

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/348003841>

Buku Ajar AI, Machine Learning & Deep Learning

Book · July 2019

CITATIONS

4

READS

27,927

2 authors:



Imam Cholissodin

Brawijaya University

112 PUBLICATIONS 393 CITATIONS

[SEE PROFILE](#)



Arief Andy Soebroto

Brawijaya University

34 PUBLICATIONS 72 CITATIONS

[SEE PROFILE](#)



Link: <http://bit.ly/3piOnnU>

AI, MACHINE LEARNING & DEEP LEARNING (Teori & Implementasi)

“from Basic Science to High Scientific Solution for Any Problem”

Versi 1.01

Oleh:
Imam Cholissodin
Sutrisno
Arief Andy Soebroto
Uswatun Hasanah
Yessica Inggit Febiola

PENGANTAR

Buku ini merupakan uraian dari pemahaman empat teknik dasar pemecahan masalah dalam Artificial Intelligence, Machine Learning dan Deep Learning (AI, ML & DL), yaitu: Searching, Reasoning, Planning dan Learning. Setiap teknik memiliki banyak metode yang dapat digunakan untuk menyelesaikan kasus tertentu. Oleh karena itu, penggunaan metode ini harus disesuaikan dengan permasalahan apa yang akan diselesaikan. Dalam perkuliahan nanti akan diberikan beberapa tugas dalam bentuk presentasi, pembuatan ilustrasi dan penyelesaian case study dengan mengimplementasikan beberapa teknik serta metode untuk membantu mempermudah pemahaman serta memberikan gambaran bagaimana memilih teknik dan metode yang tepat untuk digunakan sebagai general problem solving.

Imam Cholissodin

Dosen Pengampu MK *Stream Data Science* FILKOM UB

2012-2020

Kata Pengantar

Alhamdulillahhi robbil alamin, puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya dengan terselesaikannya penulisan buku ini untuk menunjang perkuliahan baik online untuk cegah Covid-19 maupun offline, serta mengakselerasi pencapaian kompetensi agar lebih optimal dalam kurikulum Merdeka Belajar, dengan judul "**AI, Machine Learning & Deep Learning**". Buku ini merupakan uraian dari pemahaman yang membahas empat teknik dasar pemecahan masalah dalam AI, ML dan Deep Learning, yaitu: Searching, Reasoning, Planning dan Learning serta berbagai macam Algoritma. Setiap teknik memiliki banyak metode yang dapat digunakan untuk menyelesaikan kasus tertentu. Oleh karena itu, penggunaan teknik dan metode ini harus disesuaikan dengan permasalahan apa yang akan diselesaikan. Dalam perkuliahan akan diberikan beberapa tugas dalam bentuk presentasi, pembuatan ilustrasi dan penyelesaian case study dengan mengimplementasikan beberapa teknik serta metode untuk membantu mempermudah pemahaman serta memberikan gambaran bagaimana memilih teknik dan metode yang tepat untuk digunakan sebagai general problem solving.

Penulis mengucapkan terimakasih yang sebesar-besarnya kepada beberapa pihak terkait yang telah membantu baik secara langsung maupun tidak langsung dalam penyelesaian buku ini:

1. Bapak Sutrisno, Bapak Arief Andy Soebroto, dan Para penulis artikel Kecerdasan Buatan atau AI, Machine Learning & Deep Learning di forum, web, blog dan buku yang menjadi referensi untuk memberikan masukan yang sangat berharga untuk perbaikan dan penyelesaian buku ini.
2. Mbak Uswatun Hasanah dan Mbak Yessica Inggris Febiola, yang telah banyak membantu penulisan buku ini. Semoga kontribusinya menjadi ilmu yang barokah dan bermanfaat. Aamiin.:)

Semoga menjadi ilmu yang barokah dan bermanfaat. Aamiin.:). Tidak ada gading yang tak retak, maka penulis memohon kritik dan saran untuk perbaikan dan penyempurnaan buku ini. In Syaa Allah pada edisi berikutnya, kami akan memberikan lebih banyak contoh algoritma pada AI, Machine Learning maupun penerapan yang lebih luas terkait Deep Learning. Selamat membaca buku ini dan semoga bermanfaat.

Malang, 19 Juli 2019 – 29 Desember 2020

Penulis



Daftar Isi

Judul	i
Kata Pengantar.....	ii
Daftar Isi	iii
Daftar Gambar	xi
Daftar Tabel.....	xix
BAB 1 Konsep Dasar AI.....	1
1.1 Apa itu AI.....	1
1.2 Fondasi AI	2
1.3 Sejarah AI	3
1.4 AI Saat Ini.....	4
1.5 AI Masa Depan	10
1.6 Simple Case Study (Logic 1)	11
1.7 Tugas Kelompok	11
BAB 2 Agen Cerdas	12
2.1 Agen dan Lingkungan	12
2.1.1 Agen Cerdas	12
2.1.2 Struktur Agen	19
2.2 Rasionalitas.....	20
2.3 PEAS.....	22
2.3.1 Karakteristik PEAS.....	23
2.3.2 Arsitektur Agen	24
2.3.3 Jenis atau Tipe Agen	24
2.4 Jenis Lingkungan	28
2.5 Jenis Agen	30
2.6 Tugas Kelompok	31
BAB 3 Un-Informed Searching.....	32
3.1 Agen Penyelesaian Problem.....	32
3.2 Jenis Problem atau Masalah.....	33
3.3 Formulasi Problem.....	34

3.4	Algoritma Pencarian Dasar	38
3.4.1	Tree Search	38
3.4.2	Graph Search.....	41
3.5	Strategi Pencarian Uninformed.....	42
3.6	Latihan Individu	46
3.7	Tugas Kelompok	47
BAB 4	Informed Searching.....	48
4.1	Best-first search	48
4.2	Algoritma Min-Max	49
4.3	Greedy best-first search.....	51
4.4	A* search.....	52
4.5	Heuristics	55
4.6	Hill-Climbing Search	57
4.7	Alpha Beta Pruning (Game Search)	58
4.8	Latihan Individu	65
4.9	Tugas Kelompok	67
BAB 5	Constraint Satisfaction Problem	68
5.1	Constraint Satisfaction Problems (CSP)	68
5.2	Pencarian Backtracking untuk CSP	70
5.3	Local search untuk CSPs.....	76
5.4	Latihan Individu	84
5.5	Tugas Kelompok	84
BAB 6	Agen Logika	86
6.1	Agen Logika	86
6.2	Agen Berbasis Pengetahuan	86
6.3	Logika Proposisi.....	98
6.4	Metode Pembuktian	101
6.5	Latihan Individu	103
6.6	Tugas Kelompok	103
BAB 7	Logika Order Pertama (First Order Logic)	104
7.1	Konsep dasar FOL (First Order Logic).....	104

7.2	Sintak dan Semantic FOL	105
7.2.1	Syntax FOL	105
7.2.2	Semantics FOL	106
7.3	Konteks Penggunaan FOL.....	107
7.4	Rekayasa Pengetahuan dengan FOL	109
7.5	Latihan Individu	111
7.6	Tugas Kelompok	112
BAB 8	Inferensi pada FOL	113
8.1	Mengubah FOL inference ke PL inference	113
8.2	Unification	116
8.3	Inference Rule untuk FOL.....	117
8.4	Forward chaining.....	119
8.5	Backward chaining.....	122
8.6	Resolution (The Next)	123
8.7	Tugas Kelompok	124
BAB 9	Logic Programming.....	126
9.1	Logic Programming	126
9.2	Logika Predikat	127
9.3	Bahasa Deklaratif.....	128
9.4	Pemrograman Prolog	128
9.5	Latihan Individu	137
9.6	Tugas Kelompok	137
BAB 10	Ketidakpastian (Uncertainty).....	139
10.1	Uncertainty	139
10.2	Probability	140
10.3	Semantics & Syntax	141
10.4	Inferensi Atau Penalaran	145
10.5	Aturan Bayes (just review).....	146
10.6	Latihan Individu.....	146
10.7	Tugas Kelompok.....	147
BAB 11	Bayesian Network	148

11.1	Syntax & Semantics	148
11.2	Compact conditional distributions.....	152
11.3	Efficient Inference	155
11.4	Latihan Individu.....	159
11.5	Tugas Kelompok.....	160
BAB 12	AI & Machine Learning.....	161
12.1	Review Artificial Intelligence (AI)	161
12.2	Welcome to Machine Learning	162
BAB 13	Support Vector Machine (SVM)	164
13.1	Pengertian SVM.....	164
13.2	SVM Linear	165
13.3	SVM non-linear	167
13.4	Fungsi Kernel Pada SVM	170
	13.4.1 Kernel Linear.....	170
	13.4.2 Kernel Non-Linear.....	171
13.5	Dasar-Dasar SVM.....	182
13.5.1	SVM 2 Kelas (Hyperplane Linier)	182
13.5.2	SVM 2 Kelas (Hyperplane Non-Linier).....	185
13.5.3	SVM dengan > 2 Kelas	190
13.6	Algoritma SVM.....	191
13.6.1	Simplified Sequential Minimal Optimization (SMO)	191
13.6.2	Sequential Training SVM	209
13.7	Klasifikasi SVM Multi Kelas (> 2 Kelas).....	253
13.7.1	One Against All	253
13.7.2	One-Against-One	254
13.7.3	Binary Decision Tree SVM.....	255
13.7.4	Directed Acyclic Graph SVM (DAGSVM)	260
13.8	SVM Untuk Kasus Regresi	263
13.8.1	Tentang Support Vector Regression (SVR)...	263
13.8.2	Komponen SVR (Training dan Testing)	264
13.9	Pengukuran Nilai Performa Klasifikasi	272

13.9.1	Esensi Confusion Matrix (CM)	272
13.9.2	Rumus Evaluasi untuk $n > 2$ Kelas	278
BAB 14	Algoritma Regresi.....	280
14.1	Regresi Linier.....	280
14.2	Regresi Logistik	289
14.3	Pengukuran Nilai Performa Regresi	292
14.3.1	MAE	292
14.3.2	RMSE	292
14.3.3	CC	293
14.3.4	MAPE	293
BAB 15	Machine Learning & Deep Learning	294
15.1	Tradisional Machine Learning vs Transfer Learning	294
15.2	Welcome to Deep Learning	295
15.3	Prinsip Kerja Deep Learning.....	299
15.4	How to Build Core Engine Deep Learning.....	300
15.5	Tugas Kelompok.....	301
BAB 16	Algoritma Convolutional Neural Networks (CNN)	302
16.1	Struktur Algoritma CNN	302
16.2	Studi Kasus: CNN untuk Menyelesaikan Kasus Prediksi maupun Klasifikasi	308
16.2.1	Proses Training.....	310
16.2.2	Proses Testing	323
16.3	Tugas Kelompok.....	331
BAB 17	Prototype Deep Blockchain.....	333
17.1	Pengantar	333
17.2	Studi Kasus: Blockchain Pada Isi Informasi	334
BAB 18	Recurrent Neural Networks (RNN) Native dan Modified Long Short-Term Memory (MLSTM) Berbasis ELM.....	340
18.1	Pengantar	340
18.2	Struktur Algoritma RNN	341
18.3	Studi Kasus: RNN untuk Prediksi Nilai Tukar Dollar ke Rupiah	342

18.3.1	Proses Training	344
18.3.2	Proses Testing	349
18.4	Struktur RNN Berbasis Modified Long Short-Term Memory (MLSTM) Menggunakan ELM	359
18.5	Studi Kasus: RNN berbasis Simplified/ Modified LSTM (MLSTM) menggunakan ELM pada prediksi data Teks untuk Pembuatan Karya Sastra (dengan Extend Long-Short Feature pada Mono Cell Gate)	364
18.6	Tugas Kelompok.....	432
BAB 19	Deep Reinforcement Learning (DRL)	434
19.1	Pengantar	434
19.2	Multi-Agent RL (MARL)	435
19.3	Self-Organizing Map (SOM)	435
19.4	Q-learning dan JST	436
19.5	Studi Kasus: DRL untuk Multi-Agent pada Permainan Pim Pong	438
19.6	Tugas Kelompok.....	443
BAB 20	Algoritma Deep Belief Net (DBN).....	445
20.1	Langkah dan Pembuktian DBN	446
20.2	Studi Kasus: DBN untuk Klasifikasi Data Sederhana 450	
20.2.1	Proses Training.....	450
20.2.2	Proses Testing	459
20.3	Tugas Kelompok.....	459
BAB 21	Topik Lanjut Pada Deep Learning	461
21.1	Pengantar	461
21.2	Improve Autoencoder untuk Fast Mapping Fitur dengan Kernel ELM (KELM) dan ELM Murni.....	462
21.2.1	Autoencoder (AE) vs CAE vs PCA	462
21.2.2	Evaluasi Autoencoder	464
21.2.3	Studi Kasus: Deep ELM Kernel atau non-Kernel dengan Autoencoder.....	465
21.3	Improve Deep Learning dengan Particle Swarm Optimization (PSO)	484

21.3.1	Tentang Improve Deep Learning dengan PSO (Deep PSO).....	484
21.3.1	Evaluasi Deep PSO	486
21.3.2	Studi Kasus: Prediksi Curah Hujan menggunakan Improve Deep Learning dengan Particle Swarm Optimization (PSO).....	486
21.4	Tugas Kelompok.....	488
BAB 22	Project Pilihan	489
22.1	Simulasi Reinforcement Learning untuk Pencarian Kandidat Obat Covid-19	489
22.1.1	Konsep	489
22.1.2	Tampilan Implementasi.....	490
22.1.3	Source Code	491
22.2	Ai Dino, Flappy Bird & Frozen Lake dengan Reinforcement Learning untuk Otomasi Pergerakan.....	497
22.2.1	Konsep	497
22.2.2	Tampilan Implementasi.....	499
22.2.3	Source Code	499
22.3	Generative Adversarial Network (GAN) by Example	502
22.3.1	Konsep	502
22.3.2	Tampilan Implementasi.....	502
22.3.3	Source Code	504
22.4	Prototype Style Transfer dengan CNN Net. (<i>Build from Scratch</i>) like ResNet, VGG, Inception, etc.....	505
22.4.1	Konsep	505
22.4.2	Tampilan Implementasi.....	505
22.4.3	Source Code	506
22.5	Deteksi Otomatis General Fashion (case Study: Jilbab Fashion) dengan Yolo untuk <i>Level Up</i> Market Bidang Ekonomi & Bisnis	513
22.5.1	Konsep	513
22.5.2	Tampilan Implementasi.....	514
22.5.3	Source Code	520
22.6	Prototype Teknik Multimodal Deep Learning untuk Image Captioning	522

22.6.1	Konsep	522
22.6.2	Tampilan Implementasi.....	523
22.6.3	Source Code	524
	Daftar Pustaka	525
	Biografi Penulis.....	535

Daftar Gambar

Gambar 1.1 Tampilan (Rute Optimal)	5
Gambar 1.2 Tampilan pada Permainan Catur	5
Gambar 1.3 Contoh penyelesaian dengan memanfaatkan AI pada Game Catur	6
Gambar 1.4 Contoh penyelesaian dengan memanfaatkan AI pada Game Super Mario	6
Gambar 1.5 Tampilan Permainan Dino.....	7
Gambar 1.6 Penyelesaian dengan memanfaatkan AI	7
Gambar 1.7 Hal yang mencakup pada Computer Vision.....	7
Gambar 1.8 Penggunaan AI untuk Deteksi Wajah	8
Gambar 1.9 Penggunaan AI untuk menghitung Jumlah Kendaraan ...	8
Gambar 1.10 Penggunaan AI untuk menghitung Jumlah Orang	8
Gambar 1.11 Penggunaan Algoritme Optimasi.....	9
Gambar 1.12 Hasil dari memanfaatkan AI	9
Gambar 1.13 Optimasi Penjadwalan Jaga Dokter:	10
Gambar 1.14 Tampilan Permainan	11
Gambar 2.1 Abstraksi dari Model Komputasi dari sebuah Agen	12
Gambar 2.2 Komponen Internal dari Model Agen BDI.....	13
Gambar 2.3 Contoh 1 pada Effectuator dari Human Agen	13
Gambar 2.4 Contoh 2 pada Effectuator dari Human Agen	14
Gambar 2.5 Contoh 3 pada Effectore dari Human Agen	14
Gambar 2.6 Contoh Robot Asimo	14
Gambar 2.7 Gambaran Action pada Vacuum Cleaner	15
Gambar 2.8 Alat Pemanen Pada pada Jaman Dulu	15
Gambar 2.9 Alat Pemanen Padi.....	16
Gambar 2.10 Alat Pemanen Jagung Jaman Dulu.....	16
Gambar 2.11 Alat Pemanen Jagung Jaman Sekarang.....	16
Gambar 2.12 Alat Semprot Manual	17
Gambar 2.13 Alat Semprot Basmi Hama dengan Drone	17
Gambar 2.14 Tanam Padi Jaman Dulu.....	17

Gambar 2.15 Tanam Padi Jaman Modern	18
Gambar 2.16 Agen Robot Alat Pengatur PH.....	18
Gambar 2.17 Agen Robot Pengusir Hama Tikus Jaman Dulu.....	18
Gambar 2.18 Agen Robot Pengusir Hama Tikus Modern.....	19
Gambar 2.19 Agen Software Antarmuka Pengguna Grafis	19
Gambar 2.20 Struktur Agen.....	19
Gambar 2.21 Struktur Simple Reflex Agents	25
Gambar 2.22 Model-Based Reflex Agents	26
Gambar 2.23 Goal-Based Agents	26
Gambar 2.24 Utility-Based Agents	27
Gambar 2.25 Learning Agents	28
Gambar 3.1 Contoh Permasalahan dalam Perjalanan	33
Gambar 3.2 Contoh Jenis Problem pada Vacuum Word	34
Gambar 3.3 Vacuum Cleaner World	36
Gambar 3.4 Permainan 8-Puzzle	36
Gambar 3.5 Permainan *-Queens Problems.....	37
Gambar 3.6 Gambaran Robotic Assembly.....	37
Gambar 3.7 Tree Search	38
Gambar 3.8 Tree Search.....	38
Gambar 3.9 Tree Search.....	39
Gambar 3.10 Search Tree	39
Gambar 3.11 Implementation State and Node.....	41
Gambar 3.12 Gambaran untuk Search Search.....	41
Gambar 3.13 Gambar State Space Graph.....	42
Gambar 3.14 Hasil BFS Traversal Queue.....	43
Gambar 3.15 Hasil DFS Traversal Stack	44
Gambar 4.1 Contoh Heuristic Function	48
Gambar 4.2 Contoh Implementasi Algoritme Min-Max pada Permainan Catur	50
Gambar 4.3 Penyelesaian menggunakan Algoritme Min-Max	51
Gambar 4.4 Contoh untuk Greedy Best-First Search	51
Gambar 4.5 Contoh Penelusuran A*Search	52

Gambar 4.6 Contoh Penyelesaian Algoritme A*Search.....	53
Gambar 4.7 Permainan 8-Puzzel	56
Gambar 4.8 Konsep Hill-Climbing Search.....	57
Gambar 4.9 Contoh Penyelesaian Hill-Climbing Search	58
Gambar 4.10 Contoh Implementasi MIN-MA	59
Gambar 4.11 Contoh Implemetasi Min-Max.....	59
Gambar 4.12 Min-Max untuk Estimasi Parent Node	60
Gambar 4.13 Alpha-Value dari Node Parent \geq Beta Value	61
Gambar 4.14 Alpha-Value dari Node Parent \leq Beta Value	61
Gambar 4.15 Implementasi Mini-Max dengan $\alpha - \beta$ at work.....	64
Gambar 4.16 Contoh Penyelesaian Masalah Tree	64
Gambar 5.1 Peta Mewarnai.....	68
Gambar 5.2 Solusi CSP dalam mewarnai Peta	69
Gambar 5.3 Constraint Graph pada Pewarnaan Peta	69
Gambar 5.4 Backtracking Step 1	70
Gambar 5.5 Backtracking Step 2	71
Gambar 5.6 Backtracking Step 3	71
Gambar 5.7 Backtracking Step 4	71
Gambar 5.8 Gambaran Variabel yang paling Dibatasi.....	72
Gambar 5.9 Gambaran Variabel yang Membatasi.....	72
Gambar 5.10 Gambaran Least Contraining Value.....	73
Gambar 5.11 Forward Checking pada Pewarnaan Peta	73
Gambar 5.12 Forward Checking Step 1.....	73
Gambar 5.13 Forward Checking Step 2.....	73
Gambar 5.14 Forward Checking Step 3.....	74
Gambar 5.15 Forward Checking Step 4.....	74
Gambar 5.16 Pewarnaan Peta pada permasalahan	74
Gambar 5.17 Hasil dari Forward Checking	75
Gambar 5.18 Arc Konsistency Step 1	75
Gambar 5.19 Arc Konsistency Step 2	75
Gambar 5.20 Arc Konsistency Step 3	75

Gambar 5.21 Arc Konsistency Step 4	75
Gambar 5.22 Map Coloring	76
Gambar 5.23 Map Coloring Step 1	77
Gambar 5.24 Map Coloring Step 2	77
Gambar 5.25 Map Coloring Step 3	77
Gambar 5.26 Map Coloring Step 4	78
Gambar 5.27 Map Coloring Step 5	78
Gambar 5.28 Map Coloring Step 6	78
Gambar 5.29 Map Coloring Step 7	79
Gambar 5.30 Map Coloring Step 8	79
Gambar 5.31 Map Coloring Step 9	79
Gambar 5.32 Map Coloring Step 10	80
Gambar 5.33 Map Coloring Step 11	80
Gambar 5.34 Map Coloring Step 12	80
Gambar 5.35 Map Coloring Step 13	81
Gambar 5.36 Map Coloring Step 14	81
Gambar 5.37 Map Coloring Step 15	81
Gambar 5.38 Map Coloring Step 16	82
Gambar 5.39 Map Coloring Step 17	82
Gambar 5.40 Map Coloring Step 18	82
Gambar 5.41 Map Coloring Step 19	83
Gambar 5.42 Map Coloring Step 20	83
Gambar 5.43 Map Coloring Step 21	83
Gambar 5.44 Hasil Map Coloring	84
Gambar 6.1 Permainan Wumpus Word	88
Gambar 6.2 Aturan dalam Permainan Wumpus World	89
Gambar 6.3 Contoh Aturan Permainan Wumpus Step 1	90
Gambar 6.4 Contoh Aturan Permainan Wumpus Step 2	90
Gambar 6.5 Contoh Aturan Permainan Wumpus Step 3	91
Gambar 6.6 Contoh Aturan Permainan Wumpus Step 4	91
Gambar 6.7 Contoh Aturan Permainan Wumpus Step 5	92

Gambar 6.8 Contoh Aturan Permainan Wumpus Step 6	92
Gambar 6.9 Contoh Aturan Permainan Wumpus Step 7	93
Gambar 6.10 Wumpus World dalam Kondisi Khusus	93
Gambar 6.11 Wumpus World dalam Kondisi Khusus	93
Gambar 6.12 Proses Reasoning	95
Gambar 6.13 Ilustrasi Proses Reasoning dalam Otak Manusia	96
Gambar 6.14 Contoh Model Jebakan	97
Gambar 6.15 Ilustrasi Model Checking	97
Gambar 6.16 Ilustrasi Model Checking	97
Gambar 6.17 Ilustrasi Kinerja dari <i>Forward Chaining</i>	102
Gambar 7.1 Contoh Sebuah Model.....	106
Gambar 7.2 Contoh Sebuah Model (lebih rinci)	107
Gambar 8.1 Contoh Forward Chaining FOL	120
Gambar 8.2 Contoh Forward Chaining FOL	120
Gambar 8.3 Contoh Backward Chaining FOL.....	122
Gambar 8.4 Contoh Pembuktian dengan Resolution.....	124
Gambar 9.1 Contoh Gambaran Prolog 1	129
Gambar 9.2 Contoh Gambaran Prolog 2	130
Gambar 9.3 Contoh Gambaran Query	131
Gambar 10.1 Contoh Distribution untuk Variable Real & Kontinyu..	143
Gambar 11.1 Bayesin Network Conditional Independence	148
Gambar 11.2 Bayesin Network	149
Gambar 11.3 Contoh Gambaran Probabilitas	150
Gambar 11.4 Struktur Network.....	152
Gambar 11.5 Struktur Full Joint Distribution.....	152
Gambar 11.6 Variable dengan nilai kontinyu	153
Gambar 11.7 Distribusi Integral Gaussian	154
Gambar 11.8 Ilustrasi Variable Kontinyu.....	154
Gambar 11.9 Contoh Path Efficient Inference.....	155
Gambar 11.10 Contoh Path Efficient Inference.....	155
Gambar 11.11 Contoh Sampling 1	156

Gambar 11.12 Contoh Sampling 2	157
Gambar 12.1 Ilustrasi Map AI dan ML, serta Transfer AI.....	161
Gambar 12.2 Map cakupan dari Machine Learning	162
Gambar 12.3 Map pengembangan produk App	162
Gambar 12.4 Supervised vs Reinforcement vs vs Un-Supervised Learning.....	163
Gambar 13.1 Ilustrasi SVM untuk <i>linearly separable data</i>	166
Gambar 13.2 <i>Linear Kernel</i>	169
Gambar 13.3 Non-Linear Kernel	170
Gambar 13.4 Visualisasi Dataset pada kernel Linear	182
Gambar 13.5 Visualisasi Data	184
Gambar 13.6 Visualisasi Data Uji.....	185
Gambar 13.7 Visualisasi Datset	186
Gambar 13.8 Gambar Pseudo-code <i>Simplified SMO</i>	194
Gambar 13.9 Testing 3D K(x,y)=(x.y) ² Ke-1.....	252
Gambar 13.10 Testing 3D K(x,y)=(x.y)2 Ke-2	252
Gambar 13.11 Klasifikasi dengan Metode One-Against-All	254
Gambar 13.12 Klasifikasi <i>One-Against-One</i> untuk 4 Kelas	255
Gambar 13.13 Ilustrasi Pohon Biner	259
Gambar 13.14 Ilustrasi klasifikasi dengan metode <i>One-against-One</i> , <i>One-against-All</i> , dan DDAGSVM	261
Gambar 13.15 Nilai Tukar IDR terhadap USD Periode 2006-2015 .	264
Gambar 13.16 Visualisasi Hasil Peramalan Iterasi SVR 100000....	272
Gambar 15.1 Ilustrasi Machine Learning dan Deep Learning.....	294
Gambar 15.2 (a) Tradisional Machine Learning, (b) Transfer Learning	295
Gambar 15.3 Ilustrasi arsitektur pada Deep Learning	296
Gambar 15.4 Ilustrasi arsitektur pada Non-Deep ELM	298
Gambar 16.1 Ilustrasi Ekstrasi Fitur CNN pada Objek 3D ke 2D	302
Gambar 16.2 Map contoh CNN-ELM	303
Gambar 17.1 Blockchain dan Bagaimana Cara Kerjanya.....	333
Gambar 17.2 Blockchain - previous_hash dan hash block	335

Gambar 17.3 Consensus & Decentralization based Interoperability	338
Gambar 18.1 Memori RNN untuk Sharing hasil ke Next Proses	340
Gambar 18.2 Jaringan Algoritme Recurrent Extreme Learning Machine Neural Network	341
Gambar 18.3 Arsitektur Jaringan RNN untuk Prediksi Nilai Tukar...	343
Gambar 18.4 Alur kerja Long Short-Term Memory (LSTM)	359
Gambar 18.5 Alur kerja Gated Recurrent Unit (GRU).....	360
Gambar 18.6 Jaringan Algoritme Modified Long Short-Term Memory (MLSTM)	364
Gambar 19.1 Contoh Multi-Agent dan MDP	435
Gambar 19.2 Contoh Multi-Agent dengan JST (Karpathy, 2016)	436
Gambar 19.3 Hasil arsitektur SOM	438
Gambar 20.1 Tentang skema DBN	445
Gambar 20.2 Ilustrasi Contoh Arsitektur Umum dari DBN.....	446
Gambar 21.1 Contoh arsitektur Autoencoder (AE)	462
Gambar 21.2 Contoh arsitektur CAE.....	462
Gambar 21.3 Linear vs nonlinear reduksi dimensi pada 2D	463
Gambar 21.4 Contoh Sebaran Data pada 3D nonlinear, dan Sebaran Data 2D linear pada 3D	463
Gambar 21.5 Ilustrasi arsitektur AE	464
Gambar 21.6 Bagian yang ditunjuk Panah Merah merupakan Outlier	464
Gambar 21.7 Structure ELM Autoencoder untuk generating presentations “Deep model”	465
Gambar 21.8 Map Simplified Deep Learning CNN based ELM with PSO (Deep PSO)	485
Gambar 22.1 Reinforcement Learning untuk Kandidat Obat Covid-19	489
Gambar 22.2 Reinforcement Learning pada Ai Dino	497
Gambar 22.3 Reinforcement Learning pada Ai Flappy Bird	498
Gambar 22.4 Reinforcement Learning pada Ai Frozen Lake.....	498
Gambar 22.5 Ilustrasi Generative Adversarial Network (GAN)	502
Gambar 22.6 Arsitektur algoritma GAN.....	503
Gambar 22.7 Ilustrasi Cara Kerja Style Transfer	505

Gambar 22.8 Pengambilan data dengan crop berupa bounding box pada bagian wajah dan Jilbab	513
Gambar 22.9 Multimodal DL dengan data berbeda dan ekstraksi fiturnya	522
Gambar 22.10 Ilustrasi Hasil Auto Image Captioning kata bijak:D ..	522
Gambar 22.11 Multimodal Neural Network	523

Daftar Tabel

Tabel 2.1 Percept dan Action pada Vacuum Cleaner	15
Tabel 2.2 Function Reflex-Vacuum-Agent	15
Tabel 2.3 Jenis-Jenis Agen	21
Tabel 2.4 Contoh Jenis Agen	30
Tabel 4.1 Implementasi Algoritme Min-Max pada Permainan	50
Tabel 4.2 Perbandingan Jumlah Node	55
Tabel 4.3 Menentukan Posisi	56
Tabel 6.1 Ilustrasi Entailment dalam Wumpus World	96
Tabel 6.2 Contoh Permasalahan Logika Proposisi	99
Tabel 6.3 Solusi untuk Permasalahan Logika Proposisi	100
Tabel 6.4 Contoh Tabel Kebenaran	100
Tabel 6.5 Contoh Tabel Kebenaran	101
Tabel 7.1 Contoh Jenis Logic	105
Tabel 8.1 Contoh Unifications	117
Tabel 10.1 Contoh Joint Probability	144
Tabel 10.2 Joint Probability Distribution	144
Tabel 10.3 Contoh Joint Probability Distribution	145
Tabel 13.1 Dataset	170
Tabel 13.2 Dataset	177
Tabel 13.3 Dataset perhitungan Kernel Linear.....	182
Tabel 13.4 Data	184
Tabel 13.5 Data Uji.....	184
Tabel 13.6 Dataset Kernel Polynomial	185
Tabel 13.7 Perhitungan $K(x, x_i)$	186
Tabel 13.8 Matrik NxN.....	187
Tabel 13.9 Data Uji.....	190
Tabel 13.10 Data Pelatihan	195
Tabel 13.11 Hasil Perhitungan dengan Kernel RBF	196
Tabel 13.12 Hasil <i>Error</i> pada data ke-i.....	197

Tabel 13.13 Kondisi KKT	197
Tabel 13.14 <i>Error</i> data pembanding (<i>Ej</i>)	198
Tabel 13.15 Nilai Alpha data <i>i</i> dan <i>j</i>	199
Tabel 13.16 Nilai L dan H (target sama)	200
Tabel 13.17 Nilai L dan H (target tidak sama)	200
Tabel 13.18 Nilai L dan H (Kondisi).....	201
Tabel 13.19 Hasil perhitungan <i>eta</i>	202
Tabel 13.20 Nilai Alpha Baru data ke- <i>j</i>	203
Tabel 13.21 Nilai Alpha data <i>j</i> pada segmen garis.....	203
Tabel 13.22 Kondisi nilai absolut.....	204
Tabel 13.23 Nilai Alpha Baru Data ke- <i>i</i>	205
Tabel 13.24 Nilai b1 dan b2.....	206
Tabel 13.25 Nilai bias akhir	207
Tabel 13.26 Nilai Alpha dan Bias Pada Proses Pelatihan	207
Tabel 13.27 Data Uji.....	208
Tabel 13.28 Hasil Hitung Kernel RBF Data Uji dan Data Training ...	208
Tabel 13.29 <i>Dataset</i>	211
Tabel 13.30 Pembagian <i>Dataset</i> ke dalam 3 <i>fold</i>	212
Tabel 13.31 Nilai Parameter.....	214
Tabel 13.32 Tabel Penentuan Kelas Positif dan Negatif.....	214
Tabel 13.33 Tabel Matrik Kernel	215
Tabel 13.34 Tabel Matrik <i>Hessian</i>	216
Tabel 13.35 Tabel Nilai <i>E</i>	217
Tabel 13.36 Tabel nilai $\delta\alpha_i$	218
Tabel 13.37 Tabel Pembaruan <i>Alpha</i>	218
Tabel 13.38 Tabel Data Uji yang Digunakan.....	220
Tabel 13.39 Nilai Kernel Data Uji	221
Tabel 13.40 Tabel Hasil Perhitungan Nilai Fungsi <i>f(x)</i> <i>Level 1</i>	222
Tabel 13.41 Tabel Pembagian Kelas Positif dan Negatif <i>Level 2</i>	222
Tabel 13.42 Tabel Hasil Perhitungan Nilai Fungsi <i>f(x)</i> <i>Level 2</i>	223
Tabel 13.43 Tabel Hasil Penggabungan Dua Fungsi	224

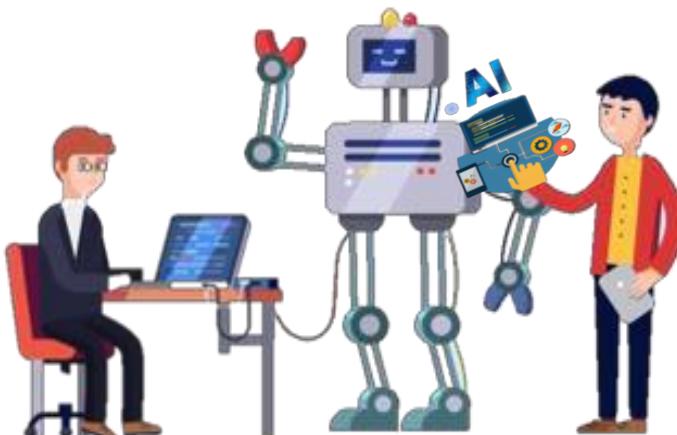
Tabel 13.44 Tabel Perbandingan Hasil Sistem dengan Hasil Aktual	224
Tabel 13.45 Contoh 3 SVM Biner dengan Metode One-Against-All	253
Tabel 13.46 Metode <i>One-Against-One</i> dengan 4 Kelas	254
Tabel 13.47 Dataset	257
Tabel 13.48 <i>Gravity Center</i> Untuk Kelas 1	257
Tabel 13.49 Data <i>Gravity Center</i> Masing-masing Kelas	258
Tabel 13.50 Jarak Euclidian dari Setiap Kelas.....	258
Tabel 13.51 Data Nilai Tukar IDR Terhadap USD Juli 2015.....	265
Tabel 13.52 Dataset dengan 4 fitur	265
Tabel 13.53 Hasil Normalisasi Data Latih	266
Tabel 13.54 Hasil Normalisasi Data Uji.....	266
Tabel 13.55 Data 1 dan 2	266
Tabel 13.56 Jarak Data Latih	267
Tabel 13.57 Matrik Hessian	267
Tabel 13.58 Nilai <i>Ei</i>	268
Tabel 13.59 Nilai $\delta\alpha_i^*$ dan $\delta\alpha_i$	268
Tabel 13.60 Hitung nilai alpha	269
Tabel 13.61 Nilai $f(x)$ Data Latih	270
Tabel 13.62 Nilai $f(x)$ Data Uji.....	270
Tabel 13.63 Hasil Denormalisasi Data Latih	270
Tabel 13.64 Hasil Denormalisasi Data Uji.....	271
Tabel 13.65 Nilai MAPE	271
Tabel 14.1 Contoh Dataset untuk Regresi Linier Sederhana.....	280
Tabel 14.2 Contoh Dataset untuk Regresi kompleks dalam Matriks.....	283
Tabel 16.1 Data Curah Hujan untuk Prediksi	309
Tabel 16.2 Data Curah Hujan untuk Klasifikasi.....	309
Tabel 18.1 Data nilai tukar Rupiah terhadap Dolar Amerika Serikat	342
Tabel 21.1 Perbandingan CNN vs Deep ELM.....	465

BAB 1 Konsep Dasar AI

1.1 Apa itu AI

Pengertian AI dari beberapa sumber dapat diketahui dari 4 pendapat pakar-pakar berikut:

- Otomasi aktivitas yang berhubungan dengan proses berpikir, pemecahan masalah dan pembelajaran (Bellman, 1978).
- Studi tentang kemampuan mengindera dengan menggunakan model komputasi. (Charniak+McDermott, 1985).
- Studi bagaimana cara melakukan sesuatu sehingga menjadi lebih baik (Rich+Knight, 1991).
- Cabang dari ilmu komputer yang fokus pada otomasi perilaku yang cerdas. (Luger+Stubblefield, 1993).



Secara garis besar AI dapat dibedakan menjadi 4 kategori yaitu:

- Thinking Humanly
Pendekatan ini dilakukan dengan dua cara yaitu pertama melalui introspeksi, mencoba menangkap pemikiran kita

sendiri saat kita berfikir. “how do you know that you understand?”. Yang kedua yaitu melalui penelitian-penelitian dari segi psikologi.

- Acting Humanly (The Turing test approach, 1950)
Pendekatan ini pada tahun 1950, Alan Turing merancang suatu ujian bagi komputer yang berinteligensi (bot Cerdas) untuk menguji apakah komputer tersebut mampu menge-labuh seorang manusia/ interrogator melalui komunikasi berbasis teks jarak jauh. Tentunya komputer tersebut harus memiliki kemampuan, Natural Language Processing, Knowledge Representation, Automated Reasoning, Machine Learning, Computer Vision, Robotics.
- Thinking rationally
Pada pendekatan ini terdapat dua masalah yaitu pertama tidak mudah membuat pengetahuan informal, lalu menyatakan dalam formal term dengan notasi-notasi logika. Yang kedua terdapat perbedaan besar antara dapat memecahkan masalah “secara prinsip” dan memecahkannya “da-lam dunia nyata”.
- Acting rationally (The Rational agent approach)
Pendekatan ini membuat inferensi logis yang merupakan bagian dari suatu rational agent. Karena untuk melakukan aksi secara rasional adalah dengan menalar secara logis, maka bisa didapatkan kesimpulan bahwa aksi yang dilakukan akan mencapai tujuan atau tidak.

Sampai saat ini, pemikiran manusia yang diluar rasionalitas, yakni refleks dan intuitif (berhubungan dengan perasaan) belum dapat sepenuhnya ditirukan oleh komputer. Kedua definisi diatas dirasa ku-rang tepat untuk saat ini. Jika menggunakan definisi ini, maka banyak produk AI saat ini yang tidak layak disebut sebagai piranti cerdas. Definisi AI yang paling tepat saat ini adalah acting rationally.

1.2 Fondasi AI

Manusia dibekali kecerdasan yang luar biasa. Pada usia 3 tahun, dia sudah mampu mengenali berbagai macam benda walaupun hanya terlihat sebagian. Ketika melihat sebagian ekor cicak, maka dia akan dengan mudah mengenali bahwa hewan tersebut adalah cicak yang sedang bersembunyi dibalik bingkai lukisan. Pada usia dewasa,

kecerdasannya terus berkembang dengan pesat, mulai dari kecerdasan kognitif, emosional dan spiritual.

Sampai saat ini belum ada satu mesinpun yang mampu menyamai kecerdasan manusia secara keseluruhan. Selama bertahun-tahun, para ilmuwan berusaha mempelajari kecerdasan manusia. Dari pemikiran para ilmuwan tersebut, maka lahirlah AI sebagai cabang ilmu yang berusaha memahami kecerdasan manusia.

Dukungan perkembangan teknologi, baik hardware maupun software yang sangat beragam. Hingga saat ini AI telah menghasilkan banyak piranti cerdas yang sangat berguna bagi kehidupan manusia. Hingga saat ini pula AI terus dipelajari dan dikembangkan secara meluas maupun mendalam.

1.3 Sejarah AI

Istilah AI pertama kali dikemukakan pada tahun 1956 di konferensi Dartmouth. Berikut adalah rangkuman singkat terkait tahapan sejarah perkembangan AI:

- Era komputer elektronik (1941)
Telah di temukan alat sebagai komputer elektronik yang dikembangkan di USA dan Jerman. Komputer tersebut memerlukan ruangan yang luas dan ruang AC yang terpisah dan melibatkan konfigurasi ribuan kabel. Penemuan ini menjadi dasar pengembangan program yang mendarah ke AI.
- Masa Persiapan AI (1943-1956)
Warren McCulloch & Walter Pitts berhasil membuat suatu model sel syaraf tiruan (1943). Dan Norbert Wiener membuat penelitian mengenai prinsip teori feedback (1950). Sedangkan John McCarthy (bapak AI) melakukan penelitian bidang Automata, JST dan pembelajaran intelijensia dengan membuat program yang mampu berfikir.
- Awal Perkembangan AI (1952-1969)
Kesuksesan Newell dan Simon dengan program “General Problem Solver”. Program ini digunakan menyelesaikan masalah secara manusiawi. McCarthy mendemokan bahasa pemrograman tingkat tinggi yaitu LISP di MIT AI Lab. Kemudian Nathaniel Rochester dari IBM dan mahasiswa-mahasiswanya mengeluarkan program AI yaitu “Geometry Theorem Prover” yang mampu membuktikan suatu teorema (1959).

- Perkembangan AI melambat (1966-1974)
Program AI yang bermunculan hanya mengandung sedikit atau bahkan tidak mengandung sama sekali pengetahuan pada subjeknya. Banyaknya permasalahan yang harus diselesaikan oleh AI, karena terlalu banyaknya masalah yang berkaitan, maka tidak jarang terjadi kegagalan ketika membuat program AI. Ada beberapa batasan pada struktur dasar yang digunakan untuk menghasilkan perilaku intelejensi, contohnya dua masukan data yang berbeda tidak dapat dilatih untuk mengenali kedua masukan yang berbeda.
- Sistem berbasis pengetahuan (1969-1979)
Ed Feigenbaum, dkk, membuat program untuk memecahkan masalah struktur molekul (Dendral Programs) yang berfokus pada segi pengetahuan kimia. Dan Saul Amarel dalam proyek “Computer in Biomedicine” membuat program dari segi pengetahuan diagnosa medis.
- AI menjadi sebuah industri (1980-1988)
Pada saat menjadi sebuah industry AI menemukan expert system (R1) yang mampu mengkonfigurasi sistem-sistem komputer. Dan booming industri AI juga melibatkan banyak perusahaan besar yang menawarkan software tools untuk membangun sistem pakar.
- Kembalinya Jaringan Syaraf Tiruan (1986-sekarang)
Pada masa tahun ini Hopfield mengembangkan teknik mekanika statistik untuk mengoptimasi jaringan syaraf tiruan (1982). Dan David Rumelhart & Geoff Hinton menemukan algoritma back-propagation. Algoritma ini berhasil diimplementasikan pada bidang ilmu komputer dan psikologi (1985).

1.4 AI Saat Ini

Berbagai produk AI berhasil dibangun dan digunakan dalam kehidupan sehari-hari. Produk-produk tersebut dikelompokkan ke dalam empat teknik yang ada di AI, yaitu:

- Searching
- Reasoning
- Planning
- Robotic
- Learning

Contoh dari produk AI yang berhasil dibangun dan digunakan dalam kehidupan sehari-hari yaitu:

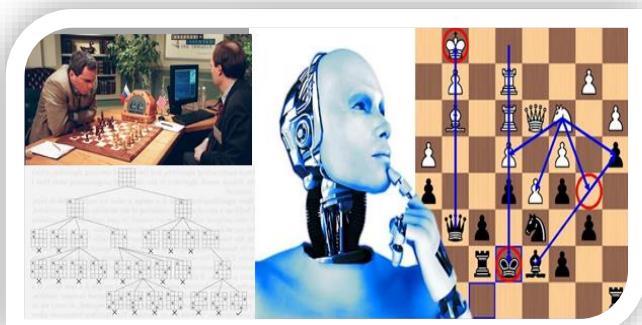
- MedicWare (Rekam medis Pasien)
- Speech Processing (Pengenalan suara, Pengenalan Pembicara)
- GPS (Rute Optimal)



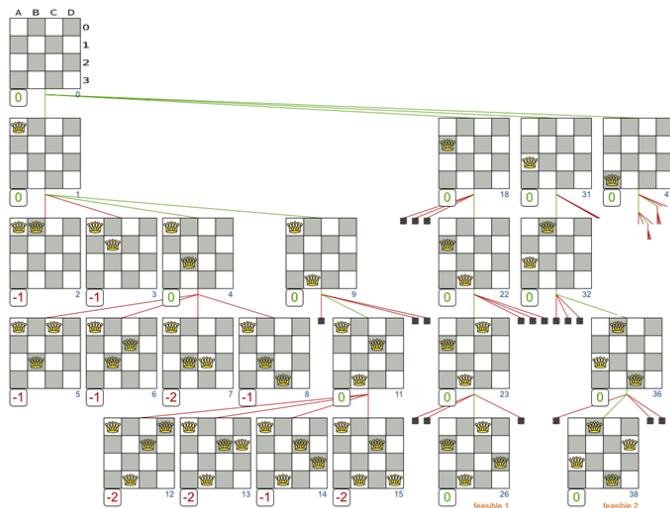
Gambar 1.1 Tampilan (Rute Optimal)

- Bagaimana anda dapat mengetahui lokasi seseorang (teman, orang lain) pada waktu berikutnya?
- Bagaimana sistem bisa memberikan rekomendasi lokasi yang sebaiknya dikunjungi (misal berangkat ke kampus) pada waktu tertentu?

- Catur

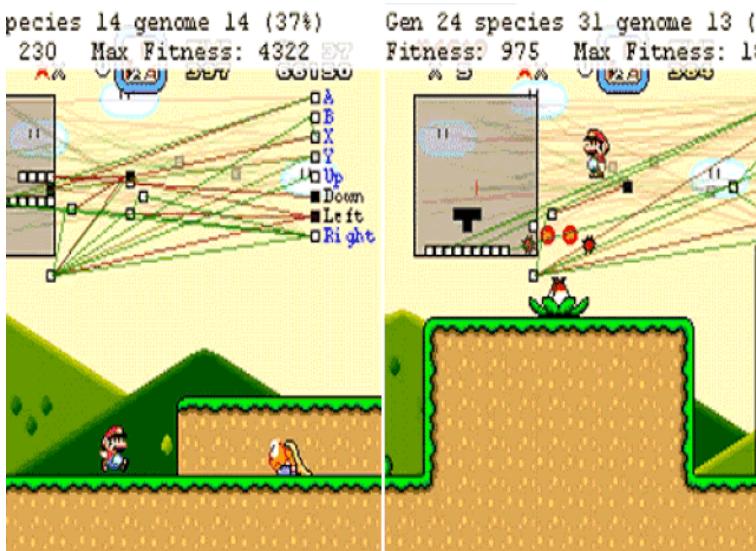


Gambar 1.2 Tampilan pada Permainan Catur



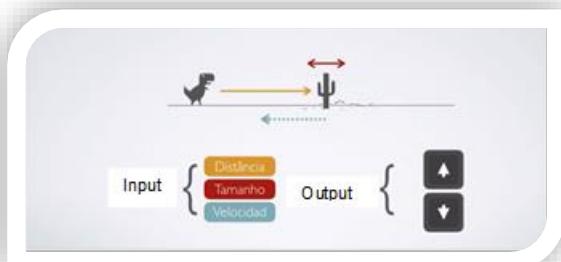
Gambar 1.3 Contoh penyelesaian dengan memanfaatkan AI pada Game Catur

- Super Mario

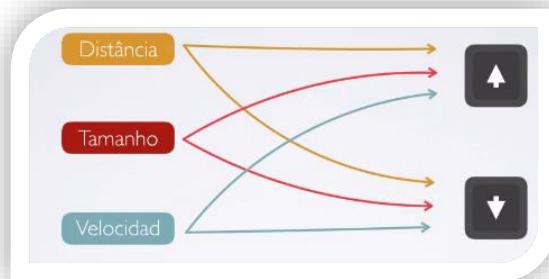


Gambar 1.4 Contoh penyelesaian dengan memanfaatkan AI pada Game Super Mario

- Game Dino AI



Gambar 1.5 Tampilan Permainan Dino

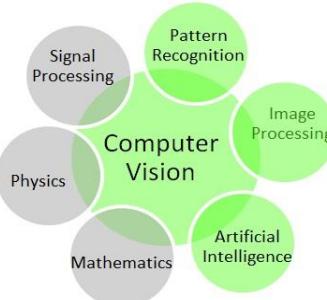


Gambar 1.6 Penyelesaian dengan memanfaatkan AI

Keterangan:

Input	= hasil dari sensor atau pengamatan disekitar agen
Warna Kuning	= jarak dengan halangan
Warna Merah	= ketinggian halangan
Warna Biru	= kecepatan halangan
Output	= bisa secara aksi otomatis

- Computer Vision



Gambar 1.7 Hal yang mencakup pada Computer Vision



Gambar 1.8 Penggunaan AI untuk Deteksi Wajah



Gambar 1.9 Penggunaan AI untuk menghitung Jumlah Kendaraaan



Gambar 1.10 Penggunaan AI untuk menghitung Jumlah Orang

- Optimasi

Pada teknik optimasi dapat menggunakan beberapa algoritma, antara lain;

- Algoritma Genetika (GA)
- Algoritma Particle Swarm Optimization (PSO)

Contoh dari permasalahan optimasi antara lain:

1. Permasalahan dalam penjadwalan kuliah, permasalahan yang dapat diambil yaitu:

- Pemilihan hari: setiap dosen dapat men-set hari dan jam biasanya dalam mengajar.
- Pemilihan SKS & MK: setiap dosen dapat set jumlah SKS & MK minimum dan maksimum yang diampu.
- Output: jadwal kuliah yang optimal.

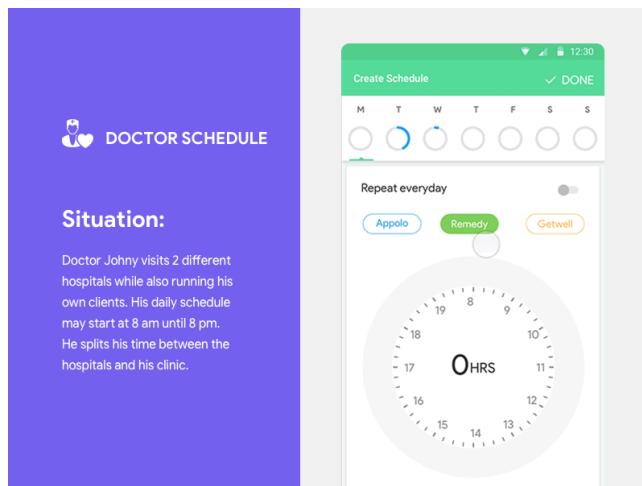


Gambar 1.11 Penggunaan Algoritme Optimasi



Gambar 1.12 Hasil dari memanfaatkan AI

2. Permasalahan penjadwalan jaga dokter, permasalahan yang dapat di ambil yaitu;
 - Hari, setiap dokter bisa men-set hari dan jam biasanya ngajar
 - Tempat jaga, misal 2 tempat, di RS dan Klinik
 - Output, jadwal jaga yang optimal pada RS dan Klinik



Gambar 1.13 Optimasi Penjadwalan Jaga Dokter:

1.5 AI Masa Depan

Tahun 2009, sebuah PC seharga \$1000 akan dapat melakukan sekitar satu triliun kalkulasi per detik dan setara dengan kemampuan komputasional otak manusia yang terdiri dari Virtual Reality dan Interaksi komputer dengan isyarat tubuh. Tahun 2029 juga kemungkinan sebuah PC \$1000 akan setara dengan kemampuan komputasional seribu otak manusia yang terdiri dari komputer dapat terhubung dengan otak manusia dan komputer mampu membaca semua literatur dan material multimedia. Pada tahun 2049, makanan diproduksi menggunakan nano technology. Kumudian pada tahun 2072, kemungkinan picoengineering atau teknologi pada skala picometer atau 10^{-12} meter berhasil diaplikasikan. Kemudian pada tahun 2099, kemungkinan ada kecenderungan untuk membuat gabungan antara pemikiran manusia dan kecerdasan mesin. Sehingga kita tidak dapat membedakan lagi apakah agent tersebut adalah mesin atau manusia.

