

Back

The Euclidean norm (also called the  $l_2$ -norm) of a vector  $\mathbf{x} \in \mathcal{R}^{n \times 1}$ , denoted by  $\|\mathbf{x}\|$ , is defined as

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \sqrt{\langle \mathbf{x} \cdot \mathbf{x} \rangle} \equiv \sqrt{\mathbf{x}^T \mathbf{x}}$$

**Plan****Introduction****Outcomes****Math**

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

For vectors  $\mathbf{u}, \mathbf{v} \in R^{n \times 1}$  and a matrix  $\mathbf{A} \in R^{n \times n}$ , we have the following rules

$$\mathbf{u}^T \mathbf{v} = \mathbf{v}^T \mathbf{u}$$

$$\frac{\partial (\mathbf{u}^T \mathbf{v})}{\partial \mathbf{v}} = \mathbf{u}$$

$$\frac{\partial (\mathbf{u}^T \mathbf{A} \mathbf{u})}{\partial \mathbf{u}} = (\mathbf{A} + \mathbf{A}^T) \mathbf{u}$$

If  $\mathbf{A} = \mathbf{I}$ , where  $\mathbf{I}$  denotes the identity matrix, then

$$\frac{\partial (\mathbf{u}^T \mathbf{A} \mathbf{u})}{\partial \mathbf{u}} = (\mathbf{A} + \mathbf{A}^T) \mathbf{u} = (\mathbf{I} + \mathbf{I}^T) \mathbf{u} = 2 \mathbf{u}$$

Back

Consider the equation

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

The non-zero vectors  $\mathbf{x}$  are called the eigenvectors of  $\mathbf{A}$ , and the corresponding values of  $\lambda$  are called the eigenvalues of  $\mathbf{A}$

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \Rightarrow \lambda\mathbf{x} - \mathbf{A}\mathbf{x} = 0 \Rightarrow (\lambda I - \mathbf{A})\mathbf{x} = 0$$

Existence of non-trivial solution requires that

$$|\lambda I - \mathbf{A}| = 0$$

whose solution gives eigenvalues  $\lambda$  of  $\mathbf{A}$

[Back](#)

Example: For a matrix  $\mathbf{A} = \begin{bmatrix} 3 & -2 \\ -1 & 4 \end{bmatrix}$ , we have

$$|\lambda I - \mathbf{A}| = \begin{vmatrix} \lambda - 3 & 2 \\ 1 & \lambda - 4 \end{vmatrix} = (\lambda - 3)(\lambda - 4) - 2 = 0$$

Solving for  $\lambda$  yields  $\lambda_1 = 5$  and  $\lambda_2 = 2$ . Substituting  $\lambda_1$  and  $\lambda_2$  back into the equation  $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$  then solving those equations yields the eigenvectors

$$\mathbf{x}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Suppose that function  $f(x)$  has maximum  $M$

The set of values of  $x$  for which  $f(x)$  attains  $M$  is expressed as

$$\operatorname{argmax}_x (f(x))$$

Example: For  $f(x) = \sin x$

$$M = \max [\sin x] = 1$$

and

$$\operatorname{argmax}_x (\sin x) = \{360^\circ k + 90^\circ\}$$

with  $k = 0, 1, 2, \dots$

Back

- Suppose a random variable  $X$  can take:

	Value	$x_1$	$x_2$	$\dots$	$x_k$
	With probability	$p_1$	$p_2$	$\dots$	$p_k$

- The expectation of  $X$  is defined as

$$E[X] = x_1 p_1 + x_2 p_2 + \dots + x_k p_k$$

$E$  is also called the expectation operator

- $E$  is linear with the following properties

$$E[X + Y] = E[X] + E[Y]$$

$$E[X + c] = E[X] + c$$

$$E[cX] = cE[X]$$

where  $c$  is a real constant

For two random variables  $X$  and  $Y$  and some value  $y$

$$E[X] = E[E[X|Y]] = \sum_y E[X|Y=y] \cdot P(Y=y)$$

- $E[X|Y=y]$  is the **conditional expectation** of  $X$  given the event  $Y=y$
- $P(Y=y)$  is the probability of  $Y$  taking value  $y$

▶ Back

Male 49%	Female 51%	Expected height of male 5'11"	Expected height of female 5'5"
-------------	---------------	----------------------------------	-----------------------------------

**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

What is the expected height of a randomly chosen individual?

$H$ : Height of individual      Female:  $G = 0$

$G$ : Gender of individual      Male:  $G = 1$

$$E[H|G = 0] = 5'5" \quad P(G = 0) = 0.51$$

$$E[H|G = 1] = 5'11" \quad P(G = 1) = 0.49$$

By the law of total expectation,

$$\begin{aligned} E[H] &= E[E[H|G]] = \sum_{g \in \{0,1\}} (E[H|G=g] \times P(G=g)) \\ &= E[H|G=0] \times P(G=0) + E[H|G=1] \times P(G=1) \\ &= 5'5" \times 0.51 + 5'11" \times 0.49 = 5'8" \end{aligned}$$

EE5904  
ME5404  
Part II

## Plan

### Introduction

### Outcomes

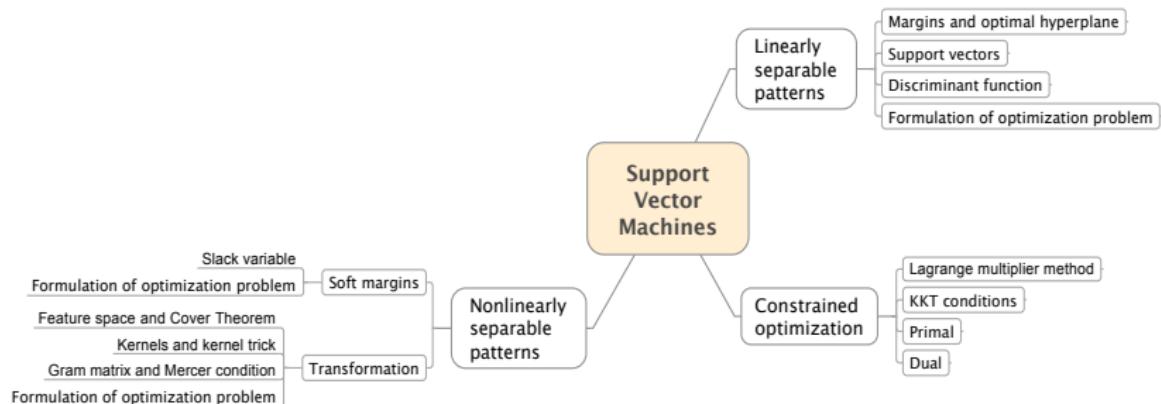
### Math

### SVM

Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

### RL

Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

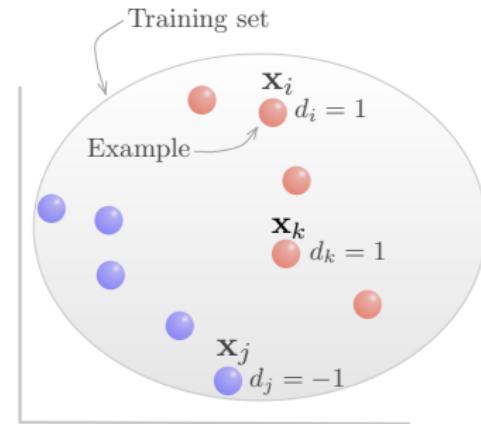


## Example: An $n$ -dimensional vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

assigned with a label

$$d = \pm 1$$



We write an example as:  $(\mathbf{x}, d)$

**Training set:** A set  $S$  of  $N$  examples, i.e.,

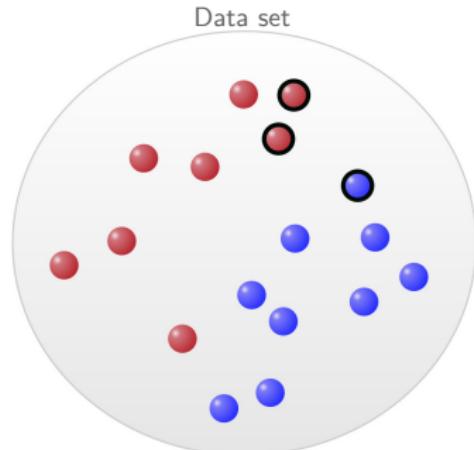
$$S = \left\{ (\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), \dots, (\mathbf{x}_N, d_N) \right\}$$

Data set  $\Sigma$  containing  $(N + \bar{N})$  examples is partitioned into

a training set:  $S = \{(\mathbf{x}_1, d_1), \dots, (\mathbf{x}_N, d_N)\}$

a test set:  $\bar{S} = \{(\bar{\mathbf{x}}_1, \bar{d}_1), \dots, (\bar{\mathbf{x}}_{\bar{N}}, \bar{d}_{\bar{N}})\}$

That is,  $\Sigma = S \cup \bar{S}$ , with  $S \cap \bar{S} = \emptyset$

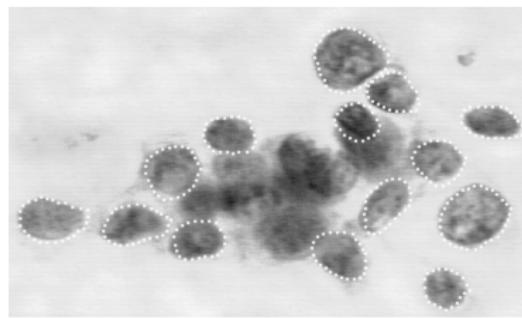


- Training data is for constructing SVM
- Test data is for evaluating performance of SVM

More on training and testing later

## Breast Cancer Wisconsin (Diagnostic) Data Set

<http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29>



883852, B, 11.3, 18.19, 73.93, 389.4, 0.09592,  
0.1325, 0.1548, 0.02854, 0.2054, 0.07669, 0.2428,  
1.642, 2.369, 16.39, 0.006663, 0.05914, 0.0888,  
0.01314, 0.01995, 0.008675, 12.58, 27.96, 87.16,  
472.9, 0.1347, 0.4848, 0.7436, 0.1218, 0.3308,  
0.1297  
842302, M, 17.99, 10.38, 122.8, 1001, 0.1184,  
0.2776, 0.3001, 0.1471, 0.2419, 0.07871, 1.095,  
0.9053, 8.589, 153.4, 0.006399, 0.04904, 0.05373,  
0.01587, 0.03003, 0.006193, 25.38, 17.33, 184.6,  
2019, 0.1622, 0.6656, 0.7119, 0.2654, 0.4601, 0.1189  
⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮

First number=Patient ID. B=Benign. M=Malignant

Each sample has 30 real-valued features. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

W.N. Street, W.H. Wolberg and O.L. Mangasarian, Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.

EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

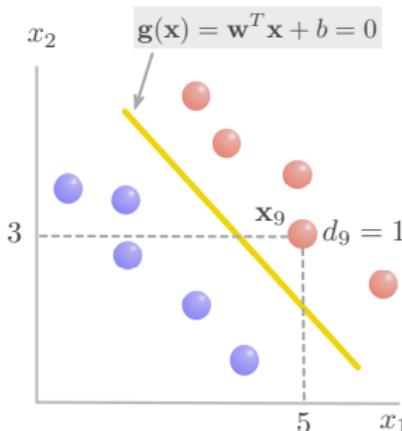
## Spam Email Data Set

<http://archive.ics.uci.edu/ml/datasets/spambase>



```
train_data = [ [ 0.00000 0.01043 0.00000 0.00000  
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000  
0.01043 0.01043 0.02105 0.00000 0.00000 0.00000  
0.00000 0.03166 0.06332 0.00000 0.02105 0.00000  
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000  
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000  
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000  
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000  
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000  
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000  
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000  
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000  
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000  
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000  
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000  
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000  
0.00000 0.00000 0.02601 0.12811 0.98827], [ ... ],  
... ]  
train_label = [ 1, ..... ]
```

Each example has a feature vector with 57 real-valued attributes.  
Most of the attributes indicate whether a particular word or character  
was frequently occurring in the e-mail.



A hyperplane, denoted by  $(\mathbf{w}, b)$ , can be expressed as

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = 0$$

Hyperplane classifies a given  $\mathbf{x}_i$  with

$$\operatorname{sgn}[g(\mathbf{x}_i)] = \begin{cases} +1 & \text{if } g(\mathbf{x}_i) > 0 \\ -1 & \text{if } g(\mathbf{x}_i) < 0 \end{cases}$$

Hyperplane classifies an example  $(\mathbf{x}_i, d_i)$  correctly if

$$\operatorname{sgn}[g(\mathbf{x}_i)] = d_i \quad \text{or} \quad \operatorname{sgn}[d_i g(\mathbf{x}_i)] = 1$$

Problem: Let  $\mathbf{w}^T = [ \ 7 \ 6 \ ]$  and  $b = -42$ . For  $\mathbf{x}_9^T = [ \ 5 \ 3 \ ]$

$$\mathbf{g}(\mathbf{x}_9) = \mathbf{w}^T \mathbf{x}_9 + b = [ \ 7 \ 6 \ ] \begin{bmatrix} 5 \\ 3 \end{bmatrix} - 42 = 11 > 0$$

$$\operatorname{sgn}[\mathbf{g}(\mathbf{x}_9)] = \operatorname{sgn}[11] = 1 = d_9$$

**Plan****Introduction****Outcomes****Math****SVM**

Data

**Classification**

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

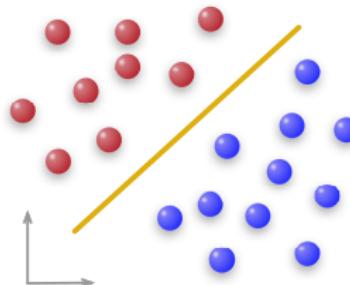
Optimal policy

Q-learning

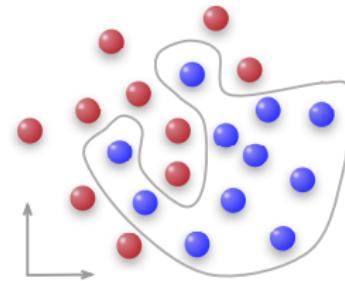
Explore/exploit

Implementation

- Two classes of data are **linearly separable** if and only if there exists a hyperplane  $\mathbf{w}^T \mathbf{x} + b = 0$  that separates the two classes
- Example in 2D:



Linearly separable



Nonlinearly separable

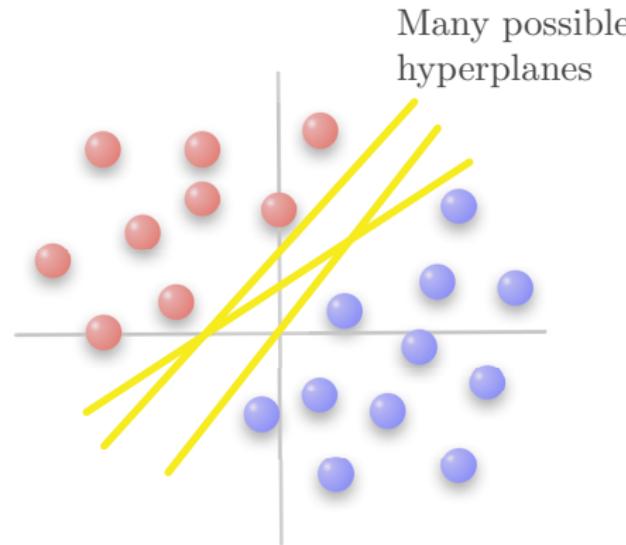
We consider the separable case first

Plan  
Introduction  
Outcomes  
Math  
**SVM**  
Data  
Classification

Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

RL  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

# Is there an “optimal” way to separate the data?



The theory of support vector machines provides a systematic method for separating the data “optimally”

It involves a key concept called **margin**

$\gamma_i^f$ : Functional margin of an example  $(\mathbf{x}_i, d_i)$   
 $\gamma_i^g$ : Geometric margin of an example  $(\mathbf{x}_i, d_i)$   
 $\gamma^f$ : Functional margin of a training set  
 $\gamma^g$ : Geometric margin of a training set  
 $\gamma$ : Margin of a training set

	Example	Training set
Functional margin	$\gamma_i^f$	$\gamma^f$
Geometric margin	$\gamma_i^g$	$\gamma^g$
Margin	—	$\gamma$

$\gamma_i^f$ ,  $\gamma^f$ ,  $\gamma_i^g$ , and  $\gamma^g$  are defined w.r.t. a given hyperplane  $(\mathbf{w}, b)$

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

The functional margin of an example  $(\mathbf{x}_i, d_i)$  with respect to a hyperplane  $(\mathbf{w}, b)$  is defined as

$$\gamma_i^f = d_i (\mathbf{w}^T \mathbf{x}_i + b)$$

### Problem:

Training set with only 1 example

Hyperplane defined by

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \quad d_1 = +1$$

$$\mathbf{w} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}, \quad b = 6$$

Determine  $\gamma_1^f$  with respect to given hyperplane

### Solution:

$$\gamma_1^f = d_1 (\mathbf{w}^T \mathbf{x}_1 + b) = 1 \times \left( \begin{bmatrix} 5 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 6 \right) = 1 \times 23 = 23$$

The geometric margin of an example  $(\mathbf{x}_i, d_i)$  with respect to a hyperplane  $(\mathbf{w}, b)$  is defined as

$$\gamma_i^g = d_i \left( \frac{1}{\|\mathbf{w}\|} \mathbf{w}^T \mathbf{x}_i + \frac{1}{\|\mathbf{w}\|} b \right)$$

### Problem:

Training set with only 1 example

Hyperplane defined by

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \quad d_1 = +1 \qquad \qquad \mathbf{w} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}, \quad b = 6$$

Determine  $\gamma_1^g$  with respect to given hyperplane

### Solution:

$$\gamma_1^g = d_1 \left( \frac{\mathbf{w}^T \mathbf{x}_1}{\|\mathbf{w}\|} + \frac{b}{\|\mathbf{w}\|} \right) = \frac{[5 \quad 3] \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 6}{\sqrt{5^2 + 3^2}} = \frac{23}{\sqrt{34}}$$

EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

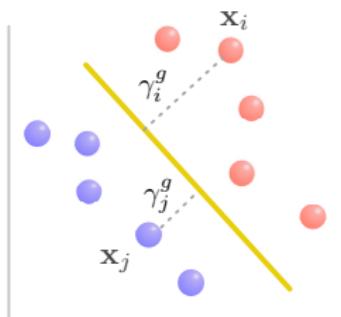
Optimal policy

Q-learning

Explore/exploit

Implementation

$$\gamma_i^g = d_i \left( \frac{1}{\|\mathbf{w}\|} \mathbf{w}^T \mathbf{x}_i + \frac{1}{\|\mathbf{w}\|} b \right)$$

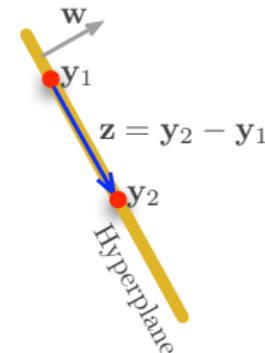


$\gamma_i^g$  is the Euclidean distance from the point defined by  $\mathbf{x}_i$  to the hyperplane  $(\mathbf{w}, b)$ .

$\gamma_i^g$  is the Euclidean distance from  $x_i$  to hyperplane  $(w, b)$

**Proof:** We first show that the unit vector  $\frac{w}{\|w\|}$  is perpendicular to hyperplane  $(w, b)$ . Consider points  $y_1$  and  $y_2$  on the hyperplane. By definition

$$\begin{aligned}\langle w \cdot y_1 \rangle + b &= w^T y_1 + b = 0 \\ \langle w \cdot y_2 \rangle + b &= w^T y_2 + b = 0\end{aligned}$$



The vector from  $y_1$  to  $y_2$  is  $z = y_2 - y_1$ . Now

$$\begin{aligned}\langle w \cdot z \rangle &= \langle w \cdot (y_2 - y_1) \rangle = \langle w \cdot y_2 \rangle - \langle w \cdot y_1 \rangle \\ &= w^T y_2 - w^T y_1 = -b - (-b) \\ &= 0\end{aligned}$$

▶ View

Hence,  $w \perp$  hyperplane  $(w, b)$ .

EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**

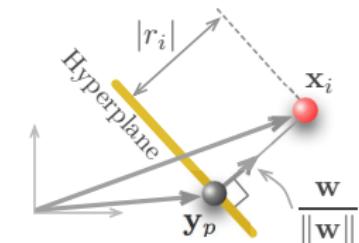
**Math**  
**SVM**  
**Data**  
**Classification**  
**Margins**  
**Primal**  
**Lagrange**  
**Dual**  
**Soft margin**  
**Kernel**  
**Application**

**RL**  
**Markov**  
**Q-function**  
**Bellman**  
**Optimality**  
**Value iteration**  
**Convergence**  
**Optimal policy**  
**Q-learning**  
**Explore/exploit**  
**Implementation**

We can now express  $\mathbf{x}_i$  as

$$\mathbf{x}_i = \mathbf{y}_p + r_i \left( \frac{\mathbf{w}}{\|\mathbf{w}\|} \right)$$

where  $|r_i|$  is the distance from  $\mathbf{x}_i$  to hyperplane.



Note that  $\mathbf{w}^T \mathbf{y}_p + b = 0$ , or,  $\mathbf{w}^T \mathbf{y}_p = -b$  (since  $\mathbf{y}_p$  is on hyperplane)

$$\begin{aligned} \mathbf{w}^T \mathbf{x}_i + b &= \mathbf{w}^T \left( \mathbf{y}_p + \frac{r_i \mathbf{w}}{\|\mathbf{w}\|} \right) + b = \mathbf{w}^T \mathbf{y}_p + \frac{r_i \mathbf{w}^T \mathbf{w}}{\|\mathbf{w}\|} + b \\ &= -b + \frac{r_i \|\mathbf{w}\|^2}{\|\mathbf{w}\|} + b = r_i \|\mathbf{w}\| \end{aligned}$$

So

$$r_i = \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|} = \left( \frac{1}{\|\mathbf{w}\|} \mathbf{w}^T \mathbf{x}_i \right) + \frac{b}{\|\mathbf{w}\|}$$

Note that  $\gamma_i^g = d_i r_i$ . Since  $d_i = \pm 1$ , and  $d_i$  and  $(\mathbf{w}^T \mathbf{x}_i + b)$  have the same sign (for a correctly classified example), we can conclude that  $\gamma_i^g = |r_i|$ , which is the Euclidian distance from  $\mathbf{x}_i$  to the hyperplane.

Plan  
Introduction  
Outcomes

Math

SVM  
Data  
Classification

Margins  
Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

The equation representing a hyperplane is invariant when scaled by an arbitrary real constant.

The hyperplane described by the equation

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = \mathbf{w}^T \mathbf{x} + b = 0$$

can also be described by

$$\langle c \mathbf{w} \cdot \mathbf{x} \rangle + cb = c \mathbf{w}^T \mathbf{x} + cb = 0$$

where  $c$  is a real number.

**Problem:** (Note that from earlier slide,  $\gamma_1^f = 23$  and  $\gamma_1^g = \frac{23}{\sqrt{34}}$ )

Training set with only 1 example      Given hyperplane

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, d_1 = +1$$

$$\mathbf{w} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}, b = 6$$

Find scaling constant  $c$  such that  $\gamma_1^f = 5$  with respect to the given hyperplane, and calculate  $\gamma_1^g$  associated with  $c\mathbf{w}$  and  $cb$ .