1.6 Simple Case Study (Logic 1)

Contoh dari permasalahan dalam permainan dibawah ini:



Gambar 1.14 Tampilan Permainan

1.7 Tugas Kelompok

1. Jelaskan pentingnya mempelajari AI & Berikan contoh penerapan AI pada bidang kesehatan, transportasi, komunikasi dan militer!
2. Review Film AI atau hacker yang ada AI-nya (Cut bagian film yang menunjukkan adanya aktifitas AI, dan Jelaskan AI-nya tentang apa), buat dengan Video editor!
3. Selesaikan contoh kasus pada “Logic 2” (Optional) dan “Logic 3” dalam bentuk langkah-langkah logis dan rasional!

BAB 2 Agen Cerdas

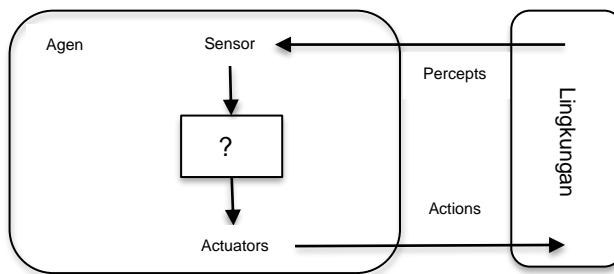
2.1 Agen dan Lingkungan

Sistem Agen Cerdas adalah sebuah program yang dapat diberi tugas dan dapat menyelesaikan tugasnya secara mandiri, serta mempunyai inteligensi. Dengan bantuan sistem agen tersebut, maka pekerjaan yang membutuhkan waktu lama dapat diselesaikan dengan baik dan lebih cepat. Dengan adanya agen cerdas pada aplikasi diharapkan aplikasi tersebut dapat berpikir dan dapat menentukan pilihan langkah terbaik sehingga hampir bisa menyamai kemampuan manusia.

2.1.1 Agen Cerdas

Agen cerdas adalah sebuah agen adalah segala sesuatu yang dapat merasakan lingkungannya melalui peralatan sensor-sensor, bertindak sesuai dengan lingkungannya dan dengan menggunakan peralatan penggeraknya /actuator (Russel dan Norvig). Sebuah agen adalah sebuah sistem komputer yang berada dalam suatu lingkungan dan memiliki kemampuan bertindak secara otonomos didalam situasi lingkungan tersebut sesuai dengan sasaran yang dirancang (Wooldridge).

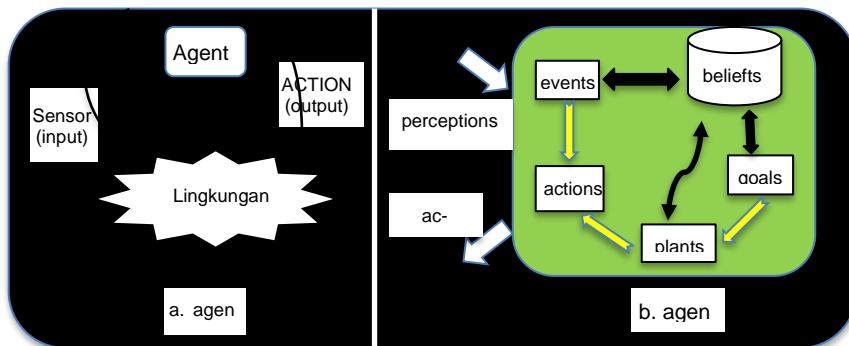
- Agen – Sensors/Actuator – Percepts/Actions – Lingkungan



Gambar 2.1 Abstraksi dari Model Komputasi dari sebuah Agen

Pada Gambar 2.1, memperlihatkan abstraksi dari model komputasi sebuah agen. Pada gambar terlihat setiap tindakan atau aktivitas akan dikerjakan oleh agen adalah untuk memenuhi atau menyesuaikan kondisi lingkungannya.

- Agen – Sensors/Actuator – Percepts/Actions (Event, Benefits,



Goals, Plans) – Lingkungan

Gambar 2.2 Komponen Internal dari Model Agen BDI

Pada Gambar 2.2, memperlihatkan komponen internal dari sebuah model agen BDI (belief-desire-intention) yang memiliki:

1. Events (pemacu indera),
2. Beliefs (pengetahuan),
3. Actions (tindakan),
4. Goals (tujuan),
5. Plans (agenda dan rencana).

Berikut beberapa contoh agent yang terdapat pada kehidupan sehari-hari:

- Human Agen memiliki mata, telinga, dan organ sejenisnya sebagai sensor. Sedangkan tangan, kaki, mulut dan anggota tubuh lainnya sebagai effector.



Gambar 2.3 Contoh 1 pada Effector dari Human Agen



Gambar 2.4 Contoh 2 pada Effectuator dari Human Agent



Gambar 2.5 Contoh 3 pada Effectuator dari Human Agent

- Agent Robot Asimo memiliki kamera dan infrared sebagai sensor, sedangkan peralatan penggerak sebagai effector.



Gambar 2.6 Contoh Robot Asimo

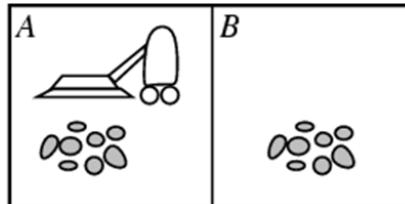
- Agent Robot Vacuum Cleaner dapat menemukan lokasi agen dan keadaan lantai yang kotor sebagai percepts. Sedangkan sebagai actions-nya yaitu ke kiri (left), ke kanan (right), hisap debu (suck), tidak ada operasi apapun (noop).

Tabel 2.1 Percept dan Action pada Vacuum Cleaner

Percept sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Right
[A, Clean], [A, Dirty]	Suck
...	...

Tabel 2.2 Function Reflex-Vacuum-Agent

Function Reflex-Vacuum-Agent([Location, status]) returns an action
If status = Dirty then return Suck
Else if location = A then return Right
Else if location = B then return Left



Gambar 2.7 Gambaran Action pada Vacuum Cleaner

- Agent Robot mesin pemanen padi dapat menemukan lokasi agen dan keadaan padi yang mana mengetahui apakah padi sudah terpotong atau belum dan tindakan tersebut sebagai percepts. Sedangkan untuk actions-nya yaitu belok kanan, kiri, maju, mundur, memisahkan padi dan yang bukan padi.



Gambar 2.8 Alat Pemanen Pada Jaman Dulu



Gambar 2.9 Alat Pemanen Padi
Jaman Sekarang

- Agent Robot mesin pemanen jagung dapat menemukan lokasi agen dan keadaan jagung yang sudah terpotong atau yang belum terpotong sebagai percepts. Sedangkan actions-nya yaitu belok kanan, kiri, maju, mundur, memisahkan jagung dan yang bukan jagung.



Gambar 2.10 Alat Pemanen Jagung Jaman Dulu



Gambar 2.11 Alat Pemanen Jagung Jaman Sekarang

- Agent Robot drone penyepot atau penabur pupuk dapat menemukan lokasi tanaman dan area yang sudah dan yang belum di berikan pupuk sebagai percepts. Sedangkan actions-nya

terbang, belok kanan, kiri, maju, mundur dan menyemprotkan pupuk cair



Gambar 2.12 Alat Semprot Manual



Gambar 2.13 Alat Semprot Basmi Hama dengan Drone

- Agent Robot penanam padi dapat menemukan lokasi keadaan tanah yang lunak dan tanah yang keras sebagai percepts. Sedangkan actions-nya menentukan kedalaman benih, serta jarak antar tanaman.



Gambar 2.14 Tanam Padi Jaman Dulu



Gambar 2.15 Tanam Padi Jaman Modern

- Agent Robot alat pengatur Ph di pertambakan ikan dan pertanian padi. Agen ini dapat mengetahui keadaan Ph air/tanah sebagai percepts. Sedangkan actions-nya mengurangi atau menambah Ph pada air/tanah.



Gambar 2.16 Agen Robot Alat Pengatur PH

- Agent Robot pengusir hama tikus tenaga surya yang mana dapat mendeteksi keberadaan tikus. Sedangkan actions-nya mengeluarkan suara yang dapat mengganggu tikus dengan jangkauan dan intensitas tertentu.

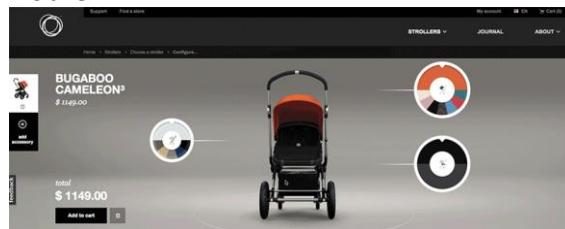


Gambar 2.17 Agen Robot Pengusir Hama Tikus Jaman Dulu



Gambar 2.18 Agen Robot Pengusir Hama Tikus Modern

- Agent Software antarmuka pengguna grafis sebagai sensor dan sekaligus sebagai penggeraknya, misal pada aplikasi web, desktop maupun mobile

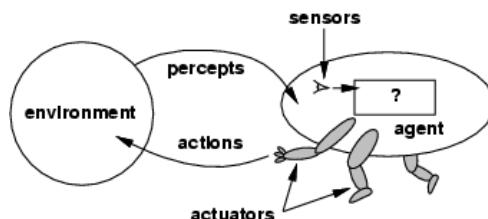


Gambar 2.19 Agen Software Antarmuka Pengguna Grafis

2.1.2 Struktur Agen

The agent function (f) maps dari percept (P^*) histories untuk actions (A)

$$[f: P^* \rightarrow A]$$



Gambar 2.20 Struktur Agen

Tugas dari mempelajari kecerdasan buatan adalah untuk membuat suatu mesin agen. Sebuah agen juga membutuhkan arsitektur dapat berupa komputer saja, atau komputer yang memiliki perangkat keras tertentu dapat melakukan suatu pekerjaan tertentu seperti memproses gambar kamera atau menyaring *input* suara. Jadi, sebuah arsitektur membuat kesan-kesan (percept) lingkungan dapat diterima dengan baik oleh sensor-sensor yang dimilikinya, lalu dapat menjalankan program agennya dan dapat memberikan tindakan

terhadap lingkungan menggunakan actuators. Hubungan agen, arsitektur, dan program dapat diasumsikan sebagai berikut:

$$\text{agen} = \text{arsitektur (hardware)} + \text{program (software)}$$

2.2 Rasionalitas

Sebuah agen selalu mencoba untuk mengoptimasikan sebuah nilai ukuran kinerja yang disebut agen memiliki rasional (*rational agent*). Sebuah agen adalah rasional jika dapat memilih kemungkinan untuk bertindak yang terbaik setiap saat, menurut apa yang ia ketahui sesuai keadaan lingkungannya pada saat itu. Untuk setiap deretan persepsi yang mungkin, sebuah agen rasional hendaklah memilih satu tindakan yang diharapkan memaksimalkan ukuran kemampuannya, dengan adanya bukti yang diberikan oleh deretan persepsi dari apapun pengetahuan yang dimiliki agen tersebut.

Jadi, agen rasional diharapkan dapat melakukan atau memberi tindakan yang benar. Tindakan yang benar adalah tindakan yang menyebabkan agen mencapai tingkat yang paling berhasil (Stuart Russel, Peter Norvig, 2003). Ukuran kinerja (dari rational agent) biasanya didefinisikan oleh perancang agen dan merefleksikan apa yang diharapkan mampu dilakukan dari agen tersebut, misal nilai fitness, waktu tercepat, jarak terpendek, hasil yang lebih akurat. Sebuah agen berbasis rasional juga disebut sebuah agen cerdas. Dari perspektif ini, bidang kecerdasan buatan dapat dipandang sebagai studi mengenai prinsip-prinsip dan perancangan dari agen-agen rasional buatan. Ada beberapa jenis perilaku dari Agent, antara lain:

- Agent rasional: agent yang mengerjakan sesuatu yang benar.
- Performance measure: bagaimana keberhasilan suatu agent.
- Diperlukan suatu standard untuk mengukur performansi, dengan mengamati kondisi yang terjadi.

Sedangkan untuk contohnya dari perilaku Agent yaitu:

- Agent (vacum cleaner), Goal untuk membersihkan lantai yang kotor. Performance dari vacum cleaner:
 - Jumlah Kotoran yang dibersihkan
 - Jumlah Konsumsi listrik
 - Jumlah Kebisingan
 - Waktu yang dibutuhkan

- Agen perubahan (mahasiswa), Goal Measure adalah Lulus Kuliah tepat waktu, Kerja di Perusahaan Besar, Buat Perusahaan Besar (Jadi Wirausaha), Kaya/Kecukupan, Menikah. Sedangkan Performancenya adalah IPK, Penghasilan/ Gaji Bulanan

Sebelum membuat suatu agen, hendaknya telah mengetahui dengan baik:

- Semua kemungkinan kesan (Percept) dan tindakan (Action) yang dapat diterima dan dilakukan oleh agen.
- Apa tujuan (Goal) atau pengukur kemampuan (Performance) agen yang ingin dicapai.
- Lingkungan (Environment) yang seperti apa yang akan dioperasikan oleh agen.

Contoh Jenis Agen dengan Kesan (Percept), Tindakan (Action), Tujuan (Goal) dan Lingkungan (Environment):

Tabel 2.3 Jenis-Jenis Agen

Jenis Agen	Kesan	Aksi	Tujuan	Lingkungan
Sistem Diagnosis Kesehatan	Gejala, jawa-ban pasien	Pertanyaan, ujian, perawatan	Kesehatan pasien, harga minimal	Pasiens, rumah sakit
Sistem analisa gambar satelit	Pixel-pixel dengan intesi-tas yang variatif, warna	Cetak penggo-longan tempat	Memperbaiki intensitas pixel	Gambar-gam-bar dari satelit
Robot pengangkat barang bagian	Pixel-pixel dengan intesi-tas yang variatif, warna	Angkat barang dan masukkan ke dalam ker-ancang	Meletakkan barang bagian ke dalam ker-ancang yang benar	Alat pengangkat barang dengan barang bagian
Pengendali kulkas	Suhu, pembaca tekanan	Buka dan tutup katup, pengaturan suhu	Memaksimal-kan kebersihan, hasil dan kesehatan	kulkas
Pengajar Bahasa Inggris interaktif	Kata-kata yang diketik	Cetak latihan himbauan, koreksi	Memaksimal-kan nilai murid saat ujian	Kumpulan murid-murid

- Rationality Vs Omniscience

- Rationality: mengetahui outcome seharusnya dari suatu tindakan dan mengerjakannya dengan benar.
- Omniscience: ketidakmungkinan dalam suatu kenyataan.

Contoh: Menyebrang jalan yang tidak ada lalin.

2.3 PEAS

PEAS singkatan dari Performance measure (ukuran kinerja), Environment (lingkungan), Actuators (aktuator), Sensors (sensor). Berikut adalah beberapa contoh dari PEAS:

- Agen “smart Car/taxi”:
 - Performance measure: aman, cepat, melalui jalan/jalur yang benar, taat lalin, nyaman ketika berkendara, tarif pas
 - Environment: jalan, lalin, pejalan kali, pelanggan
 - Actuators: roda kemudi, accelerator, rem, isyarat lampu belok kiri/kanan, klakson
 - Sensors: kamera, sonar, speedometer, GPS, meter untuk bahan bakar, mesin sensor (deteksi halangan), keyboard atau berupa layar sentuh
- Agen sistem pakar “Sistem Diagnosis Kesehatan”:
 - Performance measure: kesehatan pasien, biaya yang terjangkau, adanya lembaga yang menaungi dan legal dari segi hukum (Dinkes, BPJS, dll)
 - Environment: pasien, rumah sakit/ klinik/ puskemas, staf
 - Actuators: layar monitor (memberikan pertanyaan, diagnosis, perawatan, arahan)
 - Sensors: Keyboard (entri gejala, memasukkan hasil rating identifikasi pemeriksaan awal dari gejala, entri jawaban pasien)
- Agen “Robot pemisah dan pengambil sampah”:
 - Performance measure: prosentase tingkat kebenaran bagian yang diambil benar-benar sampah, atau sampah tertentu yang berhasil diambil dengan tepat
 - Environment: mesin berjalan dengan bagian atasnya berupa tumpukan barang-barang yang berupa sampah-sampah atau lainnya, tumpukan sampah-sampah
 - Actuators: rangkaian lengan dan tangan rakitan sebagai persendian robot
 - Sensors: kamera, sensor-sensor sudut pada persendian robot

- Agen “Interactive English tutor”:
 - Performance measure: Memaksimalkan skor hasil tes TOEFL atau lainnya
 - Environment: kelas private, atau regular yang berisi seorang atau kumpulan mahasiswa
 - Actuators: layar monitor (latihan-latihan, penjelasan dan saran, sebagai koreksi dan solusi jawaban)
 - Sensors: Keyboard

2.3.1 Karakteristik PEAS

Sebuah agen memiliki karakteristik yang menggambarkan kemampuan dari agen itu sendiri. Semakin banyak karakteristik yang dimiliki oleh suatu agen, maka akan semakin cerdas agen tersebut. Ada beberapa karakteristik dari agen:

- Autonomous : kemampuan untuk melakukan tugasnya dan mengambil keputusan secara mandiri tanpa adanya intervensi dari luar seperti agen lain, manusia ataupun entitas lain.
- Reaktif : kemampuan agen untuk cepat beradaptasi terhadap perubahan informasi yang ada pada lingkungannya.
- Proaktif: kemampuan yang berorientasi pada tujuan dengan cara selalu mengambil inisiatif untuk mencapai tujuan.
- Fleksibel : agen harus mempunyai banyak cara dalam mencapai tujuannya.
- Robust : agen harus dapat kembali ke kondisi semula (tangguh) jika mengalami kegagalan dalam hal tindakan ataupun dalam menjalankan *plan*.
- Rasional : kemampuan untuk bertindak sesuai dengan tugas dan pengetahuannya dengan tidak melakukan hal yang dapat menimbulkan konflik tindakan.
- Social : dalam melakukan tugasnya, agen memiliki kemampuan untuk berkomunikasi dan berkoordinasi baik dengan manusia maupun dengan agen lain.
- Situated : agen harus berada dan berjalan di lingkungan tertentu.

2.3.2 Arsitektur Agen

Pada konsep black box, agen menerima masukan (percepts) dari luar lalu memprosesnya sehingga bisa dihasilkan keluaran (action) yang berdasarkan masukan tadi. Brenner mengemukakan suatu model untuk proses ini yang berisi tahapan interaction, information fusional (peleburan, penyatuan), information processing dan action. BDI (kepercayaan/belief), keinginan (desire), dan kehendak (intention)):

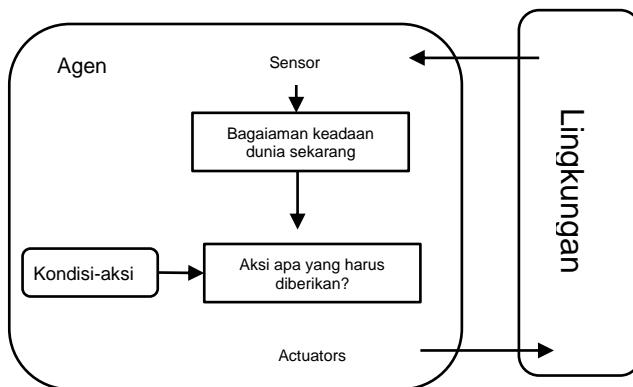
- Kepercayaan atau Belief:
 - Apa-apa saja yang diketahui dan tidak diketahui oleh agen tentang lingkungannya.
 - Atau belief merupakan pengetahuan agen atau informasi yang diperoleh agen tentang lingkungannya.
- Keinginan atau desire:
 - Tujuan, tugas yang harus diselesaikan oleh agen atau sesuatu yang ingin dicapai oleh agen.
- Kehendak atau intention:
 - Rencana-rencana yang disusun untuk mencapai tujuan.
 - Contoh dari arsitektur agen yaitu rancang bangun sistem agen cerdas untuk monitoring stok perusahaan (Percept dan Action). Adapun masukan dan tindakan adalah:
 1. Percept. Percept yang digunakan di dalam sistem antara lain data pemasok baru, data barang jual, data barang beli, dan data barang tiba.
 2. Action. Action yang terlibat adalah bandingkan harga barang pemasok, bandingkan waktu antar barang, tentukan pemasok, menentukan persediaan.

2.3.3 Jenis atau Tipe Agen

Untuk pembuatan agen cerdas, ada lima tipe agen yang dapat mengimplementasikan pemetaan dari kesan yang diterima ke tindakan yang akan dilakukan. Ada beberapa tipe Agen, antara lain:

1. Simple Reflex Agents

Agen refleks sederhana merupakan agen yang paling sederhana karena dia hanya menerapkan teknik kondisi-aksi. Jadi, jika terjadi suatu kondisi tertentu maka agen akan secara sederhana memberikan aksi tertentu. Contoh agen untuk pengendara taxi diberikan kondisi "jika mobil di depan melakukan penggereman" maka agen akan memberikan aksi "injak rem".

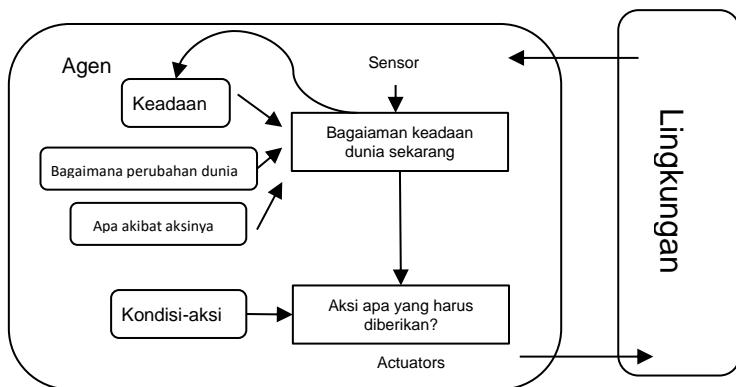


Gambar 2.21 Struktur Simple Reflex Agents

2. Model-Based Reflex Agents

Agen refleks sederhana dapat melakukan tindakannya dengan baik jika lingkungan yang memberikan kesan tidak berubah-ubah. Misalkan untuk kasus agen pengendara taxi, agen tersebut hanya dapat menerima kesan dari mobil dengan model terbaru saja. Jika ada mobil dengan model lama, agen tersebut tidak dapat menerima kesannya sehingga agen tersebut tidak melakukan tindakan pengereman. Pada kasus ini, dibutuhkan agen refleks berbasis model yang dapat terus melakukan pelacakan terhadap lingkungan sehingga lingkungan dapat dikesan dengan baik.

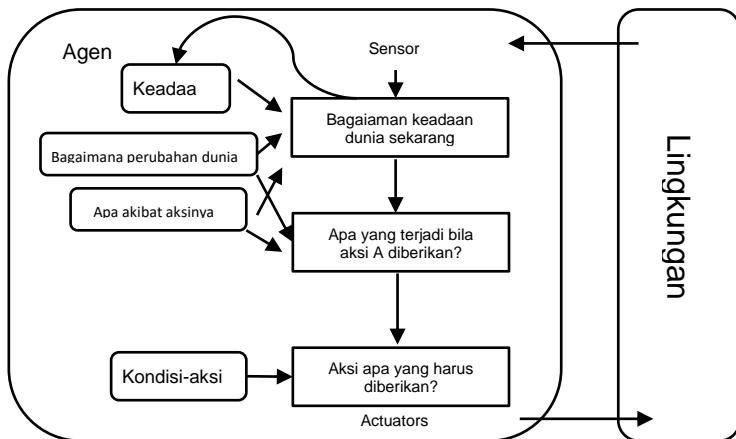
Agen ini akan menambahkan suatu model tentang dunia yaitu pengetahuan tentang bagaimana dunianya bekerja. Jadi, agen refleks berbasis model ini menjaga keadaan dunianya menggunakan model internal kemudian memilih tindakan seperti agen refleks sederhana. Misal agen pengendara taxi, hanya dapat menerima kesan dari mobil dengan model terbaru saja. Jika ada mobil dengan model lama, agen tersebut tidak dapat menerima kesannya sehingga tidak melakukan tindakan pengereman.



Gambar 2.22 Model-Based Reflex Agents

3. Goal-Based Agents

Pengetahuan agen akan keseluruhan keadaan pada lingkungan tidak selalu cukup. Suatu agen tertentu harus diberikan informasi tentang tujuan yang merupakan keadaan yang ingin dicapai oleh agen. Dengan demikian, agen akan bekerja terus menerus hingga mencapai tujuannya. Pencarian dan perencanaan adalah dua deretan pekerjaan yang dilakukan untuk mencapai tujuan agen. Agen refleks berbasis tujuan (*goal*) ini menambahkan informasi tentang tujuan tersebut.



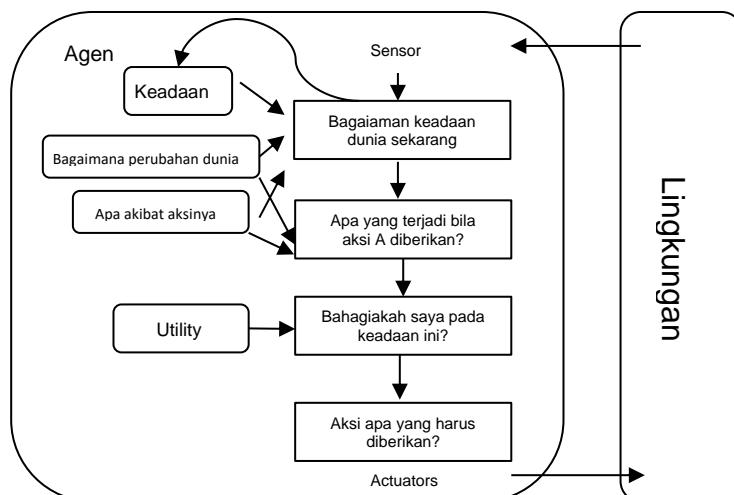
Gambar 2.23 Goal-Based Agents

4. Utility-Based Agents

Pencapaian tujuan pada agen tidak cukup untuk menghasilkan agen dengan tingkah laku berkualitas tinggi. Sebagai contoh untuk

agen pengendara taxi, ada beberapa tindakan yang dapat dilakukan oleh agen sehingga dapat mencapai tempat tujuan, namun ada yang lebih cepat, lebih aman, atau lebih murah dari yang lainnya.

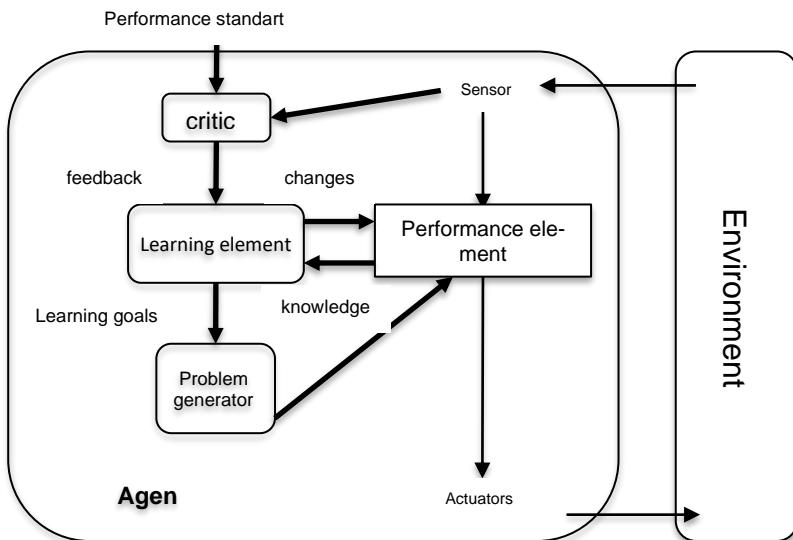
Agen refleks berbasis tujuan tidak membedakan keadaan yang bagus dengan keadaan yang tidak bagus untuk agen. Pada agen refleks berbasis kegunaan (*utility*) ini memikirkan kondisi yang bagus untuk agen sehingga agen dapat melakukan tugasnya jauh lebih baik. Walaupun untuk kasus tertentu, tidak mungkin agen dapat melakukannya semua sekaligus. Misalnya untuk agen pengendara taxi, untuk pergi ke suatu tempat tujuan dengan lebih cepat, itu bertentangan dengan keadaan lebih aman. Karena untuk perjalanan taxi yang lebih cepat, tentu saja tingkat bahaya lebih tinggi dari pada perjalanan taxi yang santai.



Gambar 2.24 Utility-Based Agents

5. Learning agents

Learning agents belajar dari pengalaman, meningkatkan kinerja bertanggung jawab untuk membuat perbaikan. Elemen kinerja bertanggung jawab untuk memilih tindakan eksternal. Kritikus memberikan umpan balik tentang bagaimana agen bekerja.



Gambar 2.25 Learning Agents

2.4 Jenis Lingkungan

Ada beberapa tipe untuk jenis lingkungan pada PEAS, antara lain:

1. Fully observable – partially observable

Apabila sensor pada sebuah agen dapat mengakses keseluruhan keadaan pada lingkungan, maka lingkungan itu dapat dikatakan fully observable terhadap agen. Lebih efektif lagi lingkungan dikatakan fully observable jika sensor dapat mendeteksi seluruh aspek yang berhubungan dengan pilihan aksi yang akan dilakukan.

Lingkungan yang fully observable biasanya sangat memudahkan, karena agen tidak perlu mengurus keadaan internal untuk terus melacak keadaan lingkungan. Suatu lingkungan bisa menjadi partially observable akibat ada gangguan dan ketidakakurasan sensor ataupun karena ada bagian keadaan yang hilang dari data sensor.

2. Deterministic – stochastic

Apabila keadaan lingkungan selanjutnya sepenuhnya bergantung pada keadaan sekarang dan juga tindakan yang akan dilakukan oleh agen, maka lingkungan tersebut bersifat deterministic.

Sedangkan stochastic adalah kebalikan dari deterministic, di mana keadaan selanjutnya tidak bergantung pada keadaan sekarang dan juga tindakan yang akan dilakukan oleh agen.

Apabila lingkungan bersifat deterministic terkecuali untuk tindakan dari agen, maka lingkungan tersebut bersifat strategic. Permainan Reversi bersifat deterministic karena keadaan selanjutnya bergantung pada keadaan sekarang (saat mengambil langkah).

3. Episodic – sequential

Untuk lingkungan yang bersifat episodic, pengalaman agen dibagi-bagi menjadi beberapa epidose pendek. Tiap episode terdiri dari apa yang dirasakan agen dan kemudian melakukan satu tindakan tertentu. Kualitas dari tindakan agen hanya tergantung pada episode itu saja, karena tindakan selanjutnya tidak tergantung pada tindakan apa yang akan dilakukan di episode sebelumnya.

Lingkungan episodic lebih sederhana karena agen tidak perlu memikirkan langkah-langkah pada keadaan selanjutnya. Sedangkan pada lingkungan sequential, tindakan saat sekarang dapat mempengaruhi tindakan selanjutnya. Permainan Reversi bersifat sequential karena agen berpikir untuk langkah-langkah selanjutnya dan seluruh langkah yang akan diambil oleh agen saling bergantung.

4. Static – dynamic

Apabila lingkungan dapat berubah saat agen sedang mengambil keputusan, maka lingkungan tersebut bersifat dynamic, sebaliknya bersifat static. Lingkungan yang bersifat static lebih mudah dihadapi karena agen tidak perlu memperhatikan lingkungannya saat dia sedang mengambil tindakan, maupun waktu yang terus berjalan.

Apabila lingkungan tidak berubah seiring waktu berjalan, namun menyebabkan nilai kemampuan agen berubah-ubah, maka lingkungan tersebut bersifat semidynamic. Permainan Reversi bersifat static karena saat agen mengambil tindakan, lingkungan tidak berubah dan juga tidak mempengaruhi nilai kemampuan agen.

5. Discrete – continuous

Apabila kesan dan tindakan yang akan diterima dan dilakukan oleh agen telah ditetapkan dengan jelas, maka lingkungan tersebut

bersifat discrete. Catur bersifat discrete, karena langkah yang akan diambil terbatas dan tertentu.

Sedangkan pengendara taxi bersifat continuous, karena kecepatan dan lokasi pada taxi untuk suatu jangka tertentu mempunyai nilai yang terus-menerus berubah. Permainan Reversi bersifat discrete karena seluruh kesan dan tindakan telah jelas ditetapkan sesuai dengan peraturan permainan Reversi

6. Single agent – multi agent

Agen pemecah permainan teka teki silang berada pada lingkungan yang bersifat single agent. Agen pemain catur berada pada lingkungan yang bersifat multiagent. Ada hal lain yang memberikan perbedaan lingkungan agen, yaitu akan hal apakah agen memberikan bantuan kepada agen lain atau apakah agen akan memaksimalkan kemampuannya bergantung pada prilaku agen lain. Permainan Reversi bersifat multi agent karena memikirkan langkah yang akan diambil oleh lawan.

2.5 Jenis Agen

Tabel 2.4 Contoh Jenis Agen

Lingkungan	Fully Observable	Deterministic	Episodic	Static	Discrete	Single Agent
Catur dengan jam	ya	ya	tidak	semi	ya	Tidak
Catur tanpa jam	ya	ya	tidak	ya	ya	Tidak
Poker	tidak	tidak	tidak	ya	ya	Tidak
Backgammon	ya	tidak	tidak	ya	ya	Tidak
Pengendara taxi	tidak	tidak	tidak	tidak	tidak	Tidak
System diagnosis kesehatan	tidak	tidak	tidak	tidak	tidak	Ya
System analisis gambar	ya	ya	ya	semi	tidak	Ya
Robot pengantuk barang bagian	tidak	tidak	ya	tidak	tidak	ya
Pengendali kulkas	tidak	tidak	tidak	tidak	tidak	ya
Pengajar Bahasa Inggris Interaktif	tidak	tidak	tidak	tidak	ya	tidak
Reversi	ya	ya	tidak	ya	ya	tidak

2.6 Tugas Kelompok

1. Jelaskan pengertian dari Agen Cerdas menurut pemahaman anda dan kapan suatu Agen dikatakan cerdas?
2. Carilah 2 contoh agen cerdas dan deskripsikan agen tersebut beserta PEAS-nya !
3. Buatlah contoh Agen berdasarkan tipenya!
4. Buatlah contoh Agen berdasarkan jenis lingkungannya!

BAB 3 Un-Informed Searching

3.1 Agen Penyelesaian Problem

Agen pemecahan masalah adalah jenis **agen berbasis tujuan**. Agen pemecahan masalah memutuskan apa yang harus dilakukan dengan **mencari urutan tindakan** yang mengarah pada keadaan (*states*) yang diinginkan. Urutan dalam penyelesaian masalah yaitu:

- Formulasi Goal.
- Perumusan masalah.
- Search → mengambil masalah sebagai masukan dan solusi pengembalian dalam bentuk urutan tindakan.
- Implementasi / Eksekusi.

```
Function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  static: seq, an action sequence, initially empty
         state, some description of the current world state
         goal, a goal initially null
         problem, a problem formulation
  state ← UPDATE STATE(state, percept)
  if seq is empty then do
    goal← FORMULATE GOAL(state)
    problem← FORMULATE PROBLEM(state, goal)
    seq, SEARCH(problem)
    action← FIRST(seq)
    seq← REST (seq)
  return action
```

Penjelasan:

Percept: persepsi yang ada

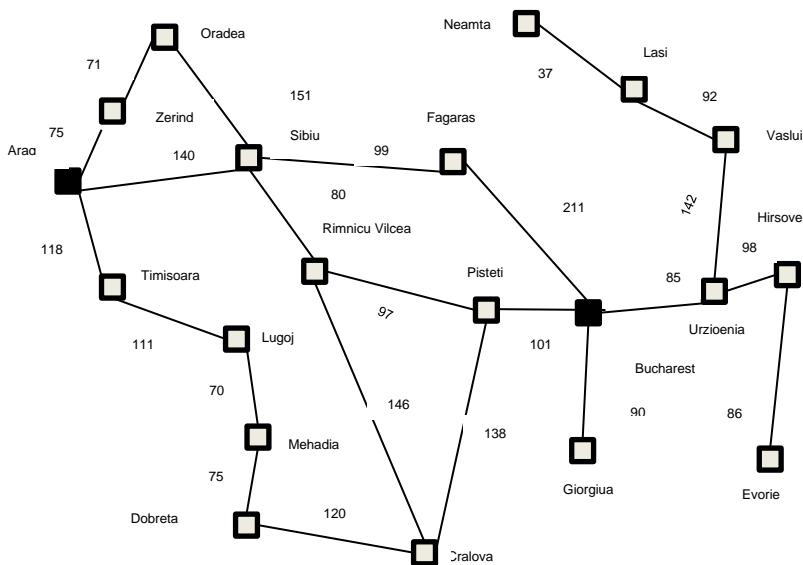
Seq : urutan tindakan

State : deskripsi dari keadaan lingkungan sekitar

Goal : tujuan yang dicapai

Problem: perumusan masalah

Contoh yang dapat di ambil untuk agen penyelesaian problem sebagai berikut:



Gambar 3.1 Contoh Permasalahan dalam Perjalanan

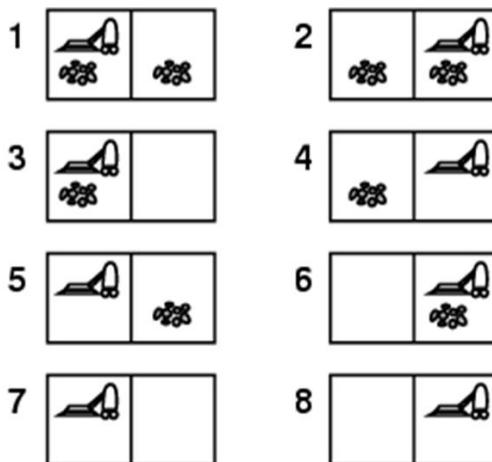
- Berlibur ke Rumania, saat ini berada di Arad. Penerbangan (keberangkatan) dilakukan besok dari Bucharest
Merumuskan tujuan (Formulate goal):
 - Berada di Bucharest
- Merumuskan masalah (Formulate problem):
 - States : berbagai kota sebagai alternatif tempat yang akan dilalui
 - Actions : drive antara kota
- Cari solusi (Find solution):
 - Urutan kota yang dilalui untuk mencapai tujuan. Misalnya; Arad, Sibiu, Fagaras, Bucharest

3.2 Jenis Problem atau Masalah

- Deterministic, fully observable → Single-state problem
 - Agen tahu persis keadaan sesuatu yang akan diamati.
- Non-observable → Sensorless problem (conformant problem)
 - Agen mungkin tidak mengetahui dimana keberadaan sesuatu yang dicari.
- Nondeterministic and/or partially observable → Contingency problem (keadaan yang tidak pasti)

- Persepsi yang dapat memberikan informasi baru tentang keadaan saat ini.
- Unknown state space → Exploration problem (Masalah eksplorasi)

Contoh dari jenis problem / masalah pada vacuum word, dan penyelesaiannya sebagai berikut:



Gambar 3.2 Contoh Jenis Problem pada Vacuum Word

- Sensorless, start in $\{1,2,3,4,5,6,7,8\}$ e.g., Right goes to $\{2,4,6,8\}$
Solution?
[Right, Suck, Left, Suck]
- Contingency
 - Nondeterministic: Suck may dirty a clean carpet
 - Partially observable: location, dirt at current location.
 - Percept: [L, Clean], i.e., start in #5 or #7
Solution?
[Right, if dirt then Suck]

3.3 Formulasi Problem

- Single-state problem formulation

Suatu problem didefinisikan dalam 4 item:

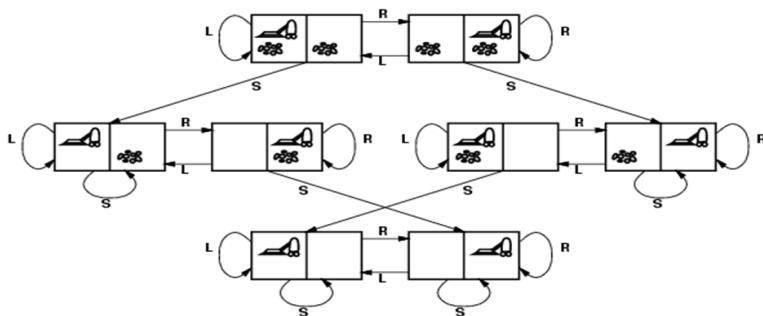
1. initial state e.g., "at Arad"
2. actions or successor function $S(x) = \text{Kumpulan dari pasangan action dan state.}$

- <action, successor(state)>
 - $S(Arad) = \{<\text{Berangkat}(Arad} \rightarrow \text{Zerind}), \text{ BeradaDi}(Zerind)>, \dots\}$
3. goal test, can be
- explicit, e.g., $x = \text{"at Bucharest"}$
 - implicit, e.g., $\text{Checkmate}(x) / \text{Skakmat}(x)$
4. path cost (additive)
- Menetapkan besarnya biaya untuk setiap jalur yang ada.
 - Mis., jumlah jarak tempuh, jumlah tindakan lain yang dilakukan, dll.
 - $c(x, a, y)$ adalah cost action a dari state x ke state y , diasumsikan ≥ 0 .

Solusinya adalah suatu urutan tindakan yang mengarah dari keadaan awal (initial state) ke keadaan tujuan (goal state). Kualitas suatu solusi dapat diukur dari nilai fungsi biaya (cost function) yang paling minimal dari jalur (path) yang dilalui. Pada saat memilih suatu state space:

- Dunia nyata luar biasa kompleks dan rumit! State space harus merupakan abstraksi masalah supaya bisa dipecahkan.
 - State = himpunan “keadaan nyata”. Mis: BeradaDi (Arad) – dengan siapa? kondisi cuaca?
 - Action = kombinasi berbagai “tindakan nyata”. Mis: Berangkat (Arad , Sibiu) – jalan tikus, isi bensin, istirahat, dll.
 - Solution = representasi berbagai “path nyata” yang mencapai tujuan
- Abstraksi ini membuat masalah yang nyata lebih mudah dipecahkan.

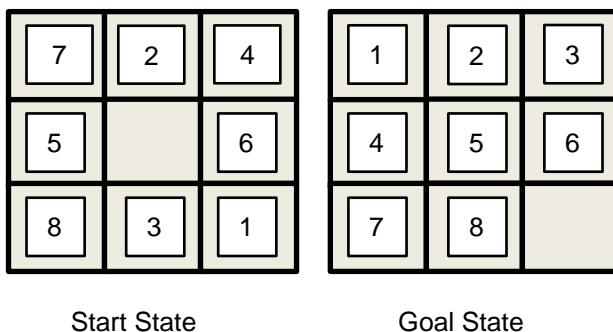
- Contoh: Vacuum Cleaner World



- State: lokasi agent, status debu.
- Possible action: DoKeKiri(L), DoKeKanan(R), DoSedot(S).

Gambar 3.3 Vacuum Cleaner World

- Goal test: semua ruangan sudah bebas debu.
- Path cost: asumsi step cost sama untuk semua action, mis: Path cost = 1 per action.
- Successor function mendefinisikan state space sbb:
- Contoh: 8-Puzzle

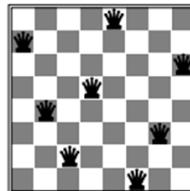


Gambar 3.4 Permainan 8-Puzzle

Penyelesaian:

- State: lokasi 8 buah angka dalam matriks 3x3
- Possible action (move, blank): left, right, up, down

- Goal test: apakah konfigurasi angka sudah seperti goal state di atas.
 - Path cost: asumsi, 1 step cost = 1 per move.
Path cost = jumlah langkah dalam path.
- Contoh: 8-Queens Problem



Gambar 3.5 Permainan *-Queens Problems

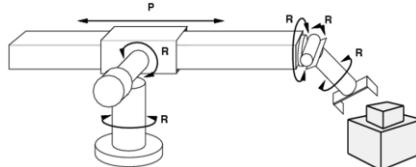
Keterangan:

Letakkan 8 bidak menteri (queen!) sedemikian sehingga tidak ada yang saling “makan” (menteri bisa makan dalam satu baris, kolom, diagonal).

- State: Papan catur dengan n bidak menteri, $0 \leq n \leq 8$.
- Initial state: Papan catur yang kosong.
- Possible action: Letakkan sebuah bidak menteri di posisi kosong.
- Goal test: 8 bidak menteri di papan, tidak ada yang saling makan.

Note: Formulasi yang lebih baik akan melarang menempatkan queen dalam setiap persegi yang sudah diserang.

- Contoh: Robotic assembly (Perakitan Robot)



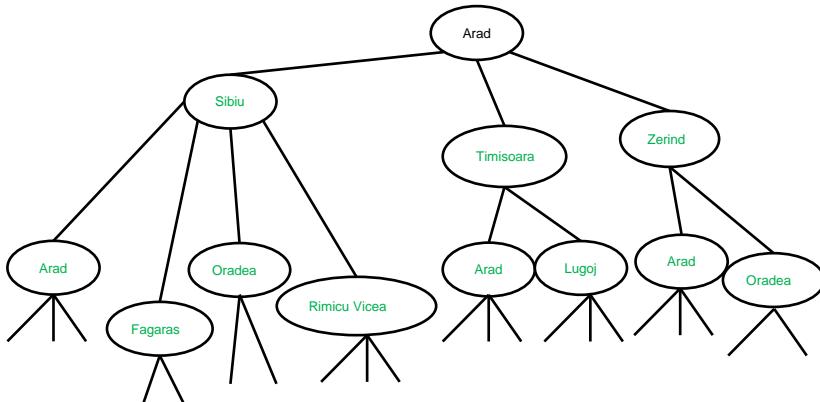
Gambar 3.6 Gambaran Robotic Assembly

- States : koordinat real-valued bagian sudut sendi robot dari obyek yang akan dirakit.
- Actions : gerakan terus menerus dari sendi robot.

- Goal test: perakitan telah lengkap (*complete assembly*).
- Path cost: waktu untuk eksekusi (*time to execute*).

3.4 Algoritma Pencarian Dasar

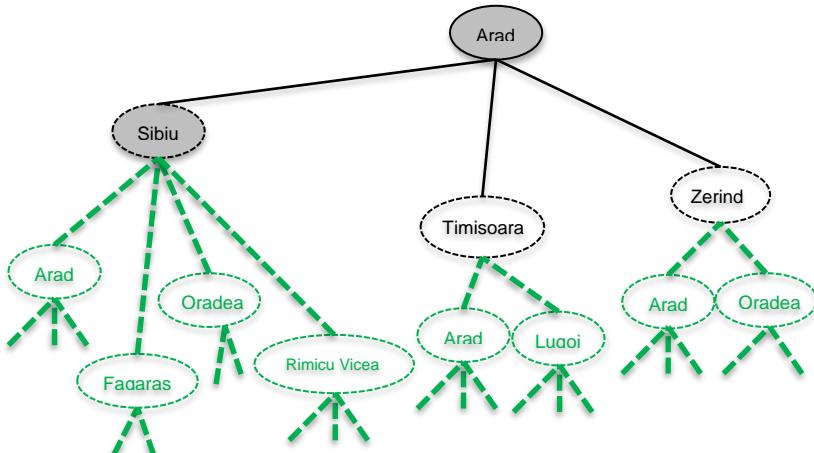
3.4.1 Tree Search



Gambar 3.7 Tree Search

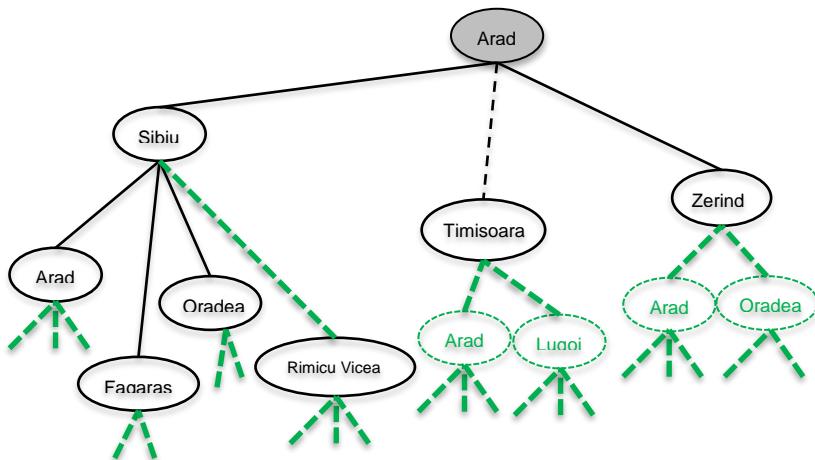
- Tree search algorithms (Basic idea):
 - Mulai dari root node (Arad) sebagai current node.
 - Lakukan node expansion terhadapnya.

Contoh Implementasi Algoritme Min-Max pada Permainan X dan O

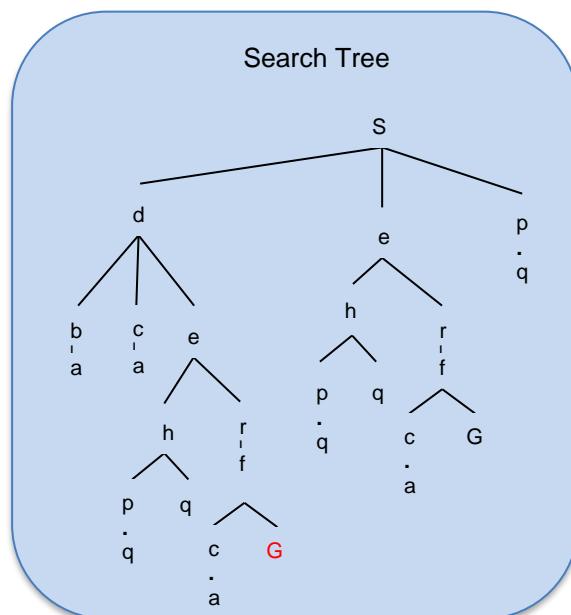


Gambar 3.8 Tree Search

- Pilih salah satu node yang di-expand sebagai current node yang baru. Ulangi langkah sebelumnya.



Gambar 3.9 Tree Search



Gambar 3.10 Search Tree

- Tree search algorithms (Basic idea):

- Pada awalnya, fringe = himpunan node yang mewakili initial state.
- Pilih satu node dari fringe sebagai current node (Kalau fringe kosong, selesai dengan gagal).
- Jika node tersebut lolos goal test, selesai dengan sukses!

```
function EXPAND(node, problem) return a set of
nodes
    successors ← the empty set
    for each action, result in SUCCESSOR-
    FN[problem] STATE[node]) do
        s ← a new NODE
        PARENT-NODE[s] ← node; ACTION[s] ← action;
        STATE[s] ← result
        PATH-COST[s] ← PATH-COST[node] + STEP-
        COST(node, action, s)
        DEPTH[s] ← DEPTH[node] + 1
        add s to successor
```

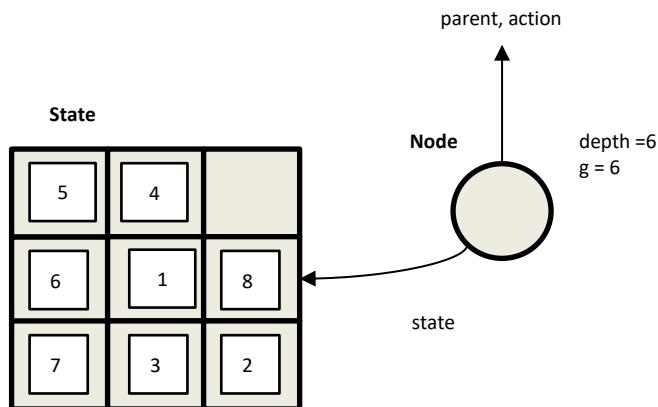
- Jika tidak, lakukan node expansion terhadap current node tsb. Tambahkan semua node yang dihasilkan ke fringe.
- Ulangi langkah 2.

```
function EXPAND(node, problem) return a set of
nodes
    successors ← the empty set
    for each action, result in SUCCESSOR-
    FN[problem] STATE[node]) do
        s ← a new NODE
        PARENT-NODE[s] ← node; ACTION[s] ← action;
        STATE[s] ← result
        PATH-COST[s] ← PATH-COST[node] + STEP-
        COST(node, action, s)
        DEPTH[s] ← DEPTH[node] + 1
        add s to successor
```

Setelah merumuskan masalah kemudian mencari solusinya dengan menggunakan sebuah search algorithm. Search tree merepresentasikan state space. Search tree terdiri dari kumpulan node: struktur data yang merepresentasikan suatu state pada suatu path, dan memiliki parent, children, depth, dan path cost. Root node merepresentasikan initial state. Penerapan successor function terhadap (state yang diwakili) node menghasilkan children baru → ini disebut

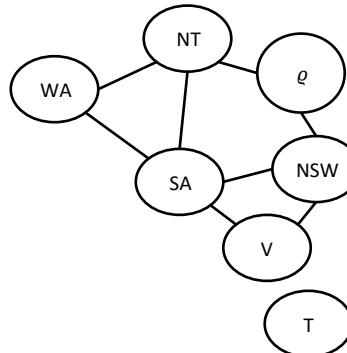
node expansion. Kumpulan semua node yang belum di-expand disebut fringe (pinggir) sebuah search tree.

- Implementation: states vs. nodes
 - Sebuah state merepresentasikan abstraksi keadaan nyata dari masalah.
 - Sebuah node adalah struktur data, bagian dari search tree.
 - State tidak memiliki parent, children, depth, path cost!
 - Node = state pada path tertentu. Dua node berbeda bisa mewakili state yang sama!

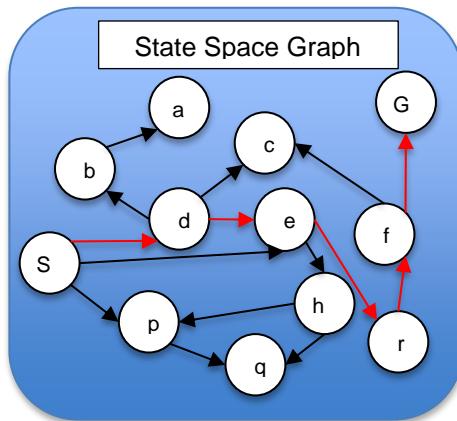


Gambar 3.11 Implementation State and Node

3.4.2 Graph Search



Gambar 3.12 Gambaran untuk Search Search



Gambar 3.13 Gambar State Space Graph

3.5 Strategi Pencarian Uninformed

Strategi pencarian terdapat berbagai jenis strategi untuk melakukan search. Semua strategi ini berbeda dalam satu hal urutan dari node expansion.

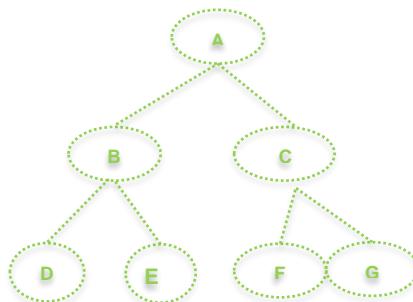
- Search strategy di-evaluasi berdasarkan:
 - Completeness: apakah solusi (jika ada) pasti ditemukan?
 - Time complexity: jumlah node yang di-expand.
 - Space complexity: jumlah maksimum node di dalam memory.
 - Optimality: apakah solusi dengan minimum cost pasti ditemukan?
- Time & space complexity diukur berdasarkan
 - b - branching factor dari search tree
 - d - depth (kedalaman) dari solusi optimal
 - m - kedalaman maksimum dari search tree (bisa infinite!)

Strategi Pencarian Uninformed hanya menggunakan informasi dari definisi masalah. Bisa diterapkan secara generik terhadap semua jenis masalah yang bisa direpresentasikan dalam sebuah state space. Ada beberapa jenis antara lain:

- Uniform-cost search
- Depth-limited search
- Iterative-deepening search
- Breadth-first search

Lakukan node expansion terhadap node di fringe yang paling dekat ke root. BFS menggunakan prinsip queue.

- Implementasi: fringe adalah sebuah queue, data struktur FIFO (First In First Out)
- Hasil node expansion (successor function) ditaruh di belakang



Gambar 3.14 Hasil BFS Traversal Queue

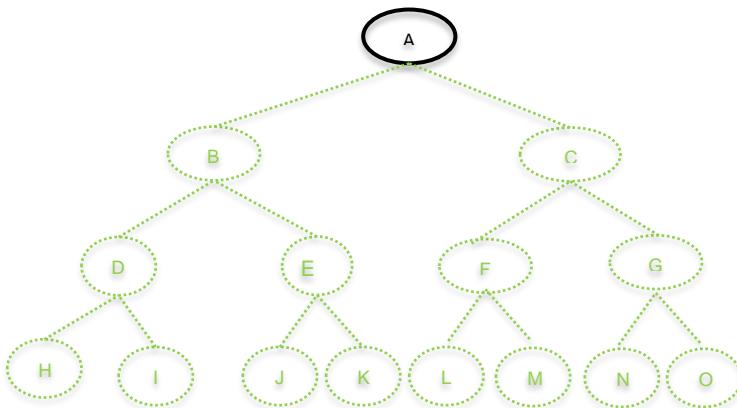
Keterangan:

BFS traversal queue = a → bc → cde → defg → efg → fg → g

- Depth-first search

Lakukan node expansion terhadap node di fringe yang paling jauh dari root. DFS menggunakan prinsip stack.

- Implementasi fringe adalah sebuah stack, data struktur LIFO (Last In First Out)
- Hasil node expansion ditaruh di depan
- Depth-first search sangat cocok diimplementasikan secara rekursif.



Gambar 3.15 Hasil DFS Traversal Stack

DFS traversal stack:

a → ab → abd → abdh → abd → abdi → abd → ab → abe → abej → abe → abek → abe → ab → a → ac → acf → acfl → acf → acfm → acf → ac → acg → acgn → acg → acgo → acg → ac → a → null

- Variasi Depth First Search
 - Backtracking search: lakukan node expansion satu-per-satu. Jika gagal backtrack dan coba nilai successor function yang lain.
 - Depth-limited search: Batasi kedalaman maksimal yang dilihat adalah k.
 - Mengatasi masalah untuk state space tak terbatas.
 - Sayangnya, ada unsur incompleteness baru, jika $k < d$.
 - Biasanya d tidak diketahui (tapi bisa ada estimasi, mis. diameter suatu graph).
- Iterative Deepening Search
 - Lakukan depth-limited search secara bertahap dengan nilai k yang incremental.
 - Strategi ini menggabungkan manfaat depth dan breadth first: space complexity linier dan completeness terjamin!
 - Lakukan depth-limited search dengan $k = 0, 1, 2, \dots$ sampai tidak cutoff.

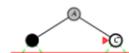
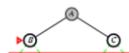
- Iterative deepening search k =0

Limit = 0



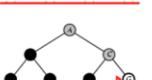
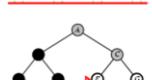
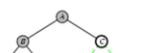
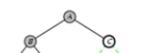
- Iterative deepening search k =1

Limit = 1



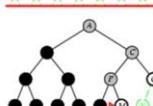
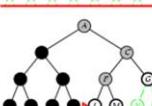
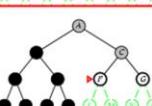
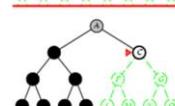
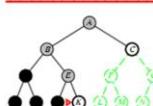
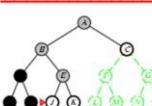
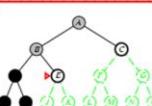
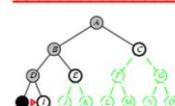
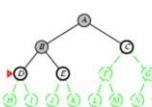
- Iterative deepening search k =2

Limit = 2



- Iterative deepening search k =3

Limit = 3



Solusinya adalah untuk mencatat state mana yang sudah pernah dicoba. Catatan ini disebut closed list (fringe = open list). Modifikasi algoritma TreeSearch dengan closed list menjadi GraphSearch.

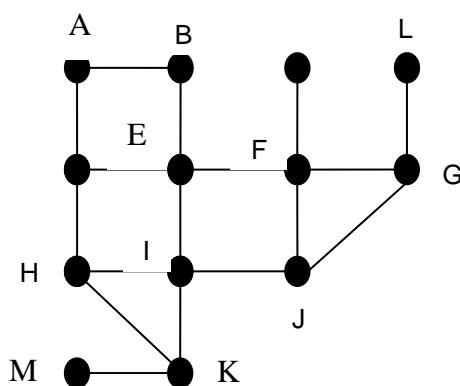
- Time complexity: sama, jika kita asumsi operasi STATE[node] ∈ closed = O (1) (implementasi dengan hashtable?)
- Space complexity: DFS dan IDS tidak lagi linier!
- GraphSearch: tidak mencatat path menuju suatu state. Ini mempengaruhi sifat optimality suatu strategi:
 - Uniform-cost dan breadth-first search dengan step cost konstan masih optimal.

- Untuk variasi Depth-first dan iterative-deepening search, jika state mengulang ditemukan, periksa apakah path cost-nya lebih kecil → update info node dan anak-anaknya!

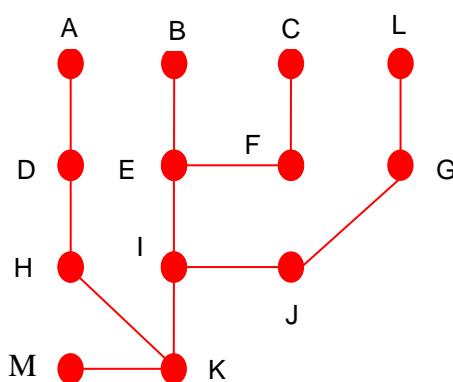
3.6 Latihan Individu

1. Selesaikan graf di bawah ini menggunakan algoritma DFS dan BFS berikut dengan hasil setiap langkah (Traversal dan Tree) dimulai dari node K!

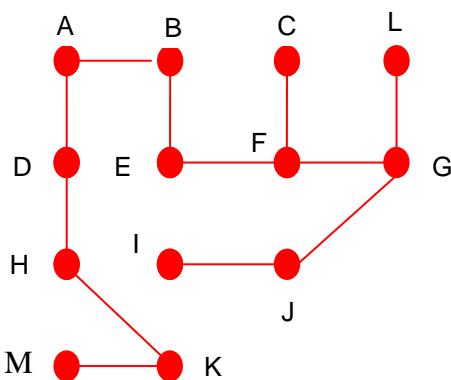
a.



b.

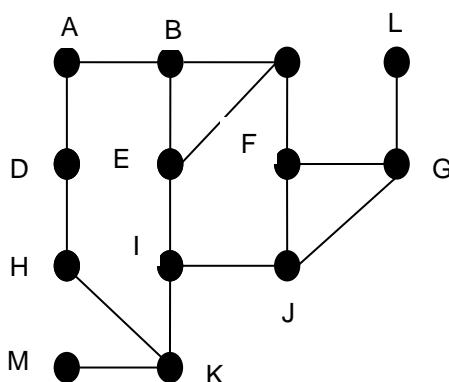


c.



3.7 Tugas Kelompok

1. Buatlah video untuk pemecahan/ solusi kasus:
 - a. 8-Puzzles
 - b. 8-Queens (Optional)Lalu tuliskan langkah-langkah penyelesaiannya ! (Initial State-nya bebas)
2. Selesaikan graf di bawah ini menggunakan algoritma IDS berikut dengan hasil setiap langkah (Traversal dan Tree) dimulai dari node A (k=3) !

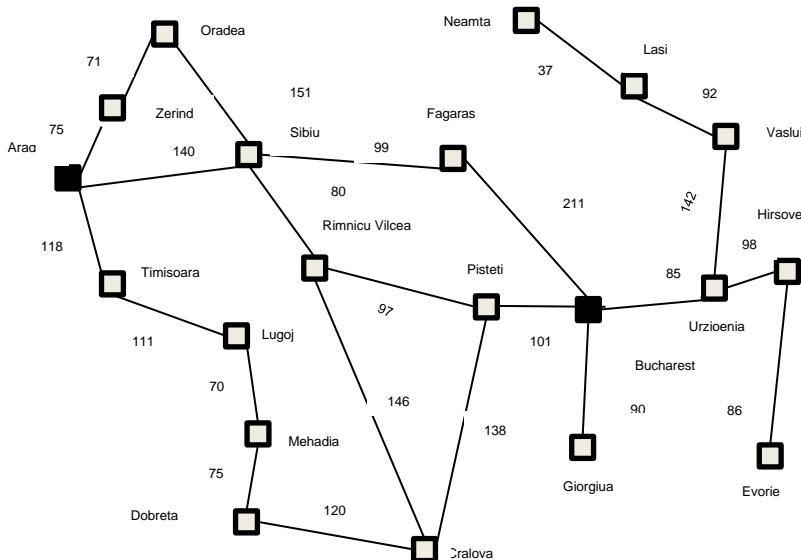


BAB 4 Informed Search

4.1 Best-first search

Prinsip best-first search dapat dilakukan node expansion terhadap node di fringe yang nilai $f(n)$ -nya paling kecil. Ide dasarnya yaitu $f(n)$ adalah sebuah evaluation function yang menyatakan perkiraan seberapa “bagus” sebuah node. Kenapa perkiraan? Kalau tidak, bukan search namanya. Implementasi dari fringe adalah sebuah priority queue di mana node disortir berdasarkan $f(n)$. Contohnya yaitu:

- Uniform-cost search
- Greedy (best-first) search
- A* search



Gambar 4.1 Contoh Heuristic Function

Kunci keberhasilan best-first search terletak di heuristic function. Heuristic adalah rule of thumb atau dapat juga sebagai “kiat-kiat sukses” atau “tips-tips keberhasilan” dengan adanya informasi tambahan bagi si agent (agar lebih sukses) untuk menjadi informed search.

Heuristic function $h(n)$ adalah fungsi yang menyatakan estimasi cost dari n ke goal state. Ada banyak kemungkinan heuristic function untuk sebuah masalah. Contoh dari *Heuristic Function* yaitu:

Straight-line distance to Bucharest:

Arab	366	Mehadia	241
Bucharest	0	Neamt	234
Cralova	160	Oradea	380
Dobreta	242	Pitesti	98
Eforie	161	Rimnicu Vilcea	193
Fagaras	178	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Lasi	226	Vaslui	199
Lugoj	244	Zerind	374

Keterangan:

Sebuah heuristic function untuk agent turis Rumania: $h_{SLD}(n) = \text{jarak straight-line distance dari } n \text{ ke Bucharest.}$

4.2 Algoritma Min-Max

Definisi algoritma min-max ini rekursif, dengan base case pada terminal state. Untuk menghitung MINIMAX VALUE pada initial state, harus depth-first search seluruh game tree.

- Complete? Ya, kalau game tree-nya finite
- Optimal? Ya, asumsi lawan musuh optimal juga. (Kalau tidak? “Lebih optimal”!)
- Time complexity? $O(bm)$
- Space complexity? $O(bm)$ (atau $O(m)$ dgn. backtracking)
- Contoh dalam catur: $b \approx 35$, $m \approx 100 \rightarrow$ pencarian strategi optimal berdasarkan Minimax tidak feasible!

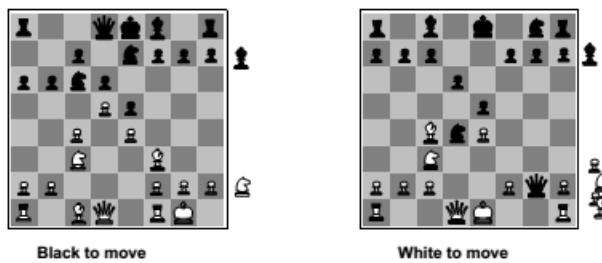
Biasanya dalam suatu permainan ada batasan waktu. Andaikan ada agent bermain catur yang diberi 100 detik untuk “berpikir” tiap langkah. Sistem bisa memproses 10^4 node/detik $\rightarrow 10^6$ node/langkah. Kita bisa melakukan aproksimasi sebagai berikut:

- Cutoff: batasi depth yang diproses (\approx IDS).

- Evaluation function: prediksi dari nilai utility function (tidak perlu sampai ke terminal state).

Biasanya, evaluation function berupa kombinasi linier dari fitur-fitur sebuah state:

- $\text{Eval}(s) = w_1f_1(s) + w_2f_2(s) + \dots + w_nf_n(s) =$
- Mis. untuk catur:
- $w_1 = 1, f_1 = \text{jumlah pion putih} - \text{jumlah pion hitam}$
- $w_2 = 3, f_2 = \text{jumlah gajah putih} - \text{jumlah gajah hitam}$



Gambar 4.2 Contoh Implementasi Algoritme Min-Max pada Permainan Catur

Contoh penerapan algoritma Min Max. Diberikan sebuah situasi permainan seperti di bawah ini:

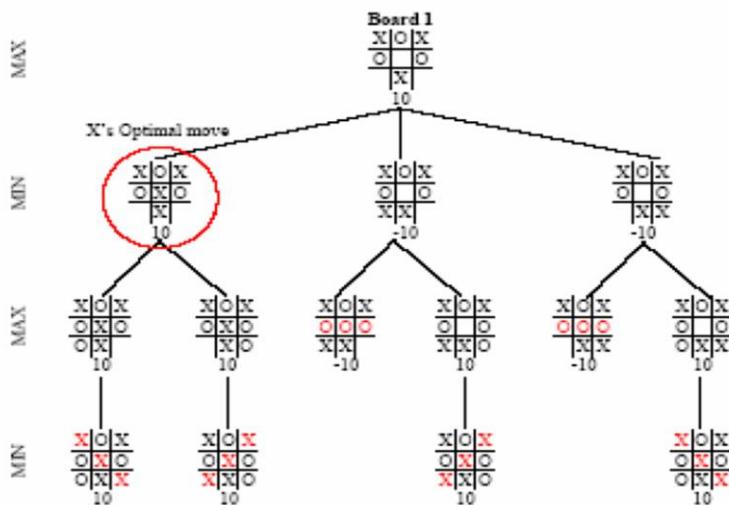
Tabel 4.1 Implementasi Algoritme Min-Max pada Permainan

X	O	X
O		O
	X	

Keterangan:

- X (max player) sedang dalam giliran untuk melanjutkan permainan. Berikan semua situasi berikutnya yang mungkin untuk X
- Pilihlah jalur yang tepat sesuai dengan algoritma minmax, jika diketahui fungsi utilitas untuk situasi menang untuk X = +10, kalah = -10, dan draw = 0.

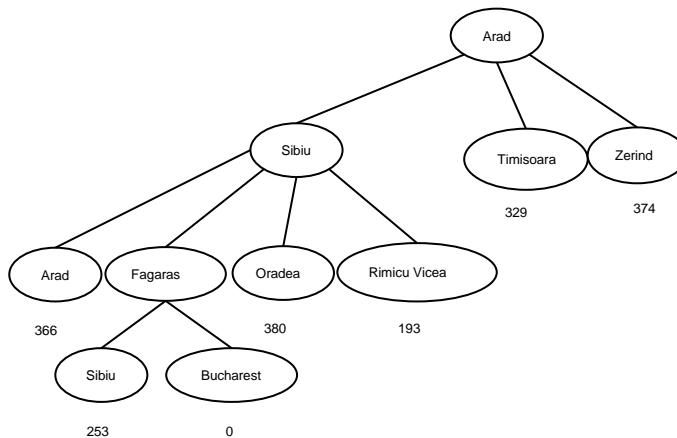
Dapat diselesaikan dengan cara berikut:



Gambar 4.3 Penyelesaian menggunakan Algoritme Min-Max

4.3 Greedy best-first search

Prinsip greedy best-first search adalah melakukan node expansion terhadap node di fringe yang nilai $h(n)$ -nya paling kecil. Greedy best-first search selalu memilih node yang kelihatannya paling dekat ke goal.



Gambar 4.4 Contoh untuk Greedy Best-First Search

- Properties of greedy best-first search:

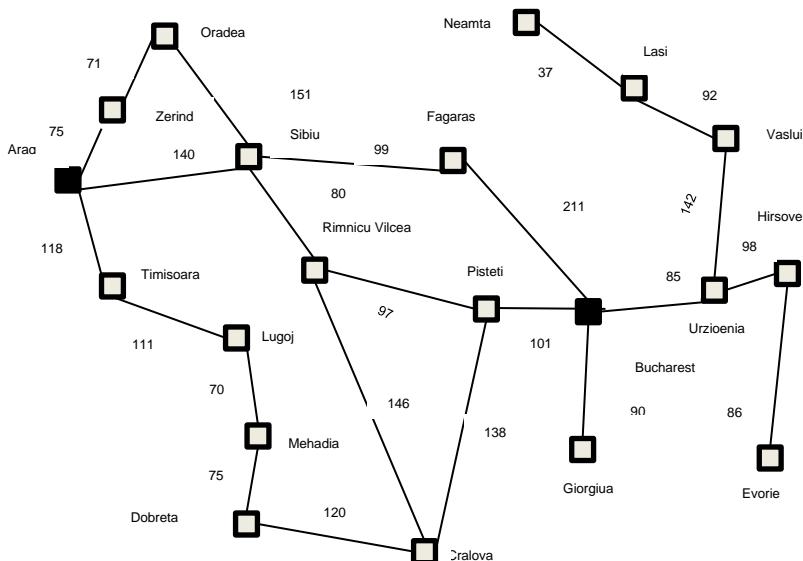
- Complete? Ya, jika state space terbatas dan pengulangan state-nya ditangani. (Lihat Neamt → Oradea)
- Time complexity? Secara teoritis, $O(b^m)$, tetapi heuristic function yang baik akan lebih mempercepat.
- Space complexity? $O(b^m) \rightarrow$ semua node disimpan di memory
- Optimal? Tidak.

4.4 A* search

Prinsip A* search: Hindari node yang berada di path yang “mahal”. Evaluation function $f(n) = g(n) + h(n)$:

- $g(n)$ = Path cost ke n
- $h(n)$ = Estimasi path cost dari n ke goal
- $f(n)$ = Estimasi total cost melalui n

Contoh penelusuran A* search:

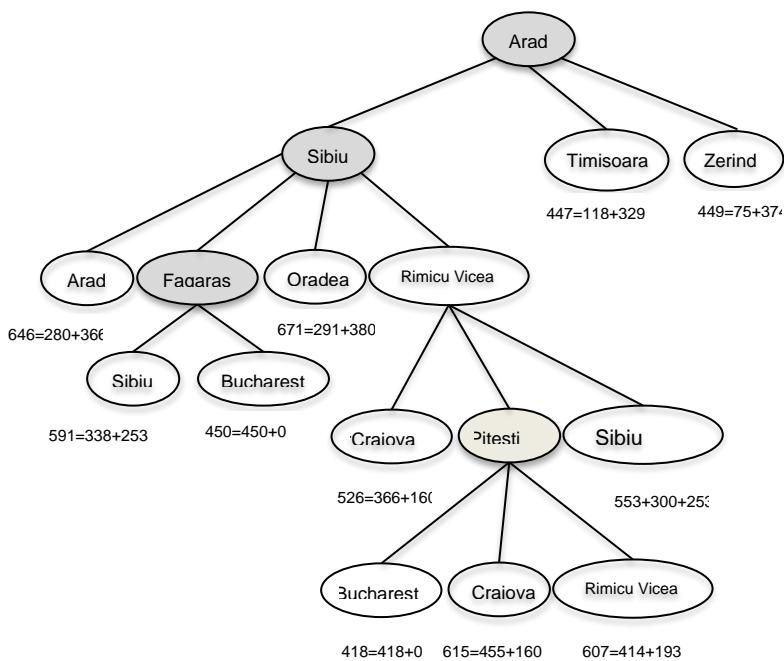


Gambar 4.5 Contoh Penelusuran A*Search

Straight-line distance to Bucharest:

Arab	366	Mehadia	241
Bucharest	0	Neamt	234

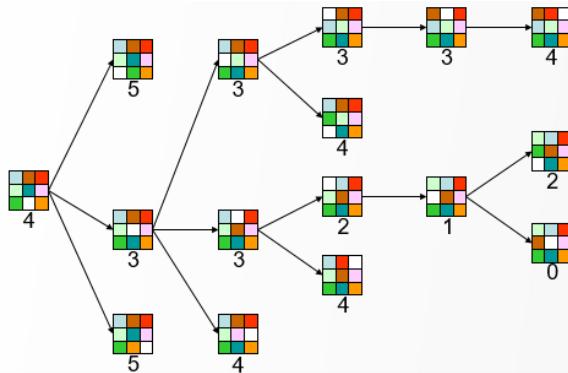
Cralova	160	Oradea	380
Dobreta	242	Pitesti	98
Eforie	161	Rimnicu Vilcea	193
Fagaras	178	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Lasi	226	Vaslui	199
Lugoj	244	Zerind	374



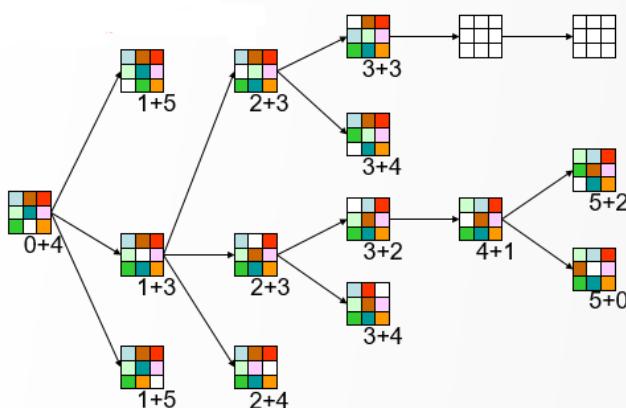
Gambar 4.6 Contoh Penyelesaian Algoritme A*Search

- Properties of A*:
 - Complete? Ya, kecuali jumlah node di mana $f \leq f(G)$ tak terbatas.
 - Time complexity? Eksponensial dalam ($\text{error } h \times \text{jumlah step solusi}$).
 - Space complexity? $O(b^m) \rightarrow$ semua node disimpan di memory.

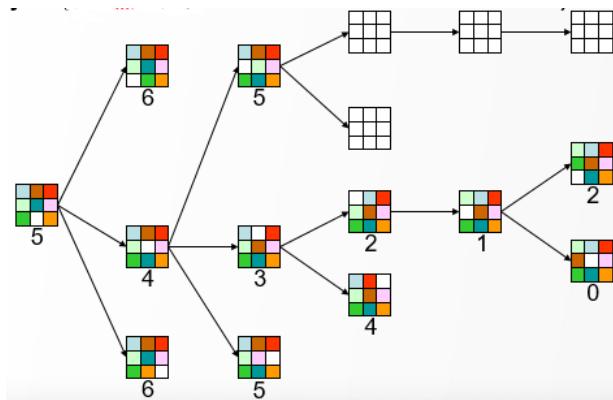
- Optimal? Ya.
 - A* meng-expand semua node di mana $f(n) < C^*$
 - A* (mungkin) meng-expand beberapa node di mana $f(n) = C^*$
 - A* tidak pernah meng-expand node di mana $f(n) > C^*$
- $f(N) = h(N) = \text{jumlah angka yang salah posisi}$



- $f(N) = g(N) + h(N)$ dengan $h(N) = \text{jumlah angka yang salah posisi}$



- $f(N) = h(N) = \sum \text{jarak semua angka dari posisi yang benar} (\text{base Manhattan Distance})$



4.5 Heuristics

Membandingkan dua heuristic

1. h_1 dan h_2 sama-sama admissible. Mana yang lebih baik?
Bandingkan jumlah node yang di-expand:

Tabel 4.2 Perbandingan Jumlah Node

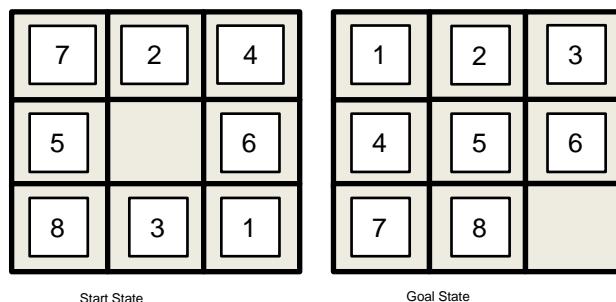
D	IDS	A*(h_1)	A*(h_2)
12	3,473,941	539	113
24	54,000,000,000	39,135	1,641

2. Jika $h_2(n) \geq h_1(n)$ untuk semua n (dan keduanya admissible), dikatakan bahwa h_2 men-*dominate* h_1 dan lebih baik untuk search.
3. Semakin besar nilai $h(n)$, semakin dekat ke $h^*(n)$, semakin banyak node yang tidak di-expand (di-prune), semakin efisien search-nya!

Contoh Admissible Heuristic

- $h(n)$ untuk 8-puzzle
 1. $h_1(n)$: jumlah angka yang salah posisi.

2. $h_2(n)$: jumlah jarak semua angka dari posisi yang benar (base Manhattan Distance)



Gambar 4.7 Permainan 8-Puzzel

Tabel 4.3 Menentukan Posisi

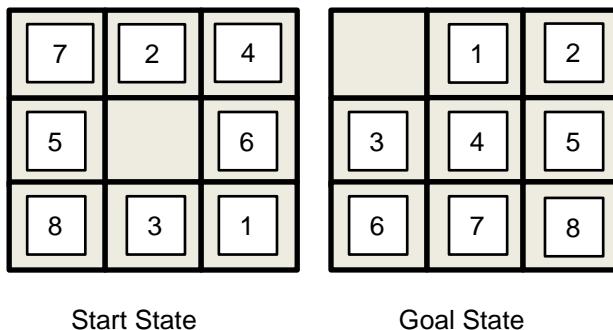
(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)
(2,0)	(2,1)	(2,2)

$$D_{man}(x, y) = \sum_{j=1}^d |x_j - y_j|$$

Diketahui posisi tile 1 di Start State (2,2) dan di Goal State (0,0)

- Hitung $D_{man}(1) = |2-0| + |2-0| = 2 + 2 = 4$
- $h_1(s) = 6$
 - $h_2(s) = D_{man}(1) + D_{man}(2) + D_{man}(3) + D_{man}(4) + D_{man}(5) + D_{man}(6) + D_{man}(7) + D_{man}(8) = 4 + 0 + 3 + 3 + 1 + 0 + 2 + 1 = 14$

- Latihan Admissible Heuristic
 1. Perhatikan 8-puzzle berikut:



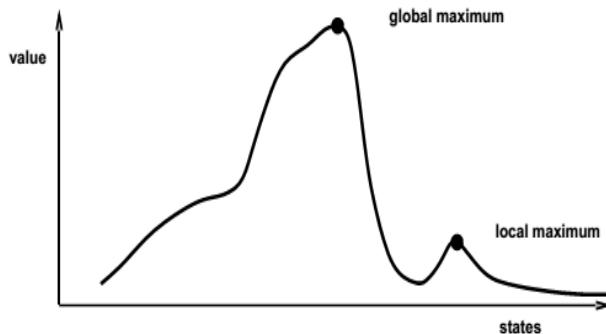
Tentukan $h_1(s)$ dan $h_2(s)$! Menggunakan rumus berikut!

$$D_{man}(x, y) = \sum_{j=1}^d |x_j - y_j|$$

4.6 Hill-Climbing Search

Nilai sebuah node adalah $h(n)$ (heuristic function). Bayangkan seorang penderita amnesia mendaki gunung dengan kabut tebal (thick fog).

- State: posisi koordinat (X, Y)
- $h(n)$: ketinggian pendaki

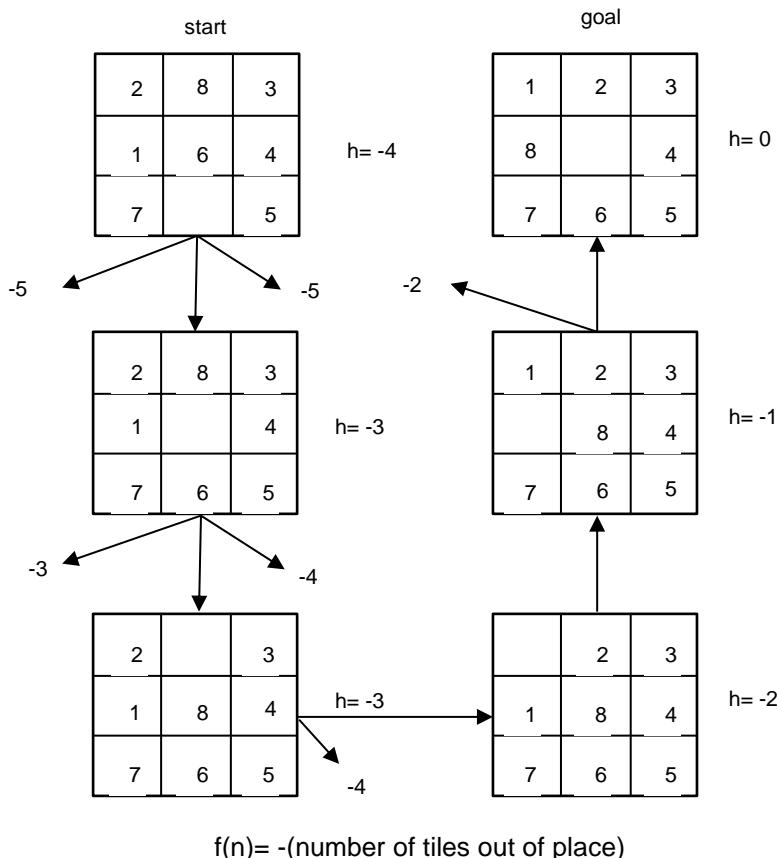


Gambar 4.8 Konsep Hill-Climbing Search

Konsep penting adalah state space sebagai landscape. Disebut juga greedy local search. Tergantung pilihan initial state, hill-climbing bisa terperangkap dalam local maximum.

- Local maximum: tidak ada tetangga yang lebih baik, tetapi bukan solusi optimal.
- Plateau (dataran): semua tetangga sama baiknya.

Contoh Hill-Climbing Search:



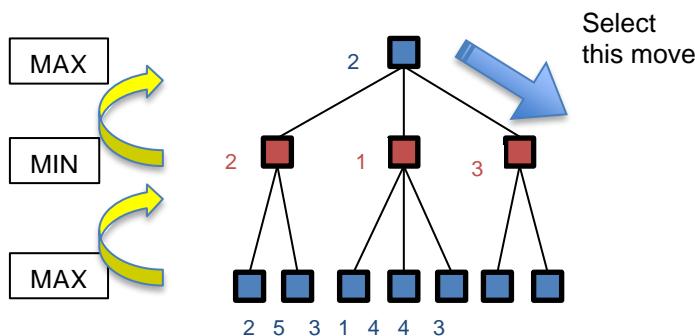
Gambar 4.9 Contoh Penyelesaian Hill-Climbing Search

4.7 Alpha Beta Pruning (Game Search)

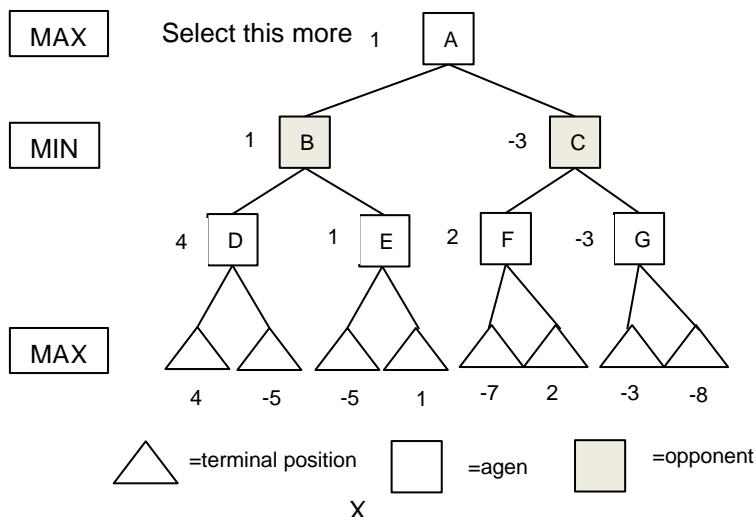
May 11th 1997, Gary Kasparov lost a six match game to Deep blue two wins for deep blue, one win for Kasparov and three draws

(Garry Kasparov and Deep Blue, 1997). Contoh alpha betha pruning pada game Minimax:

Misal ada 2 pemain MAX (computer) dan MIN (opponent) yang memiliki sifat deterministic, perfect information. Pilih suatu depth-bound (misal 2) dan lakukan evalution function. Maka yang pertama dapat dilakukan yaitu membangun tree sampai depth-bound. Kemudian hitung evaluation function untuk tiap daunnya, kemudian lakukan penelusuran evaluation function ke atas untuk taking minima in MIN dan taking maxima in MAX.

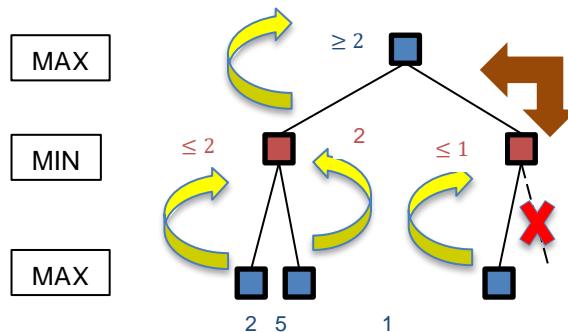


Gambar 4.10 Contoh Implementasi MIN-MA



Gambar 4.11 Contoh Implementasi Min-Max

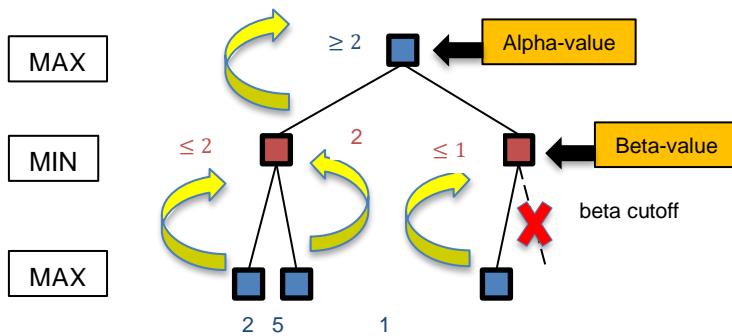
Algoritma Alpha-beta Pruning, secara umum digunakan untuk optimasi Algoritma Mini-max. Sebagai pengganti dari proses yang pertama membuat keseluruhan tree (up to depth-level). Kemudian melakukan semua penelusuran pada semua bagian tree dari antar daun pada tree, dengan penelusuran berdasarkan nilai. Ada point yang utama dalam algoritma ini yaitu beberapa bagian dari tree kemungkinan akan menyediakan informasi nilai yang sama yang sebenarnya (non-generated) atau tidak perlu ditelurusi lagi, karena bisa dikatakan (redundant). Kemudian untuk principles yaitu bangkitkan tree depth-first, dari kiri ke kanan (telusuri dari values yang paling bawah kiri sbg initial untuk estimasi parent node).



Gambar 4.12 Min-Max untuk Estimasi Parent Node

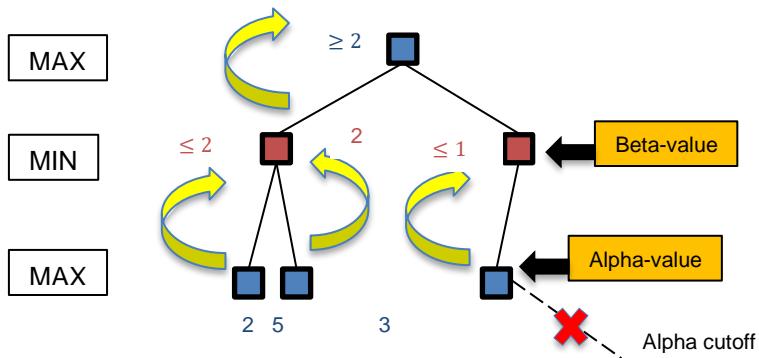
Keterangan:

- MIN-value (1) sudah lebih kecil dari MAX-value pada parent (2)
- MIN-value untuk selanjutnya hanya dapat diturunkan
- MAX-value hanya diperbolehkan untuk dinaikkan
- Tidak ada yang perlu diperhitungkan lagi untuk selanjutnya dibawah node ini
- Jika suatu ALPHA-value dari node parent \geq Beta-value dari suatu node keturunannya, maka stop pembangkitan (x) dari child dari node keturunannya tersebut seperti pada gambar di bawah ini:



Gambar 4.13 Alpha-Value dari Node Parent \geq Beta Value

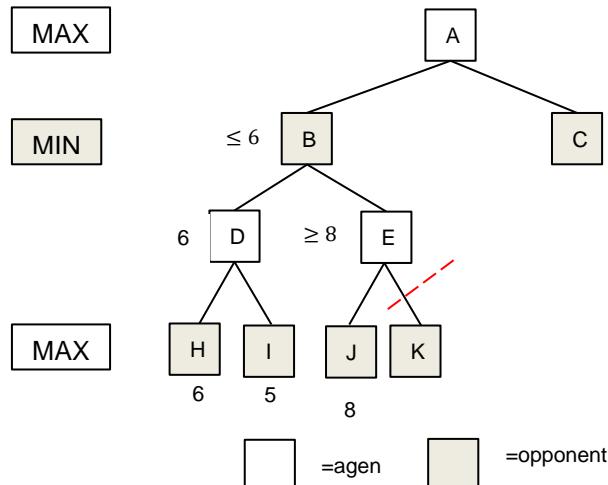
- Jika suatu Beta-value dari node parent \leq Alpha-value dari node keturunannya, maka: stop pembangkitan (x) dari child dari node keturunannya tersebut seperti pada gambar dibawah ini:



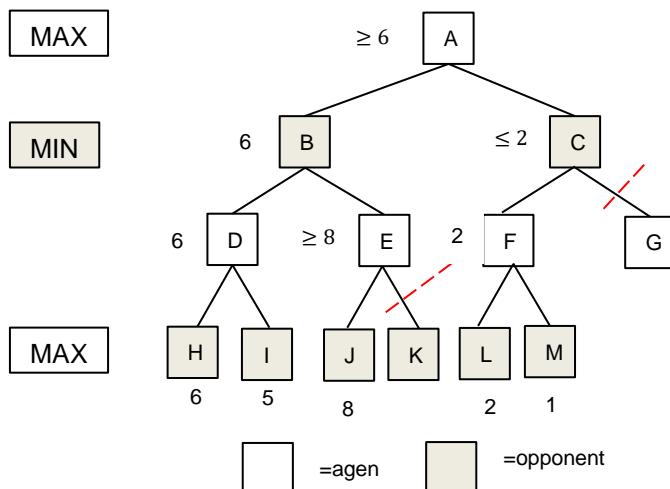
Gambar 4.14 Alpha-Value dari Node Parent \leq Beta Value

Berikut adalah contoh penyelesaian menggunakan algoritme Alpha-Beta Pruning:

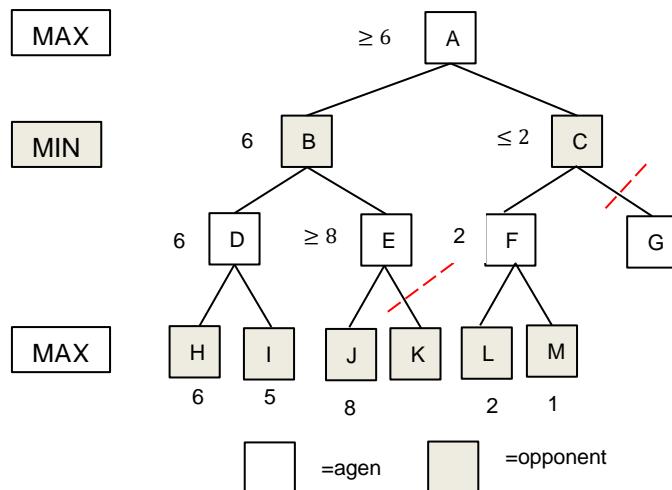
- Step 1



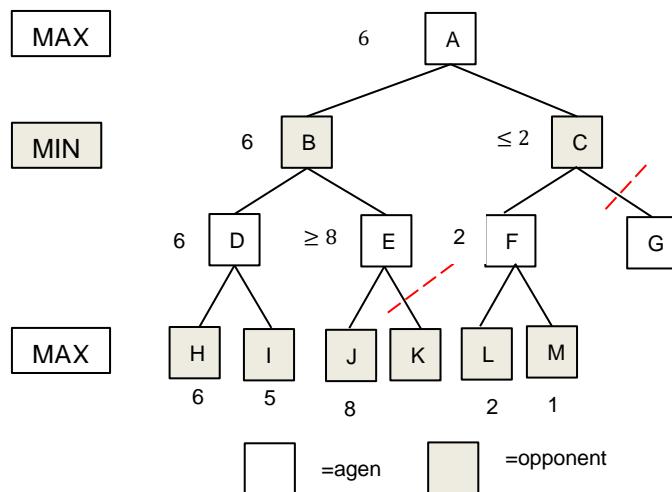
- Step 2



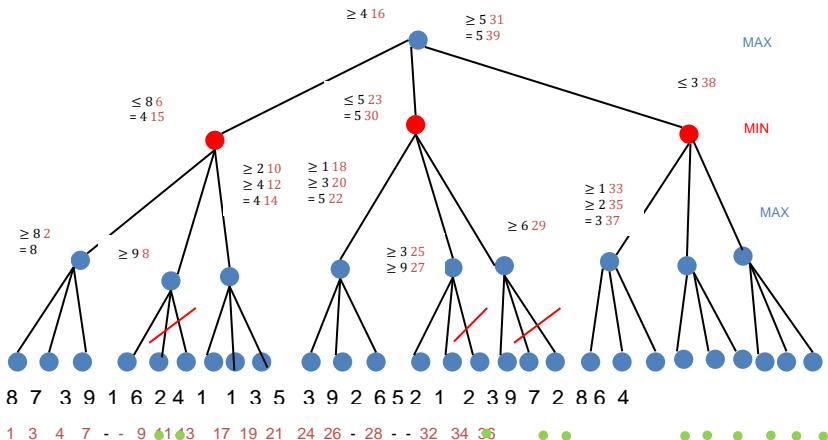
- Step 3



- Step 4

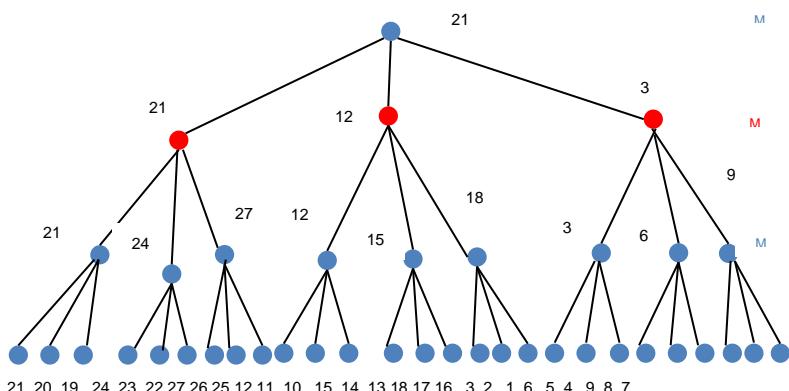


Contoh penggunaan Mini-Max dengan $\alpha - \beta$ at work:



Gambar 4.15 Implementasi Mini-Max dengan $\alpha - \beta$ at work

Contoh penyelesaian masalah tree yang sempurna:



Gambar 4.16 Contoh Penyelesaian Masalah Tree

Kesimpulan:

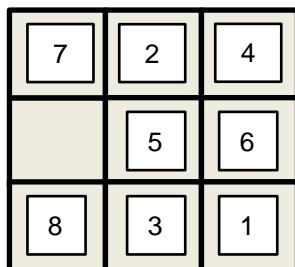
1. Alpha-beta algorithm melakukan perhitungan hampir sama dengan minimax, dan lebih efisien karena ada pemotongan pada percabangan yang tidak relevan.
2. Biasanya, keseluruhan game tree tidak diekspansi, pencarinya adalah melakukan *cut off* pada beberapa titik dan menghitung

evaluation function untuk estimasi nilai mana yang lebih baik (utility) dari suatu kondisi.

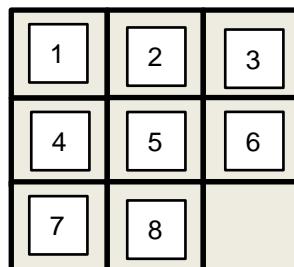
3. Sejauh ini, berikut adalah tips untuk mendapatkan kemungkinan pencarian yang baik dan efisien:
 - Pilih metode/teknik pencarian yang bagus.
 - Menyediakan info/heuristic jika memungkinkan.
 - Menerapkan pemotongan pada percabangan yang tidak relevan (alpha-beta pruning).