**Solution:**

$$d_1(c\mathbf{w}^T \mathbf{x}_1 + cb) = 1 \left( c \begin{bmatrix} 5 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 6c \right) = 23c = 5 \Rightarrow c = \frac{5}{23}$$

Thus

$$c\mathbf{w} = \frac{5}{23} \begin{bmatrix} 5 \\ 3 \end{bmatrix} = \begin{bmatrix} 25/23 \\ 15/23 \end{bmatrix}, \quad cb = \frac{30}{23}$$

and the geometric margin of the example with respect to  $c\mathbf{w}$  and  $cb$  is

$$\gamma_1^g = \frac{\gamma_1^f}{\|c\mathbf{w}\|} = \frac{5}{\sqrt{\left(\frac{25}{23}\right)^2 + \left(\frac{15}{23}\right)^2}} = \frac{5}{\frac{5\sqrt{34}}{23}} = \frac{23}{\sqrt{34}}$$

## Functional margin of an example

$$\gamma_i^f = d_i (\mathbf{w}^T \mathbf{x}_i + b)$$

## Geometric margin of an example

$$\gamma_i^g = d_i \left( \frac{1}{\|\mathbf{w}\|} \mathbf{w}^T \mathbf{x}_i + \frac{1}{\|\mathbf{w}\|} b \right)$$

Therefore, by definition:

$$\gamma_i^g = \frac{\gamma_i^f}{\|\mathbf{w}\|}$$

**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

**Margins**

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

The **functional margin of a training set  $S$** , with respect to a hyperplane  $(\mathbf{w}, b)$ , is the minimum of all the functional margins of the individual examples in  $S$ , i.e.,

$$\gamma^f = \min_{1 \leq i \leq N} \left\{ \gamma_i^f \right\}$$

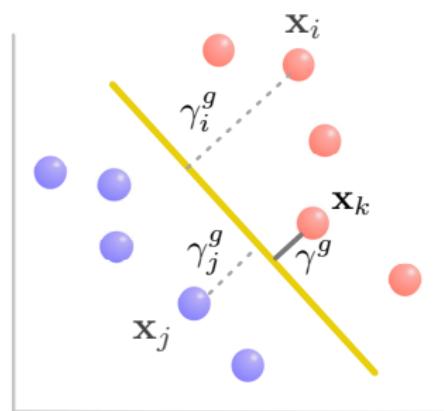
where

$$\gamma_i^f = d_i (\mathbf{w}^T \mathbf{x}_i + b)$$

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
 Data  
 Classification  
**Margins**

Primal  
 Lagrange  
 Dual  
 Soft margin  
 Kernel  
 Application

**RL**  
 Markov  
 Q-function  
 Bellman  
 Optimality  
 Value iteration  
 Convergence  
 Optimal policy  
 Q-learning  
 Explore/exploit  
 Implementation



The **geometric margin** of a training set  $S$ , with respect to a hyperplane  $(\mathbf{w}, b)$ , is the minimum of all the geometric margins of the individual examples in  $S$ , i.e.,

$$\gamma^g = \min_{1 \leq i \leq N} \{\gamma_i^g\}$$

where

$$\gamma_i^g = d_i \left( \frac{1}{\|\mathbf{w}\|} \mathbf{w}^T \mathbf{x}_i + \frac{1}{\|\mathbf{w}\|} b \right)$$

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

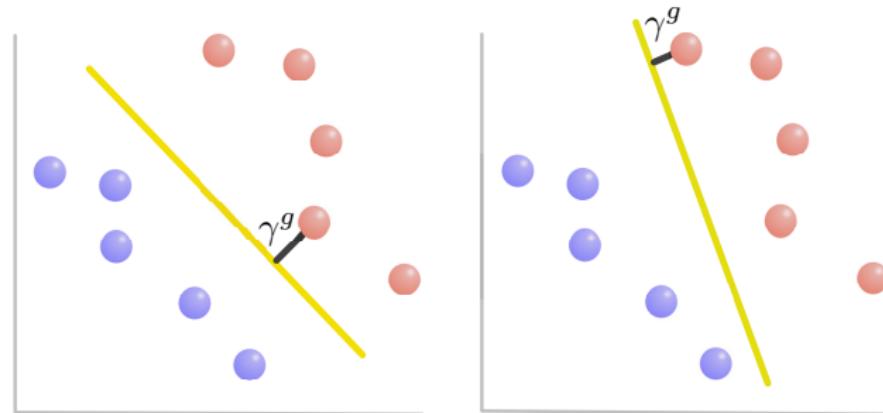
Optimal policy

Q-learning

Explore/exploit

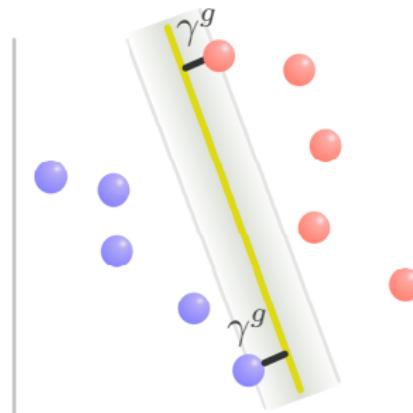
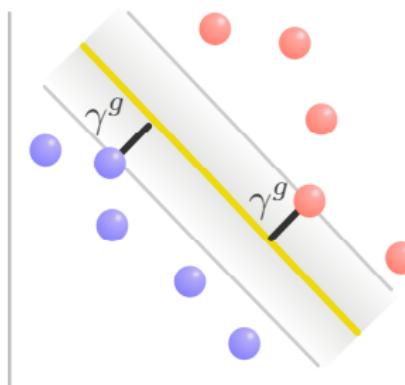
Implementation

- A training set  $S$  can have different geometric margins depending on how hyperplane is defined

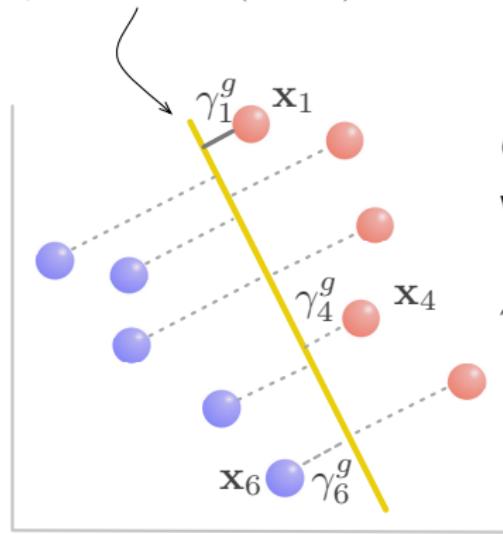


- The **optimal hyperplane** for a given  $S$  is one that gives **maximum  $\gamma^g$**  over all possible hyperplanes

- The optimal hyperplane for a given  $S$  is one that gives maximum  $\gamma^g$  over all possible hyperplanes
- This maximum  $\gamma^g$  is called the **margin of  $S$**



Hyperplane 1:  $(\mathbf{w}_1, b_1)$



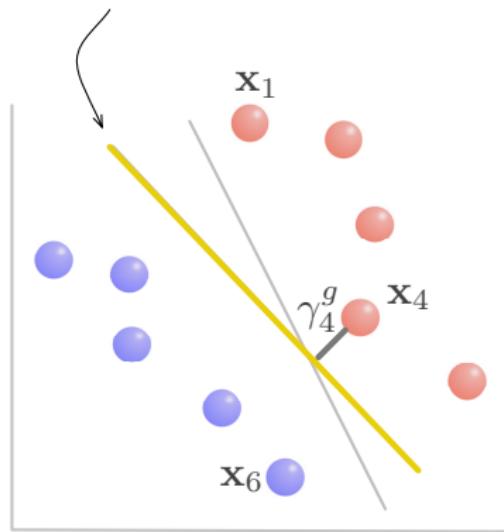
Geometric margin of **an example  $i$**  with respect to **hyperplane 1** is

$$\gamma_i^g = d_i \left( \left\langle \frac{1}{\|\mathbf{w}_1\|} \mathbf{w}_1 \cdot \mathbf{x}_i \right\rangle + \frac{1}{\|\mathbf{w}_1\|} b_1 \right)$$

Geometric margin of **training set** with respect to hyperplane 1 is

$$\min\{\gamma_1^g, \dots, \gamma_{10}^g\} = \gamma_1^g \equiv \gamma_{(\mathbf{w}_1, b_1)}^g$$

Hyperplane 2:  $(\mathbf{w}_2, b_2)$



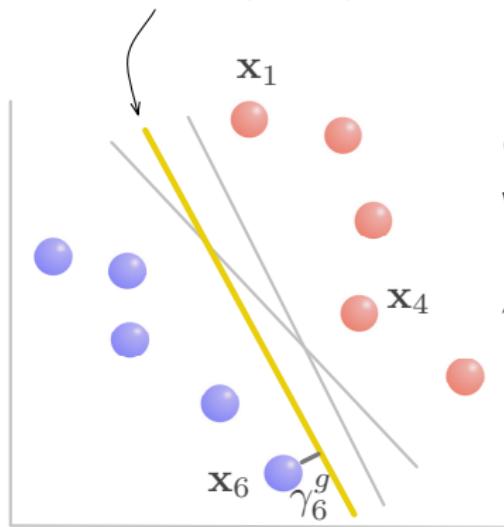
Geometric margin of an example  $i$  with respect to **hyperplane 2** is

$$\gamma_i^g = d_i \left( \left\langle \frac{1}{\|\mathbf{w}_2\|} \mathbf{w}_2 \cdot \mathbf{x}_i \right\rangle + \frac{1}{\|\mathbf{w}_2\|} b_2 \right)$$

Geometric margin of **training set** with respect to hyperplane 2 is

$$\min\{\gamma_1^g, \dots, \gamma_{10}^g\} = \gamma_4^g \equiv \gamma_{(\mathbf{w}_2, b_2)}^g$$

Hyperplane 3:  $(\mathbf{w}_3, b_3)$



Geometric margin of an example  $i$  with respect to **hyperplane 3** is

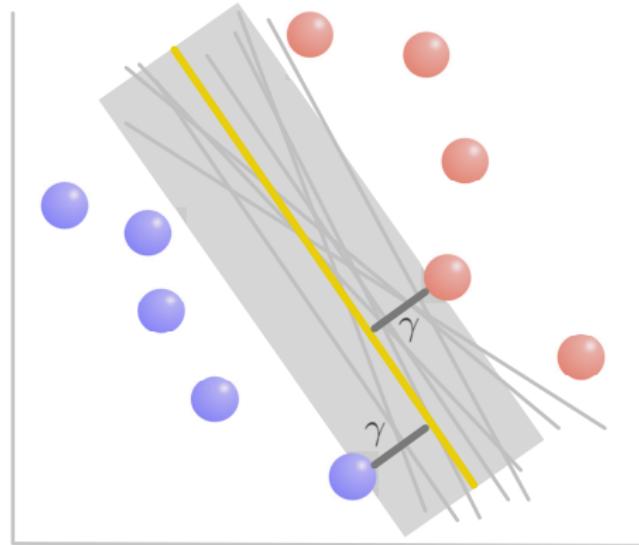
$$\gamma_i^g = d_i \left( \left\langle \frac{1}{\|\mathbf{w}_3\|} \mathbf{w}_3 \cdot \mathbf{x}_i \right\rangle + \frac{1}{\|\mathbf{w}_3\|} b_3 \right)$$

Geometric margin of **training set** with respect to hyperplane 3 is

$$\min\{\gamma_1^g, \dots, \gamma_{10}^g\} = \gamma_6^g \equiv \gamma_{(\mathbf{w}_3, b_3)}^g$$

EE5904  
ME5404  
Part II

$$\gamma = \max \underbrace{\left\{ \gamma_{(\mathbf{w}_1, b_1)}^g, \gamma_{(\mathbf{w}_2, b_2)}^g, \gamma_{(\mathbf{w}_3, b_3)}^g, \dots \right\}}_{\text{All possible hyperplanes}}$$



Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

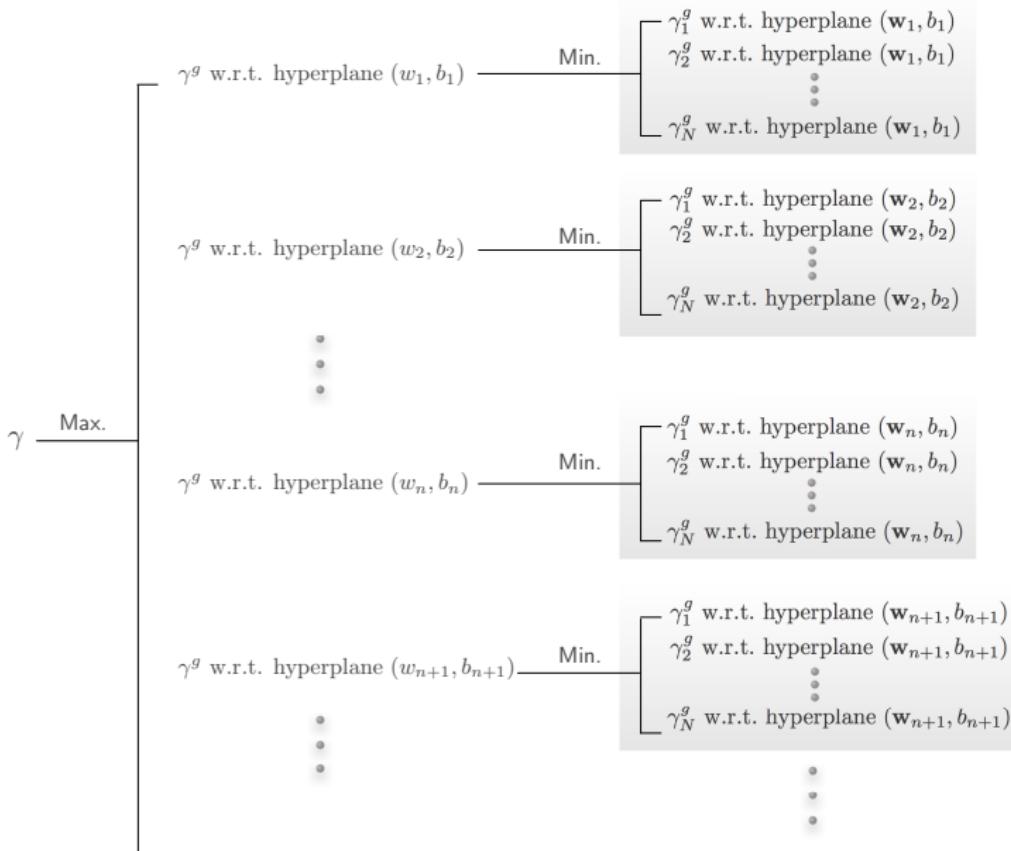
Implementation

**EE5904**  
**ME5404**  
**Part II**

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
Data  
Classification  
Margins

Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

**RL**  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation



EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

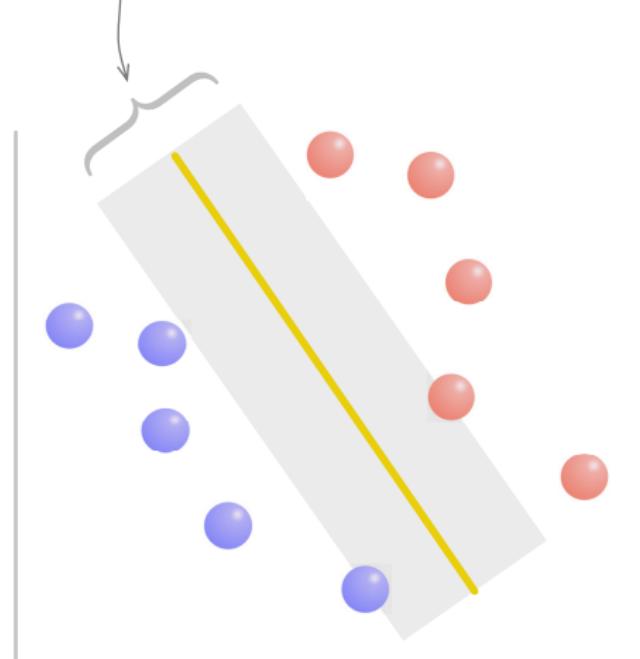
Optimal policy

Q-learning

Explore/exploit

Implementation

## Margin of separation



**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

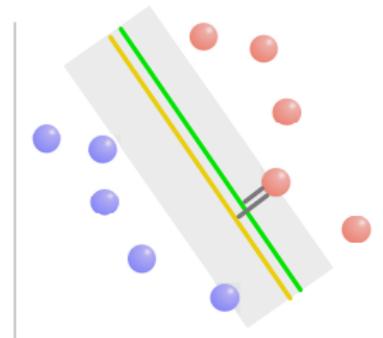
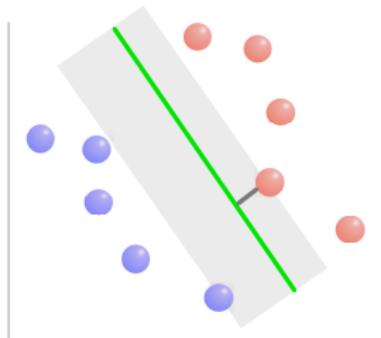
Optimal policy

Q-learning

Explore/exploit

Implementation

$$\gamma = \max_{\text{All possible hyperplanes}} \left\{ \gamma_{(\mathbf{w}_1, b_1)}^g, \gamma_{(\mathbf{w}_2, b_2)}^g, \gamma_{(\mathbf{w}_3, b_3)}^g, \dots \right\}$$



If hyperplane is optimal (with the margin as shown) but does **not** evenly divide margin of separation

Then we can always move this hyperplane to get a **larger** margin, which means the original hyperplane is not optimal

Plan  
Introduction  
Outcomes  
Math  
SVM  
Data  
Classification

**Margins**

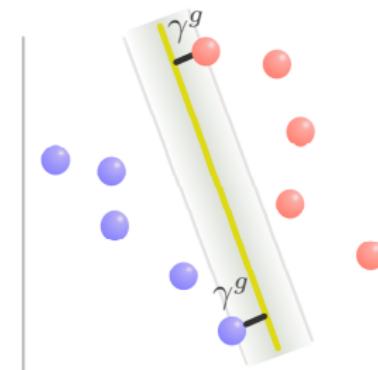
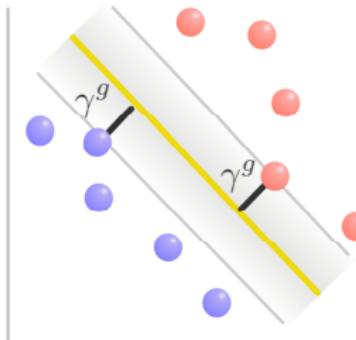
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

**RL**

Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

## Two key questions:

- Why is it important to find the margin of  $S$  ?
- How to find the margin of  $S$  ?



**Plan****Introduction**  
**Outcomes****Math****SVM**  
Data  
Classification**Margins**Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application**RL**Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

Two key questions:

- Why is it important to find the margin of  $S$  ?
- How to find the margin of  $S$  ?

Larger margin leads to lower probability of misclassification

**Theorem 4.18** (in book by Cristianini and Shawe-Taylor): Consider thresholding real-valued linear functions  $\mathcal{L}$  with unit weight vectors on an inner product space  $X$  and fix  $\gamma \in \mathbb{R}^+$ . For any probability distribution  $\mathcal{D}$  on  $X \times \{-1, 1\}$  with support in a ball of radius  $R$  around the origin, with probability  $1 - \delta$  over  $l$  random examples  $S$ , any hypothesis  $f \in \mathcal{L}$  that has margin  $m_s(f) \geq \gamma$  on  $S$  has error no more than

$$\text{err}_{\mathcal{D}}(f) \leq \varepsilon(l, \mathcal{L}, \delta, \gamma) = \frac{2}{l} \left( \frac{64R^2}{\gamma^2} \log \frac{el\gamma}{4R} \log \frac{128lR^2}{\gamma^2} + \log \frac{4}{\delta} \right)$$

provided  $l > 2/\varepsilon$  and  $64R^2/\gamma^2 < l$ .

(We will just take it as fact and will not study detailed proof in this module.)

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

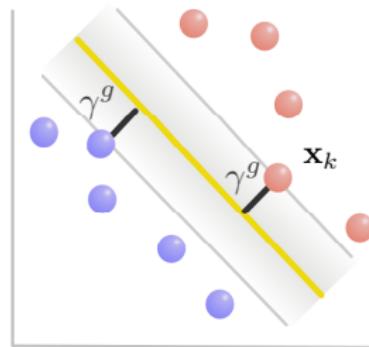
Q-learning

Explore/exploit

Implementation

## Two key questions:

- Why is it important to find the margin of  $S$  ?
- How to find the margin of  $S$  ?



Recall the relationship:  $\gamma_i^g = \frac{\gamma_i^f}{\|\mathbf{w}\|}$

For a given  $\mathbf{w}$ , the example  $k$  that yields  $\gamma_k^f = \gamma^f$  also yields  $\gamma^g = \gamma_k^g$ . So we can write

$$\gamma^g = \frac{\gamma^f}{\|\mathbf{w}\|}$$

We can **maximize  $\gamma^g$**  by **fixing  $\gamma^f$**  then **minimizing  $\|\mathbf{w}\|$**

$\gamma^g = \frac{\gamma^f}{\|\mathbf{w}\|}$ : We can maximize  $\gamma^g$  by fixing  $\gamma^f$  then minimizing  $\|\mathbf{w}\|$

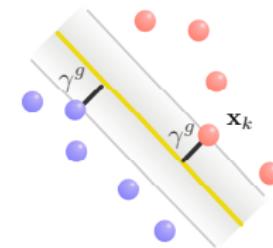
Since a hyperplane is invariant under scaling by a constant  $c$ , for any hyperplane  $(\mathbf{w}, b)$ , we can find  $c$  such that

$$\gamma^f \equiv \underbrace{\gamma_k^f = d_k (c \mathbf{w}^T \mathbf{x}_k + c b)}_{\text{functional margin of example } k} = 1$$

That is, we fix functional margin  $\gamma^f$  of training set to be 1.

Problem: Find  $c$  such that  $\gamma_k^f = 1$ , with

$$\mathbf{x}_k = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \quad d_k = +1, \quad \mathbf{w} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}, \quad b = 6$$

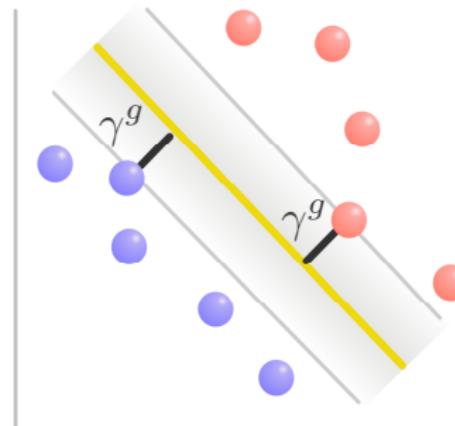


$$d_k (c \mathbf{w}^T \mathbf{x}_k + c b) = 1 \cdot c \left( \begin{bmatrix} 5 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 6 \right) = 23c = 1 \Rightarrow c = \frac{1}{23}$$

$\gamma^g = \frac{\gamma^f}{\|\mathbf{w}\|}$ : We can maximize  $\gamma^g$  by fixing  $\gamma^f$  then minimizing  $\|\mathbf{w}\|$

With  $\gamma^f$  fixed at 1, for any example  $\mathbf{x}_i$ , we have the condition

$$\gamma_i^f = \gamma_i^g \|\mathbf{w}\| = d_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$



Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

$\gamma^g = \frac{\gamma^f}{\|\mathbf{w}\|}$ : We can maximize  $\gamma^g$  by fixing  $\gamma^f$  then **minimizing  $\|\mathbf{w}\|$**

With  $\gamma^f$  fixed at 1, for any example  $\mathbf{x}_i$ , we have the condition

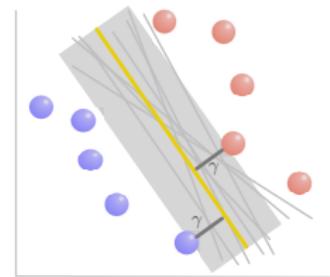
$$\gamma_i^f = \gamma_i^g \|\mathbf{w}\| = d_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

The problem of minimizing  $\|\mathbf{w}\|$  can now be expressed as a

### Constrained optimization problem

Minimizing:  $f(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 = \frac{1}{2} \mathbf{w}^T \mathbf{w}$

Subject to:  $d_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$



$$\gamma = \max \underbrace{\left\{ \gamma_{(\mathbf{w}_1, b_1)}^g, \gamma_{(\mathbf{w}_2, b_2)}^g, \dots \right\}}_{\text{All possible hyperplanes}}$$

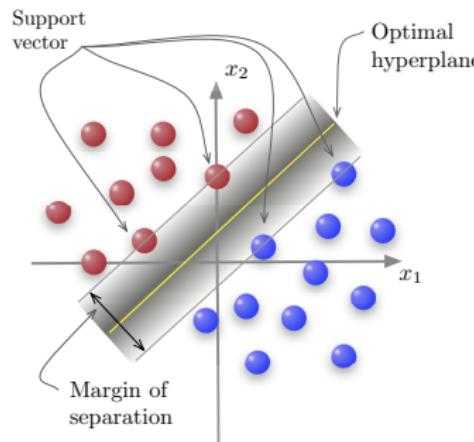
All possible hyperplanes

Solving this problem yields the optimal hyperplane  $(\mathbf{w}_o, b_o)$

EE5904  
ME5404  
Part II

For training data  $\mathbf{x}_i$

$$\begin{cases} g(\mathbf{x}_i) = \mathbf{w}_o^T \mathbf{x}_i + b_o \geq +1 & \text{for } d_i = +1 \\ g(\mathbf{x}_i) = \mathbf{w}_o^T \mathbf{x}_i + b_o \leq -1 & \text{for } d_i = -1 \end{cases}$$



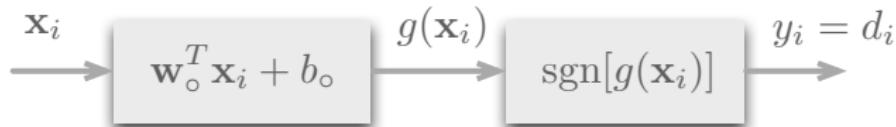
## Discriminant function

$$g(\mathbf{x}) = \mathbf{w}_o^T \mathbf{x} + b_o$$

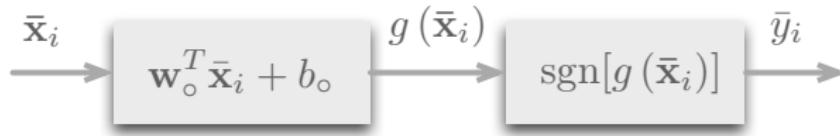
Support vector:  $\mathbf{x}_i$  that satisfies

$$g(\mathbf{x}_i) = \pm 1$$

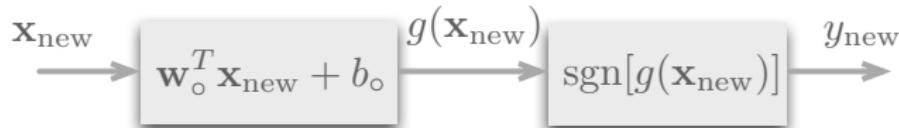
**Construction:** For a given training set  $S = \{(\mathbf{x}_1, d_1), \dots, (\mathbf{x}_N, d_N)\}$ , find optimal hyperplane  $(\mathbf{w}_o, b_o)$  such that, for all  $i \in \{1, 2, \dots, N\}$ ,



**Testing:** For a given test set  $\bar{S} = \{(\bar{\mathbf{x}}_1, \bar{d}_1), \dots, (\bar{\mathbf{x}}_{\bar{N}}, \bar{d}_{\bar{N}})\}$ , compute output  $\bar{y}_i$  of SVM (with  $\mathbf{w}_o$  and  $b_o$ ) for all  $i \in \{1, 2, \dots, \bar{N}\}$ , and compare it against the known  $\bar{d}_i$  to evaluate performance of SVM



**Application:** Given a SVM with hyperplane  $(\mathbf{w}_o, b_o)$ , classify a data point  $\mathbf{x}_{\text{new}}$  that is **not** in  $\Sigma = S \cup \bar{S}$ :



EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

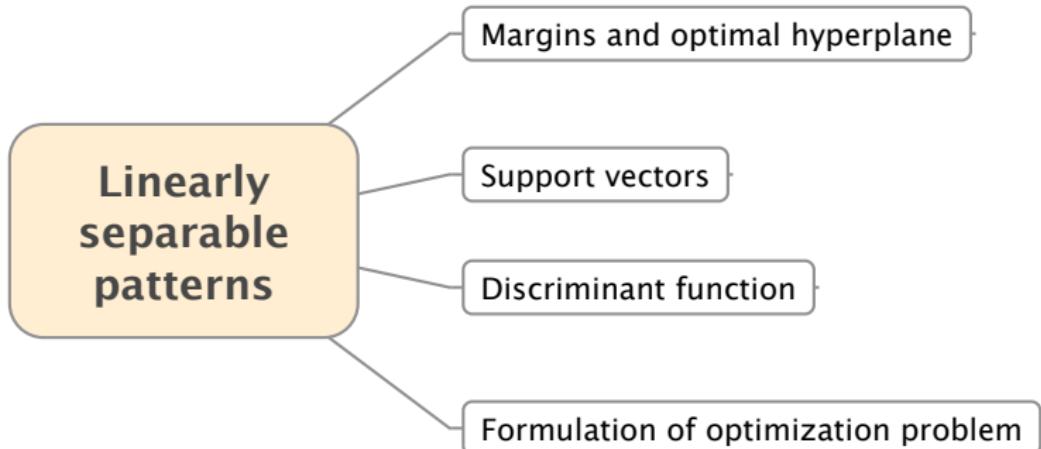
Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation



**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

**Primal**

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

## Primal problem

Given data set :  $S = \{(\mathbf{x}_i, d_i)\}, i = 1, 2, \dots, N$

Find :  $\mathbf{w}$  and  $b$

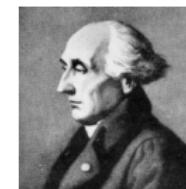
**Minimizing** :  $f(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$

Subject to :  $d_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

Well known algorithms exist for solving this problem

- R., Fletcher, *Practical Methods of Optimization*. 2nd ed., John Wiley & Sons, NY, 1987
- Arnold Neumaier, Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, Vol. 13, 271-369, 2004

Alternative formulation using method of Lagrange multipliers



Joseph-Louis Lagrange  
French mathematician  
1736–1813

## Basic class of constrained optimization problems

Minimize :  $f(\mathbf{w})$ ,  $\mathbf{w} \in \mathcal{R}^n$ ,  $f(\cdot), h(\cdot) \in \mathcal{R}$   
 Subject to :  $h_i(\mathbf{w}) = 0$ ,  $i = 1, \dots, m$

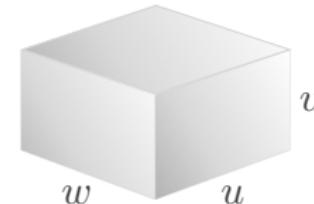
- $f$  is called the objective function and  $h_i$  the constraints
- Optimal value of  $f$  is called **value of optimization problem**
- $\mathbf{w}$  corresponding to optimal  $f$  is called **optimal solution**

If  $f$  is to be maximized instead, such a maximization problem can be expressed as a minimization problem by the transformation

$$\max_{\mathbf{w}} f(\mathbf{w}) = - \min_{\mathbf{w}} [-f(\mathbf{w})]$$

Maximize : $f(\mathbf{w})$ Subject to : $h_i(\mathbf{w}) = 0$	$\equiv$	Minimize : $-f(\mathbf{w})$ Subject to : $h_i(\mathbf{w}) = 0$
--	----------	---

**Problem:** Consider a box whose surface area is  $C$ . Formulate the optimization problem for maximizing the volume of the box.



**Solution:**

$$\begin{aligned} \text{Surface area: } & 2wu + 2uv + 2vw = C \\ \text{Volume: } & wuv \end{aligned}$$

Maximizing  $wuv$  is equivalent to minimizing  $-wuv$ .

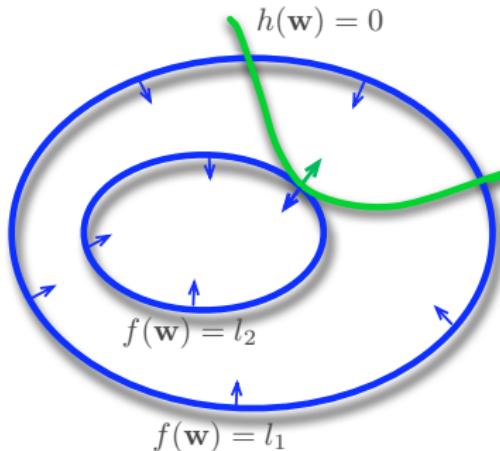
Optimization problem

$$\begin{aligned} \text{Minimize: } & f(u, v, w) = -uvw \\ \text{Subject to: } & h(u, v, w) = wu + uv + vw - \frac{C}{2} = 0 \end{aligned}$$

## Basic class of constrained optimization problems

Minimize :  $f(\mathbf{w})$

Subject to :  $h(\mathbf{w}) = 0$  (equality constraint)



Blue arrow points to direction of greatest decrease of  $f(\mathbf{w})$

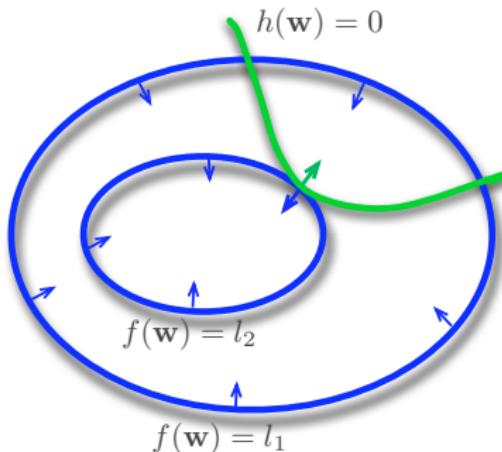
- Contours of  $f(\mathbf{w})$  correspond to fixed  $l_n$ , i.e.,  $l_1, l_2, \dots$
- $h(\mathbf{w}) = 0$  intersects many  $f$  contours. Each intersection is a solution  $\mathbf{w}$  satisfying  $h$
- Find the solution  $\mathbf{w}$  that satisfies  $h$  while gives minimum of  $f$

EE5904  
ME5404  
Part II

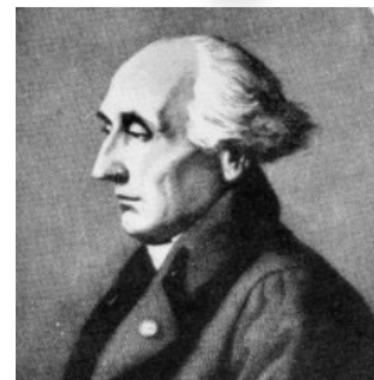
Plan  
Introduction  
Outcomes  
Math  
SVM  
Data  
Classification  
Margins  
Primal  
**Lagrange**

Dual  
Soft margin  
Kernel  
Application

RL  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation



$$\nabla f \parallel \nabla h$$



- Solution occurs at point where  $f$  and  $h$  touch tangentially
- At solution point, gradients of  $f$  and  $h$  are parallel.

**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

**Lagrange**

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

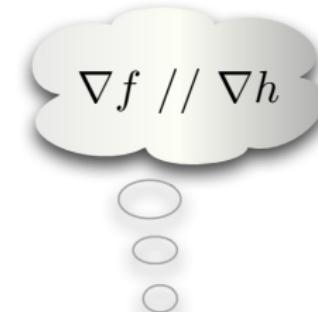
Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation



Minimize :  $f(\mathbf{w})$   
 Subject to :  $h_i(\mathbf{w}) = 0, i = 1, \dots, m$

$$\nabla f = \frac{\partial f(\mathbf{w})}{\partial \mathbf{w}}$$

$$\nabla h_i = \frac{\partial h_i(\mathbf{w})}{\partial \mathbf{w}}$$

The fact that  $\nabla f$  and  $\nabla h$  are parallel means there exist constants  $\beta_i$  such that

$$\left[ \frac{\partial f(\mathbf{w})}{\partial \mathbf{w}} \right]_{\mathbf{w}_o} + \sum_{i=1}^m \left[ \beta_i \left( \frac{\partial h_i(\mathbf{w})}{\partial \mathbf{w}} \right) \right]_{\mathbf{w}_o} = \mathbf{0}$$

where  $\mathbf{w}_o$  is an optimal solution

The constants  $\beta_i$  are called **Lagrange multipliers**

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

## Define Lagrangian function:

$$L(\mathbf{w}, \boldsymbol{\beta}) = f(\mathbf{w}) + \sum_{i=1}^m \beta_i h_i(\mathbf{w})$$

Then the fact that  $\nabla f // \nabla h$   
can be expressed compactly as

$$\frac{\partial L(\mathbf{w}, \boldsymbol{\beta})}{\partial \mathbf{w}} = \mathbf{0}$$

## Lagrange Theorem

The necessary conditions for  
the existence of an optimal  
solution  $\mathbf{w}_o$  corresponding to a  
minimum of  $f(\mathbf{w})$  subject to  
 $h_i(\mathbf{w}) = 0, i = 1, \dots, m$ , are

$$\begin{aligned}\frac{\partial L(\mathbf{w}_o, \boldsymbol{\beta}_o)}{\partial \mathbf{w}} &= \mathbf{0} \\ \frac{\partial L(\mathbf{w}_o, \boldsymbol{\beta}_o)}{\partial \beta_i} &= 0\end{aligned}$$

Theorem can be used to solve problems with equality constraints

**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

**Lagrange**

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

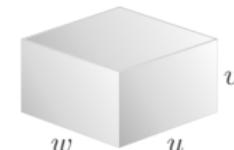
Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

Minimize:  $f(u, v, w) = -uvw$ Subject to:  $h(u, v, w) = wu + uv + vw - \frac{C}{2} = 0$ 

$$L(u, v, w) = -uvw + \beta \left( wu + uv + vw - \frac{C}{2} \right)$$

The solution is

Lagrange's Theorem yields

$$\frac{\partial L}{\partial u} = -vw + \beta(v + w) = 0$$

$$\frac{\partial L}{\partial v} = -wu + \beta(w + u) = 0$$

$$\frac{\partial L}{\partial w} = -uv + \beta(u + v) = 0$$

$$wu + uv + vw = \frac{C}{2}$$

$$u = v = w = \sqrt{\frac{C}{6}}$$

and the maximum volume is

$$\max(wuv)$$

$$= -\min(-wuv)$$

$$= -\left(-\left(\frac{C}{6}\right)^{\frac{3}{2}}\right) = \left(\frac{C}{6}\right)^{\frac{3}{2}}$$

**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

**Lagrange**

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

Lagrange Theorem can be generalized to deal with problems having both **equality and inequality** constraints

### Primal problem (Standard form)

Minimize:  $f(\mathbf{w})$

Subject to:  $q_i(\mathbf{w}) \leq 0, \quad i = 1, \dots, k$

$h_i(\mathbf{w}) = 0, \quad i = 1, \dots, m$

Constraints in the form  $q_i(\mathbf{w}) \geq 0$

can be re-written as  $q'_i(\mathbf{w}) = -q_i(\mathbf{w}) \leq 0$

**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

**Lagrange**

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

Lagrange Theorem can be generalized to deal with problems having both equality and inequality constraints

## Primal problem

Minimize:  $f(\mathbf{w})$

Subject to:  $q_i(\mathbf{w}) \leq 0, \quad i = 1, \dots, k$

$h_i(\mathbf{w}) = 0, \quad i = 1, \dots, m$

## Generalized Lagrangian function

$$L(\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\mathbf{w}) + \sum_{i=1}^k \alpha_i q_i(\mathbf{w}) + \sum_{i=1}^m \beta_i h_i(\mathbf{w})$$

where  $\alpha_i$  and  $\beta_i$  are the Lagrange multipliers

Solution characterized by Karush-Kuhn-Tucker conditions

## Kuhn-Tucker Theorem

Consider the primal problem with the Lagrangian as defined earlier. The sufficient and necessary condition for a point  $\mathbf{w}_o$  to be an optimal solution is the existence of  $\alpha_o$  and  $\beta_o$  such that

$$\frac{\partial L(\mathbf{w}_o, \alpha_o, \beta_o)}{\partial \mathbf{w}} = \mathbf{0}$$

$$\frac{\partial L(\mathbf{w}_o, \alpha_o, \beta_o)}{\partial \beta_i} = 0$$

$$\alpha_{o,i} q_i(\mathbf{w}_o) = 0, \quad i = 1, \dots, k$$

$$q_i(\mathbf{w}_o) \leq 0, \quad i = 1, \dots, k$$

$$\alpha_{o,i} \geq 0, \quad i = 1, \dots, k$$

These equations are the KKT conditions

The KKT conditions enable us to transform the primal problem into an alternative, simpler form called the **dual problem**

### Primal problem

$$\text{Minimize: } f(\mathbf{w})$$

$$\text{Subject to: } q_i(\mathbf{w}) \leq 0$$

$$h_i(\mathbf{w}) = 0$$

### Dual problem

$$\text{Maximize: } L(\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$\text{Subject to: } \frac{\partial L(\mathbf{w}_o, \boldsymbol{\alpha}_o, \boldsymbol{\beta}_o)}{\partial \mathbf{w}} = 0$$

$$\boldsymbol{\alpha} \geq 0$$

The primal and dual problems are equivalent because they have the same *value of the optimization problem*, i.e.,

$$f(\mathbf{w}_o) = L(\mathbf{w}_o, \boldsymbol{\alpha}_o, \boldsymbol{\beta}_o)$$

Proof in *Practical Methods of Optimization* by Fletcher, Wiley, 1987

## Finding optimal hyperplane (primal problem)

Given data set :  $S = \{(\mathbf{x}_i, d_i)\}$

Find :  $\mathbf{w}$  and  $b$

Minimizing :  $f(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$

Subject to :  $d_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

### In primal problem

Known parameters:  $\mathbf{x}_i, d_i$

Unknown variables:  $\mathbf{w}, b$

Apply KKT conditions to reduce unknowns to just  $\alpha_i$  to form **dual problem**

Write constraints  $d_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$  in standard form as

$$q_i = -d_i (\mathbf{w}^T \mathbf{x}_i + b) + 1 \leq 0$$

Now the Lagrangian function is

$$\begin{aligned} L(\mathbf{w}, b, \boldsymbol{\alpha}) &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i (d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1) \\ &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i \end{aligned}$$

The KKT conditions are

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{0}$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = 0$$

$$d_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

$$\alpha_i (d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1) = 0$$

$$\alpha_i \geq 0$$

Consider the first KKT condition:  $\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{0}$

$$\begin{aligned}
 \frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} &= \frac{\partial}{\partial \mathbf{w}} \left( \frac{\mathbf{w}^T \mathbf{w}}{2} - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i \right) \\
 &= \frac{1}{2} \frac{\partial (\mathbf{w}^T \mathbf{w})}{\partial \mathbf{w}} - \sum_{i=1}^N \alpha_i d_i \left( \frac{\partial (\mathbf{w}^T \mathbf{x}_i)}{\partial \mathbf{w}} \right) \\
 &= \frac{2\mathbf{w}}{2} - \sum_{i=1}^N \alpha_i d_i \left( \frac{\partial (\mathbf{x}_i^T \mathbf{w})}{\partial \mathbf{w}} \right) \quad (\text{Since } \mathbf{u}^T \mathbf{v} = \mathbf{v}^T \mathbf{u}) \\
 &= \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i \quad \left( \text{Since } \frac{\partial (\mathbf{u}^T \mathbf{v})}{\partial \mathbf{v}} = \mathbf{u} \right) \\
 &= \mathbf{0}
 \end{aligned}$$

Therefore

$$\mathbf{w} = \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i$$

▶ View

**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

Consider the second KKT condition:  $\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = 0$

$$\begin{aligned} & \frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} \\ = & \frac{\partial}{\partial b} \left( \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i \right) \\ = & \sum_{i=1}^N \alpha_i d_i = 0 \end{aligned}$$

Therefore

$$\sum_{i=1}^N \alpha_i d_i = 0$$

EE5904  
ME5404  
Part II

Plan  
Introduction  
Outcomes  
Math

SVM  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual

Soft margin  
Kernel  
Application  
RL  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

Thus from KKT conditions we have

$$\mathbf{w} = \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i \quad \text{and} \quad \sum_{i=1}^N \alpha_i d_i = 0$$

Recall that

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i$$

So

$$\frac{1}{2} \mathbf{w}^T \mathbf{w} = \frac{1}{2} \left[ \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i^T \right] \left[ \sum_{j=1}^N \alpha_j d_j \mathbf{x}_j \right] = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i = \sum_{i=1}^N \alpha_i d_i \left[ \sum_{j=1}^N \alpha_j d_j \mathbf{x}_j^T \right] \mathbf{x}_i = \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\frac{1}{2} \mathbf{w}^T \mathbf{w} = \frac{1}{2} \left[ \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i^T \right] \left[ \sum_{j=1}^N \alpha_j d_j \mathbf{x}_j \right] = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i = \sum_{i=1}^N \alpha_i d_i \left[ \sum_{j=1}^N \alpha_j d_j \mathbf{x}_j^T \right] \mathbf{x}_i = \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\sum_{i=1}^N \alpha_i d_i = 0$$

$$\begin{aligned} L(\mathbf{w}, b, \boldsymbol{\alpha}) &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i (d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1) \\ &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i \\ &= \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j \\ &\equiv Q(\boldsymbol{\alpha}) \end{aligned}$$

**Plan****Introduction****Outcomes****Math****SVM**  
**Data****Classification****Margins****Primal****Lagrange****Dual****Soft margin****Kernel****Application****RL****Markov****Q-function****Bellman****Optimality****Value iteration****Convergence****Optimal policy****Q-learning****Explore/exploit****Implementation**

## Finding optimal hyperplane (**dual problem**)

Given :  $S = \{(\mathbf{x}_i, d_i)\}$

Find : Lagrange multipliers  $\{\alpha_i\}$

Maximizing : 
$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j$$

Subject to : (1) 
$$\sum_{i=1}^N \alpha_i d_i = 0$$

(2)  $\alpha_i \geq 0$

- $\alpha_i$  are the only unknowns
- Algorithms and software available for solving this problem
- Optimal solution denoted by  $\alpha_{o,i}$
- $\mathbf{x}_i^T \mathbf{x}_j$  is called a **linear kernel** (more about kernels later)

Plan  
Introduction  
Outcomes

Math  
SVM  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual

Soft margin  
Kernel  
Application

RL  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

Suppose that a support vector machine is to be constructed using the training set below:

$i$	$\mathbf{x}_i$	$d_i$
1	$[ 1 \ -1 ]^T$	-1
2	$[ 2 \ 1 ]^T$	1
3	$[ 3 \ 1 ]^T$	1

- (i) Determine the Lagrangian function  $L$
- (ii) Show the explicit form of the dual problem

**Plan****Introduction**  
**Outcomes****Math****SVM**  
Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i (d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1)$$

From the given data, we have  $N = 3$ , and  $\mathbf{w} = [w_1, w_2]^T$ . Now

$$\begin{aligned} L(\mathbf{w}, b, \boldsymbol{\alpha}) &= \frac{1}{2} \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \\ &\quad - \alpha_1(-1) \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \alpha_1(-1)b + \alpha_1 \\ &\quad - \alpha_2(1) \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} - \alpha_2(1)b + \alpha_2 \\ &\quad - \alpha_3(1) \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} - \alpha_3(1)b + \alpha_3 \\ &= \frac{1}{2} (w_1^2 + w_2^2) + \alpha_1 (w_1 - w_2 + 1) - \alpha_2 (2w_1 + w_2 - 1) \\ &\quad - \alpha_3 (3w_1 + w_2 - 1) + (\alpha_1 - \alpha_2 - \alpha_3) b \end{aligned}$$

$$\begin{aligned}
 Q(\alpha) &= \sum_{i=1}^3 \alpha_i - \frac{1}{2} \sum_{i=1}^3 \sum_{j=1}^3 \alpha_i d_i \alpha_j d_j \mathbf{x}_i^T \mathbf{x}_j \\
 &= \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} \alpha_1 d_1 \alpha_1 d_1 \mathbf{x}_1^T \mathbf{x}_1 - \frac{1}{2} \alpha_1 d_1 \alpha_2 d_2 \mathbf{x}_1^T \mathbf{x}_2 - \frac{1}{2} \alpha_1 d_1 \alpha_3 d_3 \mathbf{x}_1^T \mathbf{x}_3 \\
 &\quad - \frac{1}{2} \alpha_2 d_2 \alpha_1 d_1 \mathbf{x}_2^T \mathbf{x}_1 - \frac{1}{2} \alpha_2 d_2 \alpha_2 d_2 \mathbf{x}_2^T \mathbf{x}_2 - \frac{1}{2} \alpha_2 d_2 \alpha_3 d_3 \mathbf{x}_2^T \mathbf{x}_3 \\
 &\quad - \frac{1}{2} \alpha_3 d_3 \alpha_1 d_1 \mathbf{x}_3^T \mathbf{x}_1 - \frac{1}{2} \alpha_3 d_3 \alpha_2 d_2 \mathbf{x}_3^T \mathbf{x}_2 - \frac{1}{2} \alpha_3 d_3 \alpha_3 d_3 \mathbf{x}_3^T \mathbf{x}_3 \\
 &= \alpha_1 + \alpha_2 + \alpha_3 \\
 &\quad - \frac{1}{2} \alpha_1(-1)\alpha_1(-1) \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \frac{1}{2} \alpha_1(-1)\alpha_2(1) \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \\
 &\quad - \frac{1}{2} \alpha_1(-1)\alpha_3(1) \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} - \frac{1}{2} \alpha_2(1)\alpha_1(-1) \begin{bmatrix} 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\
 &\quad - \frac{1}{2} \alpha_2(1)\alpha_2(1) \begin{bmatrix} 2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} - \frac{1}{2} \alpha_2(1)\alpha_3(1) \begin{bmatrix} 2 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} \\
 &\quad - \frac{1}{2} \alpha_3(1)\alpha_1(-1) \begin{bmatrix} 3 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \frac{1}{2} \alpha_3(1)\alpha_2(1) \begin{bmatrix} 3 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \\
 &\quad - \frac{1}{2} \alpha_3(1)\alpha_3(1) \begin{bmatrix} 3 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} \\
 &= \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} (2\alpha_1^2 + 5\alpha_2^2 + 10\alpha_3^2 - 2\alpha_1\alpha_2 - 4\alpha_1\alpha_3 + 14\alpha_2\alpha_3)
 \end{aligned}$$

## Explicit form of dual problem

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

Maximize

$$Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3$$

$$-\frac{1}{2} (2\alpha_1^2 + 5\alpha_2^2 + 10\alpha_3^2 - 2\alpha_1\alpha_2 - 4\alpha_1\alpha_3 + 14\alpha_2\alpha_3)$$

Subject to

$$\sum_{i=1}^3 \alpha_i d_i = \alpha_1(-1) + \alpha_2(1) + \alpha_3(1) = -\alpha_1 + \alpha_2 + \alpha_3 = 0$$

$$\alpha_1 \geq 0$$

$$\alpha_2 \geq 0$$

$$\alpha_3 \geq 0$$

Recall that for support vector  $\mathbf{x}_i$ :  $g(\mathbf{x}_i) = \mathbf{w}_o^T \mathbf{x}_i + b_o = \pm 1$

Constraint on  $\mathbf{w}_o$  and  $b_o$ :  $d_i (\mathbf{w}_o^T \mathbf{x}_i + b_o) \geq 1$

From KKT condition:  $\alpha_{o,i} (d_i (\mathbf{w}_o^T \mathbf{x}_i + b_o) - 1) = 0$

- For data point  $\mathbf{x}_i$  that is not a support vector

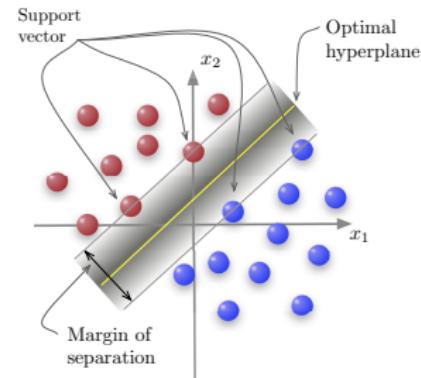
$$d_i (\mathbf{w}_o^T \mathbf{x}_i + b_o) > 1 \Rightarrow \alpha_{o,i} = 0$$

- For data point  $\mathbf{x}_i$  that is a support vector

$$d_i (\mathbf{w}_o^T \mathbf{x}_i + b_o) = 1 \Rightarrow \alpha_{o,i} \neq 0$$

Furthermore, for a support vector  $\mathbf{x}_i$

$$d_i (\mathbf{w}_o^T \mathbf{x}_i + b_o) = 1 \Rightarrow b_o = \frac{1}{d_i} - \mathbf{w}_o^T \mathbf{x}_i$$



	Primal	Dual
Find :	$\mathbf{w}, b$	$\alpha_i$
Minimizing :	$f(\mathbf{w})$	$\mathbf{w}$
Maximizing :	—	$Q(\boldsymbol{\alpha})$
Subject to :	$d_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$	$\sum_{i=1}^N \alpha_i d_i = 0$
		$\alpha_i \geq 0$

$$f(\mathbf{w}, \xi) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \quad Q(\boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j$$

After  $\alpha_{o,i}$  is obtained, we can calculate  $\mathbf{w}_o$  and  $b_o$  as follows:

$$\mathbf{w}_o = \sum_{i=1}^N \alpha_{o,i} d_i \mathbf{x}_i, \quad b_o = \frac{1}{d^{(s)}} - \mathbf{w}_o^T \mathbf{x}^{(s)}$$

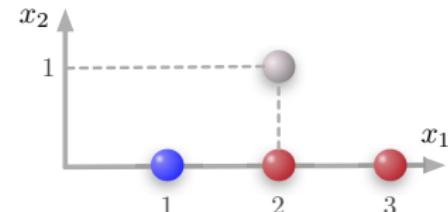
where  $\mathbf{x}^{(s)}$  is a support vector with label  $d^{(s)}$

**Problem:** Consider the training set and a support vector machine having the Lagrange multiplier values as listed in the table below:

$i$	$\alpha_{o,i}$	$\mathbf{x}_i$	$d_i$
1	2	$[1 \ 0]^T$	-1
2	2	$[2 \ 0]^T$	+1
3	0	$[3 \ 0]^T$	+1

Compute the output of this support vector machine when the input is

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix}$$



Note that  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are support vectors but  $\mathbf{x}_3$  is not. (Why?)