4.8 Latihan Individu

1. Selesaikan 8-Puzzle di bawah ini dengan algoritma A* Search berdasarkan teknik “ $f(N) = h(N) = \text{jumlah angka yang salah posisi}$ ” (min. ada 3 depth):

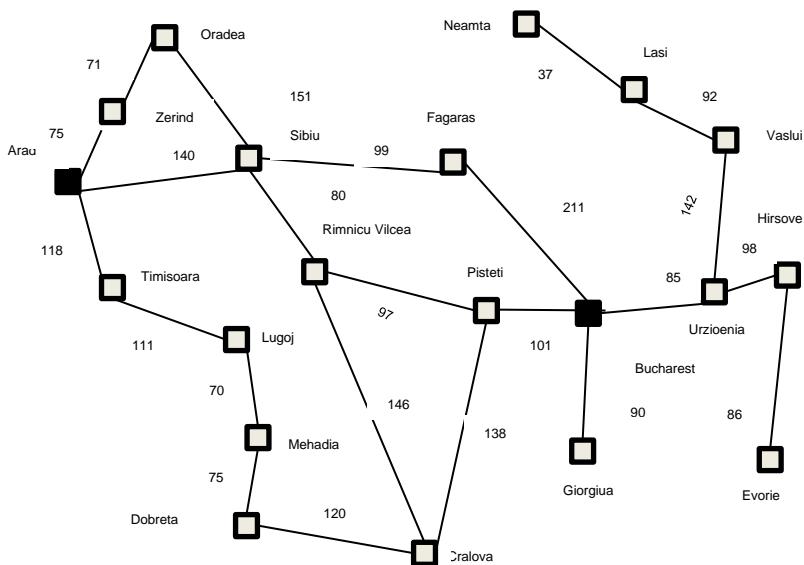


Start State



Goal State

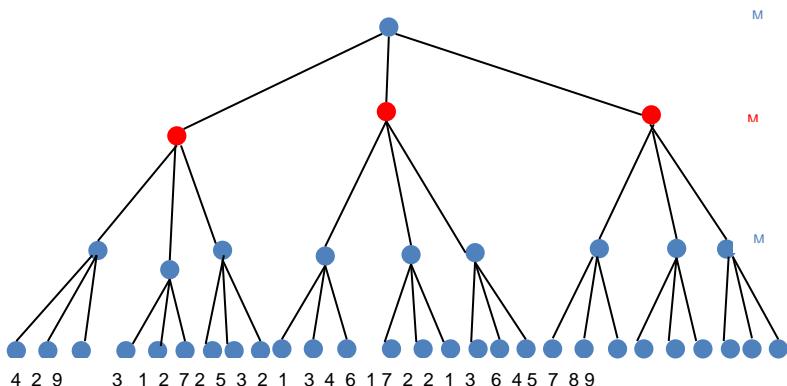
2. Perhatikan Peta berikut! (Optional)



Straight-line distance to Bucharest:

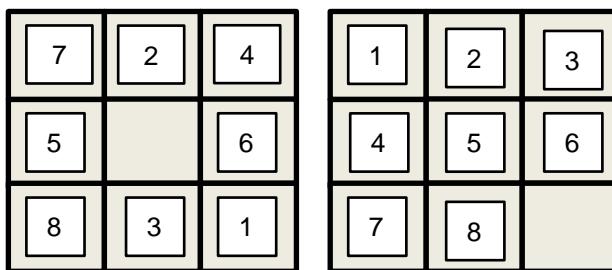
Arab	366	Mehadia	241
Bucharest	0	Neamt	234
Cralova	160	Oradea	380
Dobreta	242	Pitesti	98
Eforie	161	Rimnicu Vilcea	193
Fagaras	178	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Lasi	226	Vaslui	199
Lugoj	244	Zerind	374

- Buatlah Tree-nya, lalu selesaikan proses searching dari “Oradea ke Bucharest” menggunakan Algoritma Greedy Best-First Search!
3. Selesaikan tree berikut dengan detail seperti pada slide 14, menggunakan Alpha-Beta Pruning:



4.9 Tugas Kelompok

1. Berdasarkan latihan individu di slide 36, selesaikan proses searching dari “Oradea ke Bucharest” menggunakan Algoritma A* Search!
2. Selesaikan 8-Puzzle di bawah ini dengan algoritma A* Search berdasarkan 2 teknik berikut (setiap teknik min. ada 5 depth):



Start State

Goal State

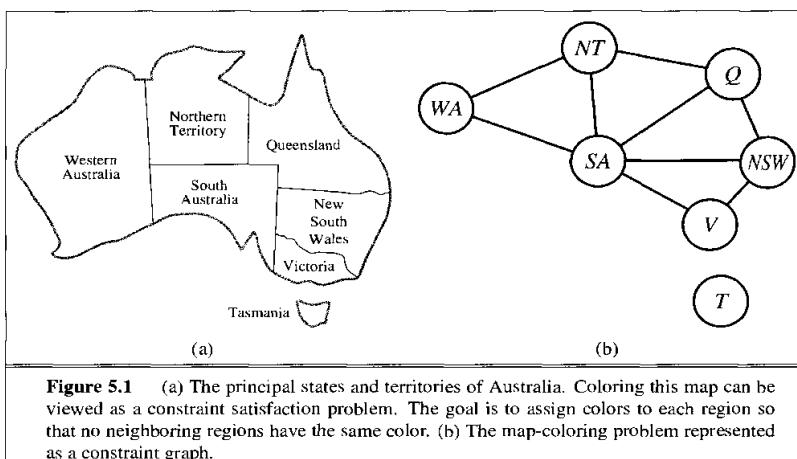
- a. $f(N) = g(N) + h(N)$ dengan $h(N) = \text{jumlah angka yang salah posisi}$
- b. $f(N) = h(N) = \text{jumlah jarak semua angka dari posisi yang benar (base Manhattan Distance)}$

Note: kerjakan 1 dari 2 (a atau b)

BAB 5 Constraint Satisfaction Problem

5.1 Constraint Satisfaction Problems (CSP)

CSP atau Constraint Satisfaction Problem adalah permasalahan yang tujuannya adalah mendapatkan suatu kombinasi variabel-variabel tertentu yang memenuhi aturan-aturan (*constraints*) tertentu. State didefinisikan dengan *variables* X_i yang mempunyai values dari domain D_i . Goal Test adalah sebuah himpunan *constraints* yang memberikan kombinasi yang diijinkan untuk mengisi variabel. Contoh CSP dapat dilihat pada Gambar 6.1.



Gambar 5.1 Peta Mewarnai

Keterangan:

- Variabel: WA, NT, Q, NSW, V, SA, T
- Ranah: $D_i = \{\text{red, green, blue}\}$
- Syarat: 2 wilayah yang berbatasan harus berbeda warna:
 1. $WA \neq NT, NT \neq SA, \dots$
 2. $(WA, NT) \in \{(\text{red, green}), (\text{red, blue}), (\text{green, red}), (\text{green, blue}), \dots\}$

Pada Gambar 6.2 merupakan contoh solusi CSP dalam mewarnai peta pada Gambar 6.1.

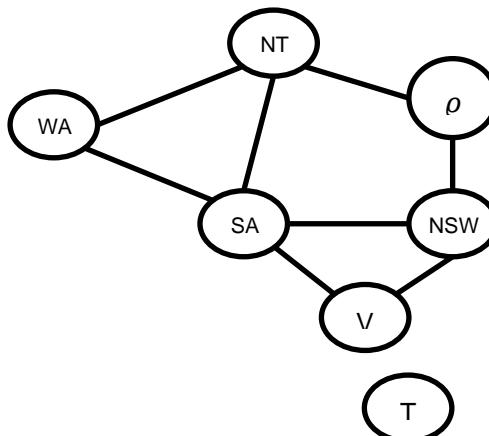


Gambar 5.2 Solusi CSP dalam mewarnai Peta

Keterangan:

Solusi adalah pemberian nilai setiap variabel yang memenuhi syarat, mis: {WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green}.

Pada Gambar 6.3 merupakan contoh CSP dalam Constraint Graph pada pewarnaan peta pada Gambar 6.1.



Gambar 5.3 Constraint Graph pada Pewarnaan Peta

Keterangan:

- Binary CSP: sebuah constraint menyangkut hubungan maks. 2 variable.
- Constraint graph: representasi di mana node adalah variable, edge adalah constraint,

5.2 Pencarian Backtracking untuk CSP

Algoritma *backtracking search* (penelusuran kembali) adalah suatu bentuk algoritma depth-first-search. Jika solusi partial melanggar constraint, backtracking melakukan langkah kembali ke solusi partial sebelumnya. Pada algoritma backtracking, teknik *look ahead* digunakan untuk meramalkan efek pemilihan variabel branching untuk mengevaluasi nilai-nilai variabel tersebut.

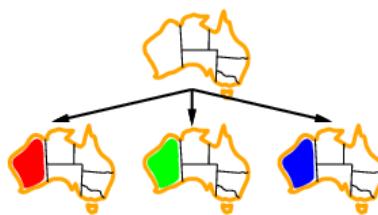
Forward checking adalah salah satu cara untuk melakukan *look ahead*. *Forward checking* mencegah terjadinya konflik dengan melarang nilai-nilai yang belum diisikan ke variable untuk dipakai. Ketika suatu nilai diisikan ke suatu variabel, nilai yang berada di domain dari variabel yang konflik tersebut akan dihilangkan dari domain.

Minimun Remaining Value adalah suatu teknik yang dikembangkan untuk menangani masalah kemungkinan besar gagal pada pencarian menggunakan CSP. MRV berkerja dengan memilih variabel yang memiliki domain legal dan paling sedikit (memiliki kemungkinan untuk membuat suatu *dead-end* paling besar) untuk diisikan terlebih dulu.

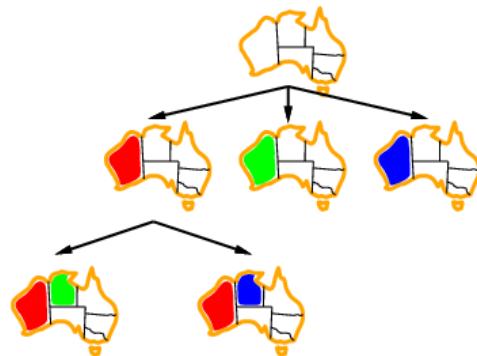
Variable assignment berlaku komutatif, dalam arti [WA=red lalu NT=green] sama saja [NT=green lalu WA=red]. Pada tiap level, hanya perlu meng-assign satu variabel b = d. Depth first search pada CSP dengan assignment satu variabel tiap level disebut backtracking search. Pada Gambar 6.4 berikut merupakan contoh dari Backtraking:



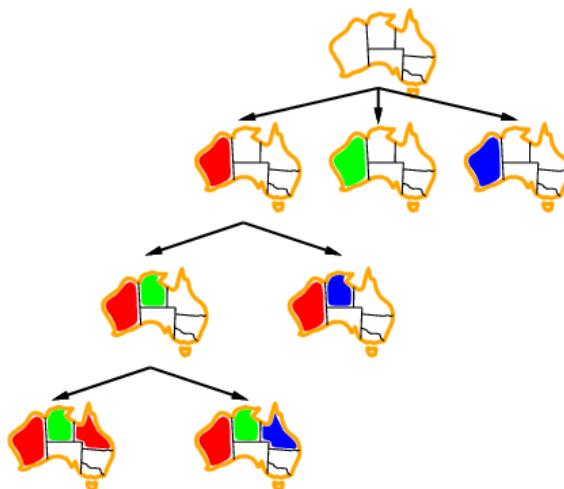
Gambar 5.4 Backtracking Step 1



Gambar 5.5 Backtracking Step 2



Gambar 5.6 Backtracking Step 3



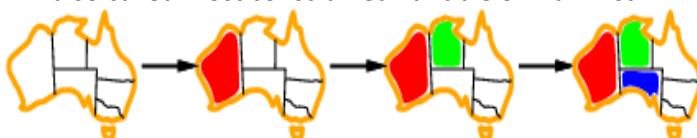
Gambar 5.7 Backtracking Step 4

- Untuk memperbaiki Kinerja Backtracking maka dapat menggunakan:

1. Urutan pemilihan variable dan nilai mempengaruhi kinerja backtracking
2. Terdapat beberapa strategi yang berlaku secara umum (general-purpose):
 - a. Variable mana yang perlu di-assign terlebih dulu?
 - b. Nilai apakah yang perlu dicoba terlebih dulu?
 - c. Apakah kita bisa mendeteksi kepastian failure lebih awal?
 - d. Apakah kita bisa memanfaatkan struktur masalah?
 - e. CSP? (Representasinya jelas!)

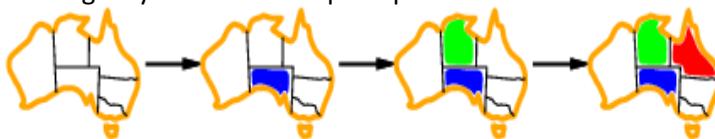
Ada 3 prinsip yang dapat digunakan pada Algoritme Backtracking, yaitu:

1. Most Cotrained Variable
 - Variabel yang paling dibatasi
 - Pilih variable yang memiliki kemungkinan nilai sah paling sedikit dan minimum remaining values (MRV) heuristic also called most constrained variable or “fail first”.



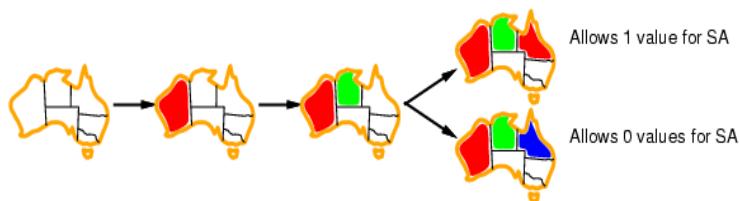
Gambar 5.8 Gambaran Variabel yang paling Dibatasi

2. Most Constraining Variable
 - Variable paling membatasi
 - Pilih variable yang terlibat constraint dengan variable lain (yang belum di-assign) yang paling banyak. Tie – breaker: gunakan kalau ada 2 atau lebih variable yang sama bagusnya berdasarkan prinsip 1.



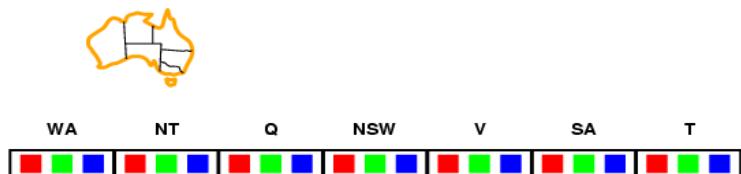
Gambar 5.9 Gambaran Variabel yang Membatasi

3. Least Constraining Value
 - Pilih nilai yang menimbulkan batasan kemungkinan nilai variable lain (yang belum di-assign) yang paling sedikit.
 - Jika ketiga prinsip ini digunakan, masalah n-queens dengan n=1000 bisa diselesaikan!



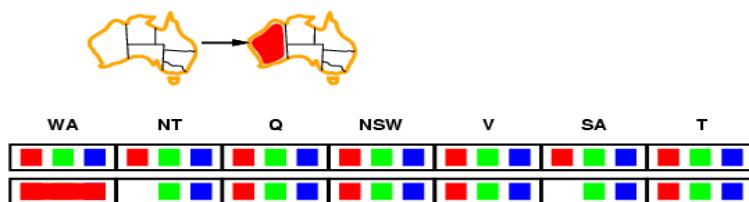
Gambar 5.10 Gambaran Least Constraining Value

Untuk Forward Checking supaya mencatat kemungkinan nilai sah untuk semua variable yang belum di-assign. Jika ada sebuah variable yang tidak ada kemungkinan nilai sah, langsung failure (backtrack).

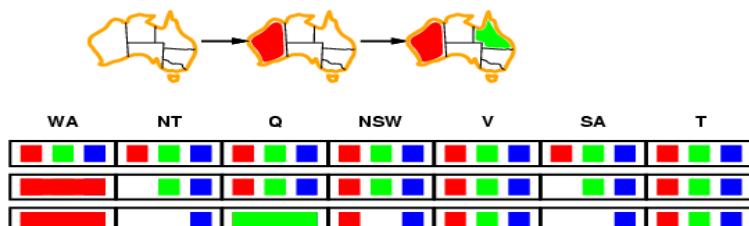


Gambar 5.11 Forward Checking pada Pewarnaan Peta

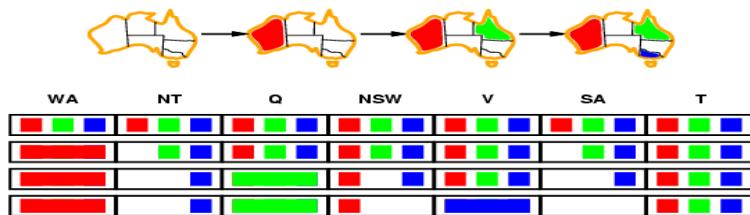
- Untuk memperbaiki kinerja Forward checking maka dapat menggunakan:
 - Simpan nilai valid untuk variable yang belum di-assign.
 - Bila salah satu variable tidak mempunyai kemungkinan nilai yang valid maka search dihentikan.



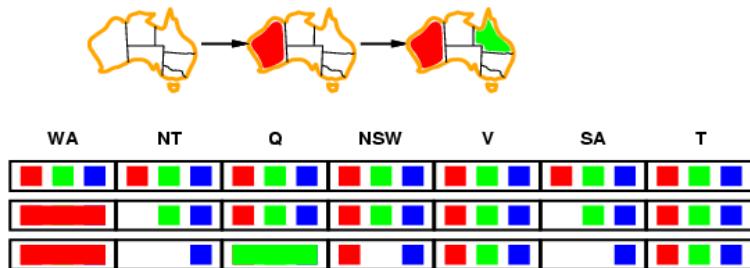
Gambar 5.12 Forward Checking Step 1



Gambar 5.13 Forward Checking Step 2



Gambar 5.14 Forward Checking Step 3



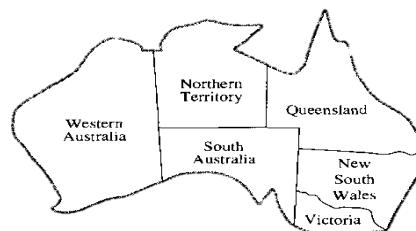
Gambar 5.15 Forward Checking Step 4

Forward checking memberikan informasi dari variabel yang dialokasi, namun tidak dapat mendeteksi kegagalan sebelumnya.

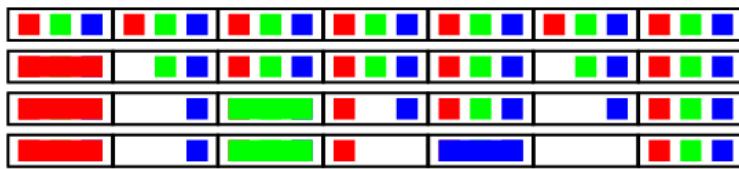
Note:

- Forward checking mem-propagasi (meneruskan)
- informasi dari variable yang sudah di-assign ke yang belum.
- Secara umum, ini disebut constraint propagation.
- Namun, tidak semua failure bisa di-deteksi secara dini.
- Arc Consistency

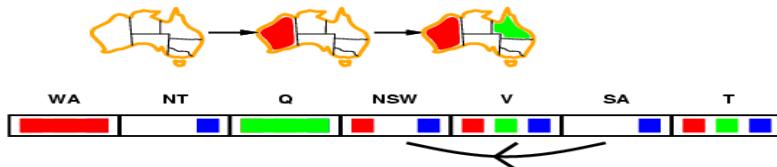
Bentuk sederhana dari propagasi, membuat arc consistent $X \rightarrow Y$ is consistent iff for every value x of X there is some allowed y .



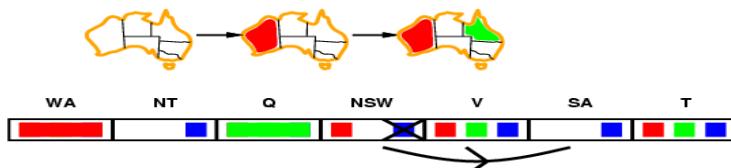
Gambar 5.16 Pewarnaan Peta pada permasalahan



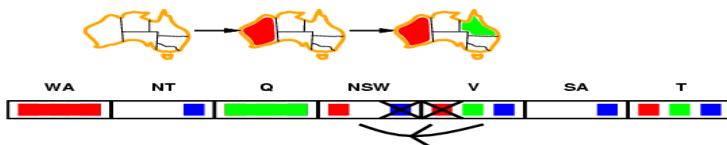
Gambar 5.17 Hasil dari Forward Checking



Gambar 5.18 Arc Konsistency Step 1

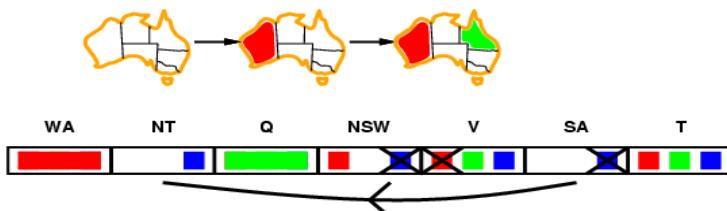


Gambar 5.19 Arc Konsistency Step 2



Gambar 5.20 Arc Konsistency Step 3

Jika X kehilangan suatu nilai, neighbors dari X perlu diperiksa ulang.



Gambar 5.21 Arc Konsistency Step 4

Arc consistency detects failure lebih dini dari pada forward checking. Dapat dijalankan sebagai preprocessor atau setelah setiap assignment.

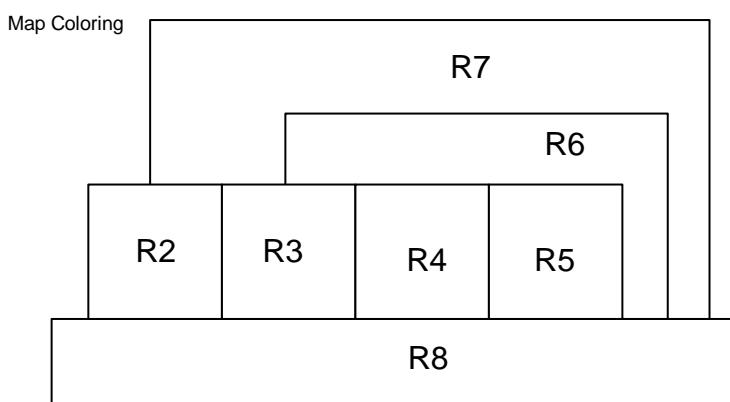
5.3 Local search untuk CSPs

Dengan heuristic yang admissible dan consistent, A* pasti complete dan optimal. Isikan bidang (R1..R7) di atas dengan warna, merah, kuning, hijau, biru. Bidang bertetangga tidak boleh memiliki warna yang sama.

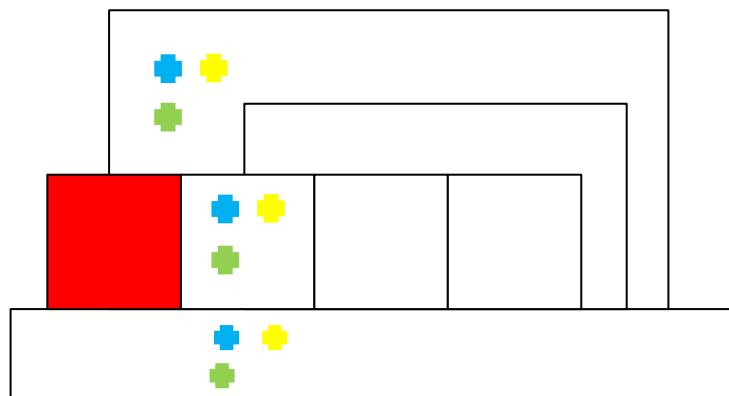
1. Apa variabel yang digunakan?
2. Apa domain yang tersedia?
3. Bagaimana Anda mengevaluasi constraints-nya?

Variabel yang harus diisi: R1,.. R7. Domain yang tersedia: warna (merah, kuning, hijau, biru). Dan berikut adalah contraints:

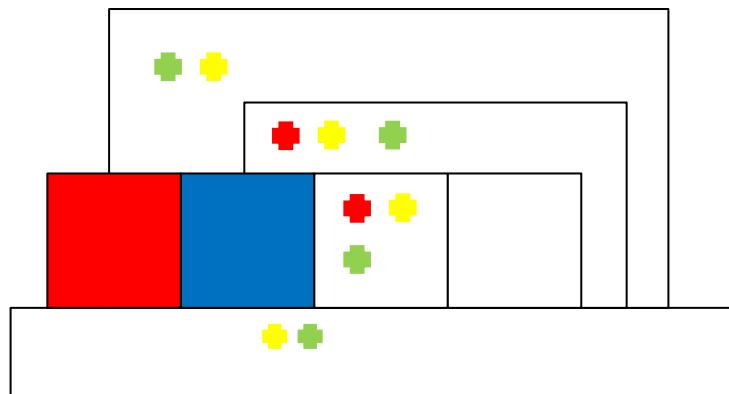
1. $R1 \neq R2, \dots, R7,$
2. $R2 \neq R3,$
3. $R3 \neq R4,$
4. $R4 \neq R5,$
5. $R5 \neq R6,$
6. $R6 \neq R7$



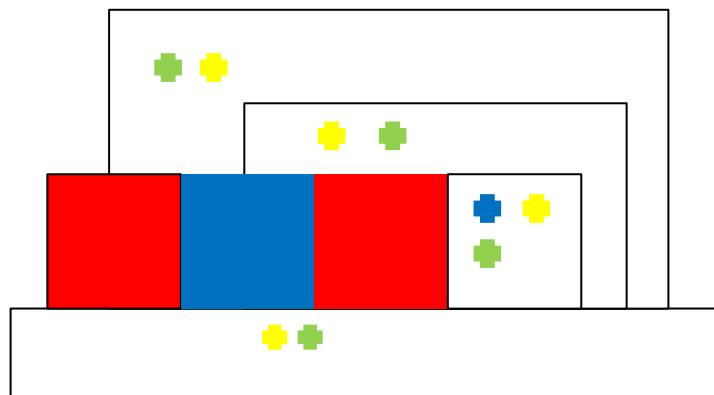
Gambar 5.22 Map Coloring



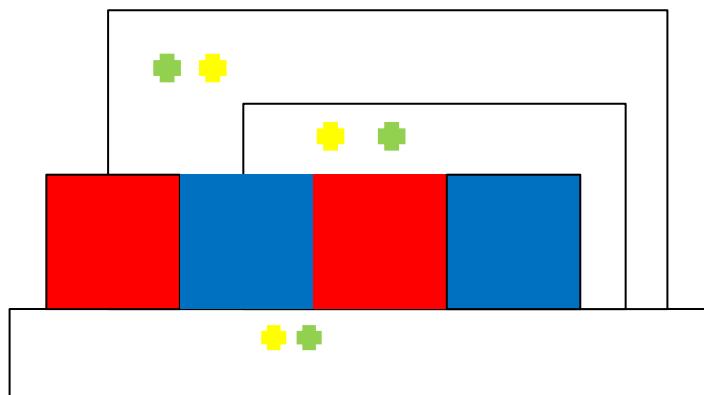
Gambar 5.23 Map Coloring Step 1



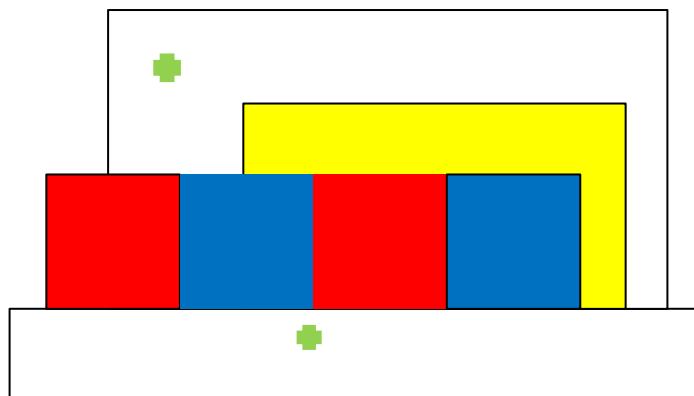
Gambar 5.24 Map Coloring Step 2



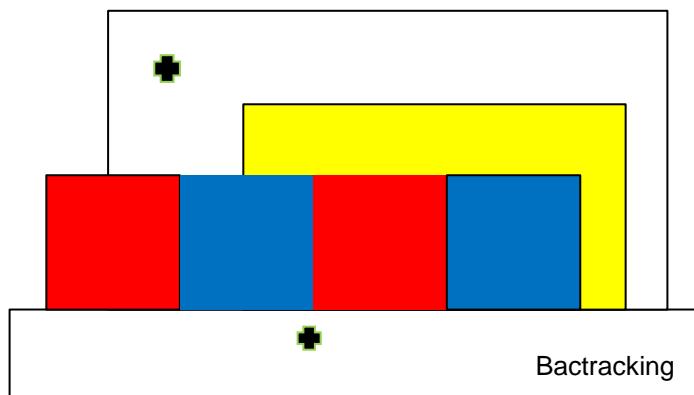
Gambar 5.25 Map Coloring Step 3



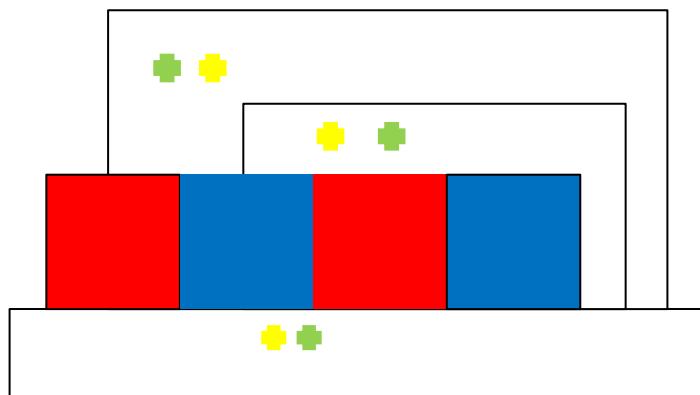
Gambar 5.26 Map Coloring Step 4



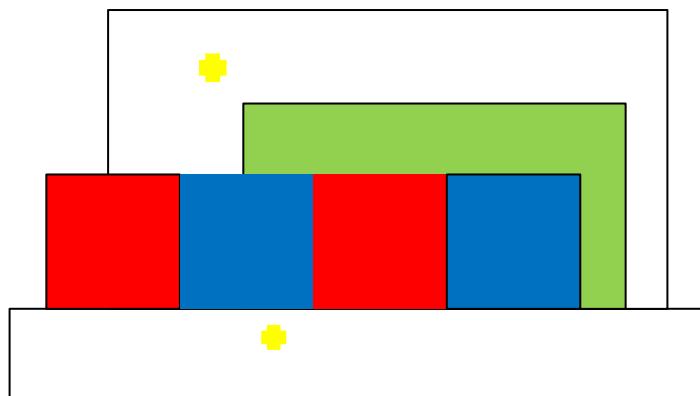
Gambar 5.27 Map Coloring Step 5



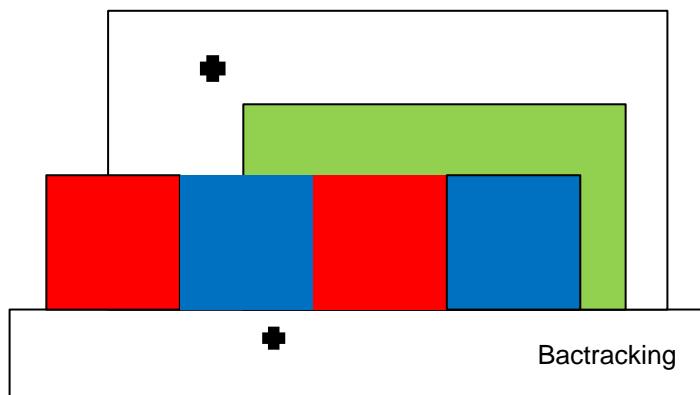
Gambar 5.28 Map Coloring Step 6



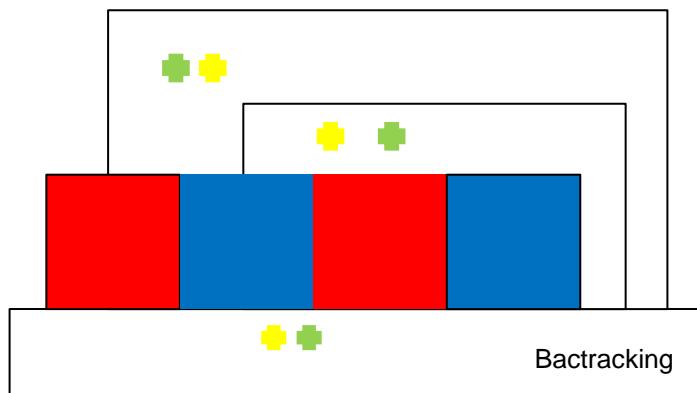
Gambar 5.29 Map Coloring Step 7



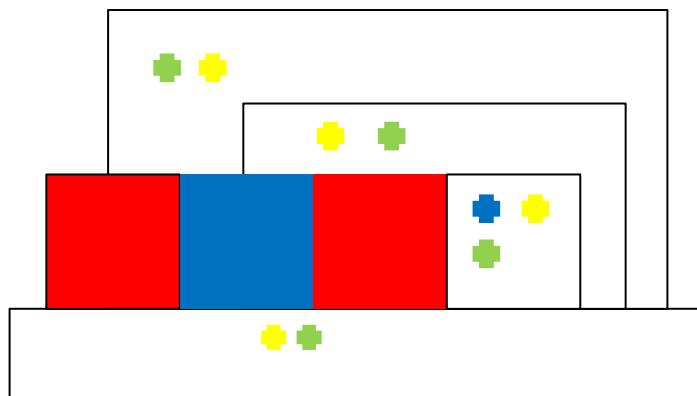
Gambar 5.30 Map Coloring Step 8



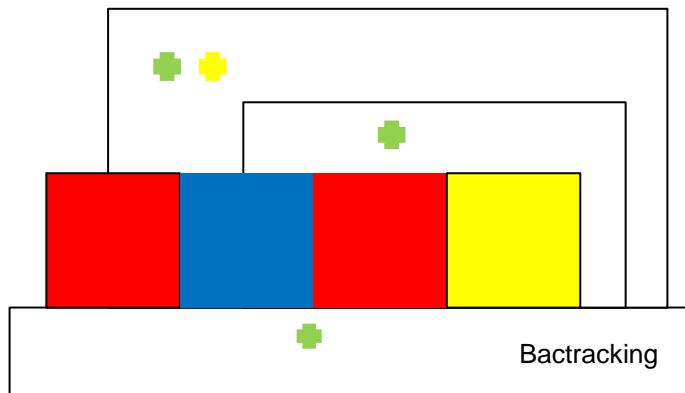
Gambar 5.31 Map Coloring Step 9



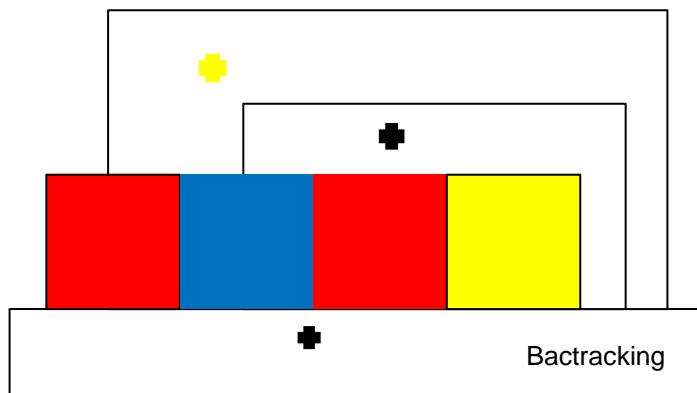
Gambar 5.32 Map Coloring Step 10



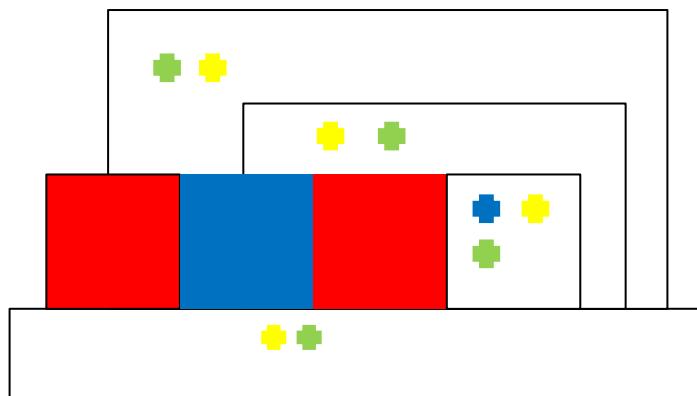
Gambar 5.33 Map Coloring Step 11



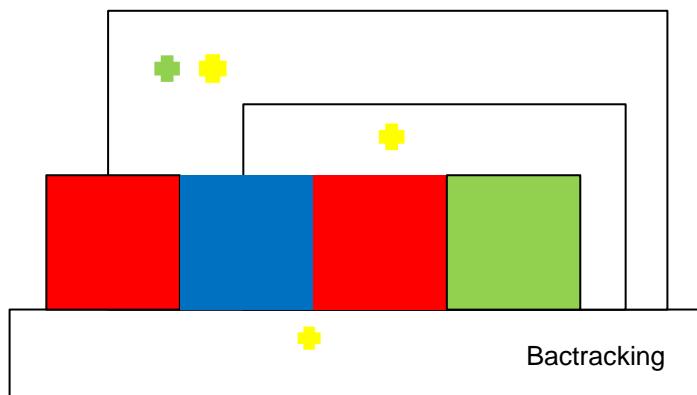
Gambar 5.34 Map Coloring Step 12



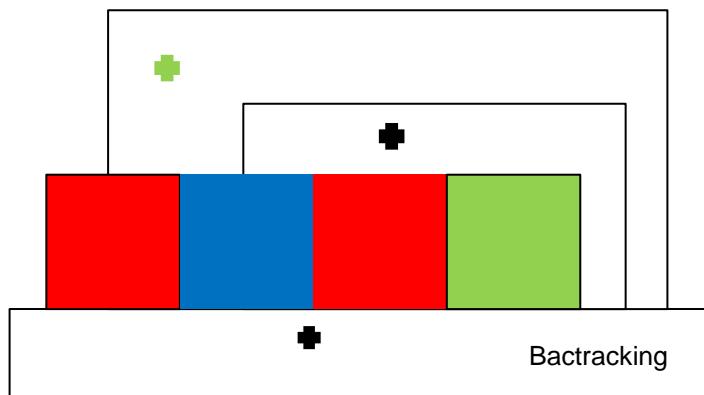
Gambar 5.35 Map Coloring Step 13



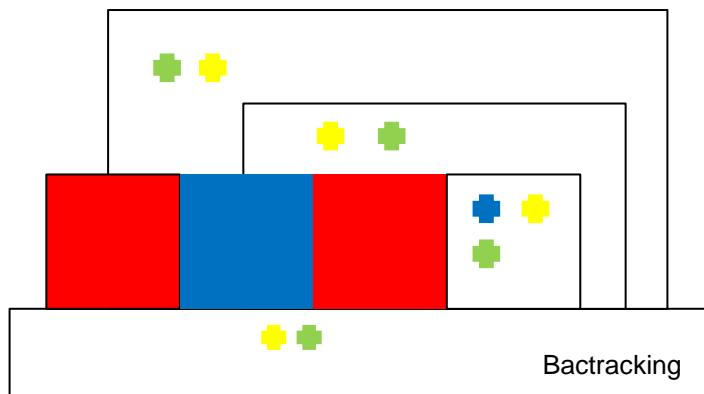
Gambar 5.36 Map Coloring Step 14



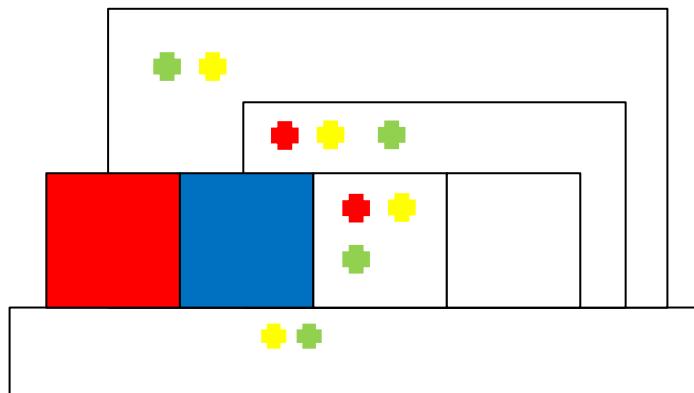
Gambar 5.37 Map Coloring Step 15



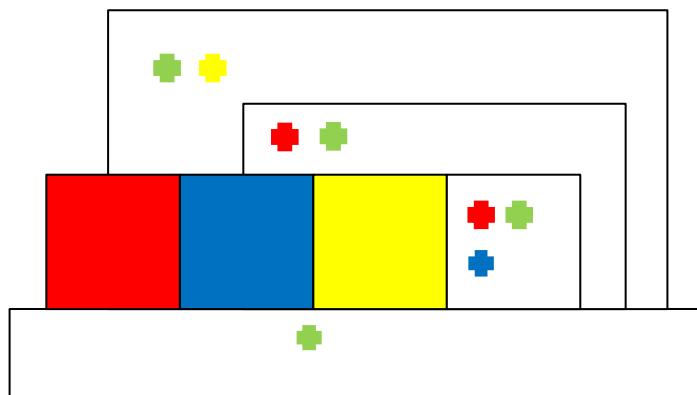
Gambar 5.38 Map Coloring Step 16



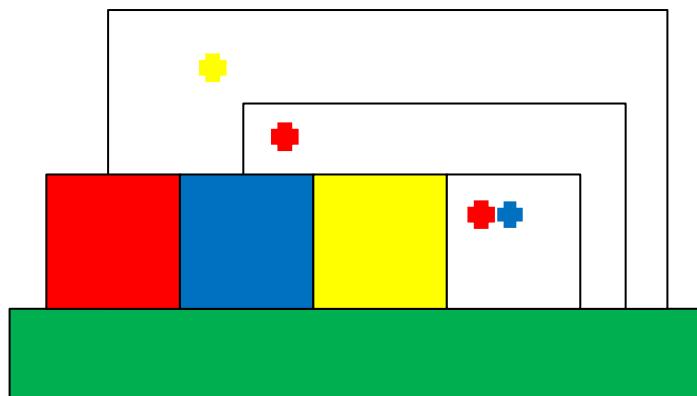
Gambar 5.39 Map Coloring Step 17



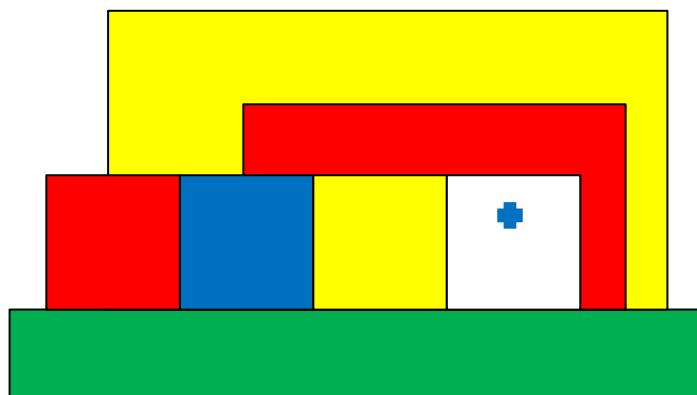
Gambar 5.40 Map Coloring Step 18



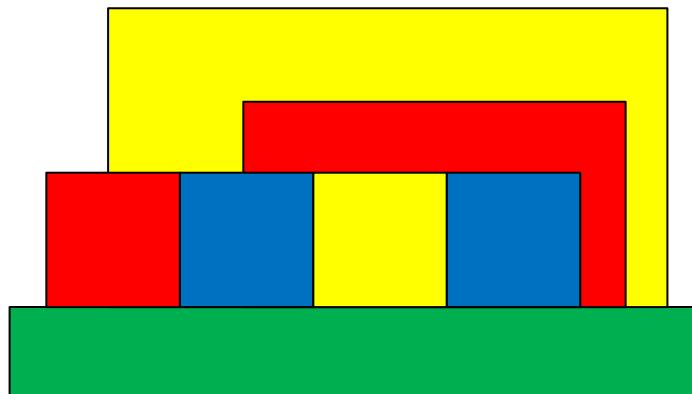
Gambar 5.41 Map Coloring Step 19



Gambar 5.42 Map Coloring Step 20



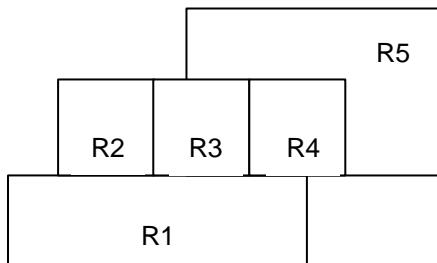
Gambar 5.43 Map Coloring Step 21



Gambar 5.44 Hasil Map Coloring

5.4 Latihan Individu

1. Isikan bidang pada gambar di bawah ini dengan warna: merah, hijau, biru. Bidang bertetangga tidak boleh memiliki warna yang sama!

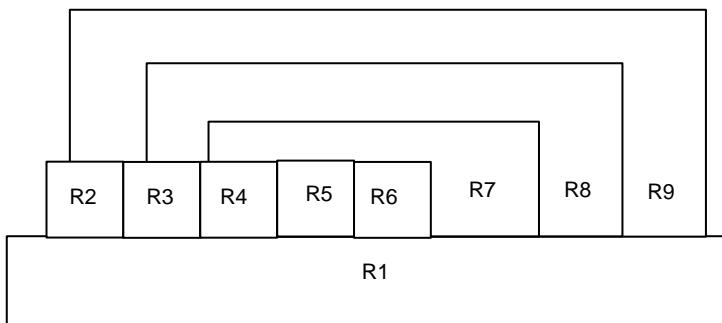


Pertanyaan:

- a) Apakah variabel, domain dan contraints yang ada?
- b) Gambarkan representasi graph coloring-nya?

5.5 Tugas Kelompok

1. Isikan bidang pada gambar di bawah ini dengan warna: merah, kuning, hijau, biru, putih, orange. Bidang bertetangga tidak boleh memiliki warna yang sama!



Pertanyaan:

- a) Apakah variabel, domain dan contraints yang ada?
- b) Gambarkan representasi graph coloring-nya?

BAB 6 Agen Logika

6.1 Agen Logika

Agen logika merupakan agen yang memiliki kemampuan bernalar secara logika. Ketika beberapa solusi tidak secara eksplisit diketahui, maka diperlukan suatu agen berbasis logika. Logika sebagai Bahasa Representasi Pengetahuan memiliki kemampuan untuk merepresentasikan fakta sedemikian sehingga dapat menarik kesimpulan (fakta baru, jawaban).

Sedangkan pengetahuan merupakan komponen yang penting, sehingga terdapat perbedaan jika diterapkan pada dua agent, yakni problem solving agent dan knowledge-based agent. Perbedaan dua agent, problem solving agent dan knowledge-based agent.

1. Problem solving agent: memilih solusi di antara kemungkinan yang ada. Apa yang ia “ketahui” tentang dunia, pengetahuannya tidak berkembang untuk mencapai problem solution (initial state, successor function, goal test)
2. Knowledge-based agent: lebih “pintar”. Ia “mengetahui” hal-hal tentang dunia dan dapat melakukan reasoning (berpikir, bernalar) mengenai:
 - Hal-hal yang tidak diketahui sebelumnya (imperfect/ partial information).
 - Tindakan yang paling baik untuk diambil (best action).

6.2 Agen Berbasis Pengetahuan

Agen berbasis pengetahuan adalah Knowledge Base (KB) menyatakan apa yang “diketahui” oleh si agent. Pendekatan deklaratif membangun agent yaitu “beritahu” informasi yang relevan, simpan dalam KB → (TELL). Agen dapat ditanya (atau bertanya diri sendiri) apa yang sebaiknya dilakukan berdasarkan KB → (ASK). Maka sebuah agen berbasis pengetahuan harus bisa:

- Merepresentasikan world, state, action, dst.

- Menerima informasi baru (dan meng-update representasinya).
- Menyimpulkan pengetahuan lain yang tidak eksplisit (hidden property).
- Menyimpulkan action apa yang perlu diambil.

Knowledge Base (KB) merupakan himpunan representasi fakta yang diketahui tentang lingkungannya. Tiap fakta disebut sebagai sentence. Fakta tersebut dinyatakan dalam bahasa formal sehingga bisa diolah. Sedangkan TELL berfungsi menambahkan sentence baru ke KB.

Inference Engine merupakan penentuan fakta baru yang dapat diturunkan dari pengetahuan yang sudah ada dalam KB dan menjawab pertanyaan (ASK) berdasarkan KB yang sudah ada. Dalam representasi, agent dapat dipandang dari knowledge level apa saja informasi yang diketahui? Misal: sebuah robot “mengetahui” bahwa gedung B di antara gedung A dan gedung C. Agent dapat dipandang dari implementation level: Bagaimana representasi informasi yang diketahuinya?

- Logical sentence: `di_antara(gdB, gdA, gdC)`.
- Natural language: “Gedung B ada di antara gedung A dan gedung C”.
- Tabel posisi koordinat gedung-gedung.
- Gambar diagram peta (dalam bitmap atau vektor).

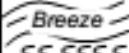
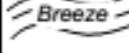
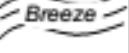
Pilihan representasi berpengaruh terhadap apa yang bisa dilakukan inference engine. Pada pendekatan deklaratif programmer memberitahu (TELL) agent informasi tentang environment. Kalau informasi kurang, agen bisa melengkapinya sendiri.

Jika dibandingkan dengan pendekatan prosedural: programmer secara eksplisit memrogram agen untuk bertindak. Sehingga bagaimana jika program tidak benar, maka akan besar kemungkinan menyebabkan kesalahan. Permasalahannya adalah bagaimana representasi yang tepat, sehingga ada dua hal yang harus diperhatikan:

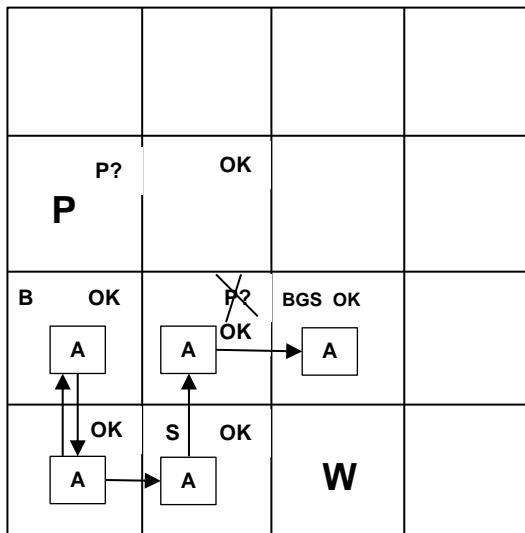
- Expressive : bisa menyatakan fakta tentang environment.
- Tractable : bisa mengolah/ memproses inference engine (dengan cepat).

Knowledge merupakan power atau kekuatan dari pemrograman secara deklaratif. Representasi dan penalaran membentuk suatu Intelligence. Contoh Aturan Permainan dalam Wumpus World:

- Performance measure: emas +1000, mati -1000, gerak -1, panah -10.
- Environment: Matriks 4x4 ruang dengan initial state [1,1]. Ada gold, wumpus, dan pit yang lokasinya dipilih secara acak.
- Percept terdiri dari:
 - Breeze: kamar di samping lubang jebakan ada hembusan angin.
 - Glitter: kamar di mana ada emas ada kilauan/ sinar.
 - Smell: kamar di samping Wumpus berbau busuk (stench).
- Action: maju, belok kiri 90° ,belok kanan 90° , tembak panah (hanya 1!), ambil benda.