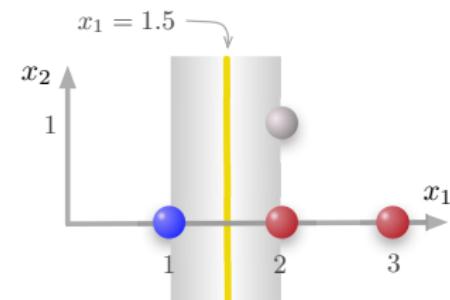
Solution: Given input  $x$ , output of SVM is

$$\text{sgn}[g(x)]$$

where

$$g(x) = w_o^T x + b_o$$

is the discriminant function. Now



$$\begin{aligned} w_o &= \sum_{i=1}^3 \alpha_{o,i} d_i x_i = \alpha_{o,1} d_1 x_1 + \alpha_{o,2} d_2 x_2 + \alpha_{o,3} d_3 x_3 \\ &= (2)(-1) \begin{bmatrix} 1 \\ 0 \end{bmatrix} + (2)(1) \begin{bmatrix} 2 \\ 0 \end{bmatrix} + (0)(1) \begin{bmatrix} 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \end{aligned}$$

To calculate  $b_o$ , we pick the support vector  $x_1^{(s)}$  which has  $d_1 = -1$ . So

$$b_o = \frac{1}{d_1} - w_o^T x_1^{(s)} = \frac{1}{-1} - \begin{bmatrix} 2 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = -3$$

Thus, for the input  $x = [2 \ 1]^T$ , the output of the support vector machine is

$$\text{sgn}[g(x)] = \text{sgn} [w_o^T x + b_o] = \text{sgn} \left[ \begin{bmatrix} 2 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} - 3 \right] = \text{sgn}[4 - 3] = 1$$

EE5904  
ME5404  
Part II

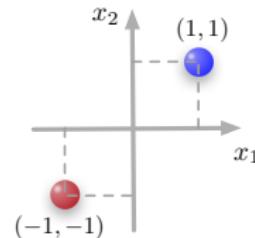
**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual

Soft margin  
Kernel  
Application

**RL**  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

Given the training set below, find  $\mathbf{w}_o$  and  $b_o$ .

$$\begin{aligned}\mathbf{x}_1 &= \begin{bmatrix} -1 \\ -1 \end{bmatrix} & \mathbf{x}_2 &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ d_1 &= -1 & d_2 &= +1\end{aligned}$$



$$\begin{aligned}Q(\alpha) &= \sum_{i=1}^2 \alpha_i - \frac{1}{2} \left( \sum_{i=1}^2 \sum_{j=1}^2 \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j \right) \\ &= \alpha_1 + \alpha_2 - \frac{1}{2} \left( \alpha_1 \alpha_1 (-1)(-1) [-1 \ -1] \begin{bmatrix} -1 \\ -1 \end{bmatrix} + \right. \\ &\quad \left. \alpha_1 \alpha_2 (-1)(1) [-1 \ -1] \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \dots \right) \\ &= \alpha_1 + \alpha_2 - \frac{1}{2} (2\alpha_1^2 + 4\alpha_1\alpha_2 + 2\alpha_2^2) \\ &= \alpha_1 + \alpha_2 - (\alpha_1 + \alpha_2)^2\end{aligned}$$

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

## Constrained optimization problem

$$\text{Maximizing : } Q(\alpha) = \alpha_1 + \alpha_2 - (\alpha_1 + \alpha_2)^2$$

$$\text{Subject to : } \sum_{i=1}^2 \alpha_i d_i = -\alpha_1 + \alpha_2 = 0$$

$$\alpha_1 \geq 0, \alpha_2 \geq 0$$

### Solution from *Mathematica* with *Global Optimization* Package

```
In[1]=<<G052A`          (* Load Global Optimization package *)
In[2]=GlobalSearch[      (* Algorithm that finds minimum *)
-(a1+a2-(a1+*a2)^2),
{-a1,-a2},              (* a1 >= 0, a2 >= 0 *)
{-a1+a2},              (* -a1 + a2 = 0 *)
{{a1,0,1},{a2,0,1}},    (* Range of variable *)
0.00001,                (* Tolerance *)
Starts->1]              (* Number of starting points for search *)
Out[2]={{a1->0.25,a2->0.25},-0.25}  (* Solution *)
```

(\*...\*) indicates comment

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

$$\text{Maximizing : } Q(\boldsymbol{\alpha}) = \alpha_1 + \alpha_2 - (\alpha_1 + \alpha_2)^2$$

$$\text{Subject to : } -\alpha_1 + \alpha_2 = 0$$

$$\alpha_1 \geq 0, \alpha_2 \geq 0$$

For optimal solution, we have

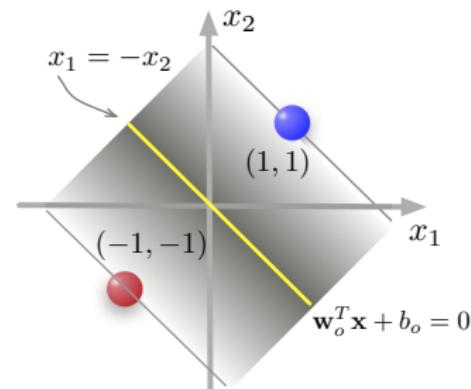
$$\frac{\partial Q(\boldsymbol{\alpha})}{\partial \alpha_1} = \frac{\partial Q(\boldsymbol{\alpha})}{\partial \alpha_2} = 0$$

$$\left. \begin{array}{l} \frac{\partial Q(\boldsymbol{\alpha})}{\partial \alpha_1} = 1 - 2(\alpha_1 + \alpha_2) = 0 \\ \frac{\partial Q(\boldsymbol{\alpha})}{\partial \alpha_2} = 1 - 2(\alpha_1 + \alpha_2) = 0 \end{array} \right\} \quad \alpha_1 + \alpha_2 = \frac{1}{2} \quad [1]$$

$$\left. \begin{array}{l} \sum_{i=1}^2 \alpha_i d_i = 0 \end{array} \right\} \quad -\alpha_1 + \alpha_2 = 0 \quad [2]$$

Solving 1 and 2 yields

$$\alpha_{o,1} = \alpha_{o,2} = \frac{1}{4}$$



$$\begin{aligned}\mathbf{w}_o &= \sum_{i=1}^N \alpha_{o,i} d_i \mathbf{x}_i = \alpha_{o,1} d_1 \mathbf{x}_1 + \alpha_{o,2} d_2 \mathbf{x}_2 \\ &= \frac{1}{4}(-1) \begin{bmatrix} -1 \\ -1 \end{bmatrix} + \frac{1}{4}(1) \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}\end{aligned}$$

To find  $b_o$ , we can take support vector  $\mathbf{x}_2$  with  $d_2 = 1$ . So

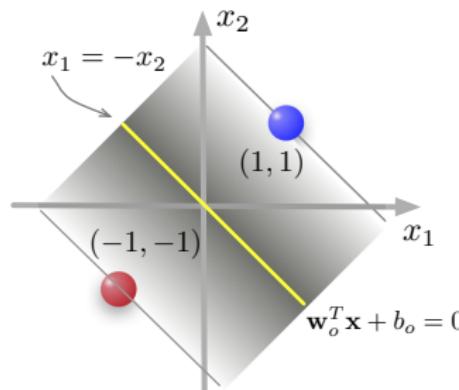
$$b_o = \frac{1}{d^{(s)}} - \mathbf{w}_o^T \mathbf{x}^{(s)} = \frac{1}{d_2} - \mathbf{w}_o^T \mathbf{x}_2 = 1 - [0.5 \ 0.5] \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 0$$

EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application  
**RL**  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

Classify a new data point  $\mathbf{x}_{\text{new}}$  as  $+1$  or  $-1$  based on

$$\begin{aligned}d_{\text{new}} &= \operatorname{sgn} [g(\mathbf{x}_{\text{new}})] \\&= \operatorname{sgn} [\mathbf{w}_o^T \mathbf{x}_{\text{new}} + b_o]\end{aligned}$$



Example: Given a SVM with

$$\mathbf{w}_o = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, b_o = 0$$

To classify  $\mathbf{x}_{\text{new}} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$

$$\begin{aligned}d_{\text{new}} &= \operatorname{sgn} [g(\mathbf{x}_{\text{new}})] \\&= \operatorname{sgn} [\mathbf{w}_o^T \mathbf{x}_{\text{new}} + b_o] \\&= \operatorname{sgn} \left[ [0.5 \ 0.5] \begin{bmatrix} 2 \\ 2 \end{bmatrix} \right] \\&= +1\end{aligned}$$

EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

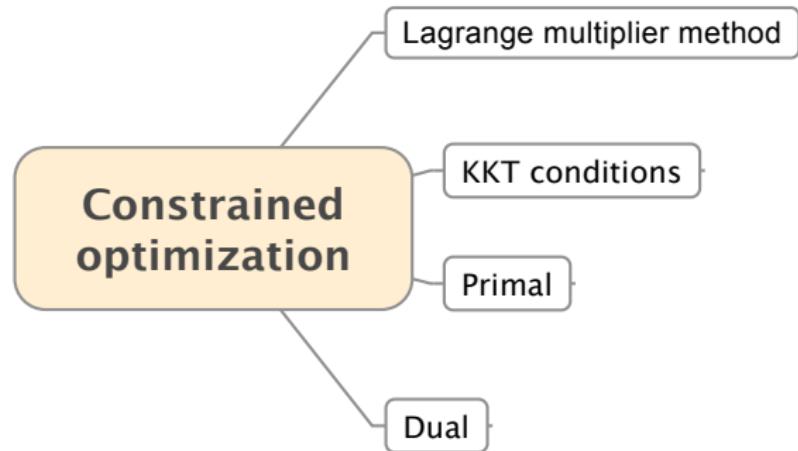
Convergence

Optimal policy

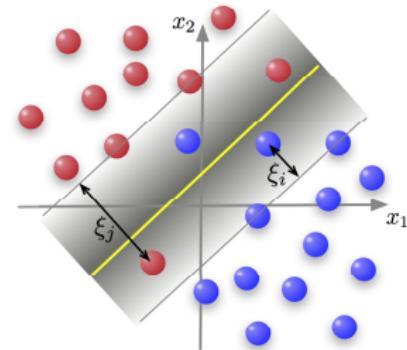
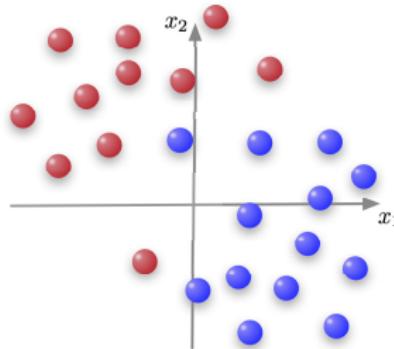
Q-learning

Explore/exploit

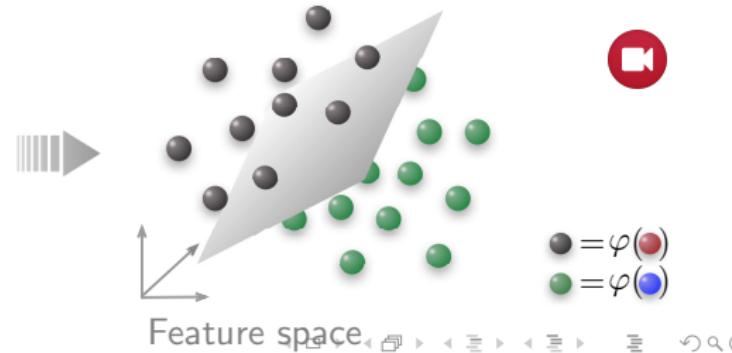
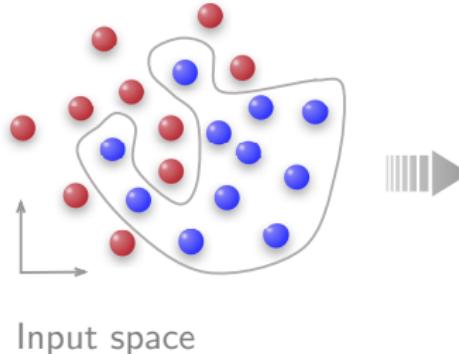
Implementation

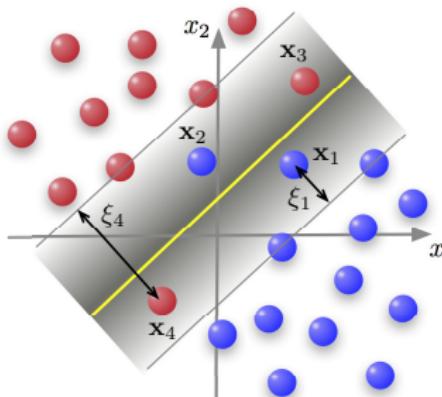


## 1. Find optimal hyperplane to minimize classification error



## 2. Transform data into higher dimension space for separation



**Plan****Introduction****Outcomes****Math****SVM****Data****Classification****Margins****Primal****Lagrange****Dual****Soft margin****Kernel****Application****RL****Markov****Q-function****Bellman****Optimality****Value iteration****Convergence****Optimal policy****Q-learning****Explore/exploit****Implementation****Nonnegative slack variables**

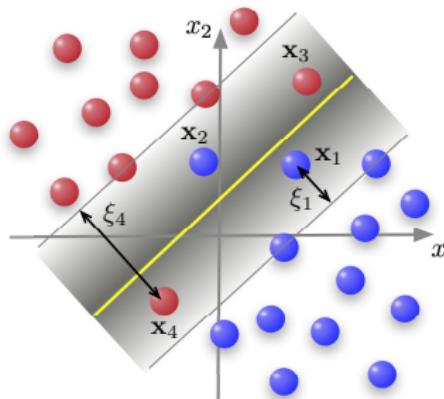
$$\xi_i, \quad i = 1, \dots, N$$

$\xi_i = 0$  Data point not in region of separation

$0 \leq \xi_i \leq 1$  Data point in region of separation and on **correct** side of hyperplane (e.g.,  $\xi_1$  and  $\xi_3$ )

$\xi_i > 1$  Data point in region of separation but on **wrong** side of hyperplane (e.g.,  $\xi_2$  and  $\xi_4$ )

Back



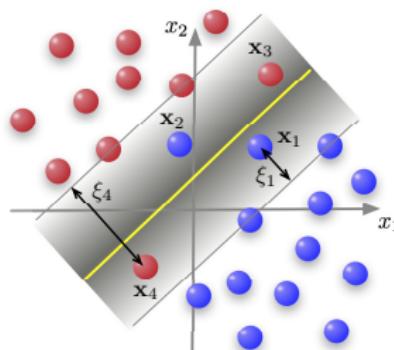
Optimal hyperplane must also minimize error penalty

$$\sum_{i=1}^N \xi_i$$

New function to be minimized

$$f(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \xi_i$$

- Value of  $C > 0$  reflects cost of violating constraints
  - A large  $C$  generally leads to smaller margin but also fewer misclassification of training data
  - A small  $C$  generally leads to larger margin but more misclassification of training data
- As a design parameter, value of  $C$  is set by user



For an example  $\mathbf{x}_i$ , we have

$$d_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$$

If  $\xi_i = 0$ , then constraint is the same as that in basic formulation (which is also known as **hard margin** classification problem.)

## Optimal hyperplane with soft margin: Primal problem

Minimize:  $\frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \xi_i$

Subject to:  $d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i \geq 0$   
 $\xi_i \geq 0$

$\xi_i$  is a variable to be optimized

**Plan****Introduction****Outcomes****Math****SVM****Data****Classification****Margins****Primal****Lagrange****Dual****Soft margin****Kernel****Application****RL****Markov****Q-function****Bellman****Optimality****Value iteration****Convergence****Optimal policy****Q-learning****Explore/exploit****Implementation**

Let  $\alpha_i$  and  $\beta_i$  be the Lagrange multipliers. Then

$$\begin{aligned}
 & L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
 = & \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i \left( d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i \right) - \sum_{i=1}^N \beta_i \xi_i \\
 = & \frac{\mathbf{w}^T \mathbf{w}}{2} + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \alpha_i \xi_i - \sum_{i=1}^N \beta_i \xi_i
 \end{aligned}$$

The KKT conditions are

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i = \mathbf{0} \quad d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i \geq 0$$

$$\frac{\partial L}{\partial b} = - \sum_{i=1}^N \alpha_i d_i = 0 \quad \alpha_i (d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i) = 0$$

$$\beta_i \xi_i = 0$$

$$\alpha_i \geq 0$$

$$\beta_i \geq 0$$

Back

**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

Since  $\mathbf{w} = \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i$ . We have

$$\mathbf{w}^T \mathbf{w} = \sum_{i=1}^N \sum_{j=1}^N \alpha_i d_i \alpha_j d_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i = \sum_{i=1}^N \sum_{j=1}^N \alpha_i d_i \alpha_j d_j \mathbf{x}_i^T \mathbf{x}_j$$

Moreover, from the KKT conditions we also have

$$C = \alpha_i + \beta_i$$

Hence,

$$L(\mathbf{w}, b, \xi, \alpha, \beta)$$

$$\begin{aligned} &= \frac{\mathbf{w}^T \mathbf{w}}{2} + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \alpha_i \xi_i - \sum_{i=1}^N \beta_i \xi_i \\ &= -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i d_i \alpha_j d_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^N \underbrace{(\alpha_i + \beta_i)}_C \xi_i + \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \alpha_i \xi_i - \sum_{i=1}^N \beta_i \xi_i \\ &= -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i d_i \alpha_j d_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^N \alpha_i \end{aligned}$$

## Dual problem (with soft margin)

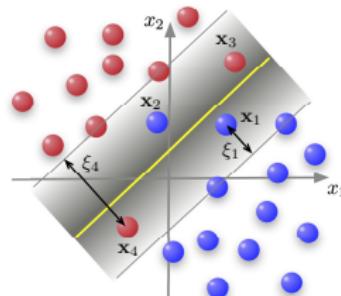
Find :  $\alpha_i$

Maximize : 
$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j$$

Subject to : 
$$\sum_{i=1}^N \alpha_i d_i = 0 \text{ and } 0 \leq \alpha_i \leq C$$

Note that  $Q(\boldsymbol{\alpha})$  is same as that for the dual problem without soft margin. The effect of the error penalty  $\sum_{i=1}^N \xi_i$  on the optimization problem is to set an upper bound for the Lagrange multiplier  $\alpha_i$ .

This can be seen by the fact that, from the KKT conditions,  
 $\alpha_i = C - \beta_i$  with  $\beta_i \geq 0$ ; that is,  $0 \leq \alpha_i \leq C$ .



From KKT conditions:

$$\begin{aligned} d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i &\geq 0 \\ \beta_i \xi_i &= 0 \\ \alpha_i + \beta_i &= C \end{aligned}$$

If the example  $\mathbf{x}_i$  is a support vector (i.e., with  $0 < \alpha_i \leq C$ ), then

$$d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i = 0 \Rightarrow d_i (\mathbf{w}^T \mathbf{x}_i + b) = 1 - \xi_i$$

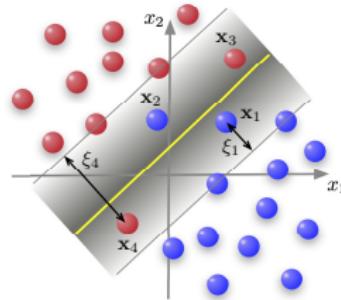
Case 1: Support vector  $\mathbf{x}_i$  with  $\alpha_i < C$

$$\beta_i = C - \alpha_i > 0$$

- To satisfy  $\beta_i \xi_i = 0$ , we must have  $\xi_i = 0$
- Support vector is located **on the border** of margin of separation

Case 2: Support vector  $\mathbf{x}_i$  with  $\alpha_i = C$

(next slide)



From KKT conditions:

$$\begin{aligned} d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i &\geq 0 \\ \beta_i \xi_i &= 0 \\ \alpha_i + \beta_i &= C \end{aligned}$$

If the example  $\mathbf{x}_i$  is a support vector (i.e., with  $0 < \alpha_i \leq C$ ), then

$$d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i = 0 \Rightarrow d_i (\mathbf{w}^T \mathbf{x}_i + b) = 1 - \xi_i$$

Case 2: Support vector with  $\alpha_i = C$

$$\beta_i = C - \alpha_i = 0$$

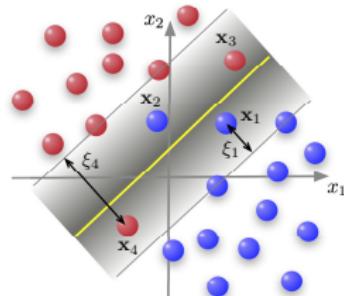
- Condition  $\beta_i \xi_i = 0$  is still satisfied when  $\xi_i \neq 0$
- Support vector is located **inside** margin of separation.

View

**Plan****Introduction****Outcomes****Math****SVM****Data****Classification****Margins****Primal****Lagrange****Dual****Soft margin****Kernel****Application****RL****Markov****Q-function****Bellman****Optimality****Value iteration****Convergence****Optimal policy****Q-learning****Explore/exploit****Implementation**

If example  $\mathbf{x}_i$  is a support vector (i.e., with  $0 < \alpha_i \leq C$ ), then

$$d_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i = 0 \Rightarrow d_i(\mathbf{w}^T \mathbf{x}_i + b) = 1 - \xi_i$$



**Case 2: Support vector with  $\alpha_i = C$**

- $\xi_i \neq 0$
- SV  $\mathbf{x}_i$  is **inside** margin of separation

SV  $\mathbf{x}_i$  is classified correctly if

$$\operatorname{sgn}[d_i(\mathbf{w}^T \mathbf{x}_i + b)] \equiv \operatorname{sgn}[1 - \xi_i] = 1$$

- If  $\xi_i \leq 1$ ,  $\mathbf{x}_i$  is classified **correctly**; e.g.,  $\mathbf{x}_1$  and  $\mathbf{x}_3$  in figure
- If  $\xi_i > 1$ ,  $\mathbf{x}_i$  is classified **incorrectly**; e.g.,  $\mathbf{x}_2$  and  $\mathbf{x}_4$  in figure

	Primal	Dual
Find :	$w, b$	$\alpha_i$
Minimizing :	$f(w, \xi)$	$\ w\ $
Maximizing :	—	$Q(\alpha)$
Subject to :	$d_i (w^T x_i + b) \geq 1 - \xi_i$ $\xi_i \geq 0$	$\sum_{i=1}^N \alpha_i d_i = 0$ $0 \leq \alpha_i \leq C$

$$f(w, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^N \xi_i \quad Q(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j x_i^T x_j$$

After  $\alpha_{o,i}$  is obtained, we can calculate  $w_o$  as follows:

$$w_o = \sum_{i=1}^N \alpha_{o,i} d_i x_i$$

After  $\mathbf{w}_o$  is obtained, we can calculate  $b_o$  as follows:

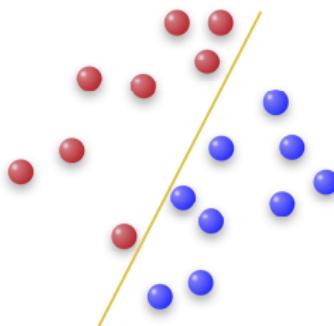
- ① For each example  $\mathbf{x}_i$  with  $0 < \alpha_i \leq C$ ,

$$b_{o,i} = \frac{1}{d_i} - \mathbf{w}_o^T \mathbf{x}_i$$

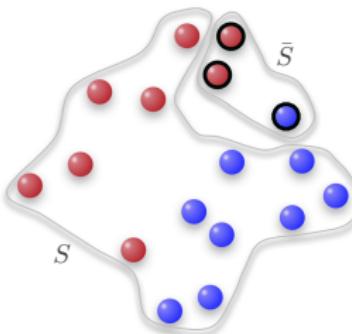
- ② Take  $b_o$  as the average of all such  $b_{o,i}$

$$b_o = \frac{\sum_{i=1}^m b_{o,i}}{m}$$

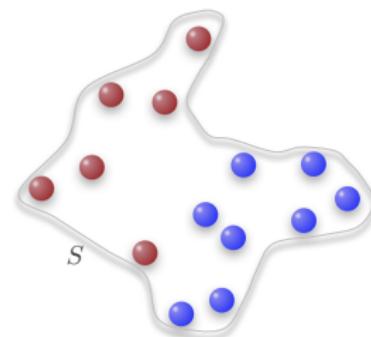
where  $m$  is the total number of  $\mathbf{x}_i$  with  $0 < \alpha_i \leq C$ .



Data set  $\Sigma$  (assumed separable here for simple illustration)



Divide  $\Sigma$  into training set  $S$  and test set  $\bar{S}$

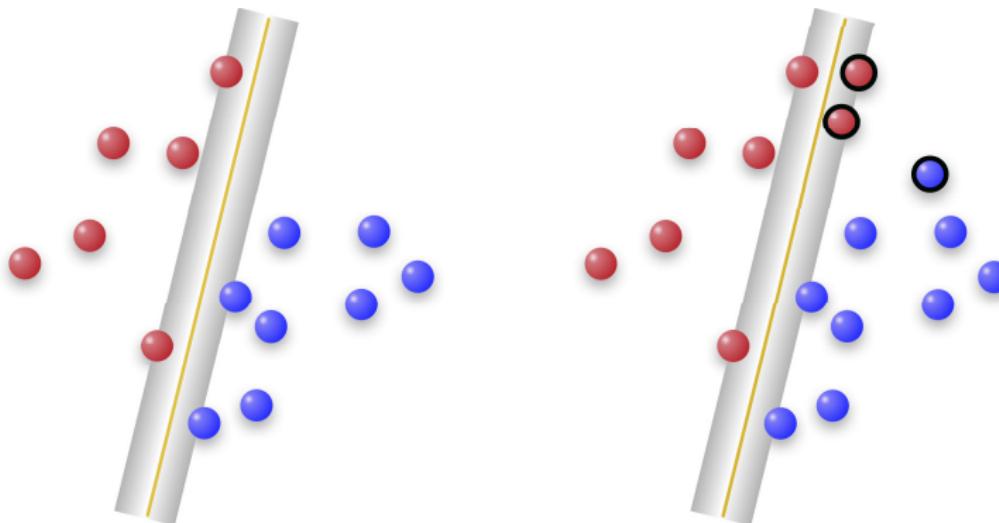


Use  $S$  to find  $(\mathbf{w}_o, b_o)$  for SVM

Performance of SVM should be evaluated based on the number of misclassifications for  $\Sigma$ , not just  $S$ .

In other words, a SVM that classifies  $S$  perfectly may not be the desired solution to the classification problem.

Why? ➤

[Plan](#)[Introduction](#)[Outcomes](#)[Math](#)[SVM](#)[Data](#)[Classification](#)[Margins](#)[Primal](#)[Lagrange](#)[Dual](#)[Soft margin](#)[Kernel](#)[Application](#)[RL](#)[Markov](#)[Q-function](#)[Bellman](#)[Optimality](#)[Value iteration](#)[Convergence](#)[Optimal policy](#)[Q-learning](#)[Explore/exploit](#)[Implementation](#)

Optimal hyperplane with **hard margin** classifies all examples in training set  $S$  correctly, but misclassifies 2 examples in the test set  $\bar{S}$ .

Hence, for the data set  $\Sigma$ , the number of misclassifications is 2.

Plan  
Introduction  
Outcomes

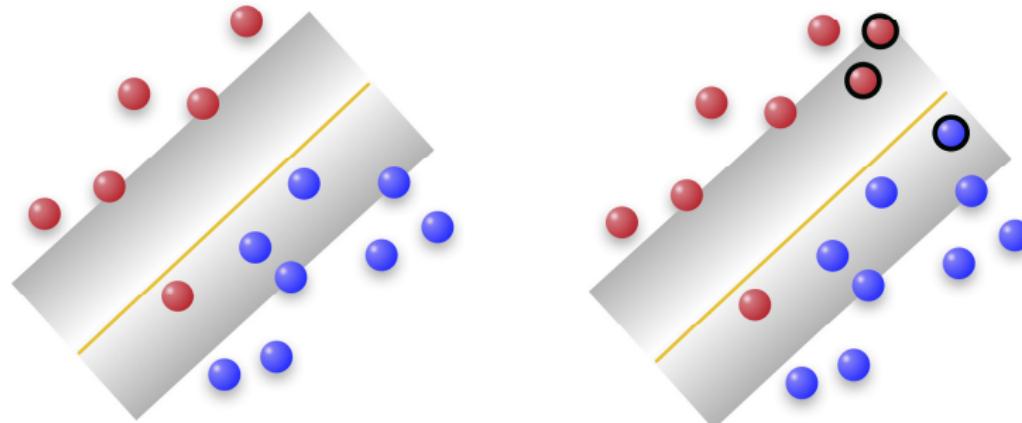
Math

SVM  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin

Kernel  
Application

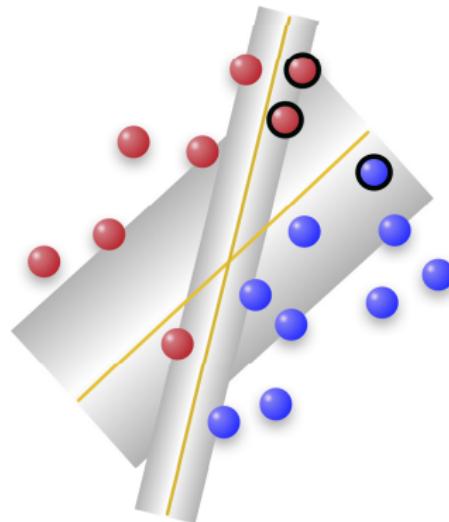
RL  
Markov  
Q-function  
Bellman  
Optimality

Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation



An optimal hyperplane with soft margin misclassifies 1 example in training set  $S$ , but it classifies correctly all examples in test set  $\bar{S}$ .

Hence, for the data set  $\Sigma$ , the number of misclassifications is 1.



## Summary of misclassifications:

Type of SVM	$S$	$\bar{S}$	$\Sigma$
With hard margin	0	2	2
With soft margin	1	0	1

A SVM that classifies training set  $S$  perfectly may not be the desired solution to the classification problem.

Plan  
Introduction  
Outcomes

Math

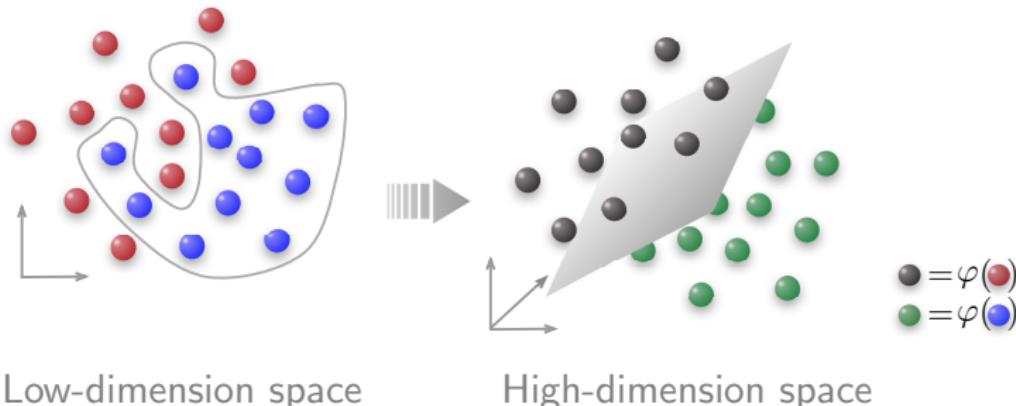
SVM  
Data  
Classification  
Margins

Primal  
Lagrange  
Dual

Soft margin

Kernel  
Application

RL  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation



## Cover's Theorem

Probability that classes are linearly separable increases when data points in input space are nonlinearly mapped to a higher dimensional feature space.

**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

Consider Newton's law of gravitation

$$f(m_1, m_2, r) = c \left( \frac{m_1 m_2}{r^2} \right) \quad (\text{For simplicity, take } r = 1)$$

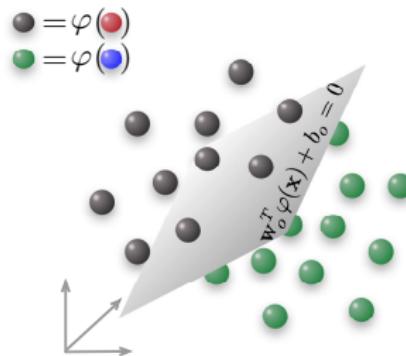
Map  $f(m_1, m_2)$  into a new function  $g(u, v)$  using nonlinear  $\varphi$

$$\mathbf{x} = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} \mapsto \varphi(\mathbf{x}) = \begin{bmatrix} \varphi_1(\mathbf{x}) \\ \varphi_2(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \ln m_1 \\ \ln m_2 \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix}$$

Solution: Applying the natural log on  $f$  yields

$$\begin{aligned} \ln f(m_1, m_2) &= \ln \left( c \left( \frac{m_1 m_2}{1^2} \right) \right) \\ &= \ln c + \ln m_1 + \ln m_2 - 2 \ln 1 \\ &= u + v + c' \quad (\text{where } c' = \ln c) \\ &= g(u, v) \end{aligned}$$

- $g(u, v)$  is linear with respect to the new coordinates  $(u, v)$ .



Optimal hyperplane in feature space

$$g(\mathbf{x}) = \mathbf{w}_o^T \varphi(\mathbf{x}) + b_o = 0$$

For training data  $\mathbf{x}_i$

$$\begin{cases} g(\mathbf{x}_i) = \mathbf{w}_o^T \varphi(\mathbf{x}_i) + b_o \geq +1 & \text{for } d_i = +1 \\ g(\mathbf{x}_i) = \mathbf{w}_o^T \varphi(\mathbf{x}_i) + b_o \leq -1 & \text{for } d_i = -1 \end{cases}$$

or, in a compact form:

$$d_i g(\mathbf{x}_i) = d_i (\mathbf{w}_o^T \varphi(\mathbf{x}_i) + b_o) \geq 1$$

$$\begin{aligned}
 L(\mathbf{w}, b, \boldsymbol{\alpha}) &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i (d_i (\mathbf{w}^T \varphi(\mathbf{x}_i) + b) - 1) \\
 &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \varphi(\mathbf{x}_i) - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i
 \end{aligned}$$

From KKT conditions:  $\mathbf{w} = \sum_{i=1}^N \alpha_i d_i \varphi(\mathbf{x}_i)$  and  $\sum_{i=1}^N \alpha_i d_i = 0$

$$\text{So } \mathbf{w}^T \mathbf{w} = \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \varphi(\mathbf{x}_i) = \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \varphi^T(\mathbf{x}_i) \varphi(\mathbf{x}_j)$$

$$\text{Let } Q(\boldsymbol{\alpha}) \equiv L(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \underbrace{\varphi^T(\mathbf{x}_i) \varphi(\mathbf{x}_j)}_{K(\mathbf{x}_i, \mathbf{x}_j)}$$

Kernel:  $\underbrace{K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_j, \mathbf{x}_i)}_{\text{symmetric}} = \varphi^T(\mathbf{x}_i) \varphi(\mathbf{x}_j) = \varphi^T(\mathbf{x}_j) \varphi(\mathbf{x}_i)$

Given the mapping

$$\mathbf{x} = [x_1 \ x_2]^T \mapsto \varphi(\mathbf{x}) = [1 \ x_1^2 \ \sqrt{2}x_1x_2 \ x_2^2 \ \sqrt{2}x_1 \ \sqrt{2}x_2]^T$$

- (i) Determine the kernel  $K(\mathbf{x}, \mathbf{y})$
- (ii) Calculate the value of the kernel if  $\mathbf{x} = [1 \ 2]^T$  and  $\mathbf{y} = [3 \ 4]^T$

Solution: (i) The kernel defined by this mapping is

$$K(\mathbf{x}, \mathbf{y}) = \varphi^T(\mathbf{x}) \varphi(\mathbf{y})$$

$$\begin{aligned}
 &= [1 \ x_1^2 \ \sqrt{2}x_1x_2 \ x_2^2 \ \sqrt{2}x_1 \ \sqrt{2}x_2] \begin{bmatrix} 1 \\ y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \\ \sqrt{2}y_1 \\ \sqrt{2}y_2 \end{bmatrix} \\
 &= 1 + x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2 + 2x_1 y_1 + 2x_2 y_2 \\
 &= (1 + \mathbf{x}^T \mathbf{y})^2
 \end{aligned}$$

(ii) With  $\mathbf{x} = [1 \ 2]^T$  and  $\mathbf{y} = [3 \ 4]^T$ , the value of the kernel is

$$\begin{aligned}
 K(\mathbf{x}, \mathbf{y}) &= 1 + x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2 + 2x_1 y_1 + 2x_2 y_2 \\
 &= 1 + 1^2 \cdot 3^2 + 2 \cdot 1 \cdot 2 \cdot 3 \cdot 4 + 2^2 \cdot 4^2 + 2 \cdot 1 \cdot 3 + 2 \cdot 2 \cdot 4 \\
 &= 1 + 9 + 48 + 64 + 6 + 16 = 144
 \end{aligned}$$

## Dual problem with soft margin

Find :  $\alpha_i$

$$\text{Maximize : } Q(\boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{Subject to : } \sum_{i=1}^N \alpha_i d_i = 0, \quad 0 \leq \alpha_i \leq C$$

## Dual problem with soft margin and transformation

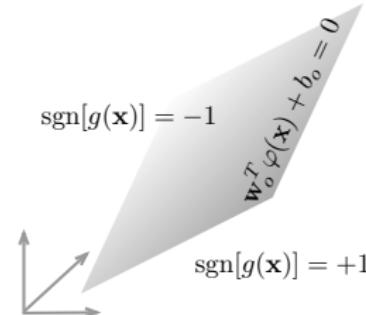
Find :  $\alpha_i$

$$\text{Maximize : } Q(\boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \boldsymbol{\varphi}^T(\mathbf{x}_i) \boldsymbol{\varphi}(\mathbf{x}_j)$$

$$\text{Subject to : } \sum_{i=1}^N \alpha_i d_i = 0, \quad 0 \leq \alpha_i \leq C$$

Given optimal value  $\alpha_{o,i}$

$$\left\{ \begin{array}{l} \mathbf{w}_o = \sum_{i=1}^N \alpha_{o,i} d_i \varphi(\mathbf{x}_i) \\ b_o = \frac{1}{d^{(s)}} - \mathbf{w}_o^T \varphi(\mathbf{x}^{(s)}) \\ (\mathbf{x}^{(s)} \text{ is a SV with label } d^{(s)}) \end{array} \right.$$



Discriminant function

$$g(\mathbf{x}) = \mathbf{w}_o^T \varphi(\mathbf{x}) + b_o = \sum_{i=1}^N \alpha_{o,i} d_i \underbrace{\varphi^T(\mathbf{x}_i) \varphi(\mathbf{x})}_{K(\mathbf{x}_i, \mathbf{x})} + b_o$$

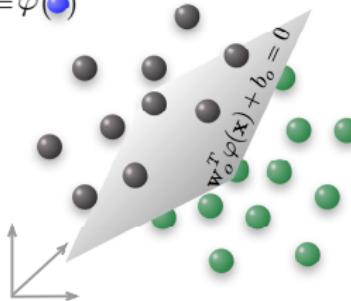
To classify a new data point  $\mathbf{x}_{\text{new}}$

$$d_{\text{new}} = \text{sgn}[g(\mathbf{x}_{\text{new}})]$$

**Main issue:** Need  $\varphi(\cdot)$

$$\bullet = \varphi(\textcolor{red}{\bullet})$$

$$\bullet = \varphi(\textcolor{blue}{\bullet})$$



Solution requires  $\varphi$  but finding explicit form of  $\varphi$  is difficult

### Kernel trick:

Find expression for  $K(\cdot, \cdot)$  directly

For instance, find

$$K(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T \mathbf{y})^2 = \varphi^T(\mathbf{x}) \varphi(\mathbf{y})$$

without first knowing the mapping

$$\mathbf{x} = [x_1 \quad x_2]^T \mapsto \varphi(\mathbf{x}) = [1 \quad x_1^2 \quad \sqrt{2}x_1x_2 \quad x_2^2 \quad \sqrt{2}x_1 \quad \sqrt{2}x_2]^T$$

Plan

Introduction

Outcomes

Math

SVM  
Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

Kernel trick Find an expression for  $K(\cdot, \cdot)$  directly without knowing  $\varphi(\cdot)$

Procedure Choose an expression for  $K(\cdot, \cdot)$ . If this expression satisfies the [Mercer's Condition](#), then it can be used as a kernel

## Mercer's condition

For training set  $S = \{(\mathbf{x}_i, d_i)\}, i = 1, 2, \dots, N$ , the [Gram matrix](#)

$$\mathbf{K} = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & \dots & K(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_N, \mathbf{x}_1) & \dots & K(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \in R^{N \times N}$$

is positive semi-definite (i.e., its eigenvalues are nonnegative)

View

**Plan****Introduction****Outcomes****Math****SVM**

## Data

## Classification

## Margins

## Primal

## Lagrange

## Dual

## Soft margin

## Kernel

## Application

**RL**

## Markov

## Q-function

## Bellman

## Optimality

## Value iteration

## Convergence

## Optimal policy

## Q-learning

## Explore/exploit

## Implementation

Given kernel  $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$  and training set below. Determine  $\mathbf{K}$ .

$i$	1	2	3	4
$\mathbf{x}_i$	$[-1, -1]^T$	$[-1, +1]^T$	$[+1, -1]^T$	$[+1, +1]^T$

$$K(\mathbf{x}_1, \mathbf{x}_1) = (1 + \mathbf{x}_1^T \mathbf{x}_1)^2 = \left(1 + \begin{bmatrix} -1 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix}\right)^2 = 9$$

$$K(\mathbf{x}_1, \mathbf{x}_2) = (1 + \mathbf{x}_1^T \mathbf{x}_2)^2 = \left(1 + \begin{bmatrix} -1 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix}\right)^2 = 1 = K(\mathbf{x}_2, \mathbf{x}_1)$$

$$K(\mathbf{x}_1, \mathbf{x}_3) = (1 + \mathbf{x}_1^T \mathbf{x}_3)^2 = \left(1 + \begin{bmatrix} -1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right)^2 = 1 = K(\mathbf{x}_3, \mathbf{x}_1)$$

$$K(\mathbf{x}_1, \mathbf{x}_4) = (1 + \mathbf{x}_1^T \mathbf{x}_4)^2 = \left(1 + \begin{bmatrix} -1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right)^2 = 1 = K(\mathbf{x}_4, \mathbf{x}_1)$$

$$\vdots \quad \vdots$$

Therefore, the Gram matrix is

$$\mathbf{K} = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & \dots & K(\mathbf{x}_1, \mathbf{x}_4) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_4, \mathbf{x}_1) & \dots & K(\mathbf{x}_4, \mathbf{x}_4) \end{bmatrix} = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

**Problem:** Consider the set of data in the table below and the kernel candidate  $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\mathbf{x}_i^T \mathbf{x}_j - 1)$ .

$i$	1	2	3	4
$\mathbf{x}_i$	$[-1, -1]^T$	$[-1, +1]^T$	$[+1, -1]^T$	$[+1, +1]^T$

- (i) Set up the equation that can be solved to obtain the eigenvalues of the Gram matrix associated with this kernel candidate.
- (ii) Determine whether the kernel candidate satisfies Mercer's condition. (Note: You might need to use some software, such as MATLAB, to find the eigenvalues.)

**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

**Solution:** (i) We first set up the Gram matrix.

$$K(\mathbf{x}_1, \mathbf{x}_1) = \tanh\left(\mathbf{x}_1^T \mathbf{x}_1 - 1\right) = \tanh\left(\begin{bmatrix} -1 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix} - 1\right) = 0.7616$$

$$K(\mathbf{x}_1, \mathbf{x}_2) = \tanh\left(\mathbf{x}_1^T \mathbf{x}_2 - 1\right) = \tanh\left(\begin{bmatrix} -1 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} - 1\right) = -0.7616$$

$$K(\mathbf{x}_1, \mathbf{x}_3) = \tanh\left(\mathbf{x}_1^T \mathbf{x}_3 - 1\right) = \tanh\left(\begin{bmatrix} -1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} - 1\right) = -0.7616$$

$$K(\mathbf{x}_1, \mathbf{x}_4) = \tanh\left(\mathbf{x}_1^T \mathbf{x}_4 - 1\right) = \tanh\left(\begin{bmatrix} -1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 1\right) = -0.9951$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots$$

The Gram matrix is

$$\mathbf{K} = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & \dots & K(\mathbf{x}_1, \mathbf{x}_4) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_4, \mathbf{x}_1) & \dots & K(\mathbf{x}_4, \mathbf{x}_4) \end{bmatrix}$$

$$= \begin{bmatrix} 0.7616 & -0.7616 & -0.7616 & -0.9951 \\ -0.7616 & 0.7616 & -0.9951 & -0.7616 \\ -0.7616 & -0.9951 & 0.7616 & -0.7616 \\ -0.9951 & -0.7616 & -0.7616 & 0.7616 \end{bmatrix}$$

The eigenvalues of  $\mathbf{K}$  are determined by solving the equation

$$|\lambda \mathbf{I} - \mathbf{K}|$$

### Plan

**Introduction**

**Outcomes**

**Math**

**SVM**

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

$$\begin{aligned} &= \left| \lambda \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.7616 & -0.7616 & -0.7616 & -0.9951 \\ -0.7616 & 0.7616 & -0.9951 & -0.7616 \\ -0.7616 & -0.9951 & 0.7616 & -0.7616 \\ -0.9951 & -0.7616 & -0.7616 & 0.7616 \end{bmatrix} \right| \\ &= \begin{vmatrix} \lambda - 0.7616 & 0.7616 & 0.7616 & 0.9951 \\ 0.7616 & \lambda - 0.7616 & 0.9951 & 0.7616 \\ 0.7616 & 0.9951 & \lambda - 0.7616 & 0.7616 \\ 0.9951 & 0.7616 & 0.7616 & \lambda - 0.7616 \end{vmatrix} = 0 \end{aligned}$$

(ii) The eigenvalues of the Gram matrix  $\mathbf{K}$  are found to be:

$$\lambda_1 = -1.7567 < 0$$

$$\lambda_2 = 1.7567$$

$$\lambda_3 = 1.7567$$

$$\lambda_4 = 1.2897$$

Since one of the eigenvalues is negative, this kernel candidate is not admissible.

Given a training set  $S = \{(\mathbf{x}_i, d_i)\}, i = 1, \dots, N$

1 Find a suitable kernel

Choose expression  
then check Mercer's  
condition

2 Choose a value for  $C$

3 Solve for  $\alpha_{o,i}$

4 Determine  $b_o$  in

$$g(\mathbf{x}) = \sum_{i=1}^N \alpha_{o,i} d_i K(\mathbf{x}, \mathbf{x}_i) + b_o$$

using the fact that for a support vector  $\mathbf{x}^{(s)}$

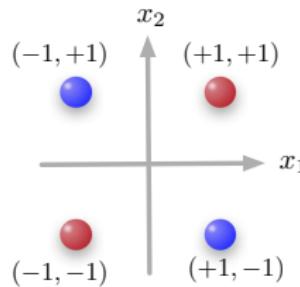
$$g(\mathbf{x}^{(s)}) = \pm 1 = d^{(s)}$$

Maximize :  $Q(\boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j K(\mathbf{x}_i, \mathbf{x}_j)$

Subject to :  $\sum_{i=1}^N \alpha_i d_i = 0, 0 \leq \alpha_i \leq C$

Support vector machine:





$i$	$\mathbf{x}_i$	$d_i$
1	$[-1, -1]^T$	-1
2	$[-1, +1]^T$	+1
3	$[+1, -1]^T$	+1
4	$[+1, +1]^T$	-1

Choose kernel:  $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$

$$\begin{aligned}
 &= 1 + x_{i,1}^2 x_{j,1}^2 + 2x_{i,1}x_{i,2}x_{j,1}x_{j,2} \\
 &+ x_{i,2}^2 x_{j,2}^2 + 2x_{i,1}x_{j,1} + 2x_{i,2}x_{j,2}
 \end{aligned}$$

The Gram matrix is

$$\mathbf{K} = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

The eigenvalues are  
*In Mathematica*

```

In[1] = Eigenvalues[
{{9, 1, 1, 1}, {1, 9, 1, 1},
 {1, 1, 9, 1}, {1, 1, 1, 9}}]
Out[1] = {12, 8, 8, 8}

```

**Plan**  
**Introduction**  
**Outcomes**

**Math**  
**SVM**  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

**RL**  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

$$\begin{aligned}
Q(\alpha) &= \sum_{i=1}^4 \alpha_i - \frac{1}{2} \sum_{i=1}^4 \sum_{j=1}^4 \alpha_i \alpha_j d_i d_j K(\mathbf{x}_i, \mathbf{x}_j) \\
&= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \\
&\quad \frac{1}{2} \left( 9\alpha_1^2 - 2\alpha_1\alpha_2 - \alpha_1\alpha_3 + 2\alpha_1\alpha_4 + \right. \\
&\quad \left. 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2 \right)
\end{aligned}$$

$$\left. \begin{array}{l} 9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 1 \\ -\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 = 1 \\ -\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 = 1 \\ \alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 = 1 \\ -\alpha_1 + \alpha_2 + \alpha_3 - \alpha_4 = 0 \end{array} \right\} \text{from KKT conditions} \quad \frac{\partial Q}{\partial \alpha_i} = 0$$

$$\alpha_{o,1} = \alpha_{o,2} = \alpha_{o,3} = \alpha_{o,4} = \frac{1}{8} \quad (\text{all } \mathbf{x}_i \text{ are SVs})$$

**Note:** This approach happens to work in this particular case. **It is not a general method.** More sophisticated numerical methods are needed to solve general constrained optimization problems

## Discriminant function

$$\begin{aligned} g(\mathbf{x}) &= \sum_{i=1}^N \alpha_{o,i} d_i K(\mathbf{x}, \mathbf{x}_i) + b_o \\ &= \alpha_{o,1} d_1 K(\mathbf{x}, \mathbf{x}_1) + \\ &\quad \alpha_{o,2} d_2 K(\mathbf{x}, \mathbf{x}_2) + \\ &\quad \alpha_{o,3} d_3 K(\mathbf{x}, \mathbf{x}_3) + \\ &\quad \alpha_{o,4} d_4 K(\mathbf{x}, \mathbf{x}_4) + b_o \end{aligned}$$

General vector:  $\mathbf{x} = [x_1, x_2]^T$

Training data: $\mathbf{x}_i = [x_{i,1}, x_{i,2}]^T$		
$i$	$\mathbf{x}_i$	$d_i$
1	$[-1, -1]^T$	-1
2	$[-1, +1]^T$	+1
3	$[+1, -1]^T$	+1
4	$[+1, +1]^T$	-1

$$K(\mathbf{x}_i, \mathbf{x}_j) = 1 + x_{i,1}^2 x_{j,1}^2 + 2x_{i,1}x_{i,2}x_{j,1}x_{j,2} + x_{i,2}^2 x_{j,2}^2 + 2x_{i,1}x_{j,1} + 2x_{i,2}x_{j,2}$$

For example,  $K(\mathbf{x}, \mathbf{x}_1)$  is calculated as follows:

$$\begin{aligned} &1 + x_1^2 x_{1,1}^2 + 2x_1 x_2 x_{1,1} x_{1,2} + x_2^2 x_{1,2}^2 + 2x_1 x_{1,1} + 2x_2 x_{1,2} \\ &= 1 + x_1^2 (-1)^2 + 2x_1 x_2 (-1)(-1) + x_2^2 (-1)^2 + 2x_1 (-1) + 2x_2 (-1) \\ &= 1 + x_1^2 + 2x_1 x_2 + x_2^2 - 2x_1 - 2x_2 \end{aligned}$$

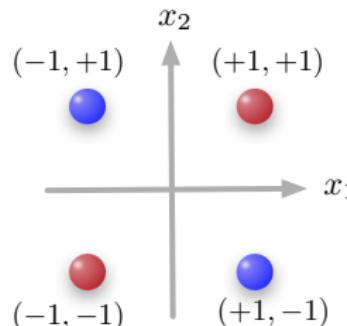
$$\begin{aligned}
 g(\mathbf{x}) &= \sum_{i=1}^N \alpha_{o,i} d_i K(\mathbf{x}, \mathbf{x}_i) + b_o && \text{General vector: } \mathbf{x} = [x_1, x_2]^T \\
 &= \alpha_{o,1} d_1 K(\mathbf{x}, \mathbf{x}_1) + && \text{Training data: } \mathbf{x}_i = [x_{i,1}, x_{i,2}]^T \\
 &\quad \alpha_{o,2} d_2 K(\mathbf{x}, \mathbf{x}_2) + \\
 &\quad \alpha_{o,3} d_3 K(\mathbf{x}, \mathbf{x}_3) + \\
 &\quad \alpha_{o,4} d_4 K(\mathbf{x}, \mathbf{x}_4) + b_o \\
 &= \frac{1}{8}(-8x_1x_2) + b_o \\
 &= -x_1x_2 + b_o
 \end{aligned}$$

$i$	$\mathbf{x}_i$	$d_i$	$\alpha_{o,i}$
1	$[-1, -1]^T$	-1	$1/8$
2	$[-1, +1]^T$	+1	$1/8$
3	$[+1, -1]^T$	+1	$1/8$
4	$[+1, +1]^T$	-1	$1/8$

To find  $b_o$ , pick a support vector, say,  $\mathbf{x}_2 = [x_{2,1} \ x_{2,2}]^T$ , then

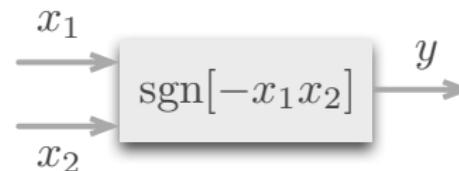
$$g(\mathbf{x}_2) = -x_{2,1}x_{2,2} + b_0 = 1$$

$$b_o = 1 + x_{2,1}x_{2,2} = 1 + (-1)1 = 0$$



$i$	$\mathbf{x}_i$	$d_i$
1	$[-1, -1]^T$	-1
2	$[-1, +1]^T$	+1
3	$[+1, -1]^T$	+1
4	$[+1, +1]^T$	-1

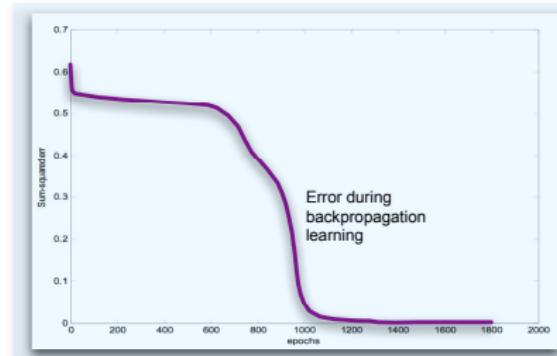
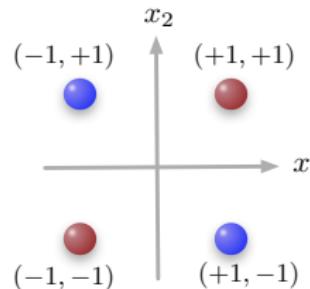
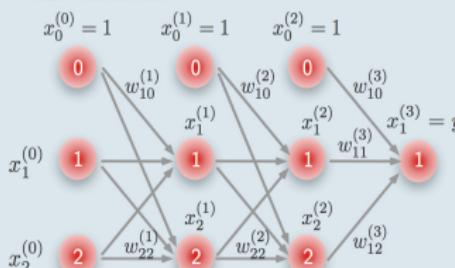
Solution in the form of a SVM:



EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
 Data  
 Classification  
 Margins  
 Primal  
 Lagrange  
 Dual  
 Soft margin  
 Kernel  
**Application**

**RL**  
 Markov  
 Q-function  
 Bellman  
 Optimality  
 Value iteration  
 Convergence  
 Optimal policy  
 Q-learning  
 Explore/exploit  
 Implementation

**MLP solution**

$$\begin{aligned} \text{bias} &= \begin{bmatrix} -1.87 & 5.17 & 5.10 \\ -4.66 & 3.06 & 3.04 \end{bmatrix} \\ \mathbf{w}^{(1)} &= \begin{bmatrix} w_{10}^{(1)} \\ w_{11}^{(2)} \\ w_{12}^{(3)} \end{bmatrix} \\ \text{bias} &= \begin{bmatrix} -1.06 & 3.69 & -5.29 \\ 2.53 & -4.67 & 3.54 \end{bmatrix} \\ \mathbf{w}^{(2)} &= \begin{bmatrix} w_{20}^{(1)} \\ w_{21}^{(2)} \\ w_{22}^{(3)} \end{bmatrix} \\ \text{bias} &= [-0.36, 6.49, -6.5147] \\ \mathbf{w}^{(3)} &= \begin{bmatrix} w_{30}^{(1)} \\ w_{31}^{(2)} \\ w_{32}^{(3)} \end{bmatrix} \end{aligned}$$