4	 Stench		 Breeze	
3		 Stench  Gold		
2	 Stench		 Breeze	
1	 START	 Breeze		 Breeze
	1	2	3	4

Gambar 6.1 Permainan Wumpus Word



Gambar 6.2 Aturan dalam Permainan Wumpus World

- Sifat dari Wumpus World:
 - Fully observable? Tidak, hanya bisa berpersepsi lokal.
 - Deterministic? Ya, hasil tindakan jelas dan pasti.
 - Episodic? Tidak, tergantung action sequence.
 - Static? Ya, gold, wumpus, pit tidak bergerak.
 - Discrete? Ya
 - Single agent? Ya

Keterangan:

A = Agent

B = Breeze

S = Smell

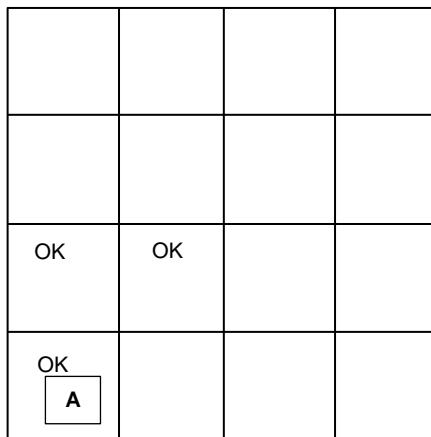
P = Pit

W = Wumpus

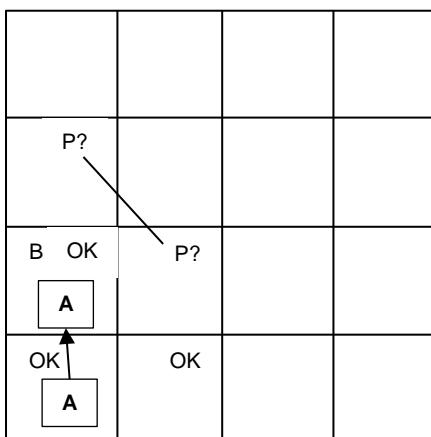
OK = Safe

V = Visited

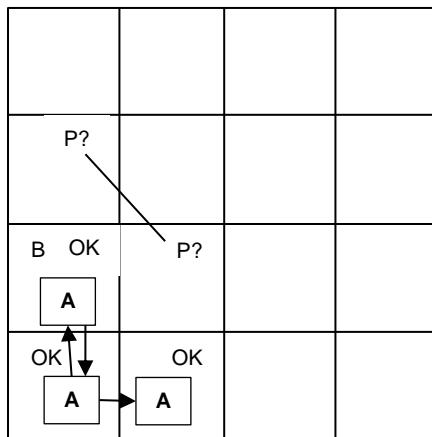
G= Glitter



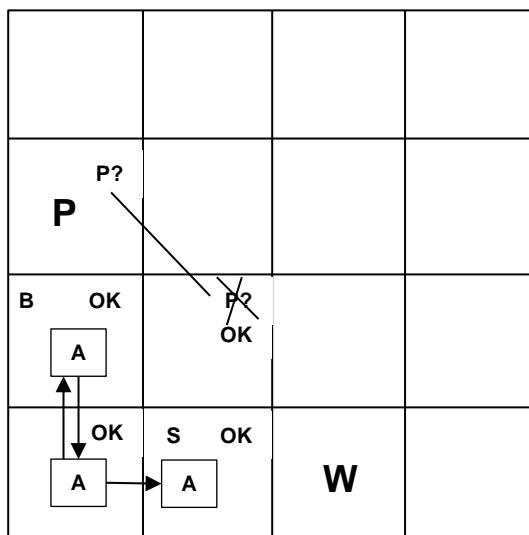
Gambar 6.3 Contoh Aturan Permainan Wumpus Step 1



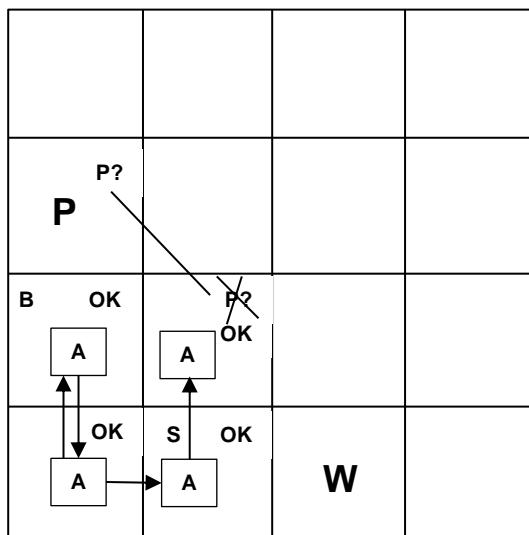
Gambar 6.4 Contoh Aturan Permainan Wumpus Step 2



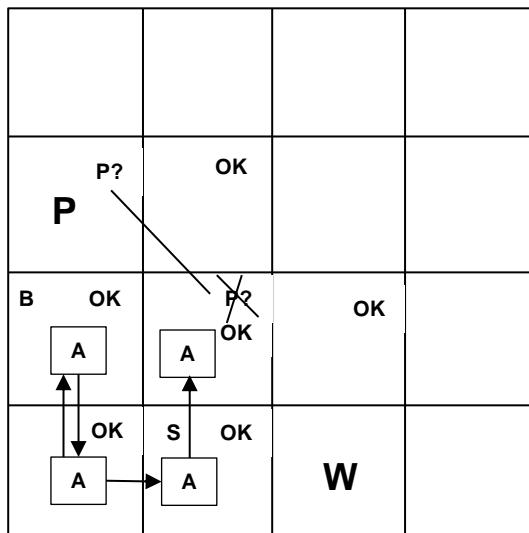
Gambar 6.5 Contoh Aturan Permainan Wumpus Step 3



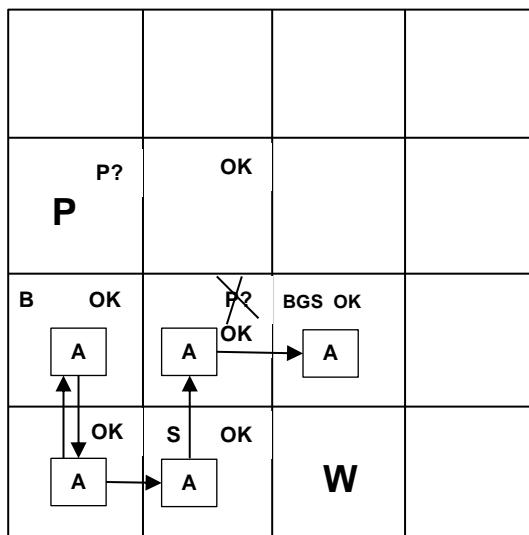
Gambar 6.6 Contoh Aturan Permainan Wumpus Step 4



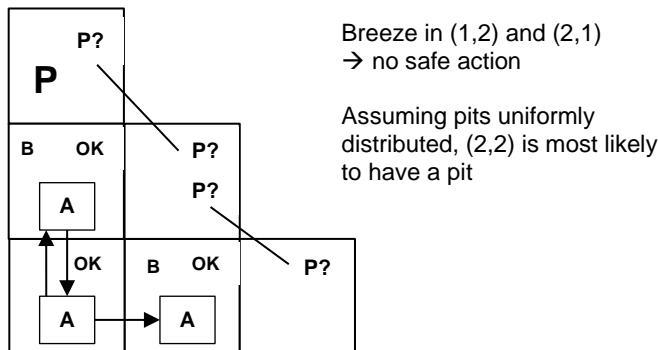
Gambar 6.7 Contoh Aturan Permainan Wumpus Step 5



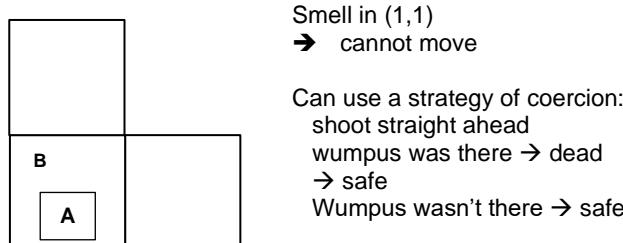
Gambar 6.8 Contoh Aturan Permainan Wumpus Step 6



Gambar 6.9 Contoh Aturan Permainan Wumpus Step 7



Gambar 6.10 Wumpus World dalam Kondisi Khusus



Gambar 6.11 Wumpus World dalam Kondisi Khusus

Bahasa Representasi Pengetahuan (Knowledge Representation Language) yang menyatakan suatu bahasa yang digunakan untuk menyatakan fakta tentang “dunia”. Atau suatu bahasa representasi pengetahuan didefinisikan dalam dua aspek, yakni:

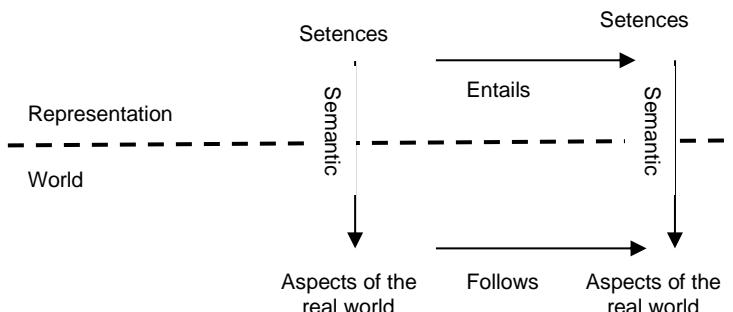
- Sintaks dari bahasa merupakan aturan yang mendefinisikan sentence yang sah dalam bahasa.
- Semantik menyatakan aturan yang mendefinisikan “arti” sebuah sentence, misalkan: kebenaran sentence dalam dunia.
- Contoh KRL (Knowledge Representation Language) dalam bahasa aritmetika yang secara Sintaks dituliskan:
 - $x + 2 \geq y$ adalah kalimat sah.
 - $x^2 + y \geq$ bukan kalimat sah.
- Contoh KRL (Knowledge Representation Language) dalam bahasa aritmetika yang secara semantik: $x + 2 \geq y$ benar jika dan hanya jika bilangan $x + 2$ tidak lebih kecil dari bilangan y :
 - $x + 2 \geq y$ benar dalam “dunia” di mana $x=7, y=1$
 - $x + 2 \geq y$ benar dalam “dunia” di mana $x=0, y=6$
- Contoh KRL dalam bahasa Indonesia yang secara Sintaks dituliskan:
 - “Jakarta adalah ibu kota Indonesia” adalah kalimat sah.
 - “Ibu Indonesia kota Jakarta adalah” bukan kalimat sah.Maka secara Semantik: “X adalah ibukota Y” benar jika dan hanya jika X adalah pusat pemerintahan negara Y.
 - “Jakarta adalah ibukota Indonesia” benar dalam “dunia” kita sekarang.
 - “Jakarta adalah ibukota Indonesia” salah dalam “dunia” tahun 1948 (Yogya? Bukittinggi?)

Logika sebagai Bahasa Representasi Pengetahuan memiliki pengertian yaitu sebagai bahasa formal untuk merepresentasikan fakta sedemikian sehingga kesimpulan (fakta baru, jawaban) dapat ditarik. Ada banyak metode inference yang diketahui. Sehingga kita bisa membangun agent Wumpus World dengan logika yang memanfaatkan perkembangan logika dari ahli matematika.

Entailment dapat diartikan sebagai suatu fakta bisa disimpulkan dari (kumpulan) fakta lain. $\text{KB} \models \alpha$ berarti KB melakukan entailment sentence α jika dan hanya jika α true dalam “dunia” di mana KB true. Contoh:

- KB mengandung dua sentence, yakni “Anto Genius” dan “Ani Cantik”.
- $\text{KB} \models \alpha_1$: “Anto Genius dan Ani Cantik” (artinya: hasil entailment bisa berupa kalimat gabungan dari dua kalimat)
- $\text{KB} \not\models \alpha_2$: “Anto Tampan”
- $x + y = 4 \models 4 = x + y$

Inferensi atau reasoning merupakan pembentukan fakta (sentence) baru yang meng-entail fakta-fakta lama. Reasoning bukan dilakukan pada fakta di dunia (berdasarkan semantik), melainkan representasi fakta dalam bahasa representasi pengetahuan si agent (secara sintaks). Otak manusia melakukan proses reasoning dalam suatu bentuk sintak dapat diilustrasikan sebagaimana gambar berikut:

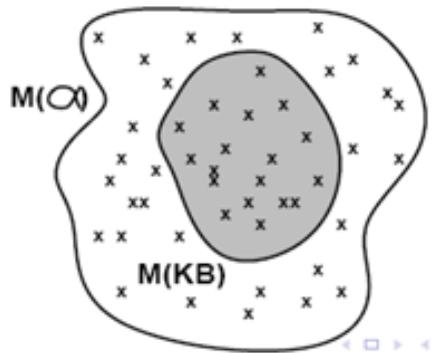


Gambar 6.12 Proses Reasoning

Model merupakan suatu “dunia” di mana kebenaran suatu sentence bisa diuji. Otak manusia melakukan proses reasoning dalam suatu bentuk sintak dapat diilustrasikan sebagaimana gambar berikut:

- Model merupakan suatu “dunia” di mana kebenaran suatu sentence bisa diuji.
- m adalah model α jika dan hanya jika true di “dalam” m.
- $M(\alpha)$ adalah himpunan semua model dari α .

- KB $\models \alpha$ jika dan hanya jika $M(KB)$ subset dari $M(\alpha)$, sehingga bisa dilihat pada ilustrasi gambar berikut:



Gambar 6.13 Ilustrasi Proses Reasoning dalam Otak Manusia

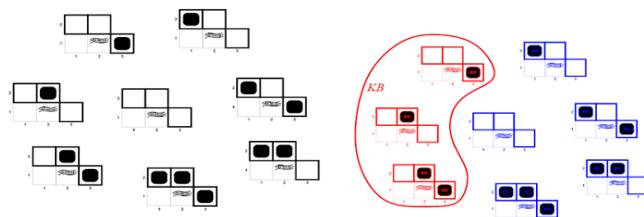
Misalkan:

- $KB = \text{Anto Genius and Ani Cantik}$
 - $\alpha = \text{Anto Genius}$
 - Hal ini bisa diartikan bahwasanya sentence Anto Genius lebih luas konotasinya dibandingkan dengan sentence Anto Genius dan Ani Cantik.
- Entailment dalam Wumpus World bisa diilustrasikan sebagaimana berikut, dengan melihat [1,1] OK, [2,1] Breeze:

Tabel 6.1 Ilustrasi Entailment dalam Wumpus World

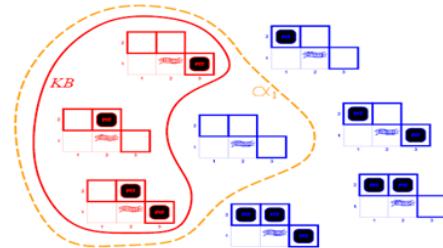
?	?		
A	B	A	?

Maka model jebakan ada 3 pilihan boolean di [2,1],[2,2],[3,1], dengan 8 kemungkinan model.



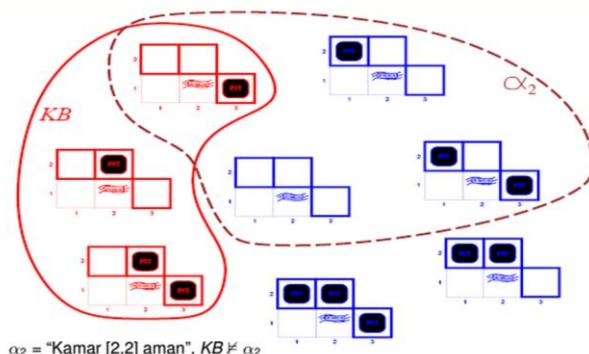
Gambar 6.14 Contoh Model Jebakan

- $KB = \text{pengamatan (percept)} + \text{aturan main Wumpus World}$, maka kita menyatakan apakah kamar [1,2] aman dengan cara melakukan entailment yang mana dibuktikan dengan menggunakan model checking, yakni memeriksa semua kemungkinan $M(KB), M(\alpha_1)$.



Gambar 6.15 Ilustrasi Model Checking

Sehingga dari ilustrasi diatas menunjukkan bahwasannya $M(KB)$ subset dari $M(\alpha_1)$, sehingga bisa disimpulkan bahwa kamar [1,2] aman. Lain halnya ketika melakukan pengamatan apakah kamar [2,2] aman, dengan ditunjukkan oleh α_2 , terlihat sebagaimana ilustrasi berikut:



Gambar 6.16 Ilustrasi Model Checking

Dari gambar tersebut menunjukkan bahwasannya $M(KB)$ bukan subset dari $M(\alpha_2)$, sehingga bisa disimpulkan bahwa $KB \not\models \alpha_2$, dengan kata lain kamar [2,2] tidak aman. Inferensi merupakan proses atau algoritma yang “menurunkan” fakta baru dari fakta-fakta lama.

- $KB \models_i \alpha$: sentence α bisa diturunkan dari KB oleh prosedur i .
- Soundness: i dikatakan sah (*sound*) jika untuk semua $KB \models_i \alpha$, $KB \models \alpha$ benar.
- Completeness: i dikatakan lengkap (*complete*) jika untuk semua $KB \models \alpha$, $KB \models_i \alpha$ benar.

6.3 Logika Proposisi

Logika proposisi merupakan logika yang paling sederhana. Sebuah sentence dinyatakan sebagai simbol proposisional P_1 , P_2 , dst. Sintaks dari logika proposisi yaitu:

- Jika S adalah kalimat, $\neg S$ adalah kalimat (negasi)
- Jika S_1 dan S_2 adalah kalimat, $S_1 \wedge S_2$ adalah kalimat (conjunction)
- Jika S_1 dan S_2 adalah kalimat, $S_1 \vee S_2$ adalah kalimat (disjunction)
- Jika Jika S_1 dan S_2 adalah kalimat, $S_1 \rightarrow S_2$ adalah kalimat (implication)
- Jika S_1 dan S_2 adalah kalimat, $S_1 \leftrightarrow S_2$ adalah kalimat (biconditional)

Semantik dari logika proposisi yaitu:

- Sebuah model memberi nilai *true/ false* terhadap setiap proposisi, misal $P_{1,2} = \text{true}$, $P_{2,2} = \text{true}$, $P_{3,1} = \text{false}$.
- Sebuah proses rekursif bisa mengevaluasi kalimat sembarang: $\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \text{true} \wedge (\text{false} \vee \text{true}) = \text{true} \wedge \text{true} = \text{true}$.
- Maka kalimat yang digunakan untuk merepresentasikan Wumpus World secara semantik:
 - $P_{i,j} = \text{true}$ menyatakan kalau ada lubang jebakan (*pit*) di $[i, j]$.

- $B_{i,j} = \text{true}$ menyatakan kalau ada hembusan angin (*breeze*).

Aturan main dalam Wumpus World: kamar di samping lubang jebakan ada hembusan angin:

- $B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$
- $B_{2,1} \leftrightarrow (P_{1,2} \vee P_{2,2} \vee P_{3,1})$

Menyatakan KB untuk dibuktikan misalnya dengan tabel kebenaran

Hasil pengamatan (percept):

- $\neg P_{1,1}$
- $\neg B_{1,1}$
- $B_{2,1}$

Menyatakan kamar i,j tersebut (α_i) aman atau tidak

- Dasar Manipulasi Rules

- $\neg(\neg A) = A$ Double negation
- $\neg(A \wedge B) = (\neg A) \vee (\neg B)$ Negated “and”
- $\neg(A \vee B) = (\neg A) \wedge (\neg B)$ Negated “or”
- $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ Distributivity of \wedge on \vee
- $A \Rightarrow B = (\neg A) \vee B$ by definition
- $\neg(A \Rightarrow B) = A \wedge (\neg B)$ using negated or
- $A \Leftrightarrow B = (A \Rightarrow B) \wedge (B \Rightarrow A)$ by definition
- $\neg(A \Leftrightarrow B) = (A \wedge (\neg B)) \vee (B \wedge (\neg A))$ using negated and & or

Contoh dalam penyelesaian masalah logika proposisi yaitu:

Let $\alpha = A \vee B$ and $KB = (A \vee C) \wedge (B \vee \neg C)$

Is it the case that $KB \models \alpha$?

Check all possible models – α must be true wherever KB is true

Tabel 6.2 Contoh Permasalahan Logika Proposisi

A	B	C	$A \vee C$	$B \vee \neg C$	KB	α
False	False	False				
False	False	True				

False	True	False				
False	True	True				
True	False	False				
True	False	True				
True	True	False				
True	True	True				

Tabel 6.3 Solusi untuk Permasalahan Logika Proposisi

A	B	C	A V C	B V $\neg C$	KB	α
False	False	False	False	True	False	False
False	False	True	True	False	False	False
False	True	False	False	True	False	True
False	True	True	True	True	True	True
True	False	False	True	True	True	True
True	False	True	True	False	False	True
True	True	False	True	True	True	True
True	True	True	True	True	True	True

Inferensi bisa dilakukan menggunakan tabel kebenaran untuk membuktikan entailment dari suatu *knowledge*. Sehingga kita dapat membuktikan apakah $KB \models \alpha_1$ menggunakan tabel kebenaran (sejenis model checking), di mana α_1 menyatakan kamar di [1, 2] aman sebagaimana tabel di bawah ini.

Tabel 6.4 Contoh Tabel Kebenaran

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	false	true						
false	false	false	true
.
.
false	true	false	false	false	false	false	false	true
false	true	false	false	false	false	true	true	true
false	true	false	false	false	true	true	true	true
false	true	false	false	true	false	false	false	true
.
.
.
true	false	false						

Tabel 6.5 Contoh Tabel Kebenaran

?	?		
A	\xrightarrow{B}	A	?

- $B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$
- $B_{2,1} \leftrightarrow (P_{1,2} \vee P_{2,2} \vee P_{3,1})$

Keterangan:

$P_{i,j}$ = true menyatakan kalau ada lubang jebakan (*pit*) di $[i,j]$

$B_{i,j}$ = true menyatakan kalau ada hembusan angina (*breeze*)

6.4 Metode Pembuktian

Metode Pembuktian secara umum, ada dua jenis metode pembuktian:

1. Pengaplikasian inference rule
 - Dihasilkan kalimat baru yang sah (sound) dari yang lama.
 - Bukti (proof) merupakan serangkaian pengaplikasian inference rule (operator \rightarrow) dari algoritma search.
 - Biasanya, kalimat harus diterjemahkan ke dalam sebuah *normal form*.
 2. Model checking
 - Penjabaran truth table (eksponensial dalam n)
 - Backtracking lebih efisien, misalkan: algoritma DPLL
 - Heuristic search dalam model space (sound tetapi incomplete), misalkan: min-conflicts hill –climbing
- Dalam Horn form, KB merupakan conjunction dari Horn Clauses. Horn Clause terdiri:
- Proposition symbol

- (Conjunction of symbol) \rightarrow symbol

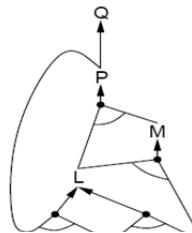
Misalkan: $C \ (B \rightarrow A) \ (C \ D \rightarrow B)$

Maka modus ponen pada Horn form (lengkap pada Horn KB):

$$\frac{\alpha_1, \dots, \alpha_n, \alpha_1 \dots \alpha_n \rightarrow \beta}{\beta}$$

Horn form bisa digunakan dengan algoritma forward chaining atau backward chaining. *Forward chaining* adalah aplikasi rule yang premise-nya diketahui benar dalam KB, kemudian tambahkan conclusionnya ke dalam KB, ulangi sampai query (Q) terbukti. Sehingga bisa dikatakan kinerja dari forward chaining merupakan metode bottom up dari fakta menuju konklusi. *Forward chaining* adalah aplikasi rule yang premise-nya diketahui benar dalam KB, kemudian tambahkan conclusionnya ke dalam KB, ulangi sampai query (Q) terbukti. Sehingga bisa dikatakan kinerja dari forward chaining merupakan metode bottom up dari fakta menuju konklusi. Misal diilustrasikan sebagai berikut:

$P \rightarrow Q$
 $L \wedge M \rightarrow P$
 $B \wedge L \rightarrow M$
 $A \wedge P \rightarrow L$
 $A \wedge B \rightarrow L$



A
B

Gambar 6.17 Ilustrasi Kinerja dari
Forward Chaining

Sedangkan konsep dasar dari algoritma backward chaining digunakan untuk membuktikan query (Q), dengan cara memeriksa Q jika sudah diketahui, atau secara rekursif, dengan membuktikan semua premise rule yang conclusion-nya Q (dikenal sebagai metode top down). Dalam backward chaining ada beberapa hal yang perlu diketahui:

- Menghindari loop: dengan cara memeriksa apakah sub-goal yang baru sudah ada di goal stack.
- Menghindari perulangan pekerjaan: periksa apakah sub-goal yang baru sudah dibuktikan benar atau sudah dibuktikan salah.

6.5 Latihan Individu

- Diketahui $KB = (\neg B_{1,1} \Leftrightarrow (P_{2,1} \wedge P_{3,1})) \wedge B_{1,1}$ dan $\alpha = \neg P_{2,1} \wedge P_{3,1}$. Buktikan dengan tabel kebenaran berikut, apa saja kondisi yang memenuhi $KB|=\alpha$!

	$B_{1,1}$	$B_{2,1}$	$P_{3,1}$	$P_{1,2} \wedge P_{3,1}$	$\neg B_{1,1} \Leftrightarrow (P_{1,2} \wedge P_{3,1})$	KB	α
1	F	F	F				
2	F	F	T				
3	F	T	F				
4	F	T	T				
5	T	F	F				
6	T	F	T				
7	T	T	F				
8	T	T	T				

6.6 Tugas Kelompok

- Jelaskan konsep dasar dari agen berbasis pengetahuan dan hal-hal apa saja yang harus dipenuhi ketika membuat agen tersebut!
- Diketahui KB = “bebas”, α = “bebas”, letak Smell, Breeze “bebas”. Buktikan dengan tabel kebenaran, apa kondisi yang memenuhi $KB|=\alpha$!

$B_{i,j}$	$P_{i,j}$	$P_{i,j}$	KB	α
F	F	F			
F	F	T			
F	T	F			
F	T	T			
T	F	F			
T	F	T			
T	T	F			
T	T	T			

BAB 7 Logika Order Pertama (First Order Logic)

7.1 Konsep dasar FOL (First Order Logic)

Beberapa konsep dasar pada First Order Logic (FOL) yaitu sebagai berikut:

- Declarative: menyatakan fakta-fakta terpisah dari mekanisme/prosedur inference.
- Memungkinkan pernyataan informasi yang partial / disjunctive / negated.
- Compositional: “arti” $P \wedge Q$ tergantung arti P dan arti Q .
- Context-independent: arti tidak tergantung konteks.
- Unambiguous: terhadap suatu model, arti sebuah sentence jelas.

Tetapi, kurang expressive. Mis.: “Kalau ada jebakan, di kamar sebelah ada hembusan angin” harus dinyatakan dengan $n \times n$ buah *sentence propositional logic* (PL). Dalam propositional logic (PL), dunia hanya mengandung fakta-fakta. Dalam first order logic (FOL), dunia bisa mengandung:

- Object: di dalam dunia ada orang, bangunan, buku, UB, ITS, UI, SBY, bilangan, warna, hari,...
- Relations: tentang object dalam dunia, ada relasi merah, bulat, cantik, positif, abang dari, lebih besar dari, di atas, terjadi sebelum,...
- Functions: fungsi yang menghasilkan object lain seperti ayah dari, babak final dari, satu lebih dari, kaki kiri dari,...

Hal ini disebut ontological commitment dari sebuah logic yaitu apa saja “isi” dunia yang dijelaskan? Ada juga epistemological commitment: “kebenaran” apa yang dapat dinyatakan tentang sebuah sentence? Contoh beberapa jenis logic lain:

Tabel 7.1 Contoh Jenis Logic

Language	Ontological (isi)	Epistemological (kebenaran)
Propositional Logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	
Fuzzy logic	degree of truth $\in [0,1]$	know interval value
Magig logic ?	?	?

7.2 Sintak dan Semantic FOL

7.2.1 Syntax FOL

Beberapa elemen-elemen dasar pada First Order Logic (FOL) yaitu sebagai berikut:

- Constants (objects): KingJohn, 2, UB, ITS, UI, Malang, Depok.
- Predicates (relations): Brother, Loves, Membenci , Mengajar ,.
- Functions (functional relations): Sqrt, LeftLegOf , Ayah,...
- Variables: x, y, a, b, \dots
- Connectives: $\wedge \vee \neg \Rightarrow \Leftrightarrow$
- Equality: =
- Quantifiers: $\forall \exists$

Definisi dari kalimat Atomic pada FOL adalah sebagai predicate ($term_1, \dots, term_n$) atau $term_1 = term_2$. Sedangkan untuk definisi term adalah sebagai Function ($term_1, \dots, term_n$) atau constant atau variable dan contoh sebagai berikut:

- Brother (KingJohn, RichardTheLionheart)
- $>(\text{Length} (\text{LeftLegOf} (\text{Richard})), \text{Length}(\text{LeftLegOf} (\text{KingJohn})))$

Sedangkan untuk Kalimat Kompleks pada FOL adalah kalimat kompleks / complex sentence terdiri dari sentence yang digabungkan dengan connective. Definisi complex sentence yaitu $\neg S$, $S_1 \wedge S_2$, $S_1 \vee S_2$, $S_1 \Rightarrow S_2$, $S_1 \Leftrightarrow S_2$. Contohnya sebagai berikut:

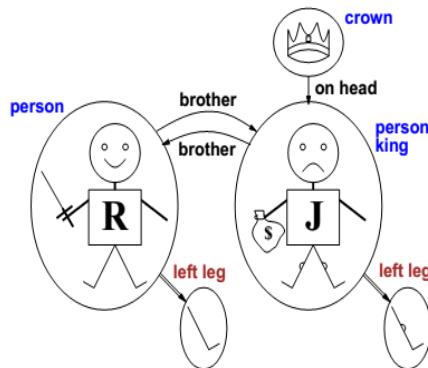
- Sibling (KingJohn, Richard) \Rightarrow Sibling(Richard , KingJohn)
- $>(1, 2) \vee \leq(1, 2)$
- $>(1, 2) \wedge \neg>(1, 2)$
- Belajar (x , SC) \Rightarrow Mengerti(x , AI)

7.2.2 Semantics FOL

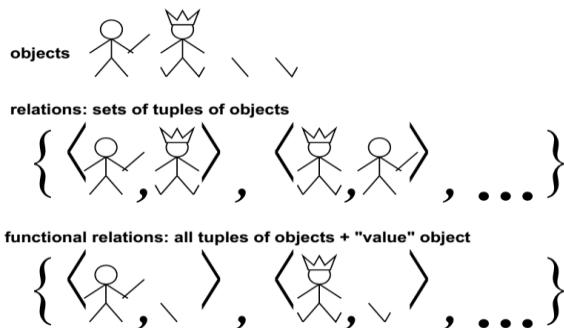
Sama halnya dengan Proposition Logic (PL), sebuah kalimat FOL bisa juga dikatakan true terhadap sebuah model. Namun, sebuah kalimat bisa diinterpretasikan banyak cara dalam sebuah model. Model berisi objects dan relations. Objects adalah elemen-elemen di dalam dunia (domain elements). Sedangkan relations adalah hubungan antara elemen-elemen tersebut. Sebuah interpretasi mendefinisikan referent (“yang dipetakan”).

- Constant symbols \rightarrow objects
- Predicate symbols \rightarrow relations
- Function symbols \rightarrow functional relations

Arti dari sebuah kalimat FOL yaitu kalimat atomik predicate(term₁,..., term_n) dikatakan bernilai true dalam model m di bawah interpretasi i iff object yang di-refer (term₁,..., term_n) (di bawah i) terhubung oleh relation yang di-refer oleh predicate (di bawah i) dalam m.



Gambar 7.1 Contoh Sebuah Model



Gambar 7.2 Contoh Sebuah Model (lebih rinci)

7.3 Konteks Penggunaan FOL

Entailment, validity, satisfiability, dll. Didefinisikan untuk semua kemungkinan interpretasi dari semua kemungkinan model! Kalau mau dijabarkan semua kemungkinannya:

For each number of domain elements n from 1 to ∞
For each k -ary predicate P_k in the vocabulary
For each possible k -ary relation on n objects
For each constant symbol C in the vocabulary
For each choice of referent for C from n objects...

Menentukan entailment berdasarkan truth-table itu adalah hal yang mustahil. Biasanya ada satu interpretasi yang “dimaksudkan” → intended interpretation.

- Universal Quantification
 - Syntax:
Jika S kalimat, \forall variables S adalah kalimat
 - Contoh:
“Semua mahasiswa FILKOM UB adalah Genius”
 $\forall x \text{ mahasiswa}(x, \text{ FILKOM UB}) \Rightarrow \text{Genius}(x)$
 - Semantics:
 $\forall x S$ bernilai true dalam model m di bawah interpretasi iff S bernilai true untuk semua kemungkinan referent dari x (setiap object di dalam m).
Dengan kata lain, $\forall x S \equiv$ conjunction dari semua instantiation S :
 $(\text{mahasiswa } (\text{Ani}, \text{ FILKOM UB}) \Rightarrow \text{Genius } (\text{Ani})) \wedge$ $(\text{mahasiswa } (\text{Anto}, \text{ FILKOM UB}) \Rightarrow \text{Genius } (\text{Anto})) \wedge$
.

(mahasiswa (Zaenal, FILKOM UB) \Rightarrow Genius (Zaenal)) \wedge

(mahasiswa (Zakky, FILKOM UB) \Rightarrow Genius (Zakky))

Biasanya, \Rightarrow adalah operator /connective yang digunakan dengan \forall . Masalah yang sering terjadi yaitu menggunakan \wedge sebagai connective untuk \forall : $\forall x \text{ mahasiswa}(x, \text{FILKOM UB}) \wedge \text{Genius}(x)$ Kalimat ini berarti “Semua orang adalah mahasiswa FILKOM UB dan Genius”.

- Existential Quantification

- Syntax:

Jika S kalimat, \exists variable S adalah kalimat

- Contoh:

“Ada mahasiswa Gunadarma yang pintar”

$\exists x \text{ mahasiswa}(x, \text{Gundarma}) \wedge \text{pintar}(x)$

- Semantics:

$\exists x S$ bernilai true dalam model m di bawah interpretasi iff S bernilai true untuk setidaknya 1 kemungkinan referent dari x (sebuah object di dalam m). Dengan kata lain, $\exists x S \equiv$ disjunction dari semua instantiation S:

(mahasiswa(Ani, Gundar) \wedge pintar (Ani)) \vee

(mahasiswa(Anto, Gundar) \wedge pintar (Anto)) \vee

(mahasiswa(Zaenal, Gundar) \wedge pintar (Zaenal)) \vee

(mahasiswa(Zakky, Gundar) \wedge pintar (Zakky))

Biasanya, \wedge adalah operator /connective yang digunakan dengan \exists . Masalah yang sering terjadi yaitu menggunakan \Rightarrow sebagai connective untuk \exists : $\exists x \text{ mahasiswa}(x, \text{Gundar}) \Rightarrow \text{pintar}(x)$. Kalimat ini true jika ada setidaknya 1 orang (object) yang tidak kuliah di Gunadarma.

7.4 Rekayasa Pengetahuan dengan FOL

Beberapa sifat \forall (For All) dan \exists (There Exist):

- $\forall x \forall y S$ sama dengan $\forall y \forall x S$, biasa ditulis $\forall x, y S$
- $\exists x \exists y S$ sama dengan $\exists y \exists x S$, biasa ditulis $\exists x, y S$
- $\exists x \forall y S$ TIDAK sama dengan $\forall y \exists x S$!
 - $\exists x \forall y Mencintai(x, y)$
“Ada (sekurang-kurangnya) seseorang yang mencintai semua orang di dunia.”
 - $\forall y \exists x Mencintai(y, x)$
“Semua orang di dunia mencintai sekurang-kurangnya satu orang”.

Quantifier bisa dinyatakan dengan yang lain:
 $\forall x Doyan(x, Bakso)$ sama dengan $\neg\exists x \neg Doyan(x, Bakso)$
 $\exists x Doyan(x, Cilok)$ sama dengan $\neg\forall x \neg Doyan(x, Cilok)$. Berikut adalah contoh kalimat dari Convert to FOL:

- “Ayah adalah orangtua”
 $\forall x, y Ayah(x, y) \Rightarrow Orangtua(x, y)$
- “Hubungan saudara berlaku simetris”
 $\forall x, y Saudara(x, y) \Leftrightarrow Saudara(y, x)$
- “Ibu adalah orangtua berjenis kelamin perempuan”
 $\forall x, y Ibu(x, y) \Leftrightarrow Orangtua(x, y) \wedge Perempuan(x)$
- “Sepupu adalah anak dari saudara orangtua”
 $\forall x, y Sepupu(x, y) \Leftrightarrow \exists o_x, o_y Orangtua(o_x, x) \wedge Saudara(o_x, o_y) \wedge Orangtua(o_y, y)$

Kalimat $term_1 = term_2$ bernilai true di bawah sebuah interpretasi iff $term_1$ and $term_2$ me-refer ke object yang sama. Contoh sebagai berikut:

- Ayah(Anto) = Abdul adalah satisfiable
- Anto = Abdul juga satisfiable!
- Anto = Anto adalah valid.

Bisa digunakan dengan negasi untuk membedakan dua term:
 $\exists x, y Mencintai(Anto, x) \wedge Mencintai(Anto, y) \wedge \neg(x = y)$ (Anto mendua!). Definisi Sibling yaitu:

- $\forall x, y \text{ Sibling}(x, y) \Leftrightarrow (\neg(x = y) \wedge \exists m, f \neg(m = f) \wedge \text{Parent}(m, x) \wedge \text{Parent}(f, x) \wedge \text{Parent}(m, y) \wedge \text{Parent}(f, y))$

Kita bisa menggunakan FOL sebagai KRL (Knowledge Representation Language) sebuah KBA. Pertama-tama, kita berikan informasi ke KB (TELL). Kalimat FOL yang ditambahkan ke KB disebut assertion. Contohnya:

- TELL(KB,King(John))
- TELL(KB, $\forall x \text{ King}(x) \Rightarrow \text{Person}(x)$)

Lalu, kita bisa memberikan query, atau bertanya, kepada KB (ASK). Contohnya:

- ASK(KB,King(John)) jawabannya adalah true.
- ASK(KB,Person(John)) jawabannya adalah true.
- ASK(KB, $\exists x \text{ Person}(x)$) jawabannya adalah $\{x / \text{John}\}$

Sebuah query dengan existential variable bertanya kepada KB: “Apakah ada x sedemikian sehingga... ?” Bisa saja jawabannya “ya” atau “tidak”, tetapi akan lebih baik jika jawabannya adalah nilai (referent) x di mana query bernilai true. Bentuk jawaban demikian disebut substitution, atau binding list: himpunan pasangan variable/term. Untuk kalimat S dan substitution σ , $S\sigma$ adalah hasil “pengeisian” S dengan σ :

- $S = \text{LebihPintar}(x, y)$
- $\sigma = \{x / \text{Ani}, y / \text{Anto}\}$
- $S\sigma = \text{LebihPintar}(\text{Ani}, \text{Anto})$

ASK(KB,S) mengembalikan (satu? semua?) σ sedemikian sehingga $KB | = S\sigma$.

- FOL sbg KRL utk KBA LA^{TM} dlm WW
Representasi hasil percept dari sensor:
Percept ([bau, angin, kilau], waktu) (perhatikan penggunaan list agar rapi):
 - TELL(KB,Percept ([None, None, None], 1))
 - TELL(KB,Percept ([Smell, None, None], 2))
 - TELL(KB,Percept ([None, Breeze, Glitter], 3))

Untuk menentukan tindakan yang diambil: $\text{ASK}(\text{KB}, \exists t \text{TindakanTerbaik}(t, 3))$. Data “mentah” dari sensor perlu diolah:

- $\forall a, k, w \text{Percept}([\text{Smell}, a, k], w) \Rightarrow \text{MenciumBau}(w)$
- $\forall b, k, w \text{Percept}([\text{b}, \text{Breeze}, k], w) \Rightarrow \text{MerasaHembus}(w)$
- $\forall b, a, w \text{Percept}([\text{b}, a, \text{Glitter}], w) \Rightarrow \text{MelihatKilauan}(w)$

Tindakan “rational reflex” bisa dinyatakan dalam kalimat, mis: $\forall w \text{ MelihatKilauan}(w) \Rightarrow \text{TindakanTerbaik}(\text{Grab}, w)$.

- Menyatakan Aturan Main Wumpus World
 - Tambah assertion mengenai kamar:
 - o $\forall k, w \text{Di}(\text{Agent}, k, w) \wedge \text{MenciumBau}(w) \Rightarrow \text{KmrBusuk}(k)$
 - o $\forall k, w \text{Di}(\text{Agent}, k, w) \wedge \text{MerasaHembus}(t) \Rightarrow \text{KmrAngin}(k)$
 - o $\forall k, w \text{Di}(\text{Agent}, k, w) \wedge \text{MelihatKilauan}(t) \Rightarrow \text{KmrEmas}(k)$
 - “Di kamar sebelah lubang jebakan ada hembusan angin”
 - o Diagnostic rule: simpulkan sebab dari akibat:
 $\forall y \text{KmrAngin}(y) \Rightarrow \exists x \text{Jebakan}(x) \wedge \text{Sebelahan}(x, y)$
 $\forall y \neg \text{KmrAngin}(y) \Rightarrow \neg \exists x \text{Jebakan}(x) \wedge \text{Sebelahan}(x, y)$
 - o Causal rule: simpulkan akibat dari sebab:
 $\forall x \text{Jebakan}(x) \Rightarrow (\forall y \text{Sebelahan}(x, y) \Rightarrow \text{KmrAngin}(y))$
 $\forall x (\forall y \text{Sebelahan}(x, y) \Rightarrow \neg \text{Jebakan}(y)) \Rightarrow \neg \text{KmrAngin}(x)$
 - Definisi predikat KmrAngin: $\forall y \text{KmrAngin}(y) \Leftrightarrow [\exists x \text{Jebakan}(x) \wedge \text{Sebelahan}(x, y)]$

Diagnostic vs. Causal (model-based) reasoning penting, mis: diagnosa medis secara AI (dulu diagnostic, sekarang model-based). Proses merancang kalimat-kalimat KRL yang dengan tepat “merekpresentasikan” sifat dunia/masalah disebut knowledge engineering. “Memrogram” secara deklaratif yaitu pengkodean fakta dan aturan domain-specific. Jargon yang biasanya digunakan adalah Agent programmer = knowledge engineer. Mekanisme/proses penjawaban query \rightarrow inference rule yang domain-independent.

7.5 Latihan Individu

1. Ubahlah “Kalimat” dibawah ini menjadi bentuk “FOL”!
 - a. “Ayah adalah orangtua berjenis kelamin laki-laki”.
 - b. “Paman adalah saudara orangtua”.

- c. $\forall x, y \ Paman(x, y) \Rightarrow \exists oy \ OrangTua(oy, y) \wedge Saudara(oy, x) \wedge Laki(x)$
 - d. "Tidak ada jamur merah yang beracun".
2. Ubahlah "FOL" dibawah ini menjadi bentuk "Kalimat" !
- a. $\forall x \ mahasiswa(x, FILKOM\ UB) \Rightarrow Genius(x)$
 - b. $\forall x (jamur(x) \wedge merah(x)) \Rightarrow beracun(x)$

7.6 Tugas Kelompok

- 1. Jelaskan perbedaan antara FOL dan PL? (Optional)
- 2. Ubahlah "Kalimat" dibawah ini menjadi bentuk "FOL"!
 - a. "Cucu adalah anak dari anak saya".
 - b. "Paman dan Bibi adalah saudara".
 - c. "Ada dua jamur merah". (Optional)
 - d. "Pohon kelapa itu tinggi". (Optional)
- 3. Ubahlah "FOL" dibawah ini menjadi bentuk "Kalimat"! (Optional)
 - a. $\exists x \forall t (person(x) \wedge time(t)) \Rightarrow can-fool(x, t)$
 - b. $(\forall x)(\forall y) \ above(x, y) \Leftrightarrow (on(x, y) \vee (\exists z) (on(x, z) \wedge above(z, y)))$
- 4. Buatlah kalimat yang menyatakan catatan berita terkini yang terdiri minimal 4 kalimat, kemudian ubah dalam bentuk "FOL".

BAB 8 Inferensi pada FOL

8.1 Mengubah FOL inference ke PL inference

Pada tahun 60-an ada ide untuk mengubah FOL inference ke PL inference, dimana kita sudah melihat mekanisme inference untuk propositional logic

- Inference rule: Modus Ponens
 - Normal form: Horn clause
 - Algoritma: Forward chaining, Backward chaining
- Inference rule: Resolution
 - Normal form: Clause Normal Form (CNF)
 - Algoritma: Proof-by-contradiction

Pendekatan-pendekatan ini sound dan complete. Cara mudah melakukan inference FOL yaitu jika KB dan query dalam FOL bisa diterjemahkan ke dalam PL selesai.

Horn clause adalah bentuk khusus dari logika predikat yang digunakan sebagai sintaks bahasa Prolog. Horn clause mempunyai head h yang disebut sebagai predikat, dan body yang disebut sebagai daftar dari predikat $p_1; p_2; \dots; p_n$. Membuat kesimpulan tunggal atau inference dari pasangan Horn clause disebut resolusi.

Definisi penugasan (assignment) variabel ke suatu nilai selama proses resolusi disebut instantiation. Unification adalah proses pencocokan pola yang menentukan instansiasi mana yang dapat dibuat untuk variabel sehingga mendapatkan resolusi secara simultan.

- Instantiation:
 - Ground term: sebuah term tanpa variable, mis: Ani, Ayah(Anto), 2007.
 - Instantiation: kalimat di mana sebuah variable diganti dengan sebuah ground term (diperoleh dengan mengaplikasikan sebuah *substitution*). Contoh sebagai berikut:

- $\alpha = \forall x \text{ mahasiswa}(x, \text{PTIIKUB}) \Rightarrow \text{Genius}(x)$
 - $\beta = \exists x \text{ mahasiswa}(x, \text{Gundar}) \wedge \text{pintar}(x)$
 - $\sigma = \{x / \text{Anto}\}$
 - SUBST(σ, α) menghasilkan instantiation:
 $\text{mahasiswa}(\text{Anto}, \text{PTIIKUB}) \Rightarrow \text{Genius}(\text{Anto})$
 - SUBST(σ, β) menghasilkan instantiation:
 $\text{mahasiswa}(\text{Anto}, \text{Gundar}) \wedge \text{pintar}(\text{Anto})$
- Universal Instantiation yaitu sebuah kalimat dengan universal quantifier (\forall) meng-*entail* semua *instantiation*-nya:
$$\frac{\forall v \alpha}{\text{SUBST } (\{v/g\}, \alpha)}$$
untuk sembarang variable v dan ground term g , maka dapat dicon-
- tohkan sebagai berikut:

 - $\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$ meng-*entail*:
 - $\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$
 - $\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$
 - $\text{King}(\text{Father}(\text{John})) \wedge \text{Greedy}(\text{Father}(\text{John})) \Rightarrow \text{Evil}(\text{Father}(\text{John}))$
 - Existential Instantiation yaitu untuk sembarang variable v , kalimat α dan constant k yang tidak muncul di knowledge-base:
$$\frac{\exists v \alpha}{\text{SUBST } (\{v/k\}, \alpha)}$$

Contoh:

- $\exists x \text{ Crown}(x) \wedge \text{OnHead}(x, \text{John})$ meng-*entail*:
 - $\text{Crown}(C_1) \wedge \text{OnHead}(C_1, \text{John})$, dengan syarat C_1 adalah *constant symbol* yang baru, disebut *Skolem constant*.
- Menghilangkan quantifier dan variable:
 - Menghilangkan \forall :
 - Universal instantiation bisa digunakan berkali-kali untuk menambahkan kalimat baru.

- KB yang baru *logically equivalent* dengan yang lama.
 - Menghilangkan \exists :
 - Existential instantiation cukup digunakan sekali untuk menggantikan kalimat existential.
 - KB yang baru tidak *logically equivalent* dengan yang lama, tetapi satisfiable iff KB yang lama juga satisfiable \rightarrow inferentially equivalent

Note: dikatakan *satisfiable* jika bernilai true pada beberapa model.

Contoh:

- Andaikan KB berisi kalimat-kalimat berikut:

$\forall x \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x) \quad \forall y \text{Greedy}(y)$

King(John) Brother (Richard, John)

- Jika kita mengambil semua kemungkinan instantiation dari kalimat universal, kita dapatkan KB sbb:

King(John) \wedge Greedy (John) \Rightarrow Evil (John)

King(Richard) \wedge Greedy (Richard) \Rightarrow Evil (Richard)

Greedy (John) Greedy (Richard)

King(John) Brother (Richard , John)

- KB yang baru dikatakan propositionalized: proposition symbol-nya: King(John), Greedy (John), Evil (John), King(Richard) etc.

- Inference FOL menggunakan inference PL:

- Ide dasar: ubah KB + query dari FOL menjadi PL, lalu gunakan resolution.

- Masalah: dengan adanya function, jumlah ground term menjadi infinite.

- Greedy (Father (John)), Greedy (Father (Father (John)))

- Greedy (Father (Father (Father (John)))), dst.

- Teorema Herbrand (1930): jika FOL KB $\models \alpha$, ada sebuah finite subset PL KB $\models \alpha$. Ide dasar: For n = 0 to ∞

- Buat propositional KB_n dengan depth-n ground term
- Periksa apakah KB_n |= α
- Masalah (lagi!): kalau α di-entail OK, kalau tidak \rightarrow infinite loop.
- Teorema Church-Turing (1936): Entailment untuk FOL bersifat semidecidable. (tidak dapat diperbandingkan)
- Masalah dengan propositionalization:
 - Propositionalization menghasilkan banyak kalimat irelevan.
 - Contohnya, dari KB berikut:
 - $\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$
 - $\forall y \text{ Greedy}(y)$
 - $\text{King}(\text{John})$
 - $\text{Brother}(\text{Richard}, \text{John})$

manusia bisa cepat mengerti kalau Evil (John), namun proposition-alization menghasilkan:

- $\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$
- $\text{Greedy}(\text{Richard})$

yang tidak relevan.

- Dengan p buah predicate k-ary dan n constant, ada $p \times n^k$ instantiation!

8.2 Unification

- Isi KB:
 - $\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$
 - $\forall y \text{ Greedy}(y)$
 - $\text{King}(\text{John})$
 - $\text{Brother}(\text{Richard}, \text{John})$

Inference bahwa KB |= Evil (John) bisa langsung disimpulkan jika kita bisa mencari substitution θ sehingga King(x) dan Greedy (x) bisa “dicocokkan” dengan King(John) dan Greedy (y).

Contoh: $\theta = \{x / \text{John}, y / \text{John}\}$

- Definisi unification
 - $\text{UNIFY}(\alpha, \beta) = \theta$ jika $\text{SUBST}(\alpha, \theta) = \text{SUBST}(\beta, \theta)$

Contoh dari Unifications adalah sebagai berikut:

Tabel 8.1 Contoh Unifications

P	Q	θ
Sayang(Anto, x)	Sayang(Anto, Ani)	$\{x / \text{Ani}\}$
Sayang(Anto, x)	Sayang(y, Ani)	$\{x / \text{Ani}, y / \text{Anto}\}$
Sayang(Anto, x)	Sayang(y, Ibu (y))	$\{y / \text{Anto}, x / \text{Ibu} (\text{Anto})\}$
Sayang(Anto, x)	Sayang(x, Ani)	fail

Standardized apart variable menghilangkan overlap, misal: Sayang (x_{101} , Ani)

8.3 Inference Rule untuk FOL

- Generalized Modus Ponens (GMP):
 - Inference rule GMP:

$$p_1', p_2', \dots, p_n', (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)$$

$$q\theta$$

di mana $p_i'\theta = p_i\theta$ untuk semua i.

$$p_1' = \text{King}(\text{John}) \quad p_1 = \text{King}(x)$$

$$p_2' = \text{Greedy}(y) \quad p_2 = \text{Greedy}(x)$$

$$\theta = \{x / \text{John}, y / \text{John}\} \quad q = \text{Evil}(x)$$

$$q\theta = \text{Evil}(\text{John})$$

- GMP dengan KB yang berisi definite clauses (seperti Horn clause pada PL): $p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q$
- Semua variable diasumsikan universally quantified
- GMP adalah hasil lifting MP: “mengangkat” inference rule PL ke FOL.
- Kelebihan dibanding propositionalization: hanya melakukan substitution yang dibutuhkan oleh inference
- Contoh knowledge base:

- Kalimat: "The law says that, it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American."
- Kalimat: "Hukum mengatakan bahwa, suatu kejahatan bagi orang Amerika adalah menjual senjata ke negara-negara yang bermusuhan. Negara Nono adalah musuh Amerika, yang memiliki beberapa rudal, dan semua rudalnya dari yang dijual oleh Kolonel West, yang merupakan orang Amerika."

Buktikan bahwa Col. West adalah criminal!

Penyelesaian:

... it is a crime for an American to sell weapons to hostile nations:

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$$

- Contoh knowledge base:

- Kalimat: "The law says that, it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American."

Buktikan bahwa Col. West adalah criminal.

Penyelesaian:

- ... it is a crime for an American to sell weapons to hostile nations:

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$$

- Nono... has some missiles: $\exists x \text{ Owns}(\text{Nono}, x) \wedge \text{Missile}(x)$

$\text{Owns}(\text{Nono}, M_1)$ and $\text{Missile}(M_1)$ (Skolemization)

- ... all of its missiles were sold to it by Colonel West:

$$\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$$

- Missiles are weapons: $\text{Missile}(x) \Rightarrow \text{Weapon}(x)$
- An enemy of America counts as “hostile”:
 $\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$
- West, who is American...:
 $\text{American}(\text{West})$
- The country Nono, an enemy of America:
 $\text{Enemy}(\text{Nono}, \text{America})$

Perhatikan:

Semua kalimat KB tersebut berbentuk definite clause.

8.4 Forward chaining

- Forward chaining pada FOL dengan GMP:
 - Mirip dengan forward chaining pada PL.
 - Mulai dari fakta yang diketahui (clause tanpa premise), mis: $\text{Owns}(\text{Nono}, M_1)$, $\text{Missile}(M_1)$
 - “Aktifkan” (trigger) rule yang premise-nya diketahui (satisfied) \rightarrow tambahkan kesimpulan rule ke KB, mis:
 $\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$
 - Ulangi sampai query terbukti, atau tidak ada fakta baru yang bisa ditambahkan ke KB.
 - “Cocokkan” premise-premise setiap rule dengan fakta yang diketahui \rightarrow pattern-matching dengan unification
- Algoritma forward chaining

```
function FOL-FC-ASK(KB, α) returns a substitution or false
repeat until new is empty
    new ← {}
    for each sentence r in KB do
        ( $p_1 \wedge \dots \wedge p_n \Rightarrow q$ ) ← STANDARDIZE-APART ( r )
        for each θ such that  $p_1 \wedge \dots \wedge p_n \Rightarrow q$  θ  $p'_1 \wedge \dots \wedge p'_n \Rightarrow q$ 
            for some  $p'_1, \dots, p'_n$  in KB
                 $q' \leftarrow \text{SUBST} (\theta, q)$ 
                if  $q'$  isn't a renaming of sentence in KB or
                    new then do add  $q'$  to new
                 $\phi \leftarrow \text{UNIFY} (q', \alpha)$ 
                if  $\phi$  is not fail then return  $\phi$ 
            add new to KB
    return false
```

Contoh Forward Chaining FOL

- . . . it is a crime for an American to sell weapons to hostile nations :

$$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$$
- Nono . . . has some missiles : $\exists x \text{ Owns}(\text{Nono}, x) \wedge \text{Missile}(x)$

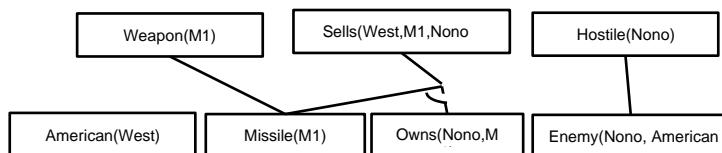
$$\text{Owns}(\text{Nono}, M_1) \text{ and } \text{Missile}(M_1)$$
 (Skolemization)
- . . . all of its missiles were sold to it by Colonel West :

$$\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$$
- Missiles are weapons : $\text{Missile}(x) \Rightarrow \text{Weapon}(x)$
- An enemy of America counts as “hostile” :

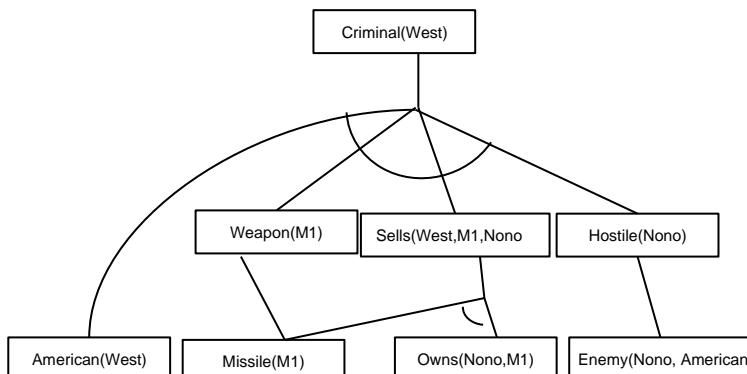
$$\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$$
- West, who is American . . . :

$$\text{American}(\text{West})$$
- The country Nono, an enemy of America . . . :

$$\text{Enemy}(\text{Nono}, \text{America})$$



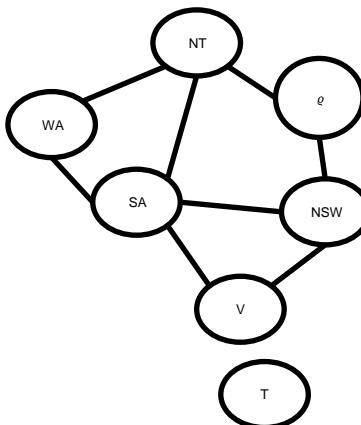
Gambar 8.1 Contoh Forward Chaining FOL



Gambar 8.2 Contoh Forward Chaining FOL

- Sifat Forward Chaining:

- Sound dan complete untuk first-order definite clause.
- Datalog = first-order definite clause tanpa function. Time complexity FC pada Datalog → polynomial
- Tapi pada kasus umum, bisa infinite loop kalau α tidak di-entail. (Konsekuensi dari teorema Church-Turing: entailment adalah semidecidable)
- Proses pattern matching pada premise NP-hard.
- Pattern matching premise NP-hard(Non-deterministic Polynomial-time hard)?



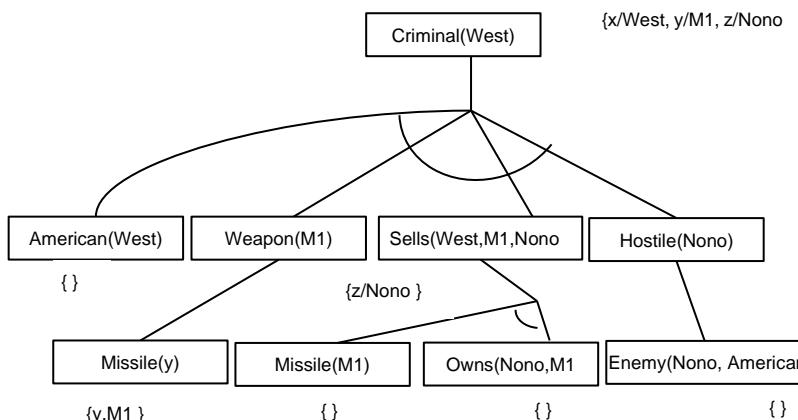
Diff(wa, nt) \wedge Diff(wa, sa) \wedge
Diff(nt, q) \wedge Diff(nt, sa) \wedge
Diff(q, nsw) \wedge Diff(q, sa) \wedge
Diff(nsw, v) \wedge Diff(nsw, sa) \wedge
Diff(v, sa) \Rightarrow Colorable()
Diff(Red, Blue)
Diff(Red, Green)
Diff(Green, Red)
Diff(Green, Blue)
Diff(Blue, Red)
Diff(Blue, Green)

- Query ASK(KB,Colorable()) jhj CSP-nya menemui solusi!
- Terdapat kasus CSP 3SAT (satisfiability pada CNF dengan clause berukuran 3 literal) yang diketahui NP-hard.

8.5 Backward chaining

- Backward chaining pada FOL dengan GMP:

```
function FOL-FC-ASK(KB, goals θ) returns a set substitution
    inputs: KB, a knowledge base
            goals, a list of conjuncts forming a query
            θ, the current substitution, initially the empty
            substitution {}
    local variables: ans, a set of substitutions,
    initially empty
    if goals is empty then return {θ}
    q' ← SUBST (θ, FIRST (goals))
    for each r in KB where STANDARDIZE-APART ( r ) = ( p1 ∧ ... ∧
    pn ⇒ q )
        and θ' ← UNIFY (q, q') succeeds
        ans ← FOL-BC-ASK (KB, [ p1 ∧ ... ∧
```



Gambar 8.3 Contoh Backward Chaining FOL

Sifat Backward Chaining:

- Depth-first search:
 - linear space complexity
 - incomplete (infinite loop)
 - repeated state
- Prinsip dasar Logic Programming

8.6 Resolution (The Next)

- Resolution pada FOL:

Resolution inference rule pada FOL (lifting resolution PL):

$$\frac{\ell_1 V \dots V \ell_k, m_1 V \dots V m_n}{(\ell_1 V \dots V \ell_{i-1} V \ell_{i+1} V \dots V \ell_k V m_1 V \dots V m_{j-1} V m_{j+1} V, \dots V \dots V m_n) \theta}$$

di mana $\text{UNIFY}(\ell_i, \neg m_j) = \theta$

Contoh:

$$\frac{\neg Kaya(x) VSedih(x) Kaya(Anto)}{Sedih(Anto)}$$

di mana $\theta = \{x / \text{Anto}\}$

Gunakan resolution rule pada CNF (KB $\wedge \neg\alpha$): complete untuk FOL

- Mengubah FOL ke CNF:

“Everyone who loves all animals is loved by someone.”

$$\forall x [\forall y \text{Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y \text{Loves}(y, x)]$$

- Eliminasi implikasi dan biimplikasi

$$\forall x [\neg\forall y \neg\text{Animal}(y) \vee \text{Loves}(x, y)] \vee [\exists y \text{Loves}(y, x)]$$

- Pindahkan \neg ke “dalam”: $\neg\forall x, p \equiv \exists x \neg p, \neg\exists x, p \equiv \forall x \neg p$:

$$\forall x [\exists y \neg(\neg\text{Animal}(y) \vee \text{Loves}(x, y))] \vee [\exists y \text{Loves}(y, x)]$$

$$\forall x [\exists y \neg\neg\text{Animal}(y) \wedge \neg\text{Loves}(x, y)] \vee [\exists y \text{Loves}(y, x)]$$

$$\forall x [\exists y \text{Animal}(y) \wedge \neg\text{Loves}(x, y)] \vee [\exists y \text{Loves}(y, x)]$$

- Standardize variables: setiap quantifier variable-nya beda

$$\forall x [\exists y \text{Animal}(y) \wedge \neg\text{Loves}(x, y)] \vee [\exists z \text{Loves}(z, x)]$$

- Skolemize: generalisasi existential instantiation. $\exists x$ diganti Skolem function universal quantified variable di “luar”:

$$\forall x [\text{Animal}(F(x)) \wedge \neg\text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$$

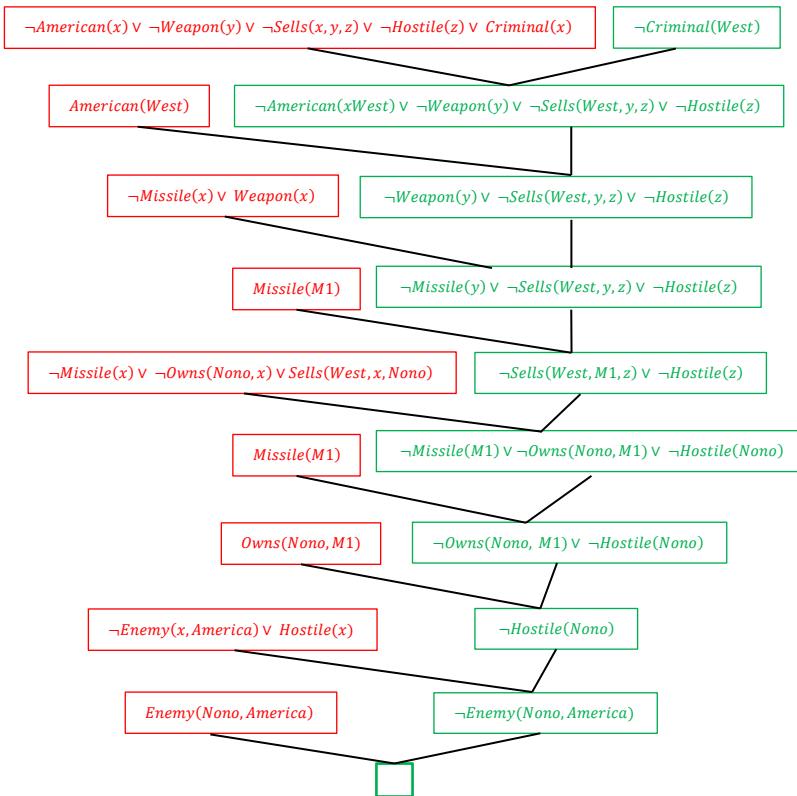
- Buang universal quantifiers:

$$[\text{Animal}(F(x)) \wedge \neg\text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$$

- Distribusi \wedge over V:

$$[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)] \wedge [\neg\text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)]$$

Contoh pembuktian dengan resolution:



Gambar 8.4 Contoh Pembuktian dengan Resolution

8.7 Tugas Kelompok

1. Jelaskan konsep Inference FOL, dan jelaskan perbedaannya dengan CNF?
2. Buatlah paragraf suatu kejadian minimal terdiri dari 3 kalimat, kemudian konversikan menjadi FOL, lalu bentuk Forward Chaining, lalu konversikan FOL tersebut menjadi CNF, lalu buktikan bahwa kesimpulan kejadian tersebut benar atau salah dengan Resolusi.

3. Berdasarkan kalimat berikut:

- a. $\exists x [\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y \text{ Loves}(y, x)]$
- b. Konversikan FOL di atas menjadi CNF
- c. Buktikan dengan resolusi bahwa terdapat seseorang yang mencintai seseorang

BAB 9 Logic Programming

9.1 Logic Programming

Sejarah singkat dari pemrograman Logika diperkenalkan oleh Robert Kowalski pada tahun 1974. Algoritma tersusun atas Logika dan control. Prolog merupakan pemrograman dalam logika. Bahasa pemrograman yang menggunakan pemrograman logika untuk komputasi. Diperkenalkan oleh Alain Colmerauer pada tahun 1970 an. Impementasi Prolog yang digunakan SWI Prolog versi 5.6.32. Bebas melakukan download di URL: <http://www.swi-prolog.org/>. Dikembangkan oleh Jan Wielemaker, Universitas Amsterdam. Ada berbagai implementasi lain seperti: SICStus Prolog, XSB, dsb.

Prolog adalah kependekan dari PROgramming in LOGic, yang berarti pemrograman. Pemrograman Prolog menggunakan bahasa deklaratif, dimana pemrogram memberi fakta dan aturan untuk selanjutnya diselesaikan oleh Prolog secara deduktif sehingga menghasilkan suatu kesimpulan.

Hal ini berbeda dengan bahasa prosedural seperti Pascal, Fortran, C, atau yang sejenis, dimana pemrogram memberi perintah atau penugasan untuk memecahkan persoalan langkah demi langkah. Prolog menggunakan relasi, bukan fungsi sehingga sangat sesuai untuk implementasi sistem pakar. Berikut merupakan contoh dari Pemrograman Logika:

- Temukan seluruh elemen yang beranggotakan dua list yang diberikan
 - o List: $[a_1, a_2, \dots, a_n]$ atau $[a_1 | [a_2, \dots, a_n]]$
 a_1 disebut dengan head dari $[a_1, a_2, \dots, a_n]$
 $[a_2, \dots, a_n]$ disebut tail dari $[a_1, a_2, \dots, a_n]$
 - o Contoh: $[1,2,3,4,5]=[1|[2,3,4,5]]$
- Kita memerlukan definisi pada saat X merupakan member dari suatu list.

- Jika X adalah sebagai head-nya, maka jawabannya positif (benar)
 $\text{member}(X,[X|List]).$
 - Sebaliknya, lakukan pengecekan apakah X adalah member dari tail.
 $\text{member}(X,[Y|List]) \leftarrow \text{member}(X,List).$
- Maka dalam bahasa Prolog bisa dituliskan terurut sebagaimana berikut:
- ```
member(X|[X|List]).
member(X,[Y|List]):-member(X,List).
member_both(X,L1,L2):-member(X,L1),member(X,L2).
```
- Kemudian jalankan program untuk memecahkan permasalahan  
`?-member_both(X,[1,2,3],[2,3,4,5]).`

## 9.2 Logika Predikat

Logika predikat (kalkulus predikat) merupakan bagian dari komputasi logika yang juga mencakup logika proposisional, dimana fakta dan aturan dinyatakan melalui predikat seperti dibawah ini:

|                                                                                                                                  |                        |
|----------------------------------------------------------------------------------------------------------------------------------|------------------------|
| <code>lelaki(Joko).</code>                                                                                                       | <code>// fakta</code>  |
| <code>menikah(Joko, Tuti).</code>                                                                                                | <code>// fakta</code>  |
| <code>  <math>\forall x \forall y [\text{menikah}(x,y) \wedge \text{lelaki}(x)] \rightarrow \neg \text{lelaki}(y).</math></code> | <code>// aturan</code> |
| <code>  <math>\forall y \exists x [\text{orang}(y) \rightarrow \text{ibu}(x,y)].</math></code>                                   | <code>// aturan</code> |

Kalimat pertama menunjukkan adanya fakta bahwa Joko adalah seorang lelaki. Kalimat kedua menyatakan bahwa Joko menikah dengan Tuti. Kalimat ketiga dan keempat menunjukkan suatu aturan atau kaidah yang umum berlaku. Simbol predikat yang digunakan dalam kalimat-kalimat tersebut adalah lelaki, menikah, orang, dan ibu yang sering disebut sebagai relasi, sedangkan Joko dan Tuti disebut sebagai simbol konstanta.

## 9.3 Bahasa Deklaratif

Sebagai bukti bahwa Prolog merupakan bahasa deklaratif adalah dalam menyatakan fakta dan aturan seperti berikut:

- Jika ingin menyatakan bahwa “Prawiro adalah bapak dari Joko”, maka dalam Prolog dituliskan sebagai:

bapak(prawiro, joko).

Jika ingin menerangkan suatu kaidah bahwa A adalah kakek dari Z maka harus dibuat dahulu logika dalam bahasa Indonesia sehingga menjadi suatu aturan seperti berikut:

A adalah kakek Z jika A adalah bapak dari X dan X adalah bapak Z

atau

A adalah kakek Z jika A adalah bapak dari X dan X adalah ibu Z

Tetapi dalam Prolog dituliskan sebagai:

kakek(A,Z):- bapak(A,X), bapak(X,Z).

atau

kakek(A,Z):- bapak(A,X), ibu(X,Z).

## 9.4 Pemrograman Prolog

Fakta adalah suatu kenyataan atau kebenaran yang diketahui, dan menyatakan hubungan (relasi) antara dua atau lebih obyek. Fakta dapat pula menunjukkan sifat suatu obyek. Contoh:

- bapak(prawiro, joko).
- merah(darah).
- asin(garam).

Aturan merupakan logika yang dirumuskan dalam bentuk relasi sebab-akibat dan hubungan implikasi. Misalnya dapat dibuat aturan bahwa jika A adalah bapak dari X dan X adalah bapak atau ibu dari Z maka dapat dipastikan bahwa A adalah kakek dari Z. Contoh:

kakek(A,Z):- bapak(A,X), bapak(X,Z).

kakek(A,Z):- bapak(A,X), ibu(X,Z).

Klausa adalah aturan yang ditulis berupa klausa (clause) dan terdiri dari head (kakek) dan tail yang dipisahkan oleh tanda (minus) - (bapak dan ibu). Klausa selalu diakhiri dengan tanda titik (.). Suatu tail klausa dapat terdiri dari beberapa sub-klausa yang dihubungkan dengan tanda koma (,) yang berarti hubungan and dan tanda titik koma (;) yang menunjukkan hubungan or. Contoh:

orangtua(P,Q):- bapak(P,Q); ibu(P,Q).

kakek(A,Z):- bapak(A,X), orangtua(X,Z).

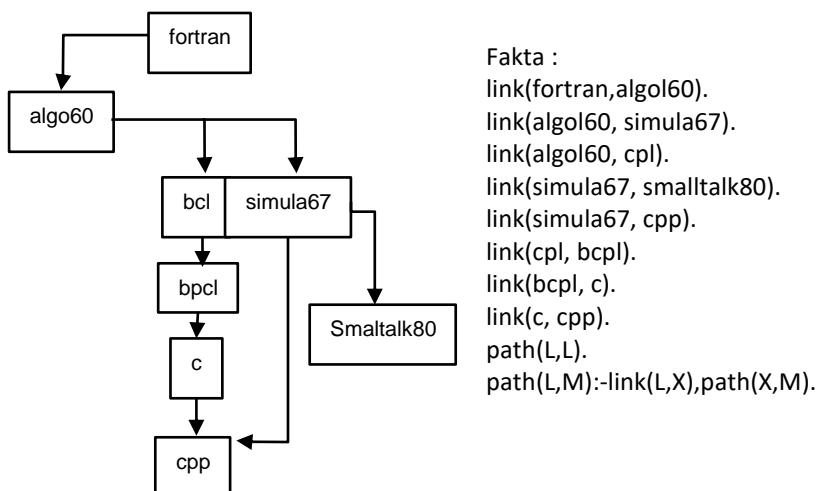
Variabel adalah argumen suatu predikat dapat berupa konstanta (atom), variabel, atau obyek lain. Suatu atom, variabel, atau obyek lain dalam Prolog disebut term, sehingga argumen selalu berupa term. Dalam Prolog terdapat dua variabel, yaitu

1. Variabel bernama, seperti X, Orang, dan sebagainya
2. Variabel tak bernama (placeholder), dilambangkan (\_).

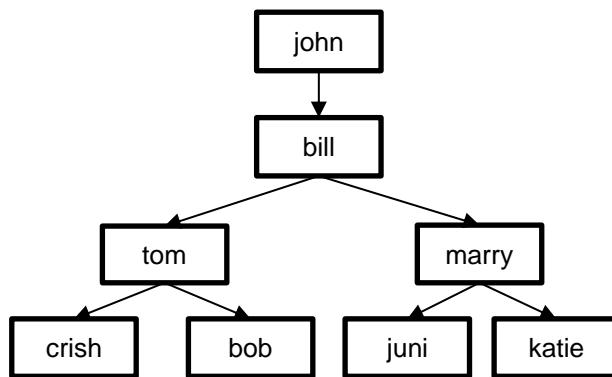
Setiap term yang ditulis dengan awalan huruf kapital selalu dianggap sebagai variabel bernama dalam Prolog, sedangkan awalan dengan huruf kecil dianggap sebagai suatu relasi atau konstanta. Variabel tak bernama digunakan untuk mengabaikan nilai suatu variabel, yang berarti bisa bernilai apa saja. Contoh:

member(X,[X|\_]).

member(X,[\_|Y]):- member(X,Y)



Gambar 9.1 Contoh Gambaran Prolog 1



Gambar 9.2 Contoh Gambaran Prolog 2

Fakta:

```
father(john,bill).
father(bill,mary).
father(bill,tom).
father(tom,chris).
father(tom,bob).
mother(mary,june).
mother(mary,katie)
```

Contoh Program Keluarga:

1. Fakta:

```
female(pat).
male(jim),
offspring(Y,X):- parent(X,Y).
mother(X,Y):- parent(X,Y),female(X).
grandparent(X,Z):- parent(X,Y),parent(Y,Z).
sister(X,Y):-parent(Z,X),parent(Z,Y),female(X), diffeent(X,Y).
predecessor(X,Z):- parent(X,Z).
predecessor(X,Z):- parent(X,Y),predecessor(Y,Z).
```

2. Fakta:

```
parent(pam,bob).
```

% Pam is a parent of Bob

parent(tom,bob).

parent(tom,liz).

parent(bob,ann).

parent(bob,pat).

parent(pat,jim).

female(pam).

% Pam is female

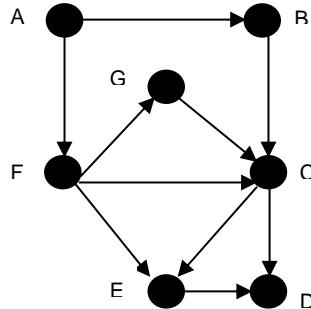
male(tom).

male(bob).

female(liz).

female(ann).

Query adalah pertanyaan yang digunakan untuk memperoleh jawaban dari suatu problem. Contoh:



Gambar 9.3 Contoh Gambaran Query

Fakta:

edge(a,b).

edge(a,e).

edge(b,d).

edge(b,c).

edge(c,a).

edge(e,b).  
edge(X,Y):- tedge(X,Y).

edge(Node1,Node2):-edge(Node1,SomeNode), edge(Some-Node,Node2).

Query:

?- edge(a,c).

?- edge(a,b).

?- edge(e,c).

- FOL form:  $\forall X (barks(X) \wedge wags\_tail(X) \Rightarrow \text{dog}(X))$
- Knowledge Base

dog(X):- barks(X), wags\_tail(X).

barks(woff).

barks(spot).

wags\_tail(woff).

Queries

?- dog(woff) => yes

?- dog(Y) => Y = woff (menggunakan Variabel)

?- dog(spot) => no

Means no more matches found.

- Knowledge Base:

big(bear).

big(elephant).

small(cat).

brown(bear).

black(cat).

gray(elephant).

dark(Z):- black(Z).

dark(Z):- brown(Z).

Queries:

?- dark(X), big(X).

- Knowledge Base:

suka(bejo, cilok).

suka(gondo, cilok).

suka(wulan, bakso).

teman(X, Y):- \+(X = Y), suka(X, Z), suka(Y, Z). Constraint diletakkan pada awal rule

atau

teman(X, Y):- suka(X, Z), suka(Y, Z), \+(X = Y). →

Queries:

?- suka(bejo, cilok).

Constraint diletakkan pada akhir rule

?- teman(bejo, gondo).

?- suka(bejo, What).

?- suka(Who, bakso).

- Knowledge Base:

written\_by(fleming, "DR NO").

written\_by(melville, "MOBY DICK").

book("MOBY DICK", 250).

book("DR NO", 310).

long\_novel>Title):-

written\_by(\_, Title),

book>Title, Length,

Length > 300.

Queries:

?- written\_by(X,Y)

Penyelesian (Unification):

?- written\_by(X,Y)

written\_by( X , Y ).

| |

written\_by(fleming, "DR NO").

written\_by( X , Y ).

| |

written\_by(melville, "MOBY DICK").

Prolog menampilkan semua solusi:

X=fleming, Y=DR NO

X=melville, Y=MOBY DICK

2 Solusi

- Knowledge Base:

written\_by(fleming, "DR NO").

written\_by(melville, "MOBY DICK").

book("MOBY DICK", 250).

book("DR NO", 310).

long\_novel>Title):-

written\_by(\_, Title),

book>Title, Length),

Length > 300.

Queries:

?- written\_by(X,Y)

?- written\_by(X, "MOBY DICK")

Penyelesian (Unification):

?- written\_by(X, "MOBY DICK")

?- written\_by(X, "MOBY DICK").

| |

written\_by(fleming, "DR NO").

FAIL

?- written\_by(X, "MOBY DICK").

| |

written\_by(melville, "MOBY DICK").

Prolog menampilkan 1 solusi:

X=melville

- Knowledge Base:

written\_by(fleming, "DR NO").

written\_by(melville, "MOBY DICK").

book("MOBY DICK", 250).

book("DR NO", 310).

long\_novel>Title):-

written\_by(\_, Title),

book>Title, Length),

Length > 300.

Queries:

?- written\_by(X,Y)

?- written\_by(X, "MOBY DICK")

?- long\_novel(X)

Penyelesian (Unification):

?- long\_novel(X)

?- long\_novel(X)

|

long\_novel>Title):-

written\_by(\_, Title),

book>Title, Length),

Length > 300.

written\_by(\_, Title)

| |

written\_by(fleming,"DR NO").

book>Title, Length)

|

book("DR NO", Length).

|

book("DR NO", 310).

Length > 300

|

310 > 300.

Prolog menampilkan 1 solusi:

X=DR NO

- Knowledge Base: (Backtracking)

likes(bill,X):-food(X),tastes(X,good).

tastes(pizza,good).

tastes(brussels\_sprouts,bad).

food(brussels\_sprouts).

food(pizza).

Queries:

?- likes(bill, What).

Penyelesaian (Backtracking):

likes(bill, What).

|

likes(bill, X)

food(X)

| backtracking point

food(brussels\_sprouts)

tastes(X,good).

|

|

FAIL

tastes(brussels\_sprouts,bad).

Backtracking

food(X)

|

food(pizza)

tastes(X, good).

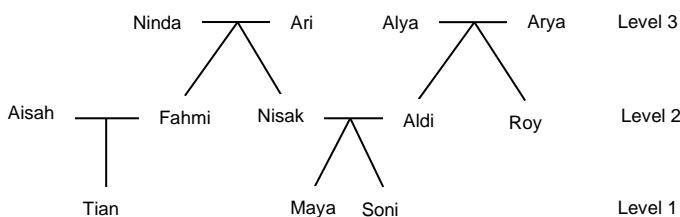
| |

tastes(pizza,good).

X => What = pizza

## 9.5 Latihan Individu

1. Perhatikan silsilah keluarga berikut:



- a. Buatlah Code Prolog sesuai fakta yang ada dari silsilah keluarga di atas (kakek, nenek, bibi, paman, sepupu, etc)!
- b. Buatlah minimal 2 query yang merepresentasikan permasalahan di atas dalam Code Prolog!

## 9.6 Tugas Kelompok

1. Jalankan dengan prolog slide ke-18 dan slide ke-21. Berikan screenshot detail coding dan hasil running querynya!
2. Buat KB dan Query dalam prolog, minimal 10 fakta, minimal 2 aturan tentang biodata minimal 2 anggota kelompok anda, dengan minimal 4 query!

- a. (Aturan bisa dibuat dari keminatan base Lab, status persahabatan, etc.)
3. Perhatikan Program Prolog berikut:

Fakta:

```
male(james1).
male(charles1).
male(charles2).
male(james2).
male(george1).
female(catherine).
female(elizabeth).
female(sophia).
parent(charles1, james1).
parent(elizabeth, james1).
parent(charles2, charles1).
parent(catherine, charles1).
parent(james2, charles1).
parent(sophia, elizabeth).
parent(george1, sophia).
```

Ubahlah pertanyaan berikut menjadi Program Prolog & Jawaban!

- a. Was George1 the parent of Charles1?
- b. Who was Charles1's parent?
- c. Who were the children of Charles1?

## BAB 10 Ketidakpastian (Uncertainty)

### 10.1 Uncertainty

Contoh ada sebuah agent yang perlu ke bandara karena akan terbang ke LN. Mis. action  $A_t$  = pergi ke bandara  $t$  menit sebelum pesawat terbang. Apakah  $A_t$  berhasil sampai dengan waktu cukup? Ada banyak masalah:

- Tidak tahu keadaan jalan, kemacetan, dll. (partially observable).
- Kebenaran informasi tidak bisa dijamin—"laporan pandangan mata" (noisy sensor).
- Ketidakpastian dalam tindakan, mis. ban kempes (nondeterministic).
- Kalaupun semua hal di atas bisa dinyatakan, reasoning akan luar biasa repot.
- Sebuah pendekatan yang murni secara logika
  - beresiko menyimpulkan dengan salah, mis: "A<sub>60</sub> berhasil dengan waktu cukup", atau
  - kesimpulan terlalu lemah, mis: "A<sub>60</sub> berhasil dengan waktu cukup asal nggak ada kecelakaan di tol, dan nggak hujan, dan ban nggak kempes,..."
  - kesimpulan tidak rational, mis: kesimpulannya A<sub>1440</sub>, tetapi terpaksa menunggu semalam di bandara (utility theory).
- Masalah ini bisa diselesaikan dengan probabilistic reasoning
  - Berdasarkan info yang ada, A<sub>60</sub> akan berhasil dengan probabilitas 0.04".

Kalimat "A<sub>60</sub> akan berhasil dengan probabilitas 0.04" disebut probabilistic assertion. Sebuah probabilistic assertion merangkum efek ketidakpastian (info tak lengkap, tak bisa dipegang, action nondeterministic, dst.) dan menyatakannya sebagai sebuah bilangan. Bentuk/syntax probabilistic assertion:

- “Kalimat X bernilai true dengan probabilitas N,  $0 \leq N \leq 1$ ”.
- Pernyataan tentang knowledge atau belief state dari agent, BUKAN berarti pernyataan tentang sifat probabilistik di dunia/environment.
- Nilai probabilitas sebuah proposition bisa berubah dengan informasi baru (“evidence”):

$$P(A_{60} | \text{tidak ada laporan kecelakaan}) = 0.06$$

$$P(A_{60} | \text{tidak ada laporan kecelakaan, jam 4 pagi}) = 0.15$$

## 10.2 Probability

- Probabilistic reasoning:
  - Percept masuk (tambahan evidence), update nilai probabilitas.
  - Prior/unconditional probability: nilai sebelum evidence.
  - Posterior/conditional probability: nilai sesudah evidence.
  - “ASK” secara probabilistik: hitung & kembalikan posterior probability terhadap  $\alpha$  berdasarkan evidence dari percept.

Contoh permasalahan untuk probability yaitu melempar dadu.

$$\alpha = \text{“Nilai lemparan } < 4\text{”}$$

Sebelum melihat dadu:

$$P(\alpha) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{1}{2}$$

Setelah melihat dadu:

$$P(\alpha) = 0 \text{ atau } 1$$

Mengambil keputusan dlm ketidakpastian, andaikan agent mempercayai nilai-nilai sebagai berikut:

$$P(A_{60} | \dots) = 0.04$$

$$P(A_{120} | \dots) = 0.7$$

$$P(A_{150} | \dots) = 0.9$$

$$P(A_{1440} | \dots) = 0.999$$

- Tindakan mana yang dipilih?
  - Tergantung prioritas, mis. ketinggalan pesawat vs. begadang di lobby bandara, dst.
  - Utility theory digunakan untuk menilai semua tindakan (mirip evaluation function).
  - Decision theory = utility theory + probability theory

Sama halnya dengan logic, pendefinisian “bahasa formal” untuk menyatakan kalimat probabilistic harus ada syntax (bagaimana bentuk kalimatnya), Semantics (apakah arti kalimatnya), Teknik & metode melakukan reasoning.

## 10.3 Semantics & Syntax

Semantics untuk kalimat probabilistic (Bayangkan semua kemungkinan dunia *possible worlds* yang terjadi). Dalam logic, salah satunya adalah dunia “nyata”. Dalam probability, kita tidak tahu pasti yang mana, tetapi satu dunia bisa lebih mungkin dari dunia yang lain.

Himpunan semua *possible worlds* disebut *sample space* ( $\Omega$ ). Masing-masing dunia alternatif disebut *sample point*, atau *atomic event* ( $\omega$ ). Contohnya yaitu sebagai berikut:

- Jika dunia hanya berisi sebuah lemparan dadu  $\Omega$  berisi 6 kemungkinan,  $\omega_1 \dots \omega_6$ .

Sebuah probability model adalah *sample space* di mana tiap *sample point* diberi nilai  $P(\omega)$  sehingga:

- Setiap nilai antara 0 s/d 1.
- Jumlah nilai seluruh sample space = 1.

Contohnya, untuk “dunia” dengan 1 lemparan dadu:

- $P(\omega_1) = P(\omega_2) = P(\omega_3) = P(\omega_4) = P(\omega_5) = P(\omega_6) = 1/6$

Biasanya, dunia memiliki  $> 1$  faktor yang tidak pasti. *Sample space* dan *probability model* menjadi multidimensi, menyatakan semua kemungkinan kombinasinya. Contohnya, untuk “dunia” dengan 2 lemparan dadu:

- $P(\omega_{1,1}) = P(\omega_{1,2}) = \dots = P(\omega_{6,5}) = P(\omega_{6,6}) = 1/36$

Di dalam dunia multidimensi, terkadang kita hanya tertarik dengan 1 dimensi (mis. lemparan dadu pertama). Sebuah event A adalah sembarang subset dari  $\Omega$ . Probability A adalah jumlah probability sample point anggotanya.

$$P(A) = \sum_{\omega \in A} P(\omega)$$

Contohnya,

$$P(\text{dadu}_1 = 5) = 6 \times \frac{1}{36} = \frac{1}{6}$$

Event juga bisa menyatakan probability dari deskripsi parsial. Contohnya untuk satu lemparan dadu:

$$P(\text{dadu} \geq 4) = 3 \times \frac{1}{6} = \frac{1}{2}$$

Nilai probabilitas diberikan kepada sebuah proposition. Agar proposition dapat diperinci, kita definisikan random variable, yang merepresentasikan suatu “aspek” dari sebuah dunia. Contohnya, dalam kasus melempar dadu:

- Bisa ada random variable bernama *hasil\_lemparan*.

Secara formal, random variable adalah fungsi yang memetakan setiap sample point ke dalam ranah, mis. boolean, integer, real. Contohnya:

- *hasil\_lemparan* adalah fungsi yang memetakan  $\omega_1$  ke integer 1,  $\omega_2$  keinteger 2,  $\omega_3$  ke integer 3, dst.

Sekarang semua proposition berupa pernyataan tentang satu atau lebih *random variable*. Nilai probabilitas diberikan kepada sebuah proposition. Agar proposition dapat diperinci, kita definisikan random variable, yang merepresentasikan suatu “aspek” dari sebuah dunia. Contohnya, dalam kasus melempar dadu:

- Bisa ada random variable bernama *hasil\_lemparan*.

Secara formal, random variable adalah fungsi yang memetakan setiap sample point ke dalam ranah, mis. boolean, integer, real. Contohnya:

- *hasil\_lemparan* adalah fungsi yang memetakan  $\omega_1$  ke integer 1,  $\omega_2$  keinteger 2,  $\omega_3$  ke integer 3, dst.

Sekarang semua proposition berupa pernyataan tentang satu atau lebih *random variable*. Domain sebuah random variable bisa:

- boolean, mis:  $\text{Ganjil}(\omega_1) = \text{true}$
- diskrit, mis:  $\text{Weather}(\omega) \in (\text{sunny}, \text{rain}, \text{cloudy}, \text{snow})$
- takhingga, mis: integer (diskrit), real (kontinyu)

Sebuah probability model  $P$  menghasilkan probability distribution untuk sembarang random variable:

$$P(X = x_i) = \sum_{\omega: X(\omega) = x_i} P(\omega)$$

Contoh dengan dadu:

$$P(\text{Ganjil} = \text{true}) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{1}{2}$$

Contoh dgn cuaca:

$$P(\text{Weather} = \text{sunny}) = 0.7$$

$$P(\text{Weather} = \text{rain}) = 0.2$$

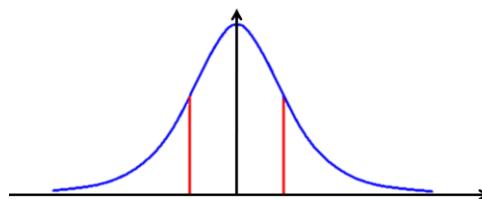
$$P(\text{Weather} = \text{cloudy}) = 0.08$$

$$P(\text{Weather} = \text{snow}) = 0.02$$

$$\text{atau disingkat } P(\text{Weather}) = (0.7, 0.2, 0.08, 0.02)$$

Contoh distribution untuk variable real & kontinyu yang banyak ditemui dalam dunia nyata adalah fungsi Gaussian:

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$



Gambar 10.1 Contoh Distribution untuk Variable Real & Kontinyu

Dalam AI, seringkali sample point didefinisikan oleh nilai sekumpulan random variable. Jadi, sample space berisi semua kemungkinan

kombinasi nilai semua variable. Joint probability distribution dari se-himpunan random variable memberikan nilai probability untuk setiap sample point tersebut. Andaikan kita tertarik mengamati hubungan cuaca dengan sakit gigi, berikut contoh joint probability distribution-nya:

Tabel 10.1 Contoh Joint Probability

| Weather =   |       | sunny | rain | cloudy | snow |
|-------------|-------|-------|------|--------|------|
| Toothache = | true  | 0.144 | 0.02 | 0.016  | 0.02 |
| Toothache = | false | 0.576 | 0.08 | 0.064  | 0.08 |

Sebuah proposition adalah pernyataan tentang nilai dari satu atau lebih random variable. Bayangkan proposition sebagai event (himpunan sample point) di mana ia bernilai true. Untuk 2 buah random variable boolean A dan B:

- Event  $a =$  himpunan sample point di mana  $A(\omega) = \text{true}$
- Event  $\neg a =$  himpunan sample point di mana  $A(\omega) = \text{false}$
- Event  $a \wedge b =$  himpunan sample point di mana  $A(\omega)$  dan  $B(\omega) = \text{true}$
- Event  $a \vee b =$  himpunan sample point di mana  $A(\omega)$  atau  $B(\omega) = \text{true}$

Contoh yang memilukan, bayangkan masalah dokter gigi, di mana ada 3 random variable:

- Cavity: apakah pasien memiliki gigi berlubang atau tidak?
- Toothache: apakah pasien merasa sakit gigi atau tidak?
- Catch: apakah pisau dokter nyangkut di gigi pasien atau tidak?

Joint probability distribution sebagai berikut:

Tabel 10.2 Joint Probability Distribution

|               | toothache |              | $\neg$ toothache |               |
|---------------|-----------|--------------|------------------|---------------|
|               | catch     | $\neg$ catch | catch            | $\neg$ cavity |
| cavity        | 0.108     | 0.012        | 0.072            | 0.008         |
| $\neg$ cavity | 0.016     | 0.64         | 0.144            | 0.576         |

- Prior vs. posterior probability
  - Prior adalah nilai probability tanpa informasi spesifik (unconditional).

Contoh:  $P(\text{cavity})$ ,  $P(\text{toothache} \wedge \text{catch})$ , dst.

- Posterior: Nilai probability jika sesuatu informasi spesifik diketahui (conditional).

Contoh:  $P(\text{cavity} | \text{toothache})$ .

("Peluang jika seseorang sakit gigi maka giginya berlubang")

- Definisi *conditional probability*:

$$P(a | b) = \frac{P(a \wedge b)}{P(b)}$$

- Perumusan alternatif (Product rule):

$$P(a \wedge b) = P(a | b)P(b) = P(b | a)P(a)$$

## 10.4 Inferensi Atau Penalaran

Dengan *joint probability distribution*, probability sembarang proposition bisa dihitung sbg. jumlah probability *sample point* di mana ia bernilai true. Contoh dapat dilihat pada Tabel 10.3:

Tabel 10.3 Contoh Joint Probability Distribution

|         | toothache |        | ¬toothache |         |
|---------|-----------|--------|------------|---------|
|         | catch     | ¬catch | catch      | ¬cavity |
| cavity  | 0.108     | 0.012  | 0.072      | 0.008   |
| ¬cavity | 0.016     | 0.64   | 0.144      | 0.576   |

Maka Penyelesaiannya:

$$\begin{aligned} P(\text{cavity} \vee \text{toothache}) &= 0.108 + 0.012 + 0.072 + 0.008 + 0.016 \\ &\quad + 0.064 \\ &= 0.28 \end{aligned}$$

Bisa juga menghitung conditional probability:

$$\begin{aligned} P(\neg\text{cavity} | \text{toothache}) &= \frac{P(\neg\text{cavity} \wedge \text{toothache})}{P(\text{toothache})} \\ &= \frac{0.016 + 0.064}{0.108 + 0.012 + 0.016 + 0.064} = 0.4 \end{aligned}$$

## 10.5 Aturan Bayes (just review)

Mendefinisikan fitur untuk setiap objek dengan:

$$P(x | \omega_1) \text{ & } P(x | \omega_2)$$

(Probabilitas kodisional objek (x) terhadap kelas ( $\omega_j$ ) / Likelihood).

$$P(\omega_j | x) = \frac{P(x | \omega_j)P(\omega_j)}{P(x)}$$

$$\text{Posterior} = \frac{\text{Likelihood} * \text{Prior}}{\text{Evidence}}$$

$$P(x) = \sum_{j=1}^2 P(x | \omega_j)P(\omega_j)$$

## 10.6 Latihan Individu

1. Ubahlah FOL berikut menjadi bentuk CNF
  - a.  $\forall x [\forall y A(y) \leftrightarrow L(x, y)] \rightarrow [\exists y L(y, x)]$
  - b.  $\forall x \forall y [P(x) \wedge Q(y)] \rightarrow [\exists z R(x, y, z)]$
  - c.  $\exists x \forall y \exists z [P(x) \rightarrow [Q(y) \rightarrow R(z)]]$
  - d. Not everybody is your friend or someone is not perfect
2. Seorang dokter mengetahui bahwa penyakit meningitis (M) menyebabkan pasien memiliki leher kaku (K) dengan kemungkinan sebesar 50%. Dokter juga mengetahui besarnya probabilitas dari pasien yang memiliki meningitis adalah 1/50000, dan probabilitas dari setiap pasien memiliki leher kaku adalah 1/20. Tentukan besarnya peluang seseorang memiliki leher kaku yang mengindikasikan penyakit meningitis!
3. Jika diketahui  $P(A) = 7/11$ ,  $P(B) = 6/11$ ,  $P(A \wedge B) = 2/11$ . Tentukan nilai peluang berikut!
  - a.  $P(A|B) \text{ & } P(B|A)$

- b.  $P(\neg A | B) \& P(B | \neg A)$
- c.  $P(\neg A | \neg B)$
- d.  $P(\neg B | \neg A)$

## 10.7 Tugas Kelompok

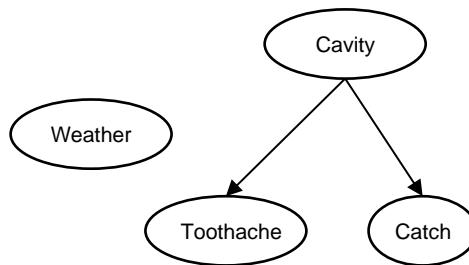
1. Buatlah Min. 2 Draft Judul Final Project(FP) + Deskripsi FP ( $\pm 10$  kalimat)
2. Buatlah minimal 1 *case study* unik dan penyelesaiannya dengan Teorema Bayes!

## BAB 11 Bayesian Network

### 11.1 Syntax & Semantics

Bayesian Network adalah notasi graf yang menyatakan conditional independence dalam suatu domain. Node menyatakan sebuah random variable. Arc (directed edge) menyatakan hubungan kausal langsung (direct influence). Arahnya dari variable “sebab” ke variable “akibat”.

Node sibling menyatakan variable yang conditionally independent karena parent-nya. Conditional distribution untuk setiap node terhadap parent-nya:  $P(X_i | \text{Parents}(X_i))$ . Tidak ada cycle di dalam Bayesian Network. Contoh kedokteran gigi yang digambarakan dengan topologi sebuah Bayesian Network menyatakan hubungan conditional independence sebagai berikut:



Gambar 11.1 Bayesin Network Conditional Independence

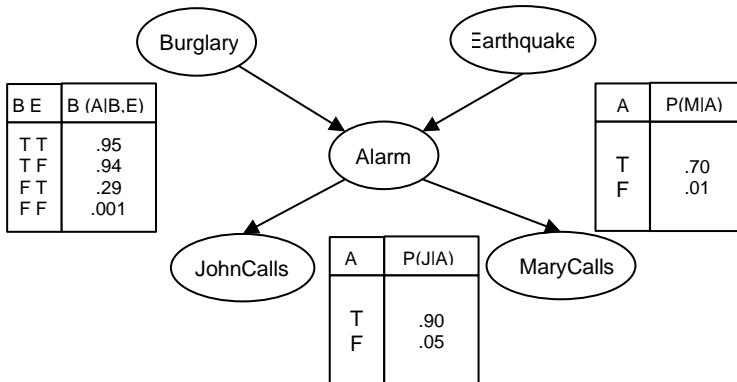
Keterangan:

- Weather independent dari semua variable lain.
- Toothache dan Catch conditionally independent karena Cavity.
- Contoh lain:
  - Anto sedang di kantor. Tetangganya, John, menelpon mengatakan alarm anti-perampoknya bunyi. Tetangganya, Mary, tidak menelpon. Kadang-kadang alarmnya nyala karena gempa bumi. Apakah ada perampok di rumah Anto?
  - Variable dalam domain:

*Burglar, Earthquake, Alarm, JohnCalls, MaryCalls*

- Hubungan sebab akibat:
  - Perampok bisa membuat alarm nyala.
  - Gempa bumi bisa membuat alarm nyala.
  - Alarm bisa membuat John menelpon.
  - Alarm bisa membuat Mary menelpon.

- Contoh Bayesian Network



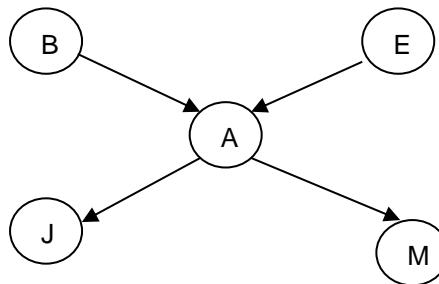
Gambar 11.2 Bayesin Network

- Rekonstruksi full joint distribution:
  - Bayesian Network adalah deskripsi lengkap sebuah domain.
  - Full joint distribution bisa diperoleh dari local conditional distribution:
$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | Parents(x_i))$$
  - Contoh: hitung probabilitas John menelpon, Mary menelpon, alarm nyala, tidak ada perampok, tidak ada gempa bumi.

$$P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) =$$

$$P(j | a)P(m | a)P(a | \neg b, \neg e)P(\neg b)P(\neg e) =$$

$$0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 = 0.00062$$



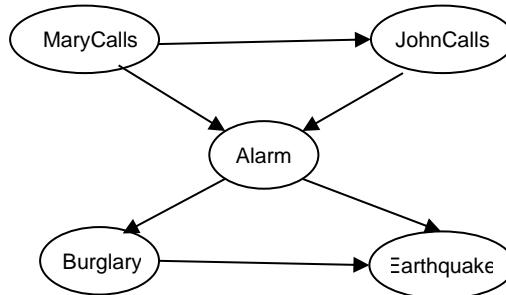
Gambar 11.3 Contoh Gambaran Probabilitas

- Membangun Bayesian Network:
  - Bagaimana membangun sebuah Bayesian Network?
  - Sebuah algoritma:
    - Pilih ordering variable  $X_1, \dots, X_n$
    - For  $i = 1$  to  $n$ 
      - Tambahkan  $X_i$  ke network
      - Pilih parent dari  $X_1, \dots, X_{i-1}$  shg.
$$P(X_i | \text{Parents}(X_i)) = P(X_i | X_1, \dots, X_{i-1})$$
      - Agar Bayesian Network sah maka:  
 $X_i$  harus conditionally independent terhadap semua  $X_1, \dots, X_{i-1}$  yang bukan anggota  $\text{Parents}(X_i)$  karena  $\text{Parents}(X_i)$ .
  - Chain rule & conditional independence:
    - Algoritma di slide sebelumnya menggunakan chain rule:
$$\begin{aligned} P(A, B, C, D) &= P(A|B, C, D)P(B, C, D) \\ &= P(A|B, C, D)P(B|C, D)P(C, D) \\ &= P(A|B, C, D)P(B|C, D)P(C|D)P(D) \end{aligned}$$
    - Ini spt. membangun Bayesian Network dengan urutan D, C, B, A tanpa conditional independence.
    - Bagaimana jika, misal:
      - A conditionally independent thd. B karena C dan D
      - B conditionally independent thd. C karena D:

$$P(A, B, C, D) = P(A|C, D)P(B|D)P(C|D)P(D)$$

- Contoh membangun Bayesian Network:

- Mis. kita pilih urutan: MaryCalls, JohnCalls, Alarm, Burglar, Earthquake.



$$P(J | M) = P(J)? \text{Tidak}$$

$$P(A | J, M) = P(A | J)? \text{P}(A | J, M) = P(A)? \text{Tidak}$$

$$P(B | A, J, M) = P(B | A)? \text{Ya}$$

$$P(B | A, J, M) = P(B)? \text{Tidak}$$

$$P(E | B, A, J, M) = P(E | A)? \text{Tidak}$$

$$P(E | B, A, J, M) = P(E | A, B)? \text{Ya}$$

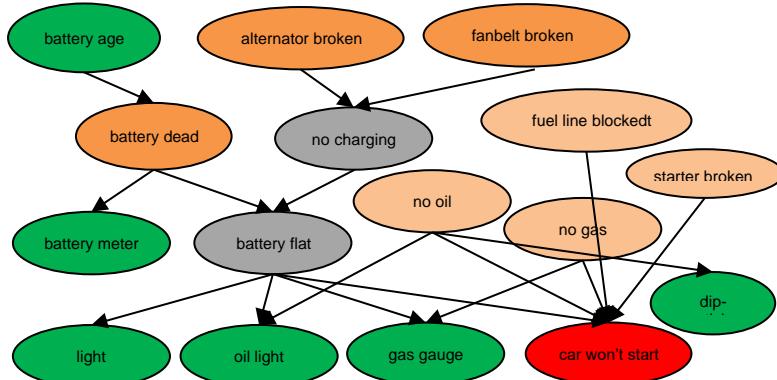
- Naive vs. paranoid:

- Naive Bayes model
  - Semua variable akibat dianggap saling conditionally independent karena variable sebab.
- Full joint distribution (paranoid?)
  - Semua random variable dianggap saling mempengaruhi

Yang kita cari adalah analisa domain-specific yang menghasilkan informasi conditional independence yang benar. Contoh yang lebih rumit:

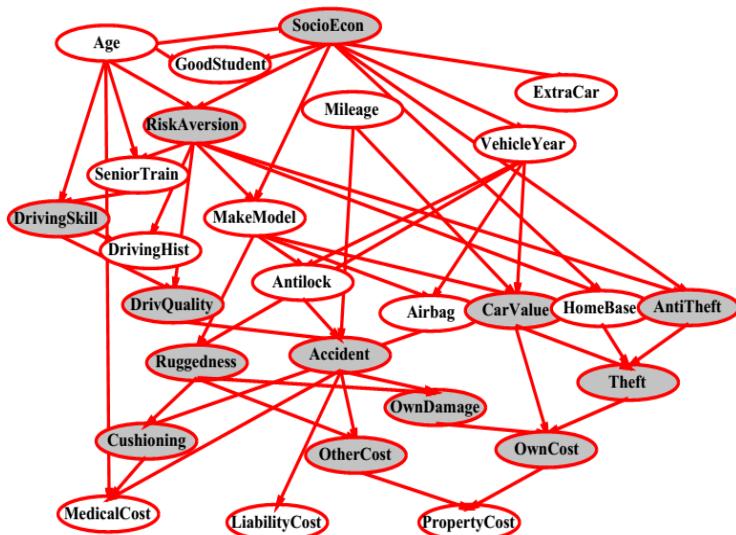
- Diagnosa awal: mobil mogok!
- Testable node: nilainya bisa diukur.
- Fixable node: nilainya bisa diatur.