**SVM solution**

$$y = \text{sgn}[-x_1 x_2]$$

EE5904  
ME5404  
Part II

## Plan

### Introduction

### Outcomes

### Math

### SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

### RL

Markov

Q-function

Bellman

Optimality

Value iteration

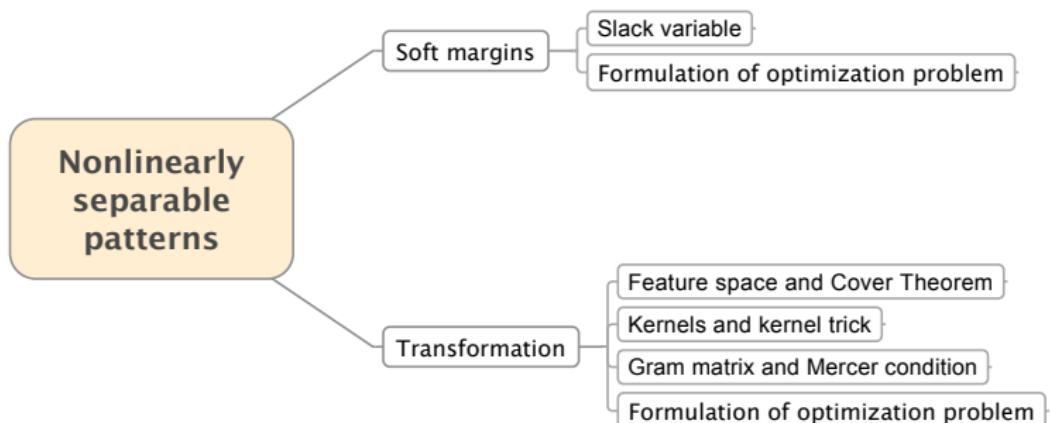
Convergence

Optimal policy

Q-learning

Explore/exploit

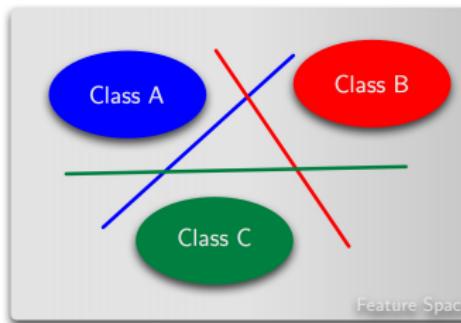
Implementation



EE5904  
ME5404  
Part II

Plan  
Introduction  
Outcomes  
Math  
SVM  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

RL  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation



- $K$  data classes:  $C_1, C_2, \dots, C_K$
- Construct  $K$  SVMs, with the  $k^{th}$  SVM trained with data from  $C_k$  as positive examples and data from remaining  $K - 1$  classes as negative examples

EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

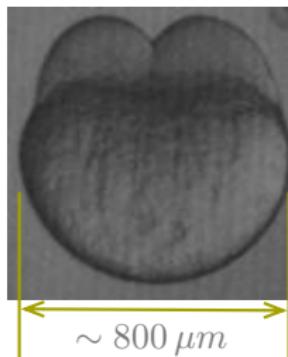
Optimal policy

Q-learning

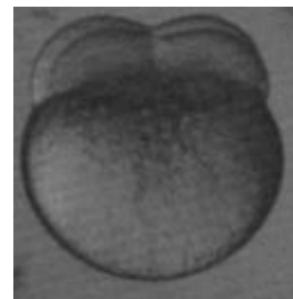
Explore/exploit

Implementation

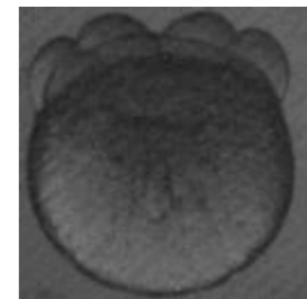
Two-cell embryo



Four-cell embryo



Eight-cell embryo





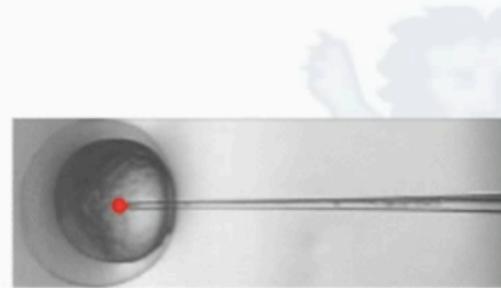
EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
**Application**

**RL**  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

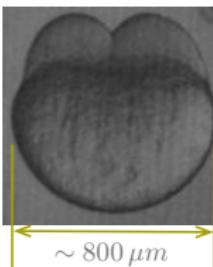
## A Micromanipulation System for Automatic Batch Microinjection

### Microinjection

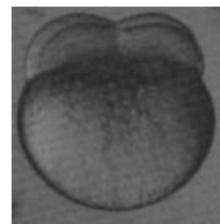


Zhe LU, Peter CHEN, Joo Hoo NAM, Ruowen GE, and Wei LIN  
2007 IEEE International Conference on Robotics and Automation, 10-14 April 2007, Roma, Italy

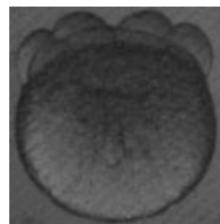
Two-cell embryo



Four-cell embryo



Eight-cell embryo

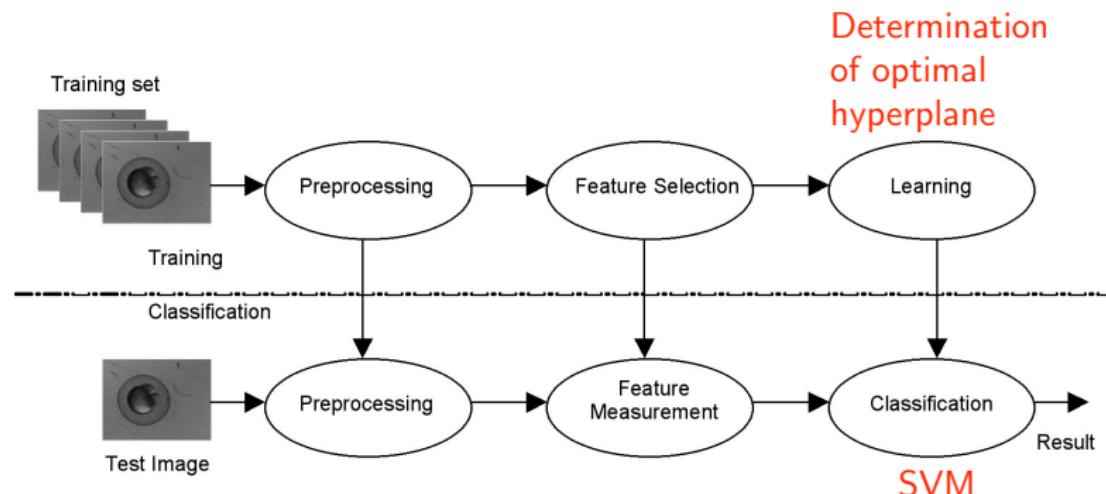


- Injecting DNA material into embryos to study development
- Injection done at the right development stage of embryo  
(Too late for injection if embryo has  $> 8$  cells)
- Depending on stage, DNA used may be different
- ✓ Important to classify stage of embryos before injection

EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

**RL**  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

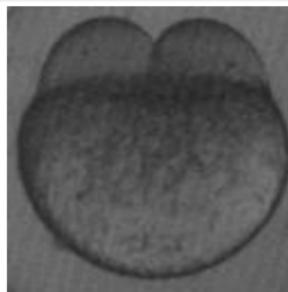
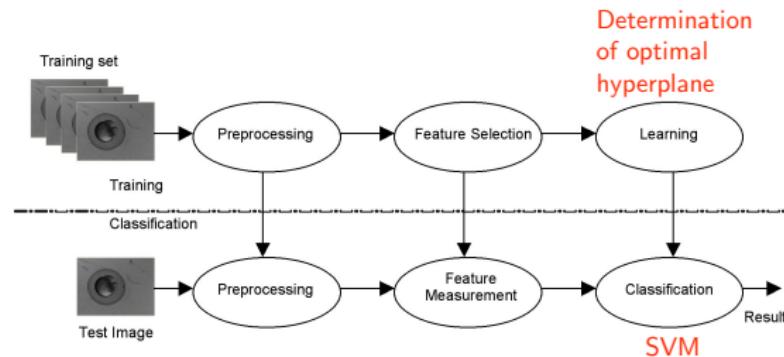


EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**

**Math**  
**SVM**  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

**RL**  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation



128 × 128 pixels  
with 8-bit gray level

$$\begin{bmatrix}
 0.06294 & 0.3865 & -0.4545 & -0.3429 \\
 0.03054 & -0.3097 & -0.3181 & 0.4041 \\
 -0.2543 & 0.3271 & -0.5241 & -0.8429 \\
 0.2073 & -0.2562 & -0.07045 & -0.3694 \\
 0.7951 & -0.1101 & 0.2174 & 0.6135 \\
 0.0488
 \end{bmatrix}$$

Vector of 21 normalized eigenbases

EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

- SVM with quadratic kernel and  $C = 1.25$
- A total of 61 images:

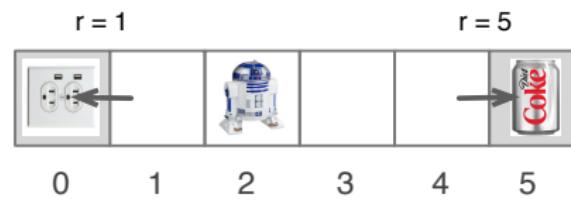
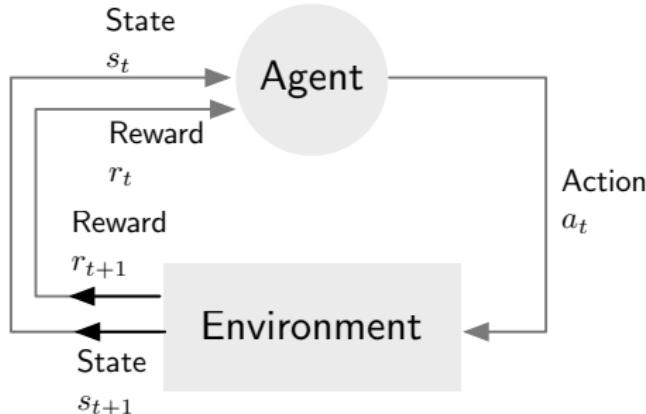
2-cell embryo	25
4-cell embryo	21
8-cell embryo	15
Training set	40
Validation set	21

- Correct classification: 71%

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
 Data  
 Classification  
 Margins  
 Primal  
 Lagrange  
 Dual  
 Soft margin  
 Kernel  
 Application

**RL**  
 Markov  
 Q-function  
 Bellman  
 Optimality  
 Value iteration  
 Convergence  
 Optimal policy  
 Q-learning  
 Explore/exploit  
 Implementation

## Agent learns to maximize reward when completing task

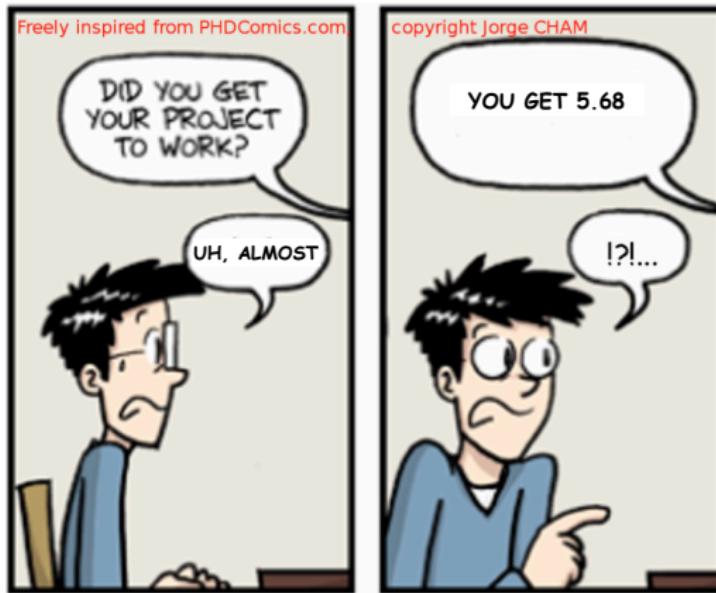


EE5904  
ME5404  
Part II

Plan  
Introduction  
Outcomes  
Math  
SVM  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

## RL

Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation



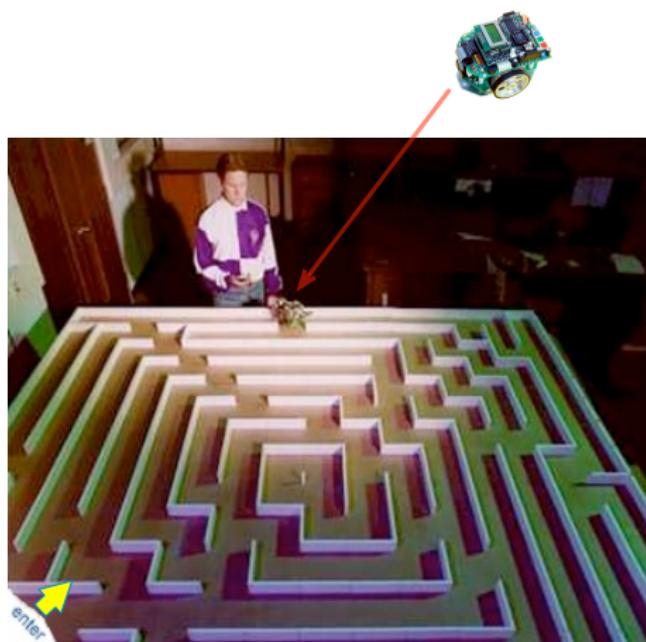
Adapted from slides by Olivier Sigaud

- Is a score of 5.68 good or bad, how good or how bad?
- The score does not tell you how to get the project to work
- You have to learn how yourself by maximizing your scores

EE5904  
ME5404  
Part II

Plan  
Introduction  
Outcomes  
Math  
SVM  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

RL  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

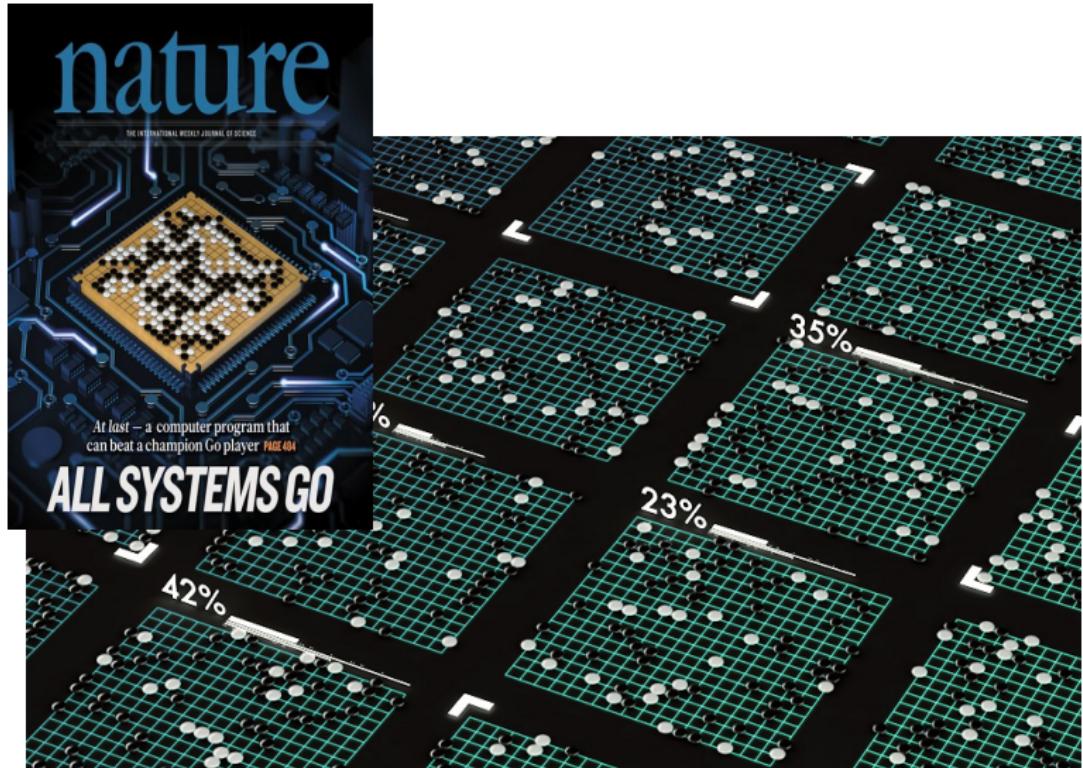


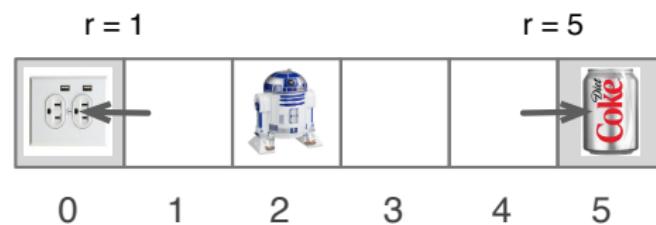
- Robot has no knowledge about maze before learning starts
- Robot explores maze and receives reward if it reaches goal
- Robot learns the “optimal” route to reach goal

# AlphaGo used RL to beat world No. 1 ranked player in 3-game match in 2017

EE5904  
ME5404  
Part II

Plan  
  
Introduction  
  
Outcomes  
  
Math  
  
SVM  
  
Data  
  
Classification  
  
Margins  
  
Primal  
  
Lagrange  
  
Dual  
  
Soft margin  
  
Kernel  
  
Application  
  
**RL**  
  
Markov  
  
Q-function  
  
Bellman  
  
Optimality  
  
Value iteration  
  
Convergence  
  
Optimal policy  
  
Q-learning  
  
Explore/exploit  
  
Implementation





- Robot can move left or right
- Robot stops when reaching state 0 or 5 and gets reward
- Future rewards are discounted by 50% per step after 1<sup>st</sup>
- What should robot do to maximize reward?

Find out maximum **reward** starting from a given state

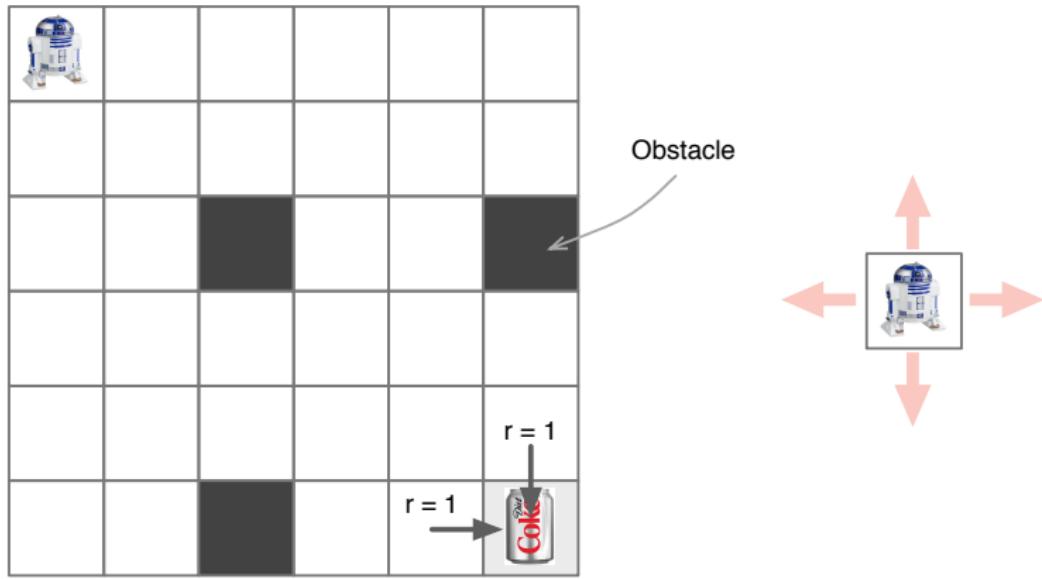
At state	Move left	Move right
1	<b>1</b>	$5 \cdot (1 - 0.5)^3 = 0.625$
2	$1 \cdot (1 - 0.5)^1 = 0.5$	$5 \cdot (1 - 0.5)^2 = \mathbf{1.25}$
3	$1 \cdot (1 - 0.5)^2 = 0.625$	$5 \cdot (1 - 0.5)^1 = \mathbf{2.5}$
4	$1 \cdot (1 - 0.5)^3 = 0.125$	<b>5</b>

Actions corresponding to values in blue are the “best” actions

Plan  
Introduction  
Outcomes  
Math  
SVM  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

RL  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

How does robot maximize reward in this case?



For **complex** situation, we need formal method

EE5904  
ME5404  
Part II

Plan  
Introduction  
Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

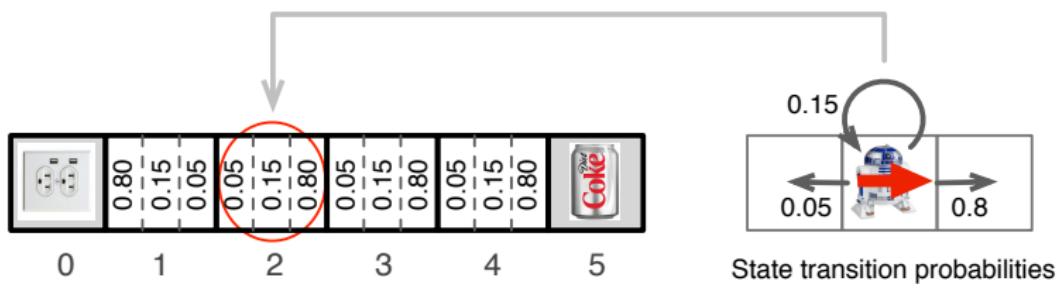
Optimal policy

Q-learning

Explore/exploit

Implementation

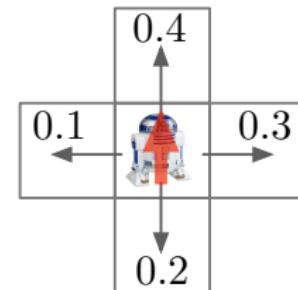
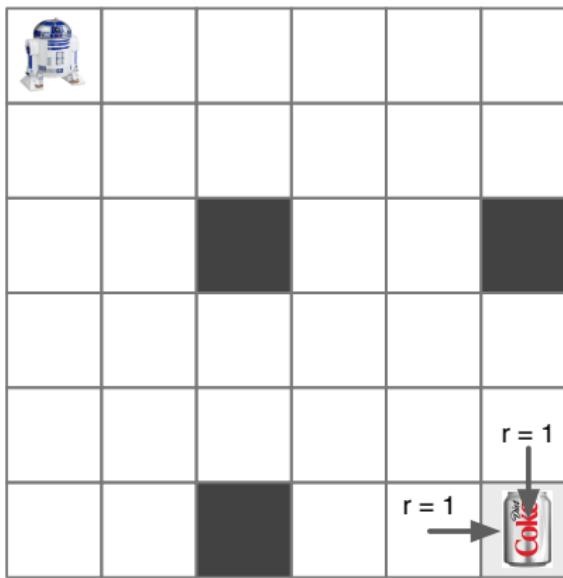
## What if robot behavior is **uncertain**?



## What if situation is both **complex** and **uncertain**?

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

**RL**  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation



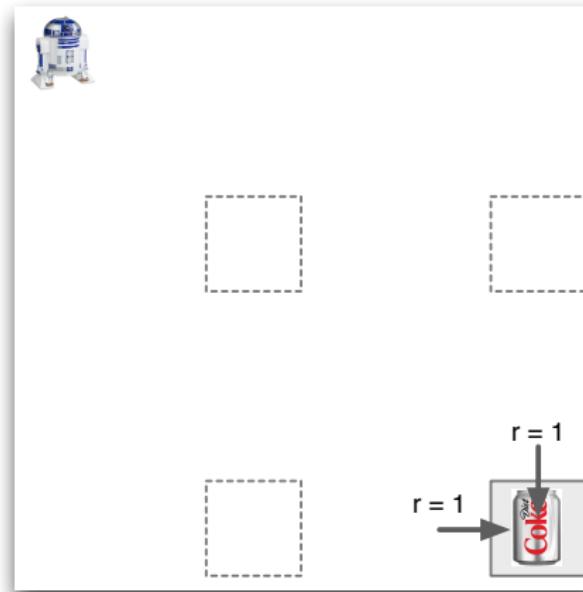
State transition probabilities

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

**RL**

Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

What if situation is **complex** and **unknown** (e.g., robot does not know where obstacles are)?