- Hidden node: hanya untuk menyederhanakan struktur network-nya.



Gambar 11.4 Struktur Network

- Menentukan nilai asuransi mobil



Gambar 11.5 Struktur Full Joint Distribution

## 11.2 Compact conditional distributions

Deterministic nodes adalah Conditional distribution sebuah node dgn. k parent exponential dalam K. Ada beberapa representasi yang lebih efisien → canonical distribution. Conditional distribution

dari suatu deterministic node bisa dihitung sepenuhnya dari nilai parent-nya. Dengan kata lain, nilai probabilitasnya bisa dinyatakan sebagai suatu fungsi:  $X = f(\text{Parents}(X))$ . Misalnya, “hidden” variable pada contoh mobil mogok:

$$\text{No\_charging} = \text{Alternator\_broken} \vee \text{Fanbelt\_broken}$$

$$\text{Battery\_flat} = \text{Battery\_dead} \vee \text{No\_charging}$$

Nilainya diperoleh dari truth table  $\vee$

Noisy-OR distribution mirip  $\vee$  dalam logic, tapi ada *uncertainty*: Berapakah ketidakpastian sebuah variable “gagal” mengakibatkan proposition bernilai true? Contohnya yaitu:

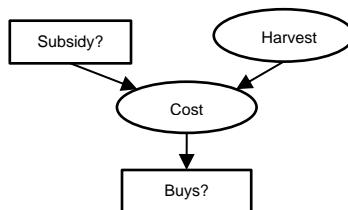
$$P(\neg\text{fever} | \text{cold}, \neg\text{flu}, \neg\text{malaria}) = 0.6$$

$$P(\neg\text{fever} | \neg\text{cold}, \text{flu}, \neg\text{malaria}) = 0.2$$

$$P(\neg\text{fever} | \neg\text{cold}, \neg\text{flu}, \text{malaria}) = 0.1$$

| Cold | Flu | Malaria | $P(\text{Fever})$ | $P(\neg\text{Fever})$               |
|------|-----|---------|-------------------|-------------------------------------|
| F    | F   | F       | 0.0               | 1.0                                 |
| F    | F   | T       | 0.9               | 0.1                                 |
| F    | T   | F       | 0.8               | 0.2                                 |
| F    | T   | T       | 0.98              | $0.02 = 0.2 \times 0.1$             |
| T    | F   | F       | 0.4               | 0.6                                 |
| T    | F   | T       | 0.94              | $0.06 = 0.6 \times 0.1$             |
| T    | T   | F       | 0.88              | $0.12 = 0.6 \times 0.2$             |
| T    | T   | T       | 0.988             | $0.012 = 0.6 \times 0.2 \times 0.1$ |

- Variable dengan nilai kontinyu:
  - Bagaimana kalau nilai variable kontinyu? Tabel?
  - Gunakan canonical distribution: fungsi dengan parameter. Contoh:



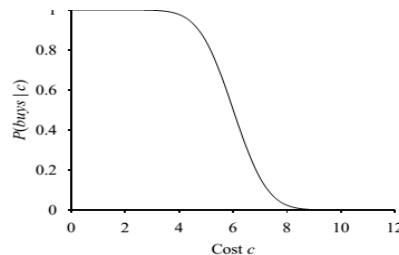
Gambar 11.6 Variable dengan nilai kontinyu

Diskrit: Subsidy?, Buys?

Kontinyu: Harvest, Cost

- Variable diskrit, parent kontinyu:

- Probabilitas dari *Buy?* jika diketahui Cost adalah “soft threshold”:



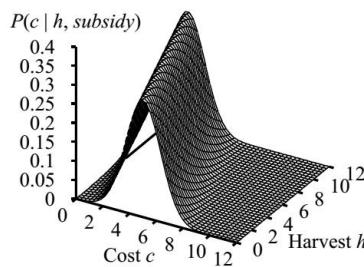
Gambar 11.7 Distribusi Integral Gaussian

$$\phi(x) = \int_{-\infty}^x N(0,1)(x) dx$$

- Distribusi prob. adalah integral dari fungsi Gaussian:

$$P(\text{Buys?} = \text{true} \mid \text{Cost} = c) = \Phi((-c + \mu)/\sigma)$$

- Variable kontinyu:



Gambar 11.8 Ilustrasi Variable Kontinyu

- Model Linear Gaussian sering dipakai:

$$P(\text{Cost} = c \mid \text{Harvest} = h, \text{Subsidy?} = \text{true}) = N(a_t h + b_t, \sigma_t)(c)$$

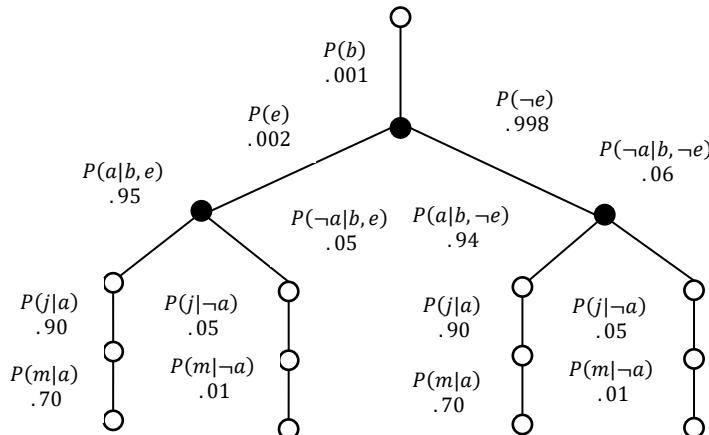
$$= \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{1}{2} \left( \frac{c - (a_t h + b_t)}{\sigma_t} \right)^2\right)$$

## 11.3 Efficient Inference

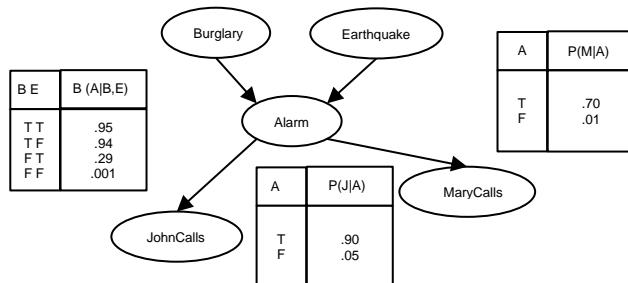
Contoh dari Inference by enumeration misal hitung probabilitas ada perampok jika John dan Mary menelpon.

$$P(b | j, m) = \alpha \sum_e \sum_a P(b)P(e)P(a | b, e)P(j | a)P(m | a)$$

$$= \alpha P(b) \sum_e P(e) \sum_a P(a | b, e)P(j | a)P(m | a)$$



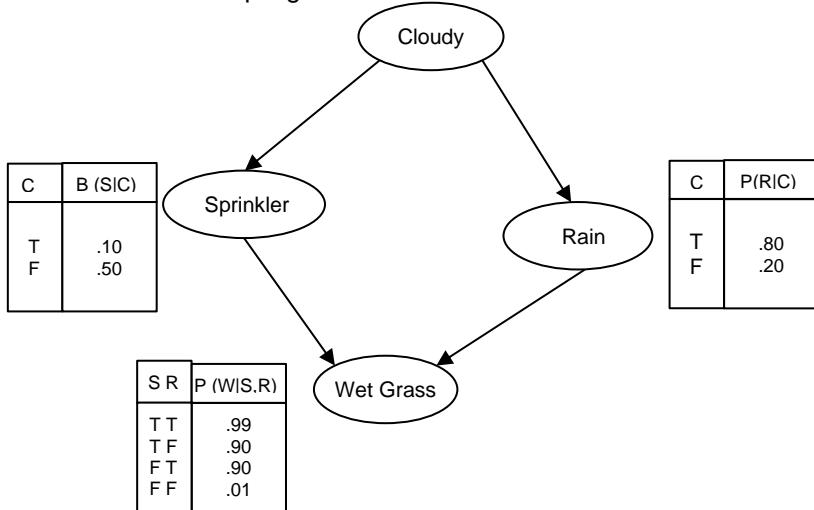
Gambar 11.9 Contoh Path Efficient Inference



Gambar 11.10 Contoh Path Efficient Inference

- Perhatikan bahwa  $P(j | a)P(m | a)$  dihitung untuk setiap nilai  $a$ . Gunakan dynamic programming: hitung sekali, simpan hasilnya!

- Approximate inference:
  - Pendekatan lain: jangan hitung nilai persis, tapi cukup di-sample (Monte Carlo).
  - Ide dasar:
    - Ambil  $N$  sample dari distribusi Bayes Net.
    - Estimasi *posterior* probability dari query event:
    - Berapa kali query event “terjadi” dari  $N$  kali sample? Dibagi  $N$ .
    - $\lim_{N \rightarrow \infty} \hat{P}$  konvergen terhadap  $P$ .
- Contoh sampling ke-1:

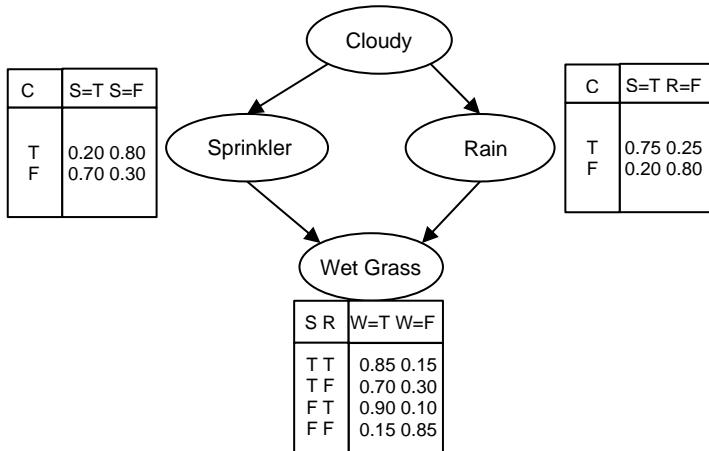


Gambar 11.11 Contoh Sampling 1

- a.  $P(C, S, R, W) = ?$
- b.  $P(C, S, R, \neg W) = ?$

- Contoh sampling ke-2:

1. Tentukan  $P(C=T|S=T)$  !



Gambar 11.12 Contoh Sampling 2

- Penyelesaian:

$$P(C = T|S = T) = \frac{P(C = T, S = T)}{P(S = T)} = \frac{P(S = T|C = T)(C = T)}{P(S = T)} = ?$$

$$P(C = T, S = T) = P(S = T|C = T)P(C = T) = (0.20) * (0.85) = 0.17$$

Atau dapat dihitung dengan cara:

$$\begin{aligned} P(C = T, S = T) &= \sum_{R=0}^1 \sum_{W=0}^1 P(C = T, S = T, W, R) \\ &= \sum_{R=0}^1 \sum_{W=0}^1 P(W|S = T, R)P(S = T|C = T)P(C = T) \end{aligned}$$

Uraian  $P(C = T, S = T)$ : (dengan Probabilitas Marginal)

| R | W | $P(W S = T, R)P(S = T C = T)P(R C = T)P(C = T)$                                                          |
|---|---|----------------------------------------------------------------------------------------------------------|
| 0 | 0 | $P(W = F S = T, R = F)P(S = T C = T)P(R = F C = T)P(C = T)$<br>$= (0.30)*(0.20)*(0.25)*(0.85) = 0.01275$ |
| 0 | 1 | $P(W = T S = T, R = F)P(S = T C = T)P(R = F C = T)P(C = T)$<br>$= (0.70)*(0.20)*(0.25)*(0.85) = 0.02975$ |

|   |   |                                                                                                                        |
|---|---|------------------------------------------------------------------------------------------------------------------------|
| 1 | 0 | $P(W = F S = T, R = T)P(S = T C = T)P(R = T C = T)P(R = T C = T)P(C = T)$<br>$= (0.15)*(0.2)*(0.75)*(0.85) = 0.019125$ |
| 1 | 1 | $P(W = T S = T, R = T)P(S = T C = T)P(R = T C = T)P(C = T)$<br>$= (0.85)*(0.2)*(0.75)*(0.85) = 0.108375$               |

+

$$P(C = T, S = T) = 0.17$$

$$P(S = T) = \sum_{R=0}^1 \sum_{W=0}^1 \sum_{C=0}^1 P(W|S = T, R)P(S = T|C)P(R = C)P(C)$$

| R | W | $P(W S = T, R)P(S = T C = T)P(R C = T)P(C = T)$                                                                        |
|---|---|------------------------------------------------------------------------------------------------------------------------|
| 0 | 0 | $P(W = F S = T, R = F)P(S = T C = T)P(R = F C = T)P(C = T)$<br>$= (0.30)*(0.20)*(0.25)*(0.85) = 0.01275$               |
| 0 | 1 | $P(W = T S = T, R = F)P(S = T C = T)P(R = F C = T)P(C = T)$<br>$= (0.70)*(0.20)*(0.25)*(0.85) = 0.02975$               |
| 1 | 0 | $P(W = F S = T, R = T)P(S = T C = T)P(R = T C = T)P(R = T C = T)P(C = T)$<br>$= (0.15)*(0.2)*(0.75)*(0.85) = 0.019125$ |
| 1 | 1 | $P(W = T S = T, R = T)P(S = T C = T)P(R = T C = T)P(C = T)$<br>$= (0.85)*(0.2)*(0.75)*(0.85) = 0.108375$               |

+

$$P(S = T) = 0.275$$

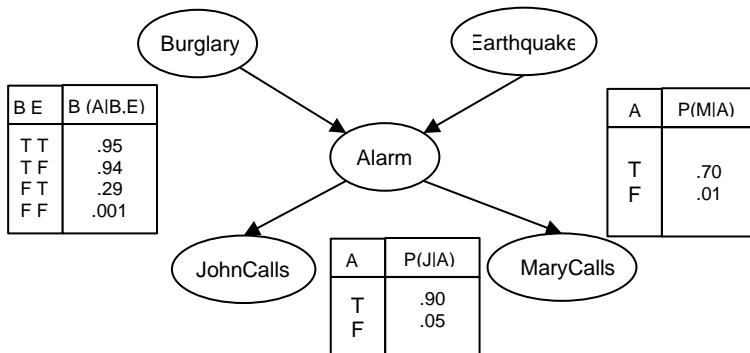
$$P(C = T|S = T) = \frac{P(C = T, S = T)}{P(S = T)}$$

$$= \frac{P(S = T|C = T)P(C = T)}{P(S = T)} = \frac{0.17}{0.275}$$

$$= 0.618182$$

## 11.4 Latihan Individu

1. Perhatikan Bayesian Network di bawah ini.

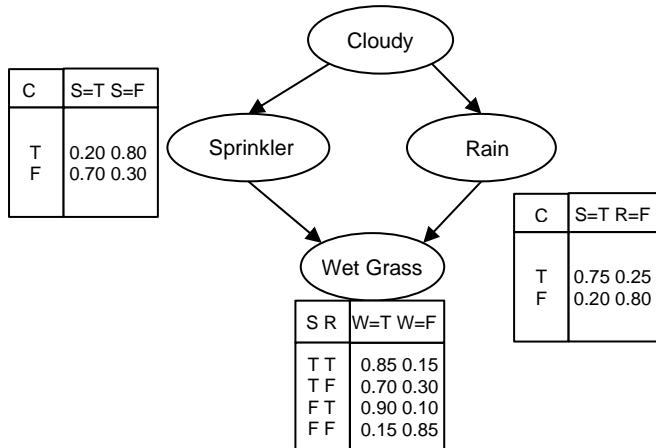


Tentukan nilai Peluang: (Note: c sampai g optional)

- John menelpon, Mary tidak menelpon, alarm nyala, ada perampok, tidak ada gempa bumi.  $P(j \wedge m \wedge a \wedge b \wedge \neg e)$ .
- John menelpon, Mary tidak menelpon, alarm nyala, ada perampok, ada gempa bumi.  $P(j \wedge \neg m \wedge a \wedge b \wedge e)$ .
- $P(B|A)$
- $P(E|A)$
- $P(A=True)$
- $P(J=True)$
- $P(M=True)$

## 11.5 Tugas Kelompok

1. Perhatikan kasus Bayesian Network berikut:



Tentukan nilai peluang di bawah ini:

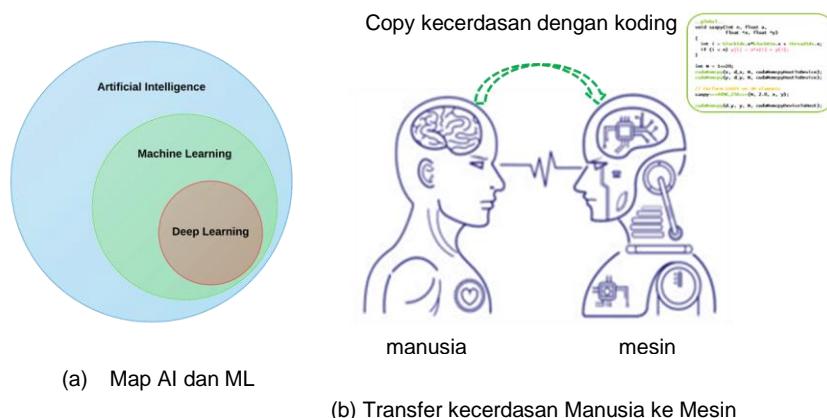
- $P(C=F, S=F, R=T, W=T)$
  - $P(C=S, R=F, W=F)$
  - $P(C=T | R=T)$
  - $P(R=F | C=F)$
2. Buatlah kasus kreatif yang menggunakan Bayesian Network, dan berikan tabel CPT-nya dan contohnya!

## BAB 12 AI & Machine Learning

### 12.1 Review Artificial Intelligence (AI)

Artificial Intelligent (AI) atau Kecerdasan Buatan adalah teknik yang menjadikan komputer dapat berpikir secerdas atau melampaui kecerdasan manusia. Tujuannya agar komputer memiliki kemampuan berperilaku, berpikir, dan mengambil keputusan layaknya manusia. Berikut beberapa pengertian dari AI dari berbagai sumber.

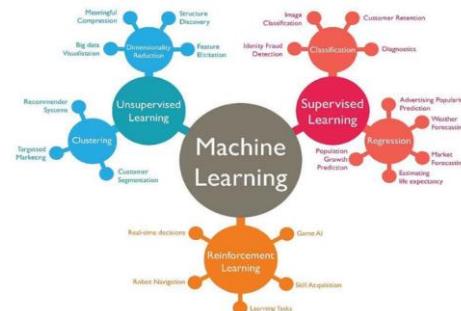
- Otomasi aktivitas yang berhubungan dengan proses berpikir, pemecahan masalah dan pembelajaran (Bellman, 1978).
- Studi tentang kemampuan mengindera dengan menggunakan model komputasi (Charniak & McDermott, 1985).
- Studi bagaimana cara melakukan sesuatu sehingga menjadi lebih baik (Rich & Knight, 1991)
- Cabang dari ilmu komputer yang fokus pada otomasi perilaku yang cerdas (Luger & Stubblefield, 1993).



Gambar 12.1 Ilustrasi Map AI dan ML, serta Transfer AI

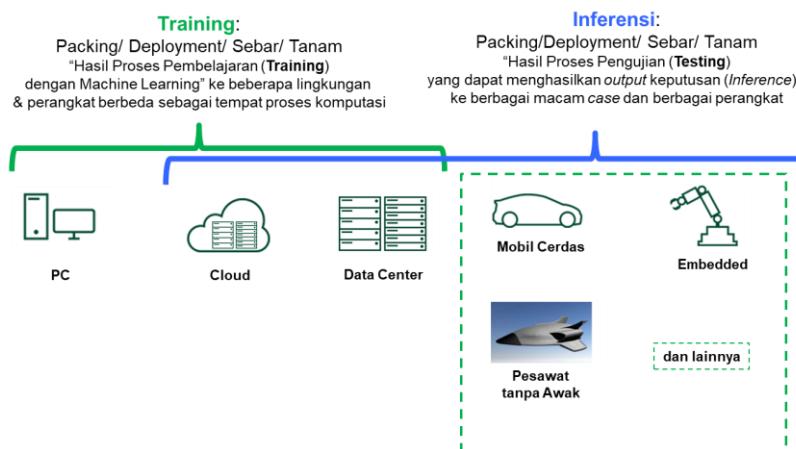
## 12.2 Welcome to Machine Learning

Machine Learning (ML) atau Mesin Pembelajar adalah cabang dari AI yang fokus belajar dari data (learn from data), yaitu fokus pada pengembangan sistem yang mampu belajar secara "mandiri" tanpa harus berulang kali diprogram manusia. ML membutuhkan Data yang valid sebagai bahan belajar (ketika proses training) sebelum digunakan ketika testing untuk hasil output yang optimal.



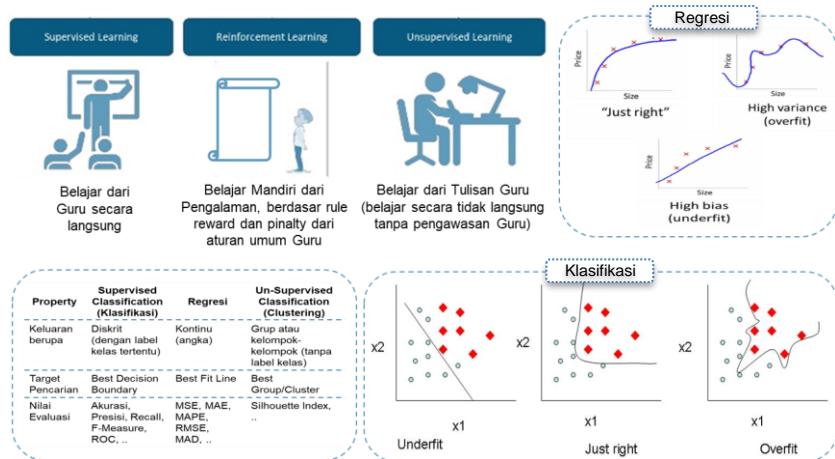
Gambar 12.2 Map cakupan dari Machine Learning

Hasil pengembangan produk bernasis AI (Machine Learning dan Teknik lainnya misal dengan Optimasi, etc) ini harapannya dapat lebih memberikan kemudahan dan langsung dapat diterapkan di masyarakat luas atau bahkan masuk ke industri dalam skala nasional dan internasional. Berikut ilustrasi Map produk untuk apply ke masyarakat.



Gambar 12.3 Map pengembangan produk App

Berikut adalah ilustrasi perbedaan Supervised vs Reinforcement (berprinsip bahwa pengalaman mandiri adalah guru terbaik) vs Un-Supervised Learning. Kemudian diberikan juga untuk beberapa rumus evaluasi yang dapat digunakan masing-masing, serta gambaran Sebagian kecil dari bentuk visualisasinya.



Gambar 12.4 Supervised vs Reinforcement vs vs Un-Supervised Learning

# BAB 13 Support Vector Machine (SVM)

## 13.1 Pengertian SVM

*Support Vector Machine* (SVM) adalah sistem pembelajaran yang menggunakan ruang hipotesis berupa fungsi-fungsi linear dalam sebuah ruang fitur (*feature space*) berdimensi tinggi, dilatih dengan algoritma pembelajaran yang berdasar pada teori optimasi dengan mengimplementasikan *learning bias* yang berasal dari teori pembelajaran statistic (Christianini, Nello dan John S. Taylor, 2000).

SVM merupakan metode klasifikasi *supervised learning* dengan Konsep dasar yang sebenarnya merupakan kombinasi harmonis dari teori-teori komputasi yang telah ada puluhan tahun sebelumnya, seperti margin *hyperplane kernel* diperkenalkan oleh Aronszajn tahun 1950, dan demikian juga dengan konsep-konsep pendukung yang lain. SVM membutuhkan *training set* positif dan negatif. *Training set* positif dan negatif ini dibutuhkan SVM untuk membuat keputusan terbaik dalam memisahkan data positif dengan data negatif di ruang  $n$ -dimensi, yang disebut dengan *hyperplane*. Secara sederhana konsep SVM dapat dijelaskan sebagai proses mencari garis pemisah *hyperplane* dengan mengoptimalkan *hyperplane*, dan memaksimalkan margin antara dua kelas.

SVM selalu mencapai solusi yang sama untuk setiap running. Dengan teknik ini kita berusaha untuk menemukan fungsi pemisah (*classifier*) optimal yang dapat memisahkan dua set data dari dua kelas yang berbeda. teknik ini menarik orang dalam bidang data mining dan machine learning karna kehandalannya dalam memprediksi kelas suatu data baru (Vapnik, 1995). Penggunaan SVM sudah banyak diterapkan dalam permasalahan klasifikasi dan mampu menghasilkan performansi yang baik seperti klasifikasi citra, klasifikasi text, klasifikasi biner dan lain-lain. Seperti Klasifikasi Citra Bibit Unggul Sapi Bali (Cholissodin & Soebroto 2015), *Classification of Campus E-Complaint Documents* (Cholissodin, Kurniawati, Indriati & arwani 2014), identifikasi telapak tangan (Nugroho, Wijanto & Magdalena 2010), klasifikasi motif batik berbasis citra digital (Arfa, Tritoasmro & Atmaja 2012), klasifikasi dokumen berita bahasa Indonesia (Noviani, Suryani & Kurniati

2008) dan lain-lain. Dari hasil yang diperoleh SVM merupakan metode yang cukup handal, sehingga perkembangan potensi kemampuan SVM terus dilakukan baik dengan mengkombinasikan dengan metode lain ataupun metode SVM itu sendiri.

## 13.2 SVM Linear

Pada dasarnya konsep SVM adalah klasifikasi linear namun dapat juga digunakan untuk permasalahan non-linear. Klasifikasi dengan SVM linear biasa diterapkan pada data yang dapat dipisahkan secara linear (*linearly separable data*). Misalkan terdapat data  $x_i = \{x_1, x_2, \dots, x_n\} \in \mathbb{R}^n$  dan  $y_i \in \{-1, +1\}$  dimana  $x_i$  merupakan titik data dan  $y_i$  merupakan kelas data dari titik data  $x_i$ . Hal pertama yang harus dilakukan adalah mencari *hyperplane* atau pemisah antara dua kelas dengan menggunakan fungsi linear yang secara matematis didefinisikan sebagai berikut:

$$f(x) = w \cdot x_i + b \quad (1.1)$$

Dimana  $w$  adalah bobot *support vector* atau *vector* yang tegak lurus dengan *hyperplane* yang dapat didefinisikan sebagai berikut:

$$w = \sum_{i=1}^n \alpha_i y_i x_i \quad (1.2)$$

Dimana:

$x_i$  = data ke-i

$y_i$  = kelas data ke-i

$\alpha_i$  = nilai  $\alpha$  dari data ke-i

Dan  $b$  adalah nilai bias (*threshold*)

$$b = -\frac{1}{2} (w \cdot x^+ + w \cdot x^-) \quad (1.3)$$

$x^+$  adalah data yang merupakan *support vector* pada kelas positif dan  $x^-$  adalah data yang merupakan *support vector* negative dengan nilai alpha paling besar. Dengan fungsi keputusan klasifikasi *sign* ( $f(x)$ ). Fungsi ini digunakan untuk mengklasifikasikan data pada kelas positif atau kelas negatif.

$$f(x) = \sum_{i=1}^m \alpha_i y_i K(x, x_i) + b \quad (1.4)$$

Dimana:  $m$  adalah jumlah *support vector*

$\alpha_i$  adalah nilai bobot tiap titik data

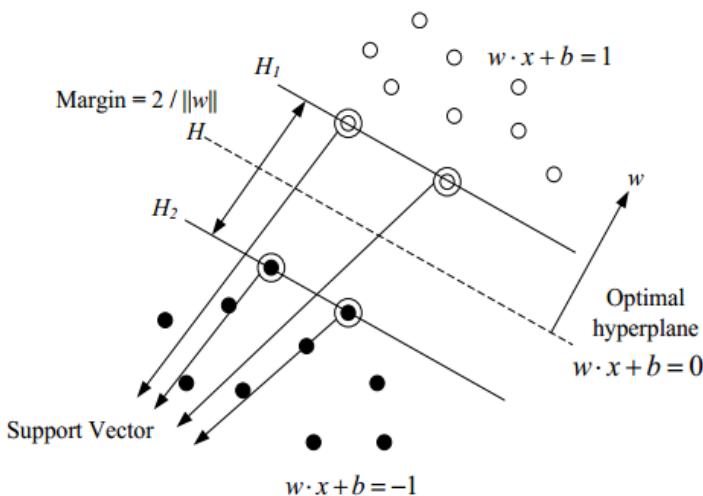
$K(x, xi)$  adalah fungsi *kernel*

Pengklasifikasian Data terhadap kelas positif dan negatif adalah sebagai berikut:

$sign(f(x)) = 1$  untuk kelas positif

$sign(f(x)) = -1$  untuk kelas negatif

Dalam SVM diusahakan untuk mencari garis pemisah *hyperplane* yang optimal dan memaksimalkan margin antara dua kelas. Hyperplane terbaik adalah hyperplane yang terletak ditengah-tengah antara dua set objek dari dua kelas. Ilustrasi Support Vector Machine untuk *linearly separable data* ditunjukkan pada Gambar 13.1.



Gambar 13.1 Ilustrasi SVM untuk *linearly separable data*

Pada Gambar 13.1 terdapat dua set data yang dipisahkan oleh hyperplane optimal dengan garis pembatas  $H_1$  untuk kelas positif dengan persamaan  $w \cdot x + b = 1$  dan  $H_2$  untuk kelas negatif dengan persamaan hyperplane  $w \cdot x + b = -1$ , dengan nilai margin (jarak) antara garis pembatas dapat dihitung dengan  $2 / \|w\|$ . Data yang paling dekat dengan hyperplane atau berada pada garis pembatas disebut dengan support vector. Selanjutnya untuk menentukan hyperplane dari dua kelas maka margin perlu dimaksimalkan dengan menggunakan persamaan berikut:

$$\text{Minimize } J_i[w] = \frac{1}{2} \|w\|^2 \quad (1.5)$$

Dimana  $y_i(x_i \cdot w + b) - 1 \geq 0$

Untuk pencarian bidang pemisah terbaik dengan nilai margin terbesar dapat diubah kedalam permasalahan langrangian dengan menggunakan langrange multiplier. Sehingga permasalahan optimasi constrain pada persamaan 1.5 dapat dirubah menjadi:

$$\min_{w,b} L_p(w,b,a) = \frac{1}{2} |w|^2 - \sum_{i=1}^n \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^n \alpha_i \quad (1.6)$$

Vector  $w$  sering kali bernilai besar , tetapi nilai  $\alpha_i$  terhingga sehingga persamaan 1.6 dirubah kedalam bentuk dual problem ( $L_d$ ) dengan persamaan sebagai berikut:

$$L_d = \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad (1.7)$$

$$\text{syarat : } 0 \leq \alpha_i \leq C \text{ dan } \sum_{i=1}^N \alpha_i y_i = 0$$

Dimana:  $L_d$  adalah Dualitas Lagrange Multiplier

$\alpha_i$  adalah nilai bobot setiap titik data

$x$  adalah titik data  $\{x_1, x_2, \dots, x_n\}$

$y$  adalah kelas data  $\{-1, +1\}$

$C$  adalah konstanta

$N$  adalah banyaknya data

Formula pencarian bidang pemisah terbaik ini Formula pencarian bidang pemisah terbaik ini adalah pemasalahan *quadratic programming*, sehingga nilai maksimum global dari  $\alpha_i$  selalu dapat ditemukan. Setelah solusi pemasalahan *quadratic programming*-ditemukan (nilai  $\alpha_i$ ), maka kelas dari data pengujian  $x$  dapat ditentukan berdasarkan nilai dari fungsi keputusan:

$$f(x_d) = \sum_{i=1}^{ns} \alpha_i y_i x_i \cdot x_d + b \quad (1.8)$$

### 13.3 SVM non-linear

Dalam beberapa kasus ada beberapa data yang tidak dapat dipisahkan secara linear. Sehingga perlu dilakukan penambahan  $\xi_i$

atau variable slack yang berfungsi untuk mengatasi kondisi ketidakmungkinan. Maka secara matematis dapat dirumuskan menjadi:

$$\min_{\frac{1}{2}} \left| |w| \right|^2 + C (\sum_{i=1}^n \xi_i) \quad (1.9)$$

Dengan  $y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0$

$\xi_i \geq 0, i = 1, \dots, n$

Dimana:

$x_i$  = data ke-i

$y_i$  = kelas data ke-i

$b$  = nilai bias

$w$  = bobot support vector

$\xi_i$  = slack variable (mengukur error dari data)

$C$  = parameter user bernilai positif (batasan error)

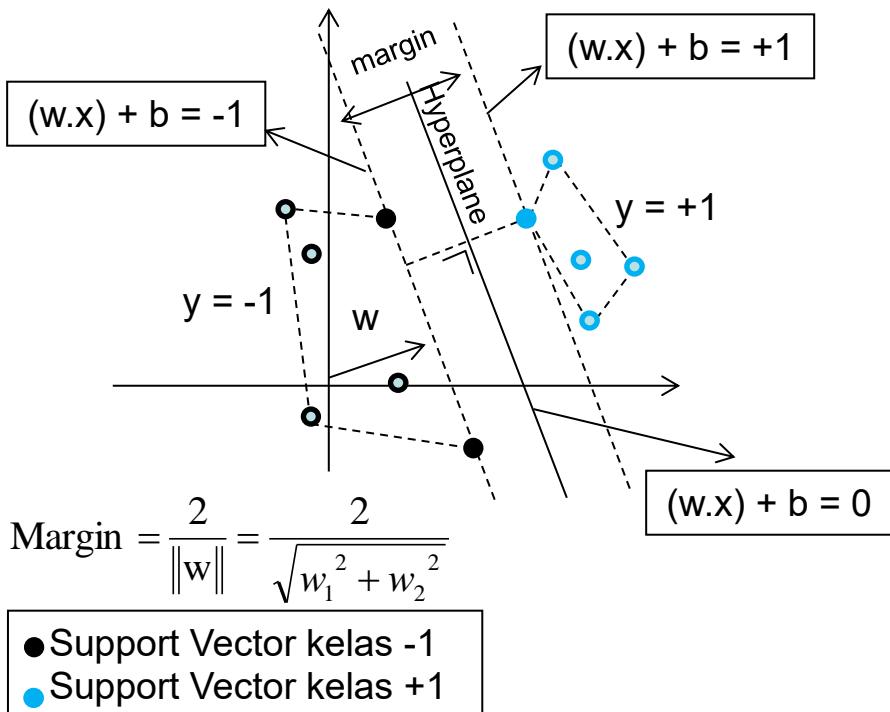
Nilai C atau *complexity* merupakan nilai batasan error yang bernilai positif yang dipilih sebelum dilakukan optimasi dengan proses *quadratic programming*. C memiliki rentang antara 0 sampai bilangan positif tak hingga ( $0 \leq C \leq \infty$ ). Tujuan dari adanya nilai C adalah untuk meminimalkan error dan memperkecil *slack variable*. Jika nilai C mendekati nol, maka lebar margin pada bidang pembatas menjadi maksimum dan pada waktu yang sama banyak jumlah data yang dilatih yang berada dalam margin atau yang ada posisi yang salah tidak akan dipedulikan. Hal ini berarti akan mengurangi tingkat akurasi pada proses training, sehingga mengakibatkan data uji tidak dapat diklasifikasikan dengan baik.

Dengan menggunakan persamaan pada 1.9, maka akan memaksimalkan margin antara dua kelas dengan meminimalkan  $\left| |w| \right|^2$ . Sama halnya pada permasalahan linear persamaan 1.9 dapat dirubah kedalam persamaan langrange sehingga persamaan 1.9 dapat dirubah kedalam bentuk primal problem menjadi.

$$\min_{w,b} L_p(w,b,a) = \frac{1}{2} |w|^2 + C \left( \sum_{i=1}^n \xi_i \right) - \sum_{i=1}^n \alpha_i \{y_i(x_i \cdot w + b) - 1 + \xi_i\} - \sum_{i=1}^n \mu_i \xi_i \quad (1.10)$$

Pengubahan formula primal kedalam dual problem akan menghasilkan persamaan yang sama dengan persamaan 1.7. metode lain untuk dapat memisahkan data non linear adalah dengan mentransformasikan data ke dalam dimensi atau ruang fitur (feature space). Berikut ilustrasi untuk data *linier*, dan *non-linear* yang ditransformasikan

kedalam ruang dimensi yang lebih tinggi yang ditunjukan pada Gambar 13.2 dan 13.3.



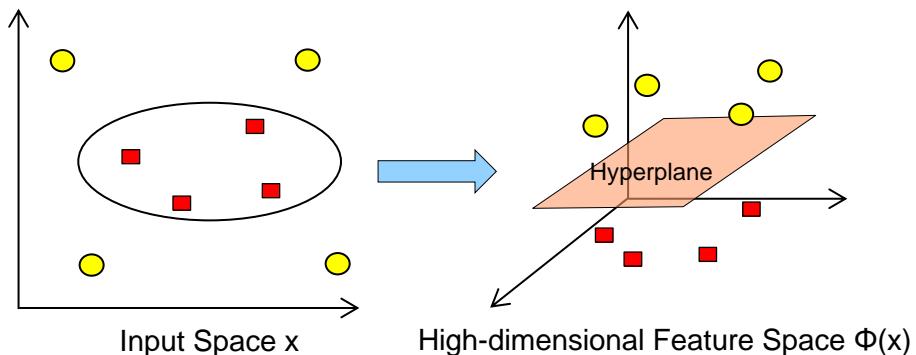
Gambar 13.2 Linear Kernel

Jarak titik ( $x$ ) ke *Hyperplane*:

$$d((w, b), x_i) = \frac{y_i(x_i \bullet w + b)}{\|w\|} \geq \frac{1}{\|w\|} \quad (1.11)$$

Pada Gambar 1.3 dapat dilihat bahwa data (kiri) tidak dapat dipisahkan secara linear. Sehingga dilakukan pemetaan terhadap dimensi input ke ruang dimensi yang lebih tinggi, sehingga membuat data dapat dipisahkan secara linear diruang dimensi baru dan setara dengan *non-linear classifier* pada dimensi asli. Dalam *machine learning*

metode ini disebut dengan *kernel trick*. Dengan *kernel* fungsi pemetaan tidak dapat diketahui secara pasti, karna ruang dimensi tinggi memungkinkan pada ruang dimensi yang tak terhingga.



Gambar 13.3 Non-Linear Kernel

## 13.4 Fungsi Kernel Pada SVM

Beberapa fungsi *kernel* yang digunakan dalam SVM terdiri dari *kernel* linear dan non linear. *Kernel* Linier digunakan ketika data yang akan diklasifikasi dapat terpisah dengan sebuah garis/*hyperplane* atau disebut *linearly separable data*. Sedangkan *Kernel* non-Linier digunakan ketika data hanya dapat dipisahkan dengan garis lengkung atau sebuah bidang pada ruang dimensi tinggi atau disebut *non-linearly sparable data*.

### 13.4.1 Kernel Linear

*Kernel* Linear merupakan salah satu fungsi *kernel* yang paling sederhana. Persamaan yang digunakan untuk *kernel* linear ini adalah:

$$K(x, y) = x^T \cdot y + c \quad (2.1)$$

Misalkan terdapat dataset yang ditunjukkan pada Tabel 13.1.

Tabel 13.1 Dataset

|   | fitur ke-1 | Fitur ke-2 |
|---|------------|------------|
| x | 1          | 1          |
| y | 1          | -1         |

$$x = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ dan } y = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Maka subsitusikan dataset pada tabel 2.1 kedalam persamaan 2.1 dengan  $C = 1$ .

$$K(x, y) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}^T \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} + 1$$

$$K(x, y) = [1 \ 1] \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} + 1$$

$$K(x, y) = ((1 \cdot 1) + (1 \cdot -1)) + 1$$

$$K(x, y) = 1$$

### 13.4.2 Kernel Non-Linear

Beberapa Macam fungsi *kernel* non linear SVM adalah sebagai berikut:

#### 1. Polynomial Kernel

*Kernel Polynomial* merupakan salah satu *kernel non linear*, dimana *kernel* ini sangat cocok untuk menyelesaikan masalah dengan semua data yang dinormalisasi. Dengan parameter konstan  $C$  dan parameter *polynomial degree d* [ROY-10]. Rumus untuk *kernel polynomial* dapat dilihat pada persamaan di bawah ini:

$$K(x, y) = (\alpha x^T y + c)^d \quad (2.2)$$

Dengan menggunakan data yang sama pada Tabel 13.1, subsitusikan kedalam persamaan 2.2 dengan parameter yang digunakan  $\alpha = 0$ ,  $c=1$  dan  $d= 2$ .

$$K(x, y) = (0 \bullet \begin{bmatrix} 1 \\ 1 \end{bmatrix}^T \begin{bmatrix} 1 \\ -1 \end{bmatrix} + 1)^2$$

$$K(x, y) = (0 \bullet [1 \ 1] \begin{bmatrix} 1 \\ -1 \end{bmatrix} + 1)^2$$

$$K(x, y) = (0 \bullet 0 + 1)^2$$

$$K(x, y) = 1$$

#### 2. Gaussian Kernel

*Gaussian RBF kernel* merupakan salah satu contoh dari *radial basis function* kernel. Parameter  $\sigma$  (sigma) memangang peranan penting dalam kinerja kernel. Jika nilai diinisialisasi berlebihan akan menyebabkan fungsi mengalami kekurangan pengaturan dan akan sangat sensitif pada data pelatihan. Fungsi untuk kernel RBF adalah sebagai berikut:

$$K(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) \quad (2.3)$$

Dengan menggunakan data pada tabel 13.1 subsitusikan kedalam persamaan 2.3 dengan parameter yang digunakan  $\sigma = 0.1$ .

$$K(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$$

$$K(x, y) = \exp\left(-\frac{\left\|\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right\|^2}{2(0.1)^2}\right) = \exp\left(-\frac{\left\|\begin{bmatrix} 0 \\ 2 \end{bmatrix}\right\|^2}{0.02}\right)$$

$$K(x, y) = \exp\left(-\frac{(\sqrt{(0)^2 + (2)^2})^2}{0.02}\right) = \exp\left(-\frac{(\sqrt{4})^2}{0.02}\right)$$

$$K(x, y) = \exp\left(-\frac{4}{0.02}\right) = \exp(-200) = 1.3839E-87$$

Selain persamaan 2.3 gaussian kernel juga dapat diimplementasikan dengan persamaan dibawah ini.

$$K(x, y) = \exp(-\gamma \|x - y\|^2) \quad (2.31)$$

Dengan menggunakan data pada Tabel 13.1 subsitusikan kedalam persamaan 2.31 dengan parameter yang digunakan  $\gamma = 0.001$ .

$$K(x, y) = \exp\left(-0.001 \left(\sqrt{(1-1)^2 + (1-(-1))^2}\right)^2\right)$$

$$K(x, y) = \exp(-0.001(4))$$

$$K(x, y) = 0.996007989$$

### 3. Exponential Kernel

*Eksponensial kernel* sangat berkaitan dengan *Gaussian kernel* hanya saja kuadrat norm pada persamaan ini tidak digunakan.

$$K(x, y) = \exp\left(-\frac{\|x-y\|}{2\sigma^2}\right) \quad (2.4)$$

Dengan menggunakan data pada tabel 13.1 dan mensubstitusikan kedalam persamaan 2.4 dengan nilai  $\sigma = 0.1$  maka diperoleh hasil.

$$K(x, y) = \exp\left(-\frac{\left\| \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\|}{2.0.1^2}\right)$$

$$K(x, y) = \exp\left(-\frac{\sqrt{(1-1)^2 + (1-(-1))^2}}{2.0.1^2}\right)$$

$$K(x, y) = \exp(-100)$$

$$K(x, y) = 3.72008E - 44$$

#### 4. Laplacian Kernel

Sama dengan *exponential kernel*, *laplacian kernel* menggunakan persamaan yang sama hanya saja kuadrat sigma dihilangkan pada persamaan *laplacian kernel*. Persamaan yang digunakan adalah:

$$K(x, y) = \exp\left(-\frac{|x-y|}{\sigma}\right) \quad (2.5)$$

Dengan mensubsitusikan data pada tabel 13.1 kedalam persamaan 2.5 maka diperoleh hasil.

$$K(x, y) = \exp\left(-\frac{\left\| \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\|}{0.1}\right)$$

$$K(x, y) = \exp\left(-\frac{\sqrt{(1-1)^2 + (1-(-1))^2}}{0.1}\right)$$

$$K(x, y) = \exp(-20)$$

$$K(x, y) = 2.06115E - 09$$

#### 5. ANOVA Kernel

Sama halnya seperti *Gaussian* dan *laplacian kernel*, ANOVA juga bagian dari *radial basis function kernel*. Persamaan yang digunakan pada ANOVA kernel adalah:

$$K(x, y) = \sum_{k=1}^n \exp(-\sigma(x^k - y^k)^2)^d \quad (2.6)$$

Dengan mensubsitusikan data pada tabel 13.1 kedalam persamaan 2.6 dengan nilai  $\sigma = 0.1$ ,  $k = 2$ ,  $d = 2$  maka diperoleh hasil.

---

$$\begin{aligned}K(x, y) &= \exp(-0.1(1^2 - 1^2)^2)^2 + \exp(-0.1(1^2 - (-1)^2)^2)^2 \\K(x, y) &= \exp(-0.1(1^2 - 1^2)^2)^2 + \exp(-0.1(1^2 - (-1)^2)^2)^2 \\K(x, y) &= \exp(0)^2 + \exp(0)^2 \\K(x, y) &= 2\end{aligned}$$

#### 6. Hyperbolic Tangent (Sigmoid) Kernel

*Hyperbolic Tangent* juga merupakan salah satu dari *non linear kernel*. Rumus untuk *Hyperbolic Tangent* dapat dilihat pada persamaan di bawah ini:

$$K(x, y) = \tanh(ax^T \cdot y + c) \quad (2.7)$$

Dengan mensubsitusikan data pada tabel 13.1 kedalam persamaan 2.7 dengan nilai  $a = 0$ ,  $c = 1$  maka diperoleh hasil.

$$K(x, y) = \tanh(0([1 \ 1] \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix}) + 1)$$

$$K(x, y) = \tanh(0([1 \ 1] \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix}) + 1)$$

$$K(x, y) = \tanh(1)$$

$$(x, y) = 0.761594156$$

#### 7. Rational Quadratic Kernel

Persamaan yang digunakan untuk *Rational Quadratic Kernel* adalah:

$$K(x, y) = 1 - \frac{\|x-y\|^2}{\|x-y\|^2 + c} \quad (2.8)$$

Dengan mensubsitusikan data pada tabel 13.1 kedalam persamaan 2.8 dengan nilai  $c = 1$  maka diperoleh hasil.

$$K(x, y) = 1 - \frac{\sqrt{(1-1)^2 + (1-(-1))^2}}{\sqrt{(1-1)^2 + (1-(-1))^2} + 1}^2$$

$$K(x, y) = 1 - \frac{4}{5}$$

$$K(x, y) = 0.761594156$$

#### 8. Multi quadric Kernel

Persamaan yang digunakan untuk *Multi Quadratic Kernel* adalah:

$$K(x, y) = \sqrt{\|x - y\|^2 + c^2} \quad (2.9)$$

Dengan mensubsitusikan data pada tabel 13.1 kedalam persamaan 2.9 dengan nilai  $c = 1$  maka diperoleh hasil.

$$K(x, y) = \sqrt{\sqrt{(1 - 1)^2 + (1 - (-1))^2} + 1^2}$$

$$K(x, y) = \sqrt{5}$$

$$K(x, y) = 2.236067977$$

#### 9. Invers Multi quadric Kernel

Persamaan yang digunakan untuk *Invers Multi Quadratic Kernel* adalah:

$$K(x, y) = \frac{1}{\sqrt{\|x-y\|^2 + c^2}} \quad (2.10)$$

Dengan mensubsitusikan data pada tabel 13.1 kedalam persamaan 2.10 dengan nilai  $c = 1$  maka diperoleh hasil.

$$K(x, y) = \frac{1}{\sqrt{5}}$$

$$K(x, y) = 0.447213595$$

#### 10. Circular Kernel

Persamaan yang digunakan untuk circular Kernel adalah:

$$K(x, y) = \frac{2}{\pi} \arccos\left(-\frac{\|x-y\|}{\sigma}\right) - \frac{2}{\pi} \frac{\|x-y\|}{\sigma} \sqrt{1 - \left(\frac{\|x-y\|}{\sigma}\right)^2} \quad (2.11)$$

Dengan mensubsitusikan data pada tabel 13.1 kedalam persamaan 2.11 dengan nilai  $\sigma = 0.1$  maka diperoleh hasil.

$$K(x, y) = \frac{2}{\pi} \arccos\left(-\frac{\sqrt{4}}{0.1}\right) - \frac{2}{\pi} \frac{\sqrt{4}}{0.1} \sqrt{1 - \left(\frac{(\sqrt{4})^2}{0.1}\right)}$$

$$K(x, y) = \frac{2}{3.14} \arccos\left(-\frac{2}{0.1}\right) - \frac{2}{3.14} \frac{2}{0.1} \sqrt{1 - \left(\frac{(\sqrt{4})^2}{0.1}\right)}$$

$$K(x, y) = (0.64) \arccos(-20) - \frac{2}{3.14} \frac{\sqrt{4}}{0.1} \sqrt{1 - \left(\frac{(\sqrt{4})^2}{0.1}\right)}$$

### 11. Spherical Kernel

Persamaan yang digunakan untuk *Spherical Kernel* adalah:

$$K(x, y) = 1 - \frac{3}{2} \frac{\|x-y\|}{\sigma} + \frac{1}{2} \left( \frac{\|x-y\|}{\sigma} \right)^3 \quad (2.12)$$

Dengan mensubsitusikan data pada tabel 13.1 kedalam persamaan 2.12 dengan nilai  $\sigma = 0.1$  maka diperoleh hasil.

$$K(x, y) = 1 - \frac{3}{2} \frac{\sqrt{(1-1)^2 + (1-(-1))^2}}{0.1} + \frac{1}{2} \left( \frac{\sqrt{(1-1)^2 + (1-(-1))^2}}{0.1} \right)^3$$

$$K(x, y) = 1 - \frac{3}{2} (20) + \frac{1}{2} (20)^3$$

$$K(x, y) = 1 - 30 + 4000$$

$$K(x, y) = 3971$$

### 12. Power Kernel

Persamaan yang digunakan untuk *power Kernel* adalah:

$$K(x, y) = -\|x-y\|^d \quad (2.14)$$

Dengan mensubsitusikan data pada tabel 13.1 kedalam persamaan 2.14 dengan nilai  $d = 2$  maka diperoleh hasil.

$$K(x, y) = -\sqrt{(1-1)^2 + (1-(-1))^2}^2$$

$$K(x, y) = -\sqrt{4}^2$$

$$K(x, y) = -4$$

### 13. Log Kernel

Persamaan yang digunakan untuk *Log Kernel* adalah:

$$K(x, y) = -\log(\|x-y\|^d + 1) \quad (2.15)$$

Dengan mensubsitusikan data pada tabel 13.1 kedalam persamaan 2.15 dengan nilai  $d = 2$  maka diperoleh hasil.

$$K(x, y) = -\log \left( \sqrt{(1-1)^2 + (1-(-1))^2} + 1 \right)$$

$$K(x, y) = -\log 5$$

$$K(x, y) = -0.698970004$$

#### 14. Cauchy Kernel

Persamaan yang digunakan untuk *Cauchy Kernel* adalah:

$$K(x, y) = \frac{1}{1 + \frac{\|x-y\|}{\sigma}} \quad (2.19)$$

Dengan mensubsitusikan data pada tabel 13.1 kedalam persamaan 2.19 dengan nilai  $\sigma = 0.1$  maka diperoleh hasil.

$$K(x, y) = \frac{1}{1 + \frac{\sqrt{(1-1)^2 + (1-(-1))^2}}{0.1}}$$

$$K(x, y) = \frac{1}{1 + \frac{4}{0.1}}$$

$$K(x, y) = \frac{1}{41}$$

$$K(x, y) = 0.024390244$$

#### 15. Chi-Square Kernel

Persamaan yang digunakan untuk *Chi-square Kernel* adalah:

$$K(x, y) = 1 - \sum_{i=1}^n \frac{(x_i - y_i)^2}{2(x_i + y_i)} \quad (2.20)$$

Misalkan terdapat dataset seperti pada Tabel 13.2.