We need systematic approach to solve such problems

EE5904  
ME5404  
Part II

Plan  
Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

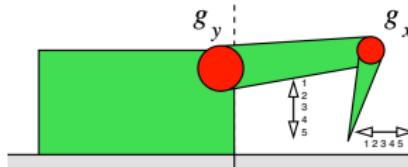
Convergence

Optimal policy

Q-learning

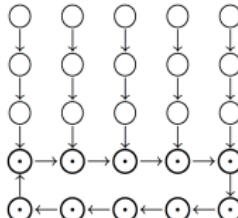
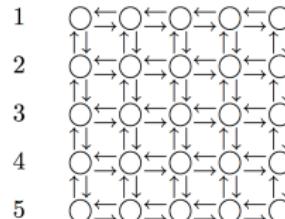
Explore/exploit

Implementation



- 2 joints for actuating upper and lower arms
- Motion of each arm is discretized into 5 states
- Reward = distance of forward motion/step time

$g_x$  1 2 3 4 5  
 $g_y$



Tokic, M., Ertel, W., and Fessler, J., The crawler, a class room demonstrator for reinforcement learning.  
Proc. of the 22<sup>nd</sup> International Florida Artificial Intelligence Research Society Conference, 160-165, 2009



EE5904  
ME5404  
Part II

**Plan**

**Introduction**

**Outcomes**

**Math**

**SVM**

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation



Adam, S., Busoniu, L. and Babuska, R., Experience replay for real-time reinforcement learning control  
IEEE Trans. on Sys., Man, and Cybern., Part C: App. and Rev., vol. 42, no. 2, 201–212, 2012

EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

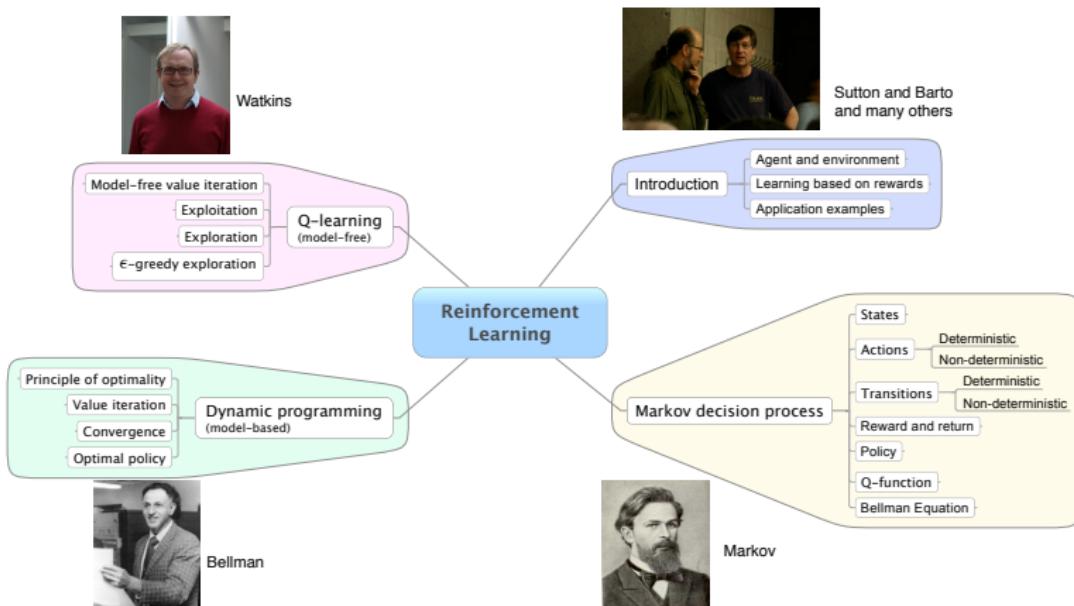
Explore/exploit

Implementation

- Unsupervised: Agent does not know the desired output
- Uncertain or unknown system behavior
- Reward received after each action is the only feedback
- **Objective:** Maximize total reward when executing a task

EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
**Data**  
**Classification**  
**Margins**  
**Primal**  
**Lagrange**  
**Dual**  
**Soft margin**  
**Kernel**  
**Application**  
**RL**  
**Markov**  
**Q-function**  
**Bellman**  
**Optimality**  
**Value iteration**  
**Convergence**  
**Optimal policy**  
**Q-learning**  
**Explore/exploit**  
**Implementation**



**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL****Markov**

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

- A set of states:  $S = \{s_1, s_2, \dots\}$

- A set of actions:  $A = \{a_1, a_2, \dots, a_n\}$
- A transition function

Deterministic:

$$\bar{f}(s, a) = s', \quad s' \in S$$

Non-deterministic:

$$f(s, a, s') = P(s_{t+1} = s' | s_t = s, a_t = a) \equiv P_{ss'}^a$$

where  $P(a|b)$  = Probability of  $a$  being true under condition  $b$

**Plan****Introduction****Outcomes****Math****SVM****Data****Classification****Margins****Primal****Lagrange****Dual****Soft margin****Kernel****Application****RL****Markov****Q-function****Bellman****Optimality****Value iteration****Convergence****Optimal policy****Q-learning****Explore/exploit****Implementation**

$$S = \{0, 1, 2, 3, 4, 5\}, \quad A = \{-1, 1\}$$

Suppose that due to uncertainties in robot's operation (such as slippery floor or error in robot position-tracking system, etc.), effect of taking  $a = 1$  at state 2 becomes stochastic

Transition probabilities are:

$$\begin{cases} f(2, 1, 3) = 0.8 \\ f(2, 1, 2) = 0.15 \\ f(2, 1, 1) = 0.05 \end{cases}$$

Note that

$$\sum_{i=1}^3 f(2, 1, i) = 1$$

**Plan****Introduction****Outcomes****Math****SVM****Data****Classification****Margins****Primal****Lagrange****Dual****Soft margin****Kernel****Application****RL****Markov****Q-function****Bellman****Optimality****Value iteration****Convergence****Optimal policy****Q-learning****Explore/exploit****Implementation**

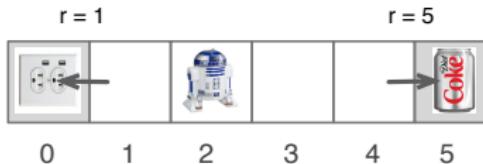
- Agent takes action  $a$  to move from state  $s$  to  $s'$
- Agent receives reward  $r$  one time-step later (i.e., at  $s'$ )

Reward for taking  $a_t$  at  $s_t$  at time step  $t$  and reaching  $s_{t+1}$

$$r_{t+1} = \rho(s_t, a_t, s_{t+1})$$

where  $\rho : S \times A \times S \rightarrow \mathcal{R}$  is called the **reward function**

Example: Robot takes action 1 at state 4 at time step  $t$



$$r_{t+1} = \rho(4, 1, 5) = 5$$

EE5904  
ME5404  
Part II

Plan  
Introduction  
Outcomes

Math  
SVM  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

RL  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

## For non-deterministic transitions:

Reward for taking  $a$  at  $s$  is characterized by **expected value** of  $r_{t+1}$  over all possible new states:

$$E[r_{t+1} \mid s_t = s] = \sum_{s'} \left( P_{ss'}^a r_{t+1} \mid s_{t+1} = s' \right) = \sum_{s'} P_{ss'}^a \rho(s, a, s')$$

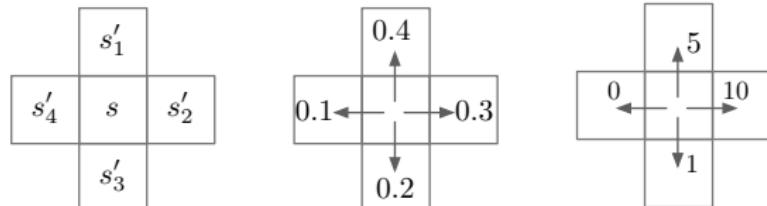
- $P_{ss'}^a$  = probability of reaching  $s'$  after taking  $a$  at  $s$ .
- Summation over  $s'$  accounts for all possible new states

- Taking action  $a$  at state  $s$

**Plan**  
**Introduction**  
**Outcomes**  
**Math**

**SVM**  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

**RL**  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation



States	Transition probabilities	Rewards
--------	--------------------------	---------

$$r_{t+1}|_{s_{t+1}=s'_1} = \rho(s, a, s'_1) = 5$$

$$r_{t+1}|_{s_{t+1}=s'_2} = \rho(s, a, s'_2) = 10$$

$$r_{t+1}|_{s_{t+1}=s'_3} = \rho(s, a, s'_3) = 1$$

$$r_{t+1}|_{s_{t+1}=s'_4} = \rho(s, a, s'_4) = 0$$

$$f(s, a, s'_1) = P(s_{t+1} = s'_1 | s_t = s, a_t = a) \equiv P_{ss'_1}^a = 0.4$$

$$f(s, a, s'_2) = P(s_{t+1} = s'_2 | s_t = s, a_t = a) \equiv P_{ss'_2}^a = 0.3$$

$$f(s, a, s'_3) = P(s_{t+1} = s'_3 | s_t = s, a_t = a) \equiv P_{ss'_3}^a = 0.2$$

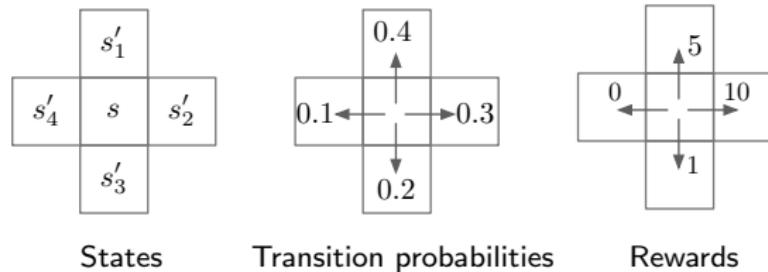
$$f(s, a, s'_4) = P(s_{t+1} = s'_4 | s_t = s, a_t = a) \equiv P_{ss'_4}^a = 0.1$$

EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**  
**Math**

**SVM**  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

**RL**  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation



$$\begin{aligned}
 E[r_{t+1}] &= \sum_{s'} \left( P_{ss'}^a r_{t+1} \right) = \sum_{s'} P_{ss'}^a \rho(s, a, s') \\
 &= P_{ss'_1}^a \rho(s, a, s'_1) + P_{ss'_2}^a \rho(s, a, s'_2) \\
 &\quad + P_{ss'_3}^a \rho(s, a, s'_3) + P_{ss'_4}^a \rho(s, a, s'_4) \\
 &= 0.4 \times 5 + 0.3 \times 10 + 0.2 \times 1 + 0.1 \times 0 \\
 &= 5.2
 \end{aligned}$$

[Plan](#)[Introduction](#)[Outcomes](#)[Math](#)[SVM](#)[Data](#)[Classification](#)[Margins](#)[Primal](#)[Lagrange](#)[Dual](#)[Soft margin](#)[Kernel](#)[Application](#)[RL](#)[Markov](#)[Q-function](#)[Bellman](#)[Optimality](#)[Value iteration](#)[Convergence](#)[Optimal policy](#)[Q-learning](#)[Explore/exploit](#)[Implementation](#)

- Agent starts from initial state and reaches  $s$  after  $t$  time steps
- **Question:** What is the total reward **from this point onward** if agent continues to make transitions?

## Return $R_t$

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

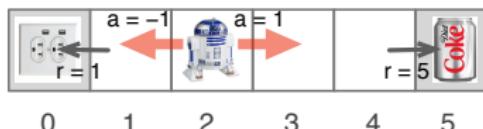
- $k$  = index of time steps after  $t$ , with  $k = 0$  being 1<sup>st</sup> step
- $\gamma$  is called the **discount rate**, with  $0 \leq \gamma \leq 1$

EE5904  
ME5404  
Part II

## Return $R_t$

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- $R_t$  determines present value of future rewards
- Reward received  $k$  steps in future is discounted by factor of  $\gamma^{k-1}$
- A small  $\gamma$  forces agent to focus more on immediate rewards from next few steps and heavily discount rewards from future steps
- A large  $\gamma$  forces agent to take into account future rewards more strongly, i.e., agent becomes more farsighted



$$S = \{0, 1, 2, 3, 4, 5\}$$

$$A = \{-1, 1\}$$

$$\gamma = 0.5$$

$$\bar{f}(s, a) = \begin{cases} s + a & \text{if } 1 \leq s \leq 4 \\ s & \text{if } s = 0 \text{ or } s = 5 \end{cases} \quad \rho(s, a, s') = \begin{cases} 5 & \text{if } s \neq 5 \text{ and } s' = 5 \\ 1 & \text{if } s \neq 0 \text{ and } s' = 0 \\ 0 & \text{otherwise} \end{cases}$$

- 3 transitions to move from state 2 to 5 without changing direction

$$\begin{aligned}
 R_t &= \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \\
 &= \rho(2, 1, 3) + 0.5 \cdot \rho(3, 1, 4) + 0.5^2 \cdot \rho(4, 1, 5) \\
 &= 0 + 0 + 0.25 \cdot 5 \\
 &= 1.25
 \end{aligned}$$

- Decision-making mechanism for controlling a Markov process
- Specifies which action to take at a state
- Can be deterministic or non-deterministic

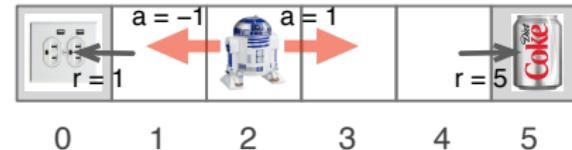
**Deterministic:** Specifies particular action to be taken at particular state

$$\pi : S \rightarrow A; \quad \pi(s) = a, s \in S, a \in A$$

Non-deterministic: Specifies probability of action being taken at particular state

$$\pi : S \times A \rightarrow [0, 1]; \quad \pi(s, a) = P(a_t = a | s_t = s)$$

In this module, we consider **deterministic policies** only



Suppose that robot must be controlled to behave as follows:

- In state 1, it must take  $a = -1$
- In state 0 or 5, it can take either action
- In any of remaining states, it must take  $a = 1$

The policy that governs behavior of robot is

$$\begin{cases} \pi(1) = -1 \\ \pi(s) = \pm 1 & \text{if } s = 0, 5 \\ \pi(s) = 1 & \text{if } s = 2, 3, 4 \end{cases}$$

Plan  
Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation



Key questions to be answered:

1. What is the “best” policy ?
2. How to find the “best” policy?

Answers require *Q*-function and Bellman Equation

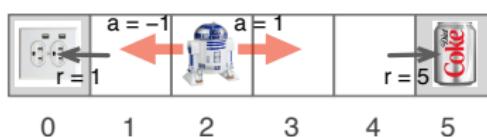
**Plan****Introduction****Outcomes****Math****SVM****Data****Classification****Margins****Primal****Lagrange****Dual****Soft margin****Kernel****Application****RL****Markov****Q-function****Bellman****Optimality****Value iteration****Convergence****Optimal policy****Q-learning****Explore/exploit****Implementation**

- Measures the “worth” of taking  $a$  at  $s$  under  $\pi$
- “Worth” is represented by

$$Q^\pi : S \times A \rightarrow \mathcal{R}$$

defined as the **expected return** from taking action  $a$  at state  $s$  at time step  $t$ , and thereafter following policy  $\pi$

$$Q^\pi(s, a) = E^\pi[R_t \mid s_t = s]$$

**Plan****Introduction****Outcomes****Math****SVM****Data****Classification****Margins****Primal****Lagrange****Dual****Soft margin****Kernel****Application****RL****Markov****Q-function****Bellman****Optimality****Value iteration****Convergence****Optimal policy****Q-learning****Explore/exploit****Implementation**

$$S = \{0, 1, 2, 3, 4, 5\}$$

$$A = \{-1, 1\}$$

$$\gamma = 0.5$$

$$\bar{f}(s, a) = \begin{cases} s + a & \text{if } 1 \leq s \leq 4 \\ s & \text{if } s = 0 \text{ or } s = 5 \end{cases} \quad \rho(s, a, s') = \begin{cases} 5 & \text{if } s \neq 5 \text{ and } s' = 5 \\ 1 & \text{if } s \neq 0 \text{ and } s' = 0 \\ 0 & \text{otherwise} \end{cases}$$

- Calculate  $Q^\pi(2, 1)$  with  $\pi(2) = 1, \pi(3) = 1$ , and  $\pi(4) = 1$

Since all transitions are deterministic,  $Q$ -function degenerates to just  $Q^\pi(2, 1) = (R_t | s_t = 2)$ , i.e.,

$$\begin{aligned} Q^\pi(2, 1) &= E[R_t | s_t = 2] = (R_t | s_t = 2) \\ &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \\ &= \rho(2, 1, 3) + 0.5 \cdot \rho(3, 1, 4) + 0.5^2 \cdot \rho(4, 1, 5) \\ &= 0 + 0 + 0.25 \cdot 5 = 1.25 \end{aligned}$$

Example about non-deterministic transitions is presented later in connection with Bellman Equation

Plan  
Introduction  
Outcomes  
Math  
SVM  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

RL  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

- $Q$ -function measures “worth” of state-action pair  $(s, a)$  in terms of rewards received when agent is executing a task
- An optimal policy is one that maximizes (with respect to a given task) values of  $Q$ -function over all possible  $(s, a)$  pairs
- Solving a reinforcement learning problems is about finding an optimal policy
- To do this requires the Bellman Equation

From definition of  $Q$ -function:

$$\begin{aligned}
 Q^\pi(s, a) &= E^\pi[R_t \mid s_t = s] \\
 &= E^\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right] \\
 &= E^\pi \left[ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \mid s_t = s \right] \\
 &= E^\pi \left[ \left( r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \right) \mid s_t = s \right] \quad (\text{Isolating } r_{t+1}) \\
 &= E^\pi[r_{t+1}] + E^\pi \left[ \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s \right]
 \end{aligned}$$

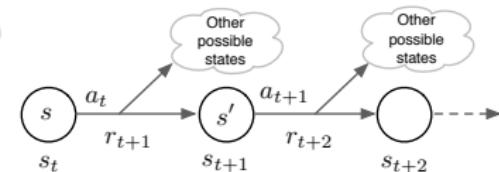
- $E[r_{t+1}] = \sum_{s'} P_{ss'}^a \rho(s, a, s')$
- Linearity of expectation operator:  $E[X + Y] = E[X] + E[Y]$

EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
**Data**  
**Classification**  
**Margins**  
**Primal**  
**Lagrange**  
**Dual**  
**Soft margin**  
**Kernel**  
**Application**  
**RL**  
**Markov**  
**Q-function**  
**Bellman**  
**Optimality**  
**Value iteration**  
**Convergence**  
**Optimal policy**  
**Q-learning**  
**Explore/exploit**  
**Implementation**

$$E[X] = E[E[X|Y]] \quad \text{▶ View}$$

$$= \sum_y E[X|Y=y] \cdot P(Y=y)$$



$$X \leftarrow \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \quad (\text{Condition } s_t = s \text{ is omitted})$$

$$Y \leftarrow s_{t+1}, \quad y \leftarrow s', \quad P(Y=y) \leftarrow P_{ss'}^a$$

$$E^\pi \left[ \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \right] = E^\pi \left[ E^\pi \left[ \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_{t+1} \right] \right]$$

$$= \sum_{s'} \gamma \underbrace{E^\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_{t+1} = s' \right]}_{\text{This is } Q^\pi(s', a') \text{ by definition}} P_{ss'}^a$$

$$= \sum_{s'} P_{ss'}^a \gamma Q^\pi(s', a') \quad (\text{Note: } a' \text{ is action taken at } s')$$

**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

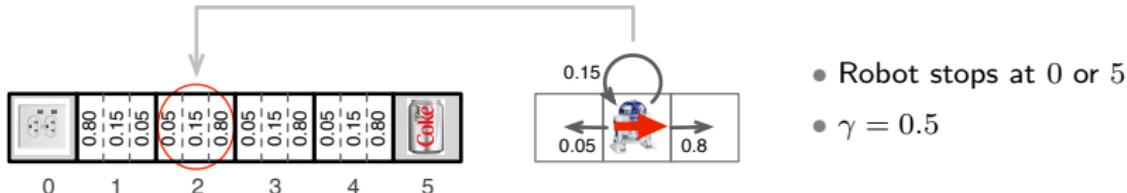
Explore/exploit

Implementation

## Bellman Equation (recursive relationship between $Q^\pi(s, a)$ and $Q^\pi(s', a')$ )

$$\begin{aligned}
 Q^\pi(s, a) &= E^\pi [r_{t+1}] + E^\pi \left[ \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_t = s \right] \\
 &= \sum_{s'} P_{ss'}^a \rho(s, a, s') + \sum_{s'} P_{ss'}^a \gamma Q^\pi(s', a') \\
 &= \sum_{s'} P_{ss'}^a \left( \rho(s, a, s') + \gamma Q^\pi(s', a') \right)
 \end{aligned}$$

Under a given policy  $\pi$ , value of taking  $a$  at  $s$  must equal to expected reward of transitioning into next state  $s'$ , i.e.,  $\rho(s, a, s')$ , plus discounted expected value of taking  $a' = \pi(s')$  at  $s'$



Reward function:

$$\rho(s, a, s') = \begin{cases} 5 & \text{if } s \neq 5 \text{ and } s' = 5 \\ 1 & \text{if } s \neq 0 \text{ and } s' = 0 \\ 0 & \text{otherwise} \end{cases}$$

Policy:

$$\begin{aligned} \pi(1) &= -1 \\ \pi(s) &= 1 \quad \text{if } s = 2, 3, 4 \\ \pi(s) &= \pm 1 \quad \text{if } s = 0, 5 \end{aligned}$$

- Determine the value of  $Q^\pi(2, 1)$

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

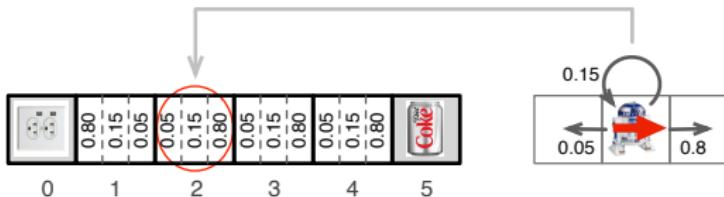
Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation



- Robot stops at 0 or 5
- $\gamma = 0.5$

$$\begin{array}{ll} f(1, -1, 0) = 0.80 \equiv P_{10}^{-1} & f(2, 1, 1) = 0.05 \equiv P_{21}^1 \\ f(1, -1, 1) = 0.15 \equiv P_{11}^{-1} & f(2, 1, 2) = 0.15 \equiv P_{22}^1 \\ f(1, -1, 2) = 0.05 \equiv P_{12}^{-1} & f(2, 1, 3) = 0.80 \equiv P_{23}^1 \end{array}$$

$$\begin{array}{ll} f(3, 1, 2) = 0.05 \equiv P_{32}^1 & f(4, 1, 3) = 0.05 \equiv P_{43}^1 \\ f(3, 1, 3) = 0.15 \equiv P_{33}^1 & f(4, 1, 4) = 0.15 \equiv P_{44}^1 \\ f(3, 1, 4) = 0.80 \equiv P_{34}^1 & f(4, 1, 5) = 0.80 \equiv P_{45}^1 \\ f(0, \pm 1, 0) = 1 \equiv P_{00}^{\pm 1} & f(5, \pm 1, 5) = 1 \equiv P_{55}^{\pm 1} \end{array}$$

$$\begin{array}{lll} \rho(0, 1, 0) = 0 & \rho(0, -1, 0) = 0 & \rho(1, 1, 2) = 0 \\ \rho(1, -1, 0) = 1 & \rho(2, -1, 1) = 0 & \rho(2, 1, 2) = 0 \\ \rho(3, -1, 2) = 0 & \rho(3, 1, 4) = 0 & \rho(4, 1, 5) = 5 \\ \rho(4, -1, 3) = 0 & \rho(5, 1, 5) = 0 & \rho(5, -1, 5) = 0 \end{array}$$

**Plan****Introduction****Outcomes****Math****SVM**  
DataClassification  
Margins

Primal

Lagrange

Dual

Soft margin  
Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

## Applying the Bellman Equation yields

$$\begin{aligned}
 Q^\pi(2, 1) &= \sum_{s'} P_{ss'}^a \left( \rho(s, a, s') + \gamma Q^\pi(s', a') \right) \\
 &= \sum_{j=1}^3 P_{2j}^1 \left( \rho(2, 1, j) + \gamma Q^\pi(j, \pi(j)) \right) \\
 &= P_{21}^1 \left( \rho(2, 1, 1) + \gamma Q^\pi(1, -1) \right) \quad (\text{Note: } j = 1 \text{ and } \pi(1) = -1) \\
 &\quad + P_{22}^1 \left( \rho(2, 1, 2) + \gamma Q^\pi(2, 1) \right) \quad (\text{Note: } j = 2 \text{ and } \pi(2) = 1) \\
 &\quad + P_{23}^1 \left( \rho(2, 1, 3) + \gamma Q^\pi(3, 1) \right) \quad (\text{Note: } j = 3 \text{ and } \pi(3) = 1) \\
 &= 0.05 \cdot \left( 0 + 0.5 Q^\pi(1, -1) \right) \\
 &\quad + 0.15 \cdot \left( 0 + 0.5 Q^\pi(2, 1) \right) \\
 &\quad + 0.80 \cdot \left( 0 + 0.5 Q^\pi(3, 1) \right) \\
 &= 0.025 Q^\pi(1, -1) + 0.075 Q^\pi(2, 1) + 0.4 Q^\pi(3, 1)
 \end{aligned}$$

Expressions for other  $Q$ -function terms are derived in supplementary notes

$$\left\{ \begin{array}{l} Q^\pi(2, 1) = 0.025Q^\pi(1, -1) + 0.075Q^\pi(2, 1) + 0.4Q^\pi(3, 1) \\ Q^\pi(1, -1) = 0.8 + 0.075Q^\pi(1, -1) + 0.025Q^\pi(2, 1) \\ Q^\pi(3, 1) = 0.025Q^\pi(2, 1) + 0.075Q^\pi(3, 1) + 0.4Q^\pi(4, 1) \\ Q^\pi(4, 1) = 0.025Q^\pi(3, 1) + 0.075Q^\pi(4, 1) + 4 \end{array} \right.$$

Solving this set of equations yields

$$Q^\pi(1, -1) = 0.888$$

$$Q^\pi(2, 1) = 0.852$$

$$Q^\pi(3, 1) = 1.915$$

$$Q^\pi(4, 1) = 4.376$$

Hence,  $Q(2, 1) = 0.852$ .

Details in supplementary notes

**Plan****Introduction****Outcomes****Math****SVM**

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

- For a given task, number of possible policies is finite
- $Q$ -function can be used to compare policies
- The policy whose  $Q$ -function values are greater or equal to that of all other policies is called an **optimal policy**

This  $Q$ -function is called the **optimal  $Q$ -function**

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$$

- Any function  $\pi^*$  that selects at  $s$  the **action  $a$  with optimal  $Q^*(s, a)$**  is an optimal policy

$$\pi^*(s) \in \arg \max_a Q^*(s, a)$$

▶ View

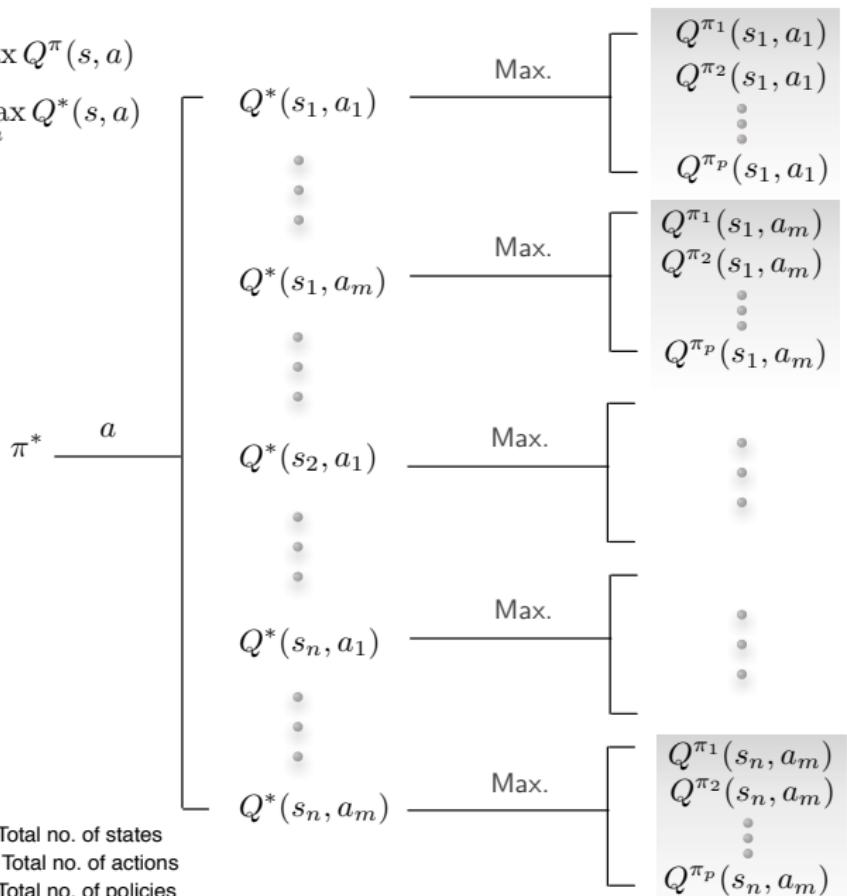
- Values of  $Q^*(s, a)$  are unique
- Possibly more than one  $\pi^*$  for the same  $Q^*(s, a)$  values

EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
 Data  
 Classification  
 Margins  
 Primal  
 Lagrange  
 Dual  
 Soft margin  
 Kernel  
 Application  
  
**RL**  
 Markov  
 Q-function  
**Bellman**  
 Optimality  
 Value iteration  
 Convergence  
 Optimal policy  
 Q-learning  
 Explore/exploit  
 Implementation

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$$

$$\pi^*(s) \in \arg \max_a Q^*(s, a)$$



$Q^*(s, a)$  satisfies Bellman Equation:

$$Q^*(s, a) = \sum_{s'} P_{ss'}^a \left( \rho(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right)$$

$Q^*(s, a)$  is the sum of

1. Reward in current state  $s$  for the chosen action  $a$ , and
2. Discounted value of  $Q$ -function for the “best” action available in the successor state  $s'$

EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

$$Q^*(s, a) = \sum_{s'} P_{ss'}^a \left( \rho(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right)$$

We can determine  $\pi^*$  recursively by picking at  $s'$  the action that yields maximum  $Q^*$  for  $s'$

## Principle of Optimality

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Use Dynamic Programming to solve for  $Q^*$  to get  $\pi^*$

- Value iteration
- Policy iteration

**Input:** state-transition probability  $f$   
 reward function  $\rho$   
 discount factor  $\gamma$

Initialize  $Q$ -function, e.g.,  $Q_0 \leftarrow 0$

Repeat for each  $l$

$$Q \leftarrow Q_l$$

For every  $(s, a)$  do

$$Q(s, a) \leftarrow \sum_{s'} P_{ss'}^a \left( \rho(s, a, s') + \gamma \max_{a'} Q(s', a') \right)$$

End for-loop

$$Q_{l+1} \leftarrow Q$$

Until  $Q_{l+1} = Q_l$

**Output:**  $Q^* = Q_l$

---

In practice, stop when  $|Q_{l+1} - Q_l|$  is below pre-defined threshold

EE5904  
ME5404  
Part II

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

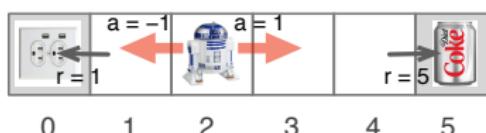
Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation



$$S = \{0, 1, 2, 3, 4, 5\}$$

$$A = \{-1, 1\}$$

$$\gamma = 0.5$$

$$\bar{f}(s, a) = \begin{cases} s + a & \text{if } 1 \leq s \leq 4 \\ s & \text{if } s = 0 \text{ or } s = 5 \end{cases} \quad \rho(s, a, s') = \begin{cases} 5 & \text{if } s \neq 5 \text{ and } s' = 5 \\ 1 & \text{if } s \neq 0 \text{ and } s' = 0 \\ 0 & \text{otherwise} \end{cases}$$

State	0	1	2	3	4	5
$Q_0$	0.0 0.0	0.000 0.000	0.000 0.000	0.000 0.000	0.000 0.000	0.0 0.0
$Q_1$	0.0 0.0	1.000 0.000	0.500 0.000	0.250 0.000	0.125 5.000	0.0 0.0
$Q_2$	0.0 0.0	1.000 0.250	0.500 0.125	0.250 0.250	1.250 5.000	0.0 0.0
$Q_3$	0.0 0.0	1.000 0.250	0.500 1.250	0.625 2.500	1.250 5.000	0.0 0.0
$Q_4$	0.0 0.0	1.000 0.625	0.500 1.250	0.625 2.500	1.250 5.000	0.0 0.0
$Q_5$	0.0 0.0	1.000 0.625	0.500 1.250	0.625 2.500	1.250 5.000	0.0 0.0

Note:  $Q^* = Q_5$

Expression " $m|n$ " indicates  $Q = m$  when  $a = -1$  and  $Q = n$  when  $a = 1$

Detailed calculations are illustrated in supplementary notes

## Value iteration: Example with non-deterministic transitions

**EE5904  
ME5404  
Part II**

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
**Data**

**Classification**

**Margins**

**Primal**

**Lagrange**

**Dual**

**Soft margin**

**Kernel**

**Application**

**RL**

**Markov**

**Q-function**

**Bellman**

**Optimality**

**Value iteration**

**Convergence**

**Optimal policy**

**Q-learning**

**Explore/exploit**

**Implementation**

$(s, a)$	$f(s, a, 0)$	$f(s, a, 1)$	$f(s, a, 2)$	$f(s, a, 3)$	$f(s, a, 4)$	$f(s, a, 5)$
$(0, -1)$	1	0	0	0	0	0
$(1, -1)$	0.8	0.15	0.05	0	0	0
$(2, -1)$	0	0.8	0.15	0.05	0	0
$(3, -1)$	0	0	0.8	0.15	0.05	0
$(4, -1)$	0	0	0	0.8	0.15	0.05
$(5, -1)$	1	0	0	0	0	1
$(0, 1)$	1	0	0	0	0	0
$(1, 1)$	0.05	0.15	0.8	0	0	0
$(2, 1)$	0	0.05	0.15	0.8	0	0
$(3, 1)$	0	0	0.05	0.15	0.8	0
$(4, 1)$	0	0	0	0.05	0.15	0.8
$(5, 1)$	0	0	0	0	0	1

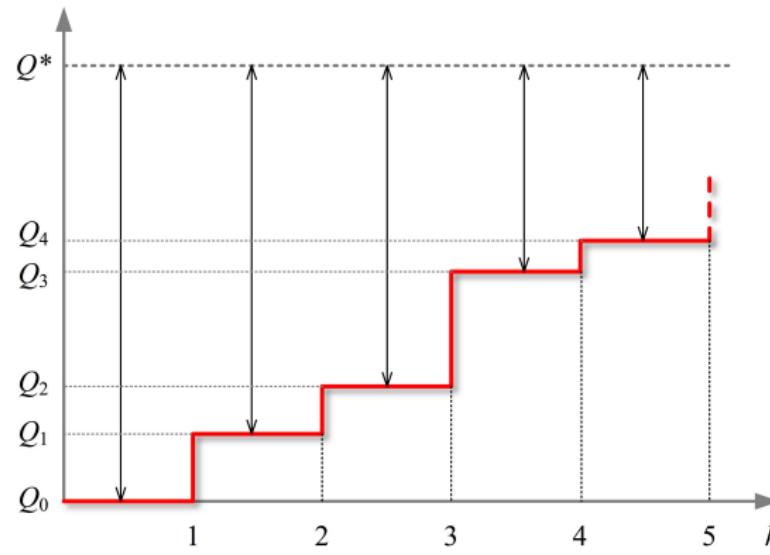
State	0	1	2	3	4	5
$Q_0$	0.0 0.0	0.000 0.000	0.000 0.000	0.000 0.000	0.000 0.000	0.0 0.0
$Q_1$	0.0 0.0	0.800 0.110	0.320 0.044	0.128 0.018	0.301 4.026	0.0 0.0
$Q_2$	0.0 0.0	8.868 0.243	0.374 0.101	0.260 1.639	1.208 4.343	0.0 0.0
$Q_3$	0.0 0.0	0.874 0.265	0.419 0.709	0.515 1.878	1.327 4.373	0.0 0.0
$Q_4$	0.0 0.0	0.883 0.400	0.453 0.826	0.581 1.911	1.342 4.376	0.0 0.0
...	...	...	...	...	...	...
$Q_{12}$	0.0 0.0	0.888 0.458	0.467 0.852	0.594 0.1915	1.344 4.376	0.0 0.0
...	...	...	...	...	...	...
$Q_{22}$	0.0 0.0	0.888 0.458	0.467 0.852	0.594 1.915	1.344 4.376	0.0 0.0

Note:  $Q^* = Q_{22}$

Detailed calculations are illustrated in supplementary notes

Proof of convergence takes two steps:

1. Show values of  $Q$ -function are **bounded** from one iteration to the next
2. Show difference between  $Q_k(s, a)$  and  $Q^*(s, a)$  **decreases** as  $k$  increases



**Plan****Introduction**  
**Outcomes****Math****SVM**  
Data**Classification**  
Margins  
Primal**Lagrange**  
Dual  
Soft margin  
Kernel  
Application**RL**  
**Markov**  
**Q-function**  
**Bellman**  
**Optimality**  
**Value iteration****Convergence**  
**Optimal policy**  
**Q-learning**  
**Explore/exploit**  
**Implementation**

Consider  $Q_k(s, a)$  at iteration  $k$

If  $Q_k(s, a) \neq Q^*(s, a)$ , then express maximum difference as  $\delta_k$

$$\delta_k = \max_{s, a} |Q^*(s, a) - Q_k(s, a)|, \text{ for all } s \text{ and } a$$

Removing maximum operator yields

$$|Q^*(s, a) - Q_k(s, a)| \leq \delta_k$$

That is

$$-\delta_k \leq Q^*(s, a) - Q_k(s, a) \leq \delta_k$$

$$\Rightarrow -Q^*(s, a) - \delta_k \leq -Q_k(s, a) \leq -Q^*(s, a) + \delta_k$$

$$\Rightarrow Q^*(s, a) + \delta_k \geq Q_k(s, a) \geq Q^*(s, a) - \delta_k$$

$$\Rightarrow Q^*(s, a) - \delta_k \leq Q_k(s, a) \leq Q^*(s, a) + \delta_k$$

Consider iteration  $(k + 1)$ . The update rule is:

$$Q_{k+1}(s, a) = \sum_{s'} P_{ss'}^a \left( \rho(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right)$$

Since  $Q_k(s', a') \leq Q^*(s', a') + \delta_k$ , we have

$$\begin{aligned} Q_{k+1}(s, a) &= \sum_{s'} P_{ss'}^a \left( \rho(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right) \\ &\leq \sum_{s'} P_{ss'}^a \left( \rho(s, a, s') + \gamma \left( \max_{a'} Q^*(s', a') + \delta_k \right) \right) \\ &= \sum_{s'} P_{ss'}^a \left( \rho(s, a, s') + \gamma \max_{a'} Q^*(s', a') + \gamma \delta_k \right) \\ &= \underbrace{\sum_{s'} P_{ss'}^a \left( \rho(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right)}_{\text{This is } Q^*(s, a)} + \gamma \delta_k \\ &= Q^*(s, a) + \gamma \delta_k \end{aligned}$$

that is

$$Q_{k+1}(s, a) \leq Q^*(s, a) + \gamma \delta_k$$

Since  $Q^*(s, a)$  is finite,  $Q_{k+1}(s, a)$  is bounded if  $\delta_k$  is bounded. But initially  $Q_0$  is set to zero, and  $\delta_k$  is the maximum absolute difference between  $Q$  and  $Q^*$  in each iteration, so  $Q_{k+1}$  is bounded

EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**  
**Math**  
**SVM**  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application  
  
**RL**  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

From Step 1:  $Q_k(s', a') \geq Q^*(s', a') - \delta_k$ . So

$$\begin{aligned}
Q_{k+1}(s, a) &= \sum_{s'} P_{ss'}^a \left( \rho(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right) \\
&\geq \sum_{s'} P_{ss'}^a \left( \rho(s, a, s') + \gamma \max_{a'} \left( Q^*(s', a') - \delta_k \right) \right) \\
&= \underbrace{\sum_{s'} P_{ss'}^a \left( \rho(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right)}_{\text{This is } Q^*(s, a)} - \gamma \delta_k \\
&= Q^*(s, a) - \gamma \delta_k
\end{aligned}$$

that is

$$Q^*(s, a) - \gamma \delta_k \leq Q_{k+1}(s, a)$$

or

$$Q^*(s, a) - Q_{k+1}(s, a) \leq \gamma \delta_k$$

Therefore

$$\delta_{k+1} \equiv \max_{s,a} |Q^*(s, a) - Q_{k+1}(s, a)| \leq \max_{s,a} |\gamma \delta_k| = \gamma \delta_k$$

i.e.,

$$\delta_{k+1} \leq \gamma \delta_k$$

Since  $0 \leq \gamma < 1$ , error measure  $\delta_k$  decreases to 0 as  $k$  goes to infinity

$$\delta_k = \max_{s,a} |Q^*(s,a) - Q_k(s,a)|$$

1.  $Q_{k+1}$  is bounded because

$$Q_{k+1}(s,a) \leq Q^*(s,a) + \gamma \delta_k$$

2. Error measure  $\delta_k \rightarrow 0$  as  $k \rightarrow \infty$  because

$$\delta_{k+1} \leq \gamma \delta_k$$

Hence, value iteration algorithm converges to  $Q^*(s,a)$

[Plan](#)[Introduction](#)[Outcomes](#)[Math](#)[SVM](#)[Data](#)[Classification](#)[Margins](#)[Primal](#)[Lagrange](#)[Dual](#)[Soft margin](#)[Kernel](#)[Application](#)[RL](#)[Markov](#)[Q-function](#)[Bellman](#)[Optimality](#)[Value iteration](#)[Convergence](#)[Optimal policy](#)[Q-learning](#)[Explore/exploit](#)[Implementation](#)

Values of  $Q$ -function are optimal if they are greater or equal to that of all other policies for all  $(s, a)$  pairs, i.e.,

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$$

### Greedy policy

At each  $s$ , select  $a$  that yields the largest value for the  $Q$ -function. When multiple choices are available, such  $a$  can be picked randomly

A greedy policy is optimal, i.e.,

$$\pi^*(s) \in \arg \max_a Q^*(s, a)$$

State	0	1	2	3	4	5
$Q_5$	0.0 0.0	1.0 0.625	0.5 1.25	0.625 2.5	1.25 5.0	0.0 0.0
$\pi^*$		-1	1	1	1	

- Optimal values are  $Q^* = Q_5$
- Take state 1 for example:
  - Optimal values are:  $Q^*(1, -1) = 1$  and  $Q^*(1, 1) = 0.625$
  - So  $\pi^*(1) = -1$
- Applying same reasoning on other states yields

$$\pi^*(1) = -1, \quad \pi^*(2) = 1, \quad \pi^*(3) = 1, \quad \pi^*(4) = 1$$



EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**

**Math**

**SVM**  
Data  
Classification  
Margins  
Primal  
Lagrange  
Dual  
Soft margin  
Kernel  
Application

**RL**  
Markov  
Q-function  
Bellman  
Optimality  
Value iteration  
Convergence  
Optimal policy  
Q-learning  
Explore/exploit  
Implementation

State	0	1	2	3	4	5
$Q_{22}$	0 0	0.888 0.458	0.467 0.852	0.594 1.915	1.344 4.376	0 0
$\pi^*$		-1	1	1	1	

- Optimal values are  $Q^* = Q_{22}$
- Optimal policy is

$$\pi^*(1) = -1, \quad \pi^*(2) = 1, \quad \pi^*(3) = 1, \quad \pi^*(4) = 1$$



Plan  
Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

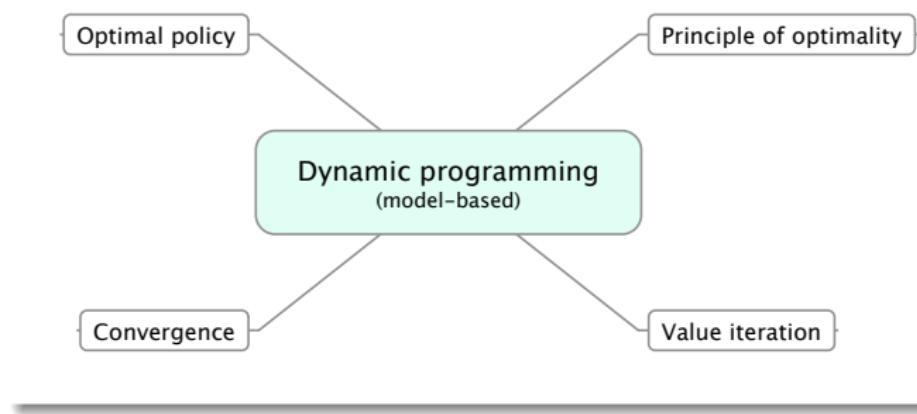
Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation



- Dynamic programming requires state-transition model
- What if no such model is available?



*Q*-learning

- No need for state-transition model
- Uses solely reward received from **observed** state transitions
- Iterative update rule

$$Q_{k+1}(s_k, a_k)$$

$$= Q_k(s_k, a_k) + \alpha_k \left( \underbrace{r_{k+1} + \gamma \max_{a'} Q_k(s_{k+1}, a') - Q_k(s_k, a_k)}_{\text{Estimate of } Q^*(s_k, a_k)} \right)$$

$\alpha_k \in (0, 1]$  is the learning rate

**Temporal difference:** Difference between estimate of  $Q^*(s_k, a_k)$  and current estimate  $Q_k(s_k, a_k)$

$Q$ -learning update rule converges to  $Q^*$  as  $k \rightarrow \infty$  if

$$(1) \sum_{k=0}^{\infty} \alpha_k^2 < \infty \quad \text{and} \quad \sum_{k=0}^{\infty} \alpha_k \rightarrow \infty$$

(2) All  $(s, a)$  pairs are (asymptotically) visited infinitely often

- (1) can be met by setting  $\alpha_k = 1/k$  with finite  $\alpha_0$
- (2) can be satisfied if agent has non-zero probability of selecting any available action in every state it visits. This is called **exploration**

Proof can be found in: Watkins, C. J. C. H. and Dayan, P.,  $Q$ -learning, Machine Learning, 8:279-292, 1992

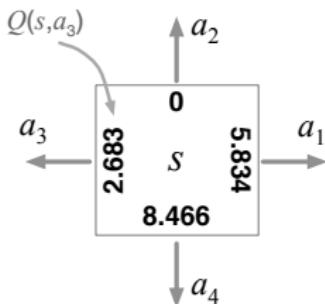
$$Q_{k+1}(s_k, a_k) = Q_k(s_k, a_k) + \alpha_k \left( r_{k+1} + \gamma \max_{a'} Q_k(s_{k+1}, a') - Q_k(s_k, a_k) \right)$$

**Exploitation:** Use greedy policy to select currently known best action

$$a_{k+1} = \max_{a'} Q_k(s_{k+1}, a')$$

**Exploration:** Try action other than currently known best action

$$a_{k+1} \neq \max_{a'} Q_k(s_{k+1}, a')$$



Exploitation: Take  $a_4$   
 Exploration: Take  $a_1, a_2, \text{ or } a_3$

- Exploration is essential for learning
- If we do not try new things, then we will never develop new capabilities and skills
- However, if we always explore and do not utilize what we have learned, then we will most likely not be able to achieve anything significant

Sometimes too much exploration can be dangerous

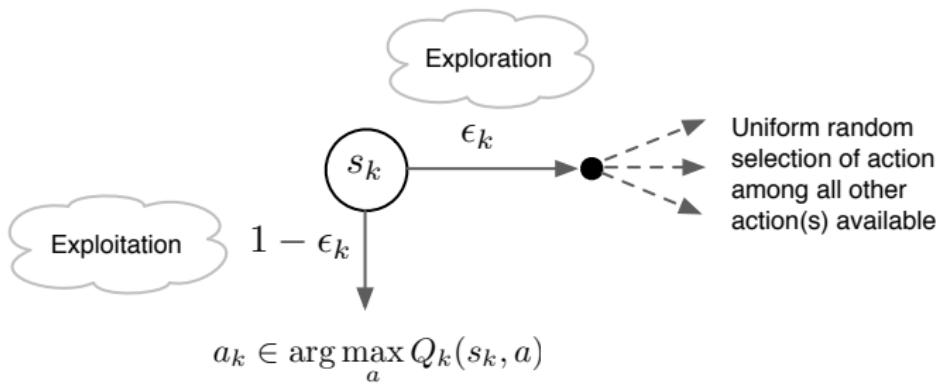


There must be **balance** between exploration and exploitation

Photo from <http://www.artlebedev.com/everything/vilcus/>

## Balancing exploration-exploitation trade-off: $\epsilon$ -greedy exploration

$$a_k = \begin{cases} a \in \arg \max_{\hat{a}} Q_k(s_k, \hat{a}) & \text{with probability } 1 - \epsilon_k \\ \text{action uniformly randomly selected from all other actions available at state } s_k & \text{with probability } \epsilon_k \end{cases}$$



Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

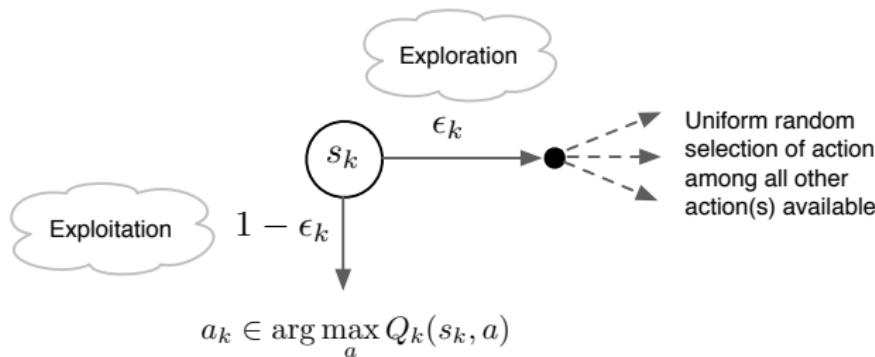
Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation



- $\epsilon$  is kept small so that agent will focus on task at hand most of the time, and will explore only occasionally
- Reduce exploration as learning continues by setting  $\epsilon_k = \frac{1}{k}$

Input: Discount factor  $\gamma$ ; exploration probability  $\epsilon_k$ ; learning rate  $\alpha_k$

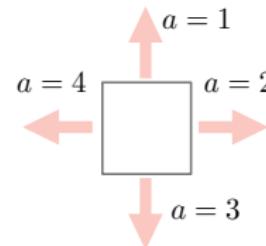
- Initialize  $Q$ -function, e.g.,  $Q_0 \leftarrow 0$
- Determine the initial state  $s_0$
- For time step  $k$ , select action  $a_k$  according to:

$$a_k = \begin{cases} a \in \arg \max_{\hat{a}} Q_k(s_k, \hat{a}) & \text{with probability } 1 - \epsilon_k \\ \text{an action uniformly randomly selected from all other actions available at state } s_k & \text{with probability } \epsilon_k \end{cases}$$

- Apply action  $a_k$ , receive reward  $r_{k+1}$ , then observe next state  $s_{k+1}$
  - Update  $Q$ -function with:
- $$Q_{k+1}(s_k, a_k) = Q_k(s_k, a_k) + \alpha_k \left( r_{k+1} + \gamma \max_{a'} Q_k(s_{k+1}, a') - Q_k(s_k, a_k) \right)$$
- Set  $k = k + 1$  and repeat for-loop for the next time step

Note: This basic algorithm may be (very) slow to converge. Some modified forms of  $\epsilon_k$  and  $\alpha$  are often used in practice to speed up the learning process.

0	1	2
3	4	5
6	7	8



$$\rho(s, a, s') = \begin{cases} 1 & \text{if } s \neq 8 \text{ and } s' = 8 \\ 0 & \text{otherwise} \end{cases}$$

$$\gamma = 0.9, \quad a_k = 1/k, \quad a_0 = 1$$

- Learning consists of series of trials
- Starting at initial state robot makes transitions according to  $\epsilon$ -greedy exploration and stops when reaching  $s = 8$
- Will **not** actually simulate probabilistic selection of action in this example
- Will simply assume sequence of actions for a trial (as if they were observed to have occurred) so as to carry out calculation of values for  $Q$ -function

Plan

Introduction

Outcomes

Math

SVM

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

RL

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

- For record keeping, define function  $N(s, a)$  which counts accumulative number of times (over a set of trials)  $a$  has been taken at  $s$
- Two types of diagrams
  - One showing values of  $N(s, a)$  at end of a trial
  - The other showing values of  $Q$ -function at end of a trial
- Initial values of  $N(s, a)$  and  $Q$ -function:

0	0	0
0	0	0
1	0	0
0	0	0
0	0	0

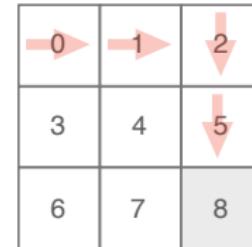
 $N(s, a)$ 

0	0	0
0	0	0
0	1	0
0	0	0
2	0	0

Values of  $Q$ -function

## Trial 1:

- Assume action sequence as shown in diagram
- $\alpha_0 = 1$
- $\gamma = 0.9$



Sample calculations:

$$\begin{aligned}
 Q_1(0, 2) &= Q_0(0, 2) + \alpha_0 \left( \rho(0, 2, 1) + \gamma \max_{a'} Q_0(1, a') - Q_0(0, 2) \right) \\
 &= 0 + 1 \cdot (0 + 0.9 \cdot 0 - 0) \\
 &= 0
 \end{aligned}$$

...

$$\begin{aligned}
 Q_4(5, 3) &= Q_3(5, 3) + \alpha_3 \left( \rho(5, 3, 8) + \gamma \max_{a'} Q_3(8, a') - Q_3(5, 3) \right) \\
 &= 0 + \frac{1}{3} (1 + 0.9 \cdot 0 - 0) \\
 &= 0.333
 \end{aligned}$$

Detailed calculations are shown in supplementary notes

EE5904  
ME5404  
Part II

**Plan**  
**Introduction**  
**Outcomes**

**Math**

**SVM**

Data

Classification

Margins

Primal

Lagrange

Dual

Soft margin

Kernel

Application

**RL**

Markov

Q-function

Bellman

Optimality

Value iteration

Convergence

Optimal policy

Q-learning

Explore/exploit

Implementation

0	0	0	0	0	0	0
0	0	<b>1</b>	0	1	0	2
0	0	0	0	<b>1</b>	0	0
0	0	0	0	0	0	0
0	3	00	4	00	5	0
0	0	0	0	<b>1</b>	0	0
0	0	0	0	0	0	0
0	6	00	7	00	8	0
0	0	0	0	0	0	0

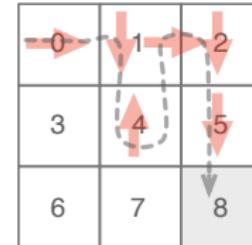
(a)  $N(s, a)$

0	0	0	0	0	0	0
0	0	00	1	00	2	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	3	00	4	00	5	0
0	0	0	<b>0.333</b>	0	0	0
0	0	0	0	0	0	0
0	6	00	7	00	8	0
0	0	0	0	0	0	0

(b)  $Q$ -function

## Trial 2:

- Assume action sequence as shown in diagram
- $\alpha_0 = 1$
- $\gamma = 0.9$



Sample calculations:

$$\begin{aligned}
 Q_1(0, 2) &= Q_0(0, 2) + \alpha_0 \left( \rho(0, 2, 1) + \gamma \max_{a'} Q_0(1, a') - Q_0(0, 2) \right) \\
 &= 0 + 1 \cdot (0 + 0.9 \cdot 0 - 0) \\
 &= 0
 \end{aligned}$$

...

...

Detailed calculations are shown in supplementary notes

**Plan****Introduction****Outcomes****Math****SVM****Data****Classification****Margins****Primal****Lagrange****Dual****Soft margin****Kernel****Application****RL****Markov****Q-function****Bellman****Optimality****Value iteration****Convergence****Optimal policy****Q-learning****Explore/exploit****Implementation**

## Values of $N(s, a)$ and $Q$ -function at end of Trial 2

0	0	0
0	<b>2</b>	0
1	2	0
2	0	0
0	<b>1</b>	<b>2</b>
0	1	0
0	3	0
0	4	0
0	5	0
0	0	<b>2</b>
0	0	0
0	6	0
0	7	0
0	8	0
0	0	0

(a)  $N(s, a)$ 

0	0	0
0	0	0
1	0	2
0	0	<b>0.075</b>
0	0	0
0	3	0
0	4	0
0	5	0
0	0	<b>0.466</b>
0	0	0
0	6	0
0	7	0
0	8	0
0	0	0

(b)  $Q$ -function

Iterative process repeats until values of  $Q$ -function converge to optimal values, upon which an optimal policy can be obtained

EE5904  
ME5404  
Part II

**Plan****Introduction****Outcomes****Math****SVM**

## Data

## Classification

## Margins

## Primal

## Lagrange

## Dual

## Soft margin

## Kernel

## Application

**RL**

## Markov

## Q-function

## Bellman

## Optimality

## Value iteration

## Convergence

## Optimal policy

## Q-learning

## Explore/exploit

## Implementation