Tabel 13.2 Dataset

|   |   |    |
|---|---|----|
| x | 3 | 1  |
| y | 1 | -2 |

Dengan mensubsitusikan data pada tabel 13.2 kedalam persamaan 2.20 maka diperoleh hasil.

$$K(x, y) = 1 - \left( \frac{(3-1)^2}{\frac{1}{2}(3+1)} + \frac{(1-(-2))^2}{\frac{1}{2}(1+(-2))} \right)$$

$$K(x, y) = 1 - \left( \frac{4}{2} + \frac{9}{-0.5} \right)$$

$$K(x, y) = 1 - (-16)$$

$$K(x, y) = 17$$

#### 16. Histogram Intersection Kernel

Persamaan yang digunakan untuk *Histogram Intersection Kernel* adalah:

$$K(x, y) = \sum_{i=1}^m \min(x_i, y_i) \quad (2.21)$$

Dengan mensubsitusikan data pada tabel 13.2 kedalam persamaan 2.21 maka diperoleh hasil.

$$K(x, y) = \min(3, 1) + \min(1, -2)$$

$$K(x, y) = 1 + (-2)$$

$$K(x, y) = -1$$

#### 17. Generalized Histogram Intersection

Persamaan yang digunakan untuk *Generalized Histogram Kernel* adalah:

$$K(x, y) = \sum_{i=1}^m \min(|x_i|^\alpha, |y_i|^\beta) \quad (2.22)$$

Dengan mensubsitusikan data pada tabel 13.2 kedalam persamaan 2.22 dimana  $\alpha = 2$ ,  $\beta = 2$  maka diperoleh hasil.

$$K(x, y) = \min(|3|^2, 1^2) + \min(|1|^2, |-2|^2)$$

$$K(x, y) = \min(9, 1) + \min(1, 4)$$

$$K(x, y) = 1 + 1$$

$$K(x, y) = 2$$

#### 18. Generalized T-Student Kernel

Persamaan yang digunakan untuk *Generalized T-Student Kernel* adalah:

$$K(x, y) = \frac{1}{1+|x-y|^d} \quad (2.23)$$

Dengan mensubsitusikan data pada tabel 13.2 kedalam persamaan 2.23 dimana  $d = 2$  maka diperoleh hasil.

$$\begin{aligned} K(x, y) &= \frac{1}{1+\sqrt{(1-1)^2+(1-(-1))^2}} \\ K(x, y) &= \frac{1}{5} \\ K(x, y) &= 0.2 \end{aligned}$$

### 19. Additive Kernel

*Additive Kernel SVM* merupakan salah satu fungsi *kernel* pada *non linear SVM*. Salah satu kelebihan yang dimiliki *Additive Kernel SVM* adalah tingkat akurasi yang tinggi dibandingkan dengan *kernel* lainnya. Bila dibandingkan antara *Linear* dan *Non Linear SVM*, *Linear SVM* lebih cepat dalam proses pelatihan dan *testing* dibandingkan *Non Linear SVM*. Namun *Non Linear SVM* memiliki keunggulan dimana tingkat akurasi yang diberikan lebih tinggi. Berikut ini tabel perbandingan antara *Linear SVM* dan *Non Linear SVM*.

**Tabel 2.1** Perbandingan *Linear* dan *Non Linear SVM*

|                     | Linear       | Non-Linear    |
|---------------------|--------------|---------------|
| Kecepatan Pelatihan | Sangat cepat | Sangat lambat |
| Kecepatan Pengujian | Sangat cepat | Sangat lambat |
| Tingkat Akurasi     | Rendah       | Tinggi        |

Dengan menggunakan *Additive Kernel SVM* dapat memberikan tingkat akurasi yang tinggi dan proses pelatihan dan pengujian menjadi lebih cepat. Karena *Additive Kernel SVM* merupakan *non linear SVM*, maka menggunakan fungsi *kernel*  $K(x, x_i)$ . Persamaan *hyperplane* yang optimal, didapatkan dengan menggunakan persamaan berikut:

Dan persamaan (2-1) berubah menjadi:

$$h(x) = \sum_{i=1}^n h_i(x_i) + b \quad (2.24)$$

Dimana:

$$h_i(x_i) = \sum_{l=1}^m \alpha_l y_l K_i(x_i, x_{l,i})$$

Dengan  $x_{l,i}$  merupakan data uji ke  $l$  pada *kernel* ke- $i$ . Dalam kasus *non linear* SVM vektor  $w$  dapat dihitung dengan rumus  $w = \sum_{i=1}^n \alpha_i y_i \phi(x_i)$  dan bias  $b$  dengan rumus  $b = \frac{1}{2} (w \bullet \phi(x^+) + w \bullet \phi(x^-))$ . Dimana  $K(x, y) = \sum_{i=1}^n K_i(K_i(x_i, y_i))$  dan  $m$  adalah banyaknya *support vector*. Namun dengan *additive kernel* nilai  $w$  tidak perlu dicari terlebih dahulu, tetapi dengan memodifikasi nilai  $b$  seperti persamaan di bawah ini:

$$\begin{aligned} b &= -\frac{1}{2} \left[ \left( \sum_{i=1}^m \alpha_i y_i x_i \bullet x^+ \right) + \left( \sum_{i=1}^m \alpha_i y_i x_i \bullet x^- \right) \right] \\ b &= -\frac{1}{2} \left[ \left( \sum_{i=1}^m \alpha_i y_i K(x_i, x^+) \right) + \left( \sum_{i=1}^m \alpha_i y_i K(x_i, x^-) \right) \right] \end{aligned} \quad (2.25)$$

Salah satu *kernel* yang termasuk *Additive Kernel* yaitu *Intersection Kernel*. Perbedaan dengan *Additive Kernel* yaitu pada fungsi  $h(x)$  yang digunakan dan juga fungsi *kernel* itu sendiri. Pada

*Additive Kernel* dilakukan dengan menggabungkan beberapa fungsi *kernel*, berbeda dengan *Intersection Kernel* yang mencari nilai minimum dimensi data dan menggabungkannya. Berikut adalah fungsi *Intersection Kernel*:

$$\begin{aligned} h(s) &= \sum_{i=1}^n h_i(s_i) + b \\ h_i(s_i) &= \sum_{l=1}^m \alpha_l y_l \min(s_i, x_{l,i}) + b \\ h_i(s_i) &= \sum_{x_{l,i} < s_i} \alpha_l y_l x_{l,i} + s_i \sum_{x_{l,i} \geq s_i} \alpha_l y_l \end{aligned} \quad (2-26)$$

Dengan  $s_i$  merupakan dimensi data uji ke  $i$  dan  $m$  adalah banyaknya data *Support Vector*. Dimana  $K \min(x, y) = \sum_{i=1}^n \min(x_i, y_i)$  dan  $m$  adalah banyaknya *support vector*. Dengan *Intersection Kernel* nilai  $w$  tidak perlu dicari terlebih dahulu, tetapi dengan memodifikasi nilai  $b$  seperti persamaan di bawah ini:

$$\begin{aligned} b &= -\frac{1}{2} \left[ \left( \sum_{i=1}^m \alpha_i y_i K \min(x_i, x^+) \right) + \left( \sum_{i=1}^m \alpha_i y_i K \min(x_i, x^-) \right) \right] \\ b &= -\frac{1}{2} \left[ \left( \sum_{i=1}^m \alpha_i y_i \sum_{i=1}^n \min(x_i, x^+) \right) + \left( \sum_{i=1}^m \alpha_i y_i \sum_{i=1}^n \min(x_i, x^-) \right) \right] \end{aligned} \quad (2-27)$$

Misal terdapat 2 buah data,  $x = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$  dan  $y = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix}$

$$\begin{aligned} K \min(x, y) &= \sum_{i=1}^n \min(x_i, y_i) = \sum_{i=1}^2 \min(x_i, y_i) \\ &= \min(x_1, y_1) + \min(x_2, y_2) \\ &= \min(5, 7) + \min(6, 4) \\ &= 5 + 4 \\ &= 9 \end{aligned}$$

## 13.5 Dasar-Dasar SVM

### 13.5.1 SVM 2 Kelas (Hyperplane Linier)

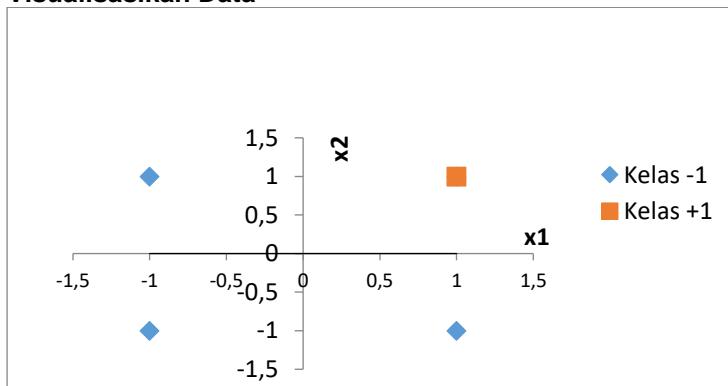
Berikut merupakan contoh perhitungan dasar untuk memahami SVM yang menerapkan kernel linear, dengan asumsi sudah diketahui titik-titik data yang merupakan support vector.

#### 1. Misalkan diketahui dataset sebagai berikut

Tabel 13.3 Dataset perhitungan Kernel Linear

| X1 | X2 | Kelas (y) | Support Vector |
|----|----|-----------|----------------|
| 1  | 1  | 1         | 1              |
| 1  | -1 | -1        | 1              |
| -1 | 1  | -1        | 1              |
| -1 | -1 | -1        | 0              |

#### 2. Visualisasikan Data



Gambar 13.4 Visualisasi Dataset pada kernel Linear

#### 3. Mencari nilai parameter w dan b

- Karena ada dua fitur ( $x_1$  dan  $x_2$ ), maka  $w$  juga akan memiliki dua fitur ( $w_1$  dan  $w_2$ ). Formulasi yang digunakan adalah sebagai berikut:

$$\frac{1}{2} \|w\|^2 = \frac{1}{2} (w_1^2 + w_2^2)$$

Dengan syarat:

$$\begin{aligned}y_i(w \cdot x_i + b) &\geq 1, \quad i = 1, 2, 3, \dots, N \\y_i(w_1 \cdot x_1 + w_2 \cdot x_2 + b) &\geq 1,\end{aligned}$$

Sehingga didapatkan persamaan sebagai berikut:

1.  $(w_1 + w_2 + b) \geq 1$ , untuk  $y_1 = 1, x_1 = 1, x_2 = 1$
2.  $(-w_1 + w_2 - b) \geq 1$ , untuk  $y_2 = -1, x_1 = 1, x_2 = -1$
3.  $(w_1 - w_2 - b) \geq 1$ , untuk  $y_3 = -1, x_1 = -1, x_2 = 1$
4.  $(w_1 + w_2 - b) \geq 1$ , untuk  $y_4 = -1, x_1 = -1, x_2 = -1$

- Menjumlahkan Persamaan 1 dan 2

$$\begin{array}{r} (w_1 + w_2 + b) \geq 1 \\ (-w_1 + w_2 - b) \geq 1 \\ \hline \end{array} +$$

$$2w_2 = 2$$

Maka  $w_2 = 1$

- Menjumlahkan Persamaan 1 dan 3

$$\begin{array}{r} (w_1 + w_2 + b) \geq 1 \\ (w_1 - w_2 - b) \geq 1 \\ \hline \end{array} +$$

$$2w_1 = 2$$

Maka  $w_1 = 1$

- Menjumlahkan Persamaan 2 dan 3

$$\begin{array}{r} (-w_1 + w_2 - b) \geq 1 \\ (w_1 - w_2 - b) \geq 1 \\ \hline \end{array} +$$

$$-2b = 2$$

Maka  $b = -1$

- Sehingga diperoleh Persamaan *Hyperplane*

$$w_1x_1 + w_2x_2 + b = 0$$

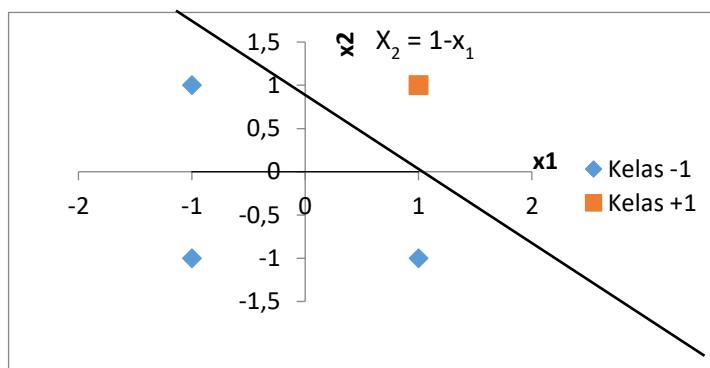
$$x_1 + x_2 - 1 = 0$$

$$x_2 = 1 - x_1$$

#### 4. Visualisasikan Garis Hyperplane (Sebagai Fungsi Klasifikasi)

Tabel 13.4 Data

| X1 | X2 = 2-x1 |
|----|-----------|
| -2 | 3         |
| -1 | 2         |
| 0  | 1         |
| 1  | 0         |
| 2  | -1        |



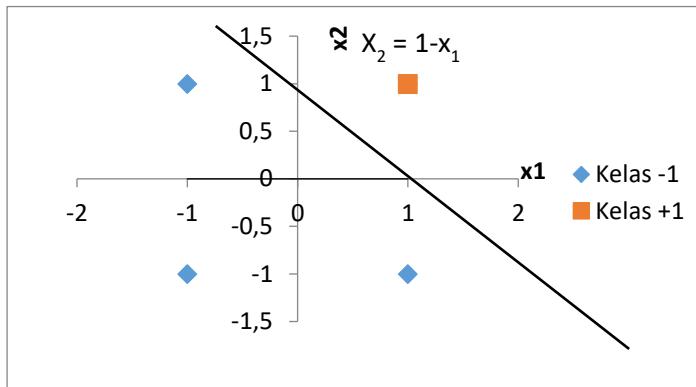
Gambar 13.5 Visualisasi Data

#### 5. Mengklasifikasikan Data Uji atau Data Testing

Tabel 13.5 Data Uji

| No | Data Uji |       | Hasil Klasifikasi<br>Kelas = sign( $x_1 + x_2 - 1$ ) |
|----|----------|-------|------------------------------------------------------|
|    | $x_1$    | $x_2$ |                                                      |
| 1  | 1        | 5     | sign (1 + 5 - 1) = +1                                |
| 2  | -1       | 4     | sign (-1 + 4 - 1) = +1                               |
| 3  | 0        | 7     | sign (0 + 7 - 1) = +1                                |

|   |    |    |                        |
|---|----|----|------------------------|
| 4 | -9 | 0  | sign (-9 + 0 - 1) = -1 |
| 5 | 2  | -2 | sign (2 - 2 - 1) = -1  |



Gambar 13.6 Visualisasi Data Uji

### 13.5.2 SVM 2 Kelas (Hyperplane Non-Linier)

Kernel Polynomial merupakan kernel Non-Linear merupakan salah satu fungsi kernel yang paling sederhana dan sangat representatif untuk memahami konsep dasar alur mapping data pada dimensi tinggi pada contoh kasus yang sederhana. Persamaan yang digunakan untuk kernel polynomial ini adalah:

$$K(x, y) = (x^T \cdot y + c)^d, \text{ dengan } c=1 \text{ dan } d=2$$

Berikut merupakan contoh perhitungan SVM yang menerapkan kernel polynomial, dengan asumsi sudah diketahui titik-titik data yang merupakan *support vector*, beserta nilai alphanya.

#### 1. Misalkan diketahui dataset sebagai berikut

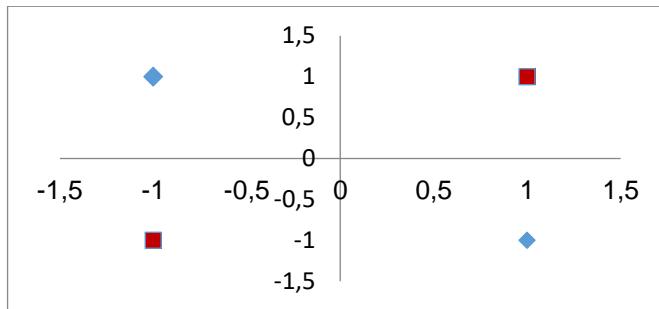
Tabel 13.6 Dataset Kernel Polynomial

| x <sub>1</sub> | x <sub>2</sub> | Kelas (y) | Support Vector |
|----------------|----------------|-----------|----------------|
| 1              | 1              | -1        | 1              |

|    |    |    |   |
|----|----|----|---|
| 1  | -1 | 1  | 1 |
| -1 | 1  | 1  | 1 |
| -1 | -1 | -1 | 1 |

## 2. Visualisasikan Data

Berdasarkan dataset pada Tabel 13.6 maka dapat digambarkan bentuk visualisasi data sebagai berikut:



Gambar 13.7 Visualisasi Datset

Dari bentuk visualisasi data terlihat bahwa data tidak dapat dipisahkan secara linear, sehingga digunakan kernel non linear. Pada contoh perhitungan ini digunakan kernel polynomial ordo 2. Sehingga persamaan yang digunakan adalah:

$$K(x, y) = (x^T \cdot y + c)^d, \text{ dengan } c=1 \text{ dan } d=2$$

## 3. Menghitung Matrik kernel

Setelah diketahui persamaan yang digunakan maka akan dilakukan perhitungan Matrik Kernel  $K(x, x_i) = \Phi(x) \cdot \Phi(x_i)$ . Berikut contoh perhitungan untuk seluruh dataset.

Tabel 13.7 Perhitungan  $K(x, x_i)$

| Perhitungan $K(x, x_i)$ |       |                                                                               |
|-------------------------|-------|-------------------------------------------------------------------------------|
| $x_1$                   | $x_1$ | $K(1,1) = (x_1 \cdot x_1 + 1)^2 = (1 \cdot 1 + 1 \cdot 1 + 1)^2 = 3^2 = 9$    |
|                         | $x_2$ | $K(1,2) = (x_1 \cdot x_2 + 1)^2 = (1 \cdot 1 + 1 \cdot (-1) + 1)^2 = 1^2 = 1$ |

|       |       |                                                                                        |
|-------|-------|----------------------------------------------------------------------------------------|
|       | $x_3$ | $K(1,3) = (x_1 \cdot x_3 + 1)^2 = (1 \cdot (-1) + 1 \cdot 1 + 1)^2 = 1^2 = 1$          |
|       | $x_4$ | $K(1,4) = (x_1 \cdot x_4 + 1)^2 = (1 \cdot (-1) + 1 \cdot (-1) + 1)^2 = (-1)^2 = 1$    |
| $x_2$ | $x_1$ | $K(2,1) = (x_2 \cdot x_1 + 1)^2 = (1 \cdot 1 + (-1) \cdot 1 + 1)^2 = 1^2 = 1$          |
|       | $x_2$ | $K(2,2) = (x_2 \cdot x_2 + 1)^2 = (1 \cdot 1 + (-1) \cdot (-1) + 1)^2 = 3^2 = 9$       |
|       | $x_3$ | $K(2,3) = (x_2 \cdot x_3 + 1)^2 = (1 \cdot (-1) + (-1) \cdot 1 + 1)^2 = 1^2 = 1$       |
|       | $x_4$ | $K(2,4) = (x_2 \cdot x_4 + 1)^2 = (1 \cdot (-1) + (-1) \cdot (-1) + 1)^2 = 1^2 = 1$    |
| $x_3$ | $x_1$ | $K(3,1) = (x_3 \cdot x_1 + 1)^2 = ((-1) \cdot 1 + 1 \cdot 1 + 1)^2 = 1^2 = 1$          |
|       | $x_2$ | $K(3,2) = (x_3 \cdot x_2 + 1)^2 = ((-1) \cdot 1 + 1 \cdot (-1) + 1)^2 = 1^2 = 1$       |
|       | $x_3$ | $K(3,3) = (x_3 \cdot x_3 + 1)^2 = ((-1) \cdot (-1) + 1 \cdot 1 + 1)^2 = 3^2 = 9$       |
|       | $x_4$ | $K(3,4) = (x_3 \cdot x_4 + 1)^2 = ((-1) \cdot (-1) + 1 \cdot (-1) + 1)^2 = 1^2 = 1$    |
| $x_4$ | $x_1$ | $K(4,1) = (x_4 \cdot x_1 + 1)^2 = ((-1) \cdot 1 + (-1) \cdot 1 + 1)^2 = 1^2 = 1$       |
|       | $x_2$ | $K(4,2) = (x_4 \cdot x_2 + 1)^2 = ((-1) \cdot 1 + (-1) \cdot (-1) + 1)^2 = 1^2 = 1$    |
|       | $x_3$ | $K(4,3) = (x_4 \cdot x_3 + 1)^2 = ((-1) \cdot (-1) + (-1) \cdot 1 + 1)^2 = 1^2 = 1$    |
|       | $x_4$ | $K(4,4) = (x_4 \cdot x_4 + 1)^2 = ((-1) \cdot (-1) + (-1) \cdot (-1) + 1)^2 = 3^2 = 9$ |

Dari perhitungan Matrik Kernel diatas diperoleh Matrik Kernel dengan ukuran NxN.

Tabel 13.8 Matrik NxN

|   |   |   |   |
|---|---|---|---|
| 9 | 1 | 1 | 1 |
| 1 | 9 | 1 | 1 |
| 1 | 1 | 9 | 1 |
| 1 | 1 | 1 | 9 |

Setiap elemen matrik  $kernel K(x, x_i)$  digunakan untuk menggantikan dot-product  $x_i \cdot x_j$  dalam persamaan dualitas *lagrange multiplier*.

#### 4. Memaksimalkan Ld (*Dualitas Lagrange Multiplier*)

Setelah diperoleh matrik *kernel* dengan ukuran  $N \times N$  maka langkah selanjutnya adalah memaksimalkan nilai *Ld* dengan menggunakan persamaan:

$$Ld = \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(x_i, x_j) \text{ syarat : } 0 \leq \alpha_i \leq C \text{ dan } \sum_{i=1}^N \alpha_i y_i = 0$$

Dengan menggunakan persamaan diatas maka diperoleh

$$Ld = (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4) - \frac{1}{2} \begin{pmatrix} 9\alpha_1\alpha_1 - 1\alpha_1\alpha_2 - 1\alpha_1\alpha_3 + 1\alpha_1\alpha_4 - 1\alpha_2\alpha_1 + 9\alpha_2\alpha_2 + 1\alpha_2\alpha_3 \\ -1\alpha_2\alpha_4 - 1\alpha_3\alpha_1 + 1\alpha_3\alpha_2 + 9\alpha_3\alpha_3 - 1\alpha_3\alpha_4 + 1\alpha_4\alpha_1 - 1\alpha_4\alpha_2 \\ -\alpha_4\alpha_3 + 9\alpha_4\alpha_4 \end{pmatrix}$$

Syarat 1:  $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \geq 0$

Syarat 2:  $-\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 0$

Misalkan didapatkan nilai  $\max Ld$   $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.125$  maka didapatkan nilai *Ld* sebagai Berikut:

$$Ld = (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4) - \frac{1}{2} \begin{pmatrix} 9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 + 9\alpha_2^2 + 2\alpha_2\alpha_3 \\ -2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2 \end{pmatrix}$$

$$\begin{aligned} Ld &= (0.125 + 0.125 + 0.125 + 0.125) - \frac{1}{2} (9(0.125)^2 - 2(0.125)(0.125) - \\ &\quad 2(0.125)(0.125) + 2(0.125)(0.125) + 9(0.125)^2 + 2(0.125)(0.125) - 2(0.125) \\ &\quad (0.125) + 9(0.125)^2 - 2(0.125)(0.125) + 9(0.125)^2 \end{aligned}$$

$$Ld = 0.5 - 0.5/2$$

$$= 0.25$$

#### 5. Menghitung Nilai *w* dan *b*

$$X_i = \begin{pmatrix} x_1^i \\ x_2^i \end{pmatrix}, \text{ jika } X_1 \text{ adalah data ke-1,}$$

$$X_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \phi(X_i) = \begin{pmatrix} x_1^{i^2} & \sqrt{2}x_1^ix_2^i & x_2^{i^2} & \sqrt{2}x_1^i & \sqrt{2}x_2^i & 1 \end{pmatrix}^T$$

Dimana  $x_1^i$  adalah nilai pada dimensi ke-1 pada data ke-i.

$$w = \sum_{i=1}^N \alpha_i y_i \phi(X_i)$$

$$w = \sum_{i=1}^N \alpha_i y_i \phi(X_i) = \sum_{i=1}^4 \alpha_i y_i \phi(X_i) = \alpha_1 y_1 \phi(X_1) + \alpha_2 y_2 \phi(X_2) + \alpha_3 y_3 \phi(X_3) + \alpha_4 y_4 \phi(X_4)$$

$$w = -0.125 \begin{pmatrix} x_1^1 = 1^2 = 1 \\ \sqrt{2}x_1^1 x_2^1 = \sqrt{2}(1)(1) = \sqrt{2} \\ x_2^1 = 1^2 = 1 \\ \sqrt{2}x_1^1 = \sqrt{2}(1) = \sqrt{2} \\ \sqrt{2}x_2^1 = \sqrt{2}(1) = \sqrt{2} \\ 1 \end{pmatrix} + 0.125 \begin{pmatrix} 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \\ 1 \end{pmatrix} + 0.125 \begin{pmatrix} 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \\ 1 \end{pmatrix} - 0.125 \begin{pmatrix} 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -0.71 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Sehingga diperoleh nilai w

$$w = \sum_{i=1}^N \alpha_i y_i \phi(X_i) = \begin{pmatrix} 0 \\ -0.71 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Kemudian pilih salah satu support vector dari kelas +1 dan -1 untuk menghitung nilai b.

$$b = -\frac{1}{2}(w \cdot x^+ + w \cdot x^-) = -\frac{1}{2} \left( \begin{pmatrix} 0 \\ -0.71 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ -0.71 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \\ 1 \end{pmatrix} \right)$$

$$b = -\frac{1}{2}((-0.71)(-\sqrt{2}) + (-0.71)(\sqrt{2})) = -\frac{1}{2}((-0.71)(-\sqrt{2}) + (-0.71)(\sqrt{2})) = 0$$

Setelah diperoleh nilai w dan b maka model SVM siap digunakan untuk klasifikasi.

## 6. Melakukan Perhitungan Data Uji

$$f(\phi(x)) = \text{sign}(w \cdot \phi(x) + b) = \text{sign}\left(\sum_{i=1}^N \alpha_i y_i \phi(x_i) \cdot \phi(x) + b\right)$$

Misalkan data uji/ data test  $x_t = (1, 5)$  maka  $K(x_i, x_t) = \phi(x_i) \cdot \phi(x_t)$

Tabel 13.9 Data Uji

|                |       |                                                                           |                |
|----------------|-------|---------------------------------------------------------------------------|----------------|
| $x_t = (1, 5)$ | $x_1$ | $K(1, t) = (x_1 \cdot x_t + 1)^2 = (1.1 + 5.1 + 1)^2 = 7^2 = 49$          | $(-0.125)(49)$ |
|                | $x_2$ | $K(2, t) = (x_2 \cdot x_t + 1)^2 = (1.1 + 5.(-1) + 1)^2 = 3^2 = 9$        | $(0.125)(9)$   |
|                | $x_3$ | $K(3, t) = (x_3 \cdot x_t + 1)^2 = (1.(-1) + 5.1 + 1)^2 = 5^2 = 25$       | $(0.125)(25)$  |
|                | $x_4$ | $K(4, t) = (x_4 \cdot x_t + 1)^2 = (1.(-1) + 5.(-1) + 1)^2 = (-5)^2 = 25$ | $(-0.125)(25)$ |

$f(\phi(x_t)) = sign(w \cdot \phi([1, 5]) + b) = sign(-6.125 + 1.125 + 3.125 - 3.125 + 0) = sign(-5) \Rightarrow -1$   
Jadi data  $x_t = (1, 5)$  tersebut masuk ke kelas negatif.

### 13.5.3 SVM dengan > 2 Kelas

SVM pada saat diperkenalkan pertama kali hanya dapat menyelesaikan masalah klasifikasi data ke dalam dua kelas saja. Seiring perkembangannya, dilakukan penelitian lebih lanjut untuk dapat mengklasifikasikan data lebih dari dua kelas atau *multi-class problem*.

Pada masalah *multi-class problem*, SVM memiliki dua pendekatan, yaitu menggabungkan beberapa SVM biner (*binary classifier*) atau menggabungkan semua data yang terdiri dari beberapa kelas ke dalam sebuah bentuk permasalahan optimasi. Namun, pada pendekatan yang kedua permasalahan optimasi yang harus diselesaikan jauh lebih rumit.

Berikut metode yang biasa digunakan untuk mengimplementasikan *multi-class SVM* dengan pendekatan yang pertama, yaitu *one-against-all*, *one-against-one*, dan *Directed Acyclic Graph Support Vector Machine* (DAGSVM) (Sembiring, 2007), yang akan dibahas pada beberapa bab selanjutnya.

## 13.6 Algoritma SVM

### 13.6.1 Simplified Sequential Minimal Optimization (SMO)

*Simplified Sequential Minimal Optimization* (SSMO) merupakan algoritma yang dihasilkan dari modifikasi algoritma SMO. Pada algoritma ini memilih dua parameter  $\alpha$ , nilai objektif,  $\alpha_i$  dan  $\alpha_j$  akan dioptimalkan secara bersama-sama untuk kedua nilai alpha. Akhirnya menyesuaikan parameter  $b$  berdasarkan nilai  $\alpha$  yang baru, proses ini diulang sampai nilai  $\alpha$  konvergen. Pada algoritma SMO memilih secara heuristik untuk memilih  $\alpha_i$  dan  $\alpha_j$  yang akan dioptimasikan sehingga memaksimalkan fungsi objektif sebanyak mungkin. Pada jumlah data set yang besar, ini sangat penting untuk kecepatan dari proses algoritma, karena ada  $m(m - 1)$  kemungkinan untuk  $\alpha_i$  dan  $\alpha_j$  dan beberapa akan menghasilkan jauh lebih sedikit peningkatan dari yang lain. Algoritma *simplified* SMO akan menyederhanakan pemilihan secara heuristik pada  $\alpha$ . Algoritma ini akan melakukan iterasi untuk semua  $\alpha_i, i = 1, \dots, m$ . Jika  $\alpha_i$  memenuhi kondisi *Karush-Kuhn-Tucker* (KKT) dengan nilai toleransi tertentu maka memilih  $\alpha_j$  secara acak dari sisa  $m - 1$  dan mencoba mengoptimalkan secara bersama-sama  $\alpha_i$  dan  $\alpha_j$ . Jika  $\alpha$  tidak ada yang berubah dari beberapa iterasi maka algoritma berakhir.

Setelah memilih  $\alpha_i$  dan  $\alpha_j$  untuk dioptimalkan, menghitung batasan dari nilai parameter yang digunakan. Pertama mencari batasan pada L dan H sehingga  $L \leq \alpha_j \leq H$  untuk  $\alpha_j$  memenuhi batasan  $0 \leq \alpha_j \leq C$ . Hal ini dapat ditunjukkan pada persamaan (5.1) dan (5.2).

- Jika  $y_i \neq y_j$ ,  $L = \max(0, \alpha_j - \alpha_i)$ ,  $H = \min(C, C + \alpha_j - \alpha_i)$ .....(4.1)
  - Jika  $y_i = y_j$ ,  $L = \max(0, \alpha_j + \alpha_i - C)$ ,  $H = \min(C, \alpha_j + \alpha_i)$ .....(4.2)

dimana:

$\alpha_i, \alpha_j$  = Lagrange Multipliers pada data ke-i dan ke-j

$y_i, y_j$  = kelas (sampel positif (+1) dan sampel negatif(-1))

$C$  = batasan *error* bernilai positif

Mencari nilai  $\alpha_j$  sehingga memaksimalkan fungsi objektif. Jika nilai ini diluar L dan H maka nilai  $\alpha_j$  berada pada rentangan ini. Persamaan untuk mencari  $\alpha_j$  ditunjukkan pada persamaan (4.3).

$$\alpha_j = \alpha_j - \frac{y_j(E_i - E_j)}{\eta} \dots \quad (4.3)$$

dimana:

$$E_k = f(x) - y_k \dots \quad (4.5)$$

$$\eta = 2K(x_i, x_j) - K(x_i, x_i) - K(x_j, x_j) \dots \quad (4.6)$$

Selanjutnya memotong  $\alpha_j$  pada rentangan nilai L dan H dengan persamaan (5.7).

$$\alpha_j = \begin{cases} H, & \text{if } \alpha_j > H; \\ \alpha_j, & \text{if } L < \alpha_j < H; \\ L, & \text{if } \alpha_j < L; \end{cases} \dots \quad (4.7)$$

Akhirnya, setelah menyelesaikan  $\alpha_j$  maka nilai akhir untuk  $\alpha_j$  ditunjukkan pada persamaan (4.8).

dimana:

$\alpha_j^{lama}$  = nilai dari  $\alpha_j$  sebelum dilakukan optimasi pada persamaan (4.3) dan (4.7).

Setelah mengoptimalkan  $\alpha_i$  dan  $\alpha_j$ , memilih b sedemikian hingga kondisi KKT terpenuhi pada data  $i$  dan  $j$ . Jika setelah optimasi,  $\alpha_i$  tidak berada pada batas  $0 \leq \alpha_i \leq C$  maka nilai  $b_1$  dapat dihitung dengan menggunakan persamaan (4.9). Jika  $\alpha_j$  tidak berada pada batas  $0 \leq \alpha_j \leq C$  maka nilai  $b_2$  dapat dihitung dengan menggunakan persamaan (4.10). Jika  $\alpha_i$  dan  $\alpha_j$  tidak memenuhi kedua kondisi tersebut maka nilai b didapatkan dengan  $b = (b_1 + b_2)/2$ .

$$b_1 = b - E_i - y_i(\alpha_i - \alpha_i^{l a m a}) K(x_i, x_i) - y_j(\alpha_j - \alpha_j^{l a m a}) K(x_j, x_i) \dots \quad (4.9)$$

$$b_2 = b - E_j - y_i(\alpha_i - \alpha_i^{l a m a}) K(\vec{x}_i, \vec{x}_j) - y_j(\alpha_j - \alpha_j^{l a m a}) K(x_i, x_j) \dots \quad (4.10)$$

$$b = \begin{cases} b_1, & \text{if } 0 < \alpha_i < C \\ b_2, & \text{if } 0 < \alpha_j < C \\ \frac{(b_1+b_2)}{2} & \text{otherwise} \end{cases} \dots \quad (4.11)$$

dimana:

$\alpha_i, \alpha_j$  = Lagrange Multipliers  
 $y_i, y_j$  = kelas (sampel positif (+1) dan sampel negatif(-1))  
 $K(\vec{x}_1, \vec{x}_1)$  = fungsi kernel  
 $b$  = bias  
 $E_i, E_j$  = error pada data training ke- $i$  dan data ke- $j$

### Pseudo-code *Simplified SMO*

### Input:

C: nilai complexity

tol: nilai toleransi

max\_iter: jumlah maksimum iterasi semua data

latih tanpa adanya perubahan nilai  $\alpha$

$(x_1, y_1), \dots, (x_n, y_n)$ : data latih

Output :

$\alpha$ : untuk semua data latih

*b:* nilai bias

- Inisialisasi awal  $\alpha = 0$ ,  $b = 0$ , pass = 0
    - while (pass < max\_iter)
      - alpha\_berubah = 0.
      - for i = 1 to m
    - hitung  $E_i = f(x_i) - y_i$  menggunakan persamaan (5.5)
    - if(  $(y_i E_i < -tol \&\& \alpha_i < C) || (y_i E_i > tol \&\& \alpha_i > 0)$ )
      - memilih j != i (random)
    - hitung  $E_j = f(x_j) - y_j$  menggunakan persamaan (5.5)
    - simpan nilai  $\alpha$ :  $\alpha_{i\_lama} = \alpha_i$ ,  $\alpha_{j\_lama} = \alpha_j$

```
 • hitung L dan H menggunakan
 persamaan (5.1) dan (5.2)
 • if (L==H)
 continue i++.
 • hitung η menggunakan persamaan
 (5.6)
 • if (η >= 0)
 continue i++.
 • hitung α_j baru menggunakan persamaan
 (5.3) dan (5.7)
 • if ($|\alpha_j - \alpha_j_{lama}| < 10^{-5}$)
 continue i++.
 • menentukan nilai α_i menggunakan
 persamaan (5.8)
 • menghitung bias berdasarkan kondisi
 pada persamaan (5.11)
 • alpha_berubah = alpha_berubah + 1
 • if berakhir.
 • for berakhir.
 • if (alpha_berubah == 0)
 pass = pass + 1
 • else
 pass = 0
 while berakhir.
```

Gambar 13.8 Gambar Pseudo-code *Simplified SMO*

**Contoh perhitungan manual** menggunakan algoritma *Simplified SMO* dalam proses pelatihan untuk mendapatkan nilai optimal dari  $\alpha$  dan *bias* pada kasus pemilihan kualitas citra. Data latih yang digunakan dalam perhitungan ini menggunakan data sebanyak 6 data yang terbagi ke dalam dua kelas (Baik dan Buruk).

Pada proses perhitungan dibutuhkan fitur rata-rata *red*, *green* dan *blue* (RGB) untuk melakukan proses pelatihan. Berikut proses perhitungan fitur rata-rata *red* (R) dengan mengacu representasi *cropping* citra.

$$\begin{aligned}\bar{X}_R &= \frac{X_{11} + X_{12} + X_{13} + X_{21} + \dots + X_{3232}}{1024} \\ &= \frac{62+62+62+63+\dots+65}{1024}\end{aligned}$$

$$= \frac{71680}{1024} \\ = 70$$

Proses perhitungan fitur rata-rata *green* (G) dan *blue* (B) sama seperti proses perhitungan fitur rata-rata *red* (R).

Tabel 13.10 Data Pelatihan

| Data Citra ke- | Id Gambar | Fitur |     |     | Kelas | Keterangan Kelas | Target (y) |
|----------------|-----------|-------|-----|-----|-------|------------------|------------|
|                |           | R     | G   | B   |       |                  |            |
| X1             | SB-2      | 70    | 67  | 55  | 1     | Baik             | 1          |
| X2             | SB-5      | 57    | 61  | 68  | 1     | Baik             | 1          |
| X3             | SB-17     | 133   | 88  | 50  | 2     | Baik             | 1          |
| X4             | SB-20     | 164   | 124 | 96  | 2     | Baik             | 1          |
| X5             | SB-40     | 170   | 130 | 86  | 3     | Buruk            | -1         |
| X6             | SB-42     | 173   | 156 | 125 | 3     | Buruk            | -1         |

Pada tabel di atas terdapat masing-masing dua data gambar untuk tiap kelas (Baik dan Buruk). Pada pelatihan data gambar untuk kelas Baik memiliki target ( $y=+1$ ) sedangkan data gambar untuk kelas Buruk memiliki target ( $y= -1$ ). Pelatihan ini akan menghasilkan fungsi keputusan berdasarkan nilai optimal dari  $\alpha$  dan *bias* pada masing-masing data latih. Berikut adalah proses pelatihan menggunakan algoritma *Simplified SMO*.

### Proses Pelatihan (Baik vs Buruk)

Pada proses pelatihan menggunakan algoritma *Simplified SMO* terlebih dahulu diberikan nilai parameter-parameter yang dibutuhkan yaitu  $\alpha = 0$ ,  $bias\_awal (b) = 0$ ,  $tol = 0,01$ ,  $tol2 = 0,0001$ ,  $C = 1$ ,  $gamma = 0,001$ , dan  $maxIter = 1$ . Nilai parameter  $C$ ,  $gamma$  dan  $maxIter$  merupakan parameter bebas yang akan digunakan dalam proses pengujian.

Tahap awal proses pelatihan mencari nilai *dot product* dari masing-masing data latih menggunakan kernel *Radial Basis Function* (RBF) yang ditunjukkan oleh persamaan (2.31). Sebagai contoh nilai kernel pada data X1 dan X2 ( $K(x_1, x_2)$ ) dengan  $gamma = 0,001$  adalah

$$K(x_1, x_2) = \exp\left(-\gamma \|x_i - x_j\|^2\right) \\ = \exp(-0,001 * ((70 - 57)^2 + (67 - 61)^2 + (55 - 88)^2))$$

$$= \exp (-0,374)$$

$$= \mathbf{0,687976912}$$

Berdasarkan perhitungan kernel diatas maka didapatkan nilai *dot product* untuk data latih seperti yang ditunjukan oleh Tabel berikut.

Tabel 13.11 Hasil Perhitungan dengan Kernel RBF

| K(xi,xj) | X1          | X2          | X3          | .... | X6          |
|----------|-------------|-------------|-------------|------|-------------|
| X1       | 1           | 0,687976912 | 0,011855066 | .... | 6,67545E-11 |
| X2       | 0,687976912 | 1           | 0,00108194  | .... | 6,69273E-12 |
| X3       | 0,011855066 | 0,00108194  | 1           | .... | 7,1457E-06  |
| X4       | 1,05076E-06 | 9,19519E-08 | 0,012613344 | .... | 0,142844308 |
| X5       | 3,28083E-07 | 1,76241E-08 | 0,01192641  | .... | 0,11014033  |
| X6       | 6,67545E-11 | 6,69273E-12 | 7,1457E-06  | .... | 1           |

- Tahap ke – 1

Mencari nilai *error* pada setiap data training (data ke-i) dengan menggunakan persamaan (4.5). Contoh perhitungan nilai *error* (Ei) pada data latih adalah misal mencari *error(i)* pada perulangan ke-1 dengan data ke-1, alpha = 0, bias = 0,target(y) = 1

$$f(x) = \sum_{i=1}^n \alpha_i y_i K(x, x_i) + b$$

$$E_i = f(x) - y_i$$

$$\begin{aligned} f(x) = & ((0 * 1 * 1) + (0 * 1 * 0,687976912) + (0 * 1 * 0,011855066) \\ & + (0 * 1 * 1,05076E - 06) \\ & + (0 * -1 * 3,28083E - 07) \\ & + (0 * -1 * 6,67545E - 11) + 0 = 0 \end{aligned}$$

$$E_1 = 0 - 1 = -1$$

Berdasarkan perhitungan *error(i)* di atas maka didapatkan nilai *error* pada tiap data training untuk masing-masing perulangan seperti yang ditunjukan pada Tabel berikut.

Tabel 13.12 Hasil *Error* pada data ke-i

| Perulangan ke- | data ke-i | f(x)         | y <sub>i</sub> | E <sub>i</sub> |
|----------------|-----------|--------------|----------------|----------------|
| 1              | 1         | 0            | 1              | -1             |
| 2              | 2         | 0            | 1              | -1             |
| 3              | 3         | 0,001074794  | 1              | -0,99892521    |
| 4              | 4         | -0,142844216 | 1              | -1,14284422    |
| 5              | 5         | -0,983648048 | -1             | 0,016351952    |
| 6              | 6         | -0,983648066 | -1             | 0,016351934    |
| 7              | 1         | 0,814468848  | 1              | -0,18553115    |
| 8              | 2         | 1,126492338  | 1              | 0,126492338    |
| 9              | 3         | 0,128253991  | 1              | -0,87174601    |
| 10             | 4         | 0,141668875  | 1              | -0,85833112    |
| 11             | 5         | -0,141668875 | -1             | 0,858331125    |
| 12             | 6         | -0,840803758 | -1             | 0,159196242    |

- Tahap ke – 2

Melakukan pengecekan terhadap kondisi *Karush-Kuhn-Tucker* (KKT) pada data training dan nilai  $E_i$ , jika tidak memenuhi kondisi tersebut maka perulangan kembali pada tahap ke – 1 dengan melakukan perhitungan error pada data ke-i selanjutnya.

Hasil dari kondisi KKT pada tiap data latih untuk masing-masing perulangan dapat dilihat pada Tabel berikut.

Tabel 13.13 Kondisi KKT

| Perulangan ke- | Hasil | Perulangan ke- | Hasil |
|----------------|-------|----------------|-------|
| 1              | True  | 7              | True  |
| 2              | True  | 8              | True  |
| 3              | True  | 9              | True  |
| 4              | True  | 10             | False |
| 5              | False | 11             | False |
| 6              | False | 12             | False |

- Tahap ke – 3

Tahap berikutnya adalah mencari data pembanding dari data latih secara *random* (acak) dengan ketentuan  $j \neq i$  dan mencari nilai *error* ( $E_j$ ) pada data tersebut.

Proses perhitungan nilai *error* pada data ke- $j$  sama seperti perhitungan nilai *error* pada data ke- $i$  menggunakan persamaan (4.5), tetapi untuk menentukan data pembanding dilakukan secara acak. Berikut contoh perhitungan nilai *error(j)* pada perulangan ke-1, misalkan nilai  $j = 3$ .

$$f(x) = \sum_{i=1}^n \alpha_i y_i K(x, x_i) + b$$

$$E_j = f(x) - y_j$$

$$\begin{aligned} f(x) = & ((0 * 1 * 0,011855066) + (0 * 1 * 0,00108194) + (0 * 1 * 1) \\ & + (0 * 1 * 0,012613344) + (0 * -1 * 0,01192641) \\ & + (0 * -1 * 7,1457E - 06) + 0 = 0 \end{aligned}$$

$$E_3 = 0 - 1 = -1$$

Berdasarkan perhitungan *error* ( $E_j$ ) di atas maka didapatkan nilai *error* pada tiap data latih untuk masing-masing perulangan seperti yang ditunjukkan pada Tabel berikut.

Tabel 13.14 *Error* data pembanding ( $E_j$ )

| Perulangan ke- | Data ke- $j$ | $f(x)$   | $y_j$ | $E_j$    |
|----------------|--------------|----------|-------|----------|
| 1              | 3            | 0        | 1     | -1       |
| 2              | 6            | 0        | -1    | 1        |
| 3              | 6            | -1       | -1    | 6.69E-12 |
| 4              | 5            | -0,11014 | -1    | 0.88986  |
| 5              | null         | null     | null  | null     |
| 6              | null         | null     | null  | null     |
| 7              | 5            | -0,98365 | -1    | 0.016352 |
| 8              | 4            | 0,141669 | 1     | -0.85833 |
| 9              | 5            | -0,14167 | -1    | 0.858331 |
| 10             | null         | null     | null  | null     |
| 11             | null         | null     | null  | null     |

|    |      |      |      |      |
|----|------|------|------|------|
| 12 | null | null | null | null |
|----|------|------|------|------|

- Tahap ke – 4

Men-set nilai alpha pada data  $i$  dan data  $j$  ke dalam variabel `alpha_lama`. Hasil men-set nilai alpha ke dalam variabel `alpha_lama` ditunjukkan pada tabel berikut.

Tabel 13.15 Nilai Alpha data  $i$  dan  $j$

| Perulangan ke- | i    | j    | Alpha_lama-i | Alpha_lama-j |
|----------------|------|------|--------------|--------------|
| 1              | 1    | 3    | 0            | 0            |
| 2              | 2    | 6    | 0            | 0            |
| 3              | 3    | 6    | 0            | 1            |
| 4              | 4    | 5    | 0            | 0            |
| 5              | null | null | null         | null         |
| 6              | null | null | null         | null         |
| 7              | 1    | 5    | 0            | 1            |
| 8              | 2    | 4    | 1            | 1            |
| 9              | 3    | 5    | 0            | 1            |
| 10             | null | null | null         | null         |
| 11             | null | null | null         | null         |
| 12             | null | null | null         | null         |

- Tahap ke – 5

Menghitung segmen garis diagonal dengan mencari nilai L (*Low*) dan H (*High*) sesuai dengan persamaan (4.1) dan (4.2). Pencarian nilai L dan H ditentukan dari target yang dimiliki dari data  $i$  dan data  $j$ .

Contoh perhitungan mencari nilai L dan H berdasarkan perhitungan pada tahap 1 dan 3 target yang didapatkan adalah  $y[i] = 1$  dan  $y[j] = 1$  (sama).

$$L = \max(0, \alpha[i] + \alpha[j] - C) \quad H = \min(C, \alpha[j] + \alpha[i])$$

$$L = \max(0, 0 - 0 - 1)$$

$$H = \min(1, 0 + 0)$$

$$L = 0$$

$$H = 0$$

Berdasarkan perhitungan di atas maka didapatkan nilai L dan H pada tiap data latih untuk masing-masing perulangan seperti yang ditunjukkan pada tabel berikut.

Tabel 13.16 Nilai L dan H (target sama)

| Perulangan ke- | L    | H    | target(y) i | target (y) j |
|----------------|------|------|-------------|--------------|
| 1              | 0    | 0    | 1           | 1            |
| 2              | 0    | 0    | 1           | -1           |
| 3              | 0    | 1    | 1           | -1           |
| 4              | 0    | 0    | 1           | -1           |
| 5              | null | Null | null        | null         |
| 6              | null | Null | null        | null         |
| 7              | 0    | 1    | 1           | -1           |
| 8              | 1    | 1    | 1           | 1            |
| 9              | 0    | 1    | 1           | 1            |
| 10             | null | null | null        | null         |
| 11             | null | null | null        | null         |
| 12             | null | null | null        | null         |

Tabel 13.17 Nilai L dan H (target tidak sama)

| Perulangan ke- | L    | H    | target i | target j |
|----------------|------|------|----------|----------|
| 1              | 0    | 1    | 1        | 1        |
| 2              | 0    | 1    | 1        | -1       |
| 3              | 1    | 1    | 1        | -1       |
| 4              | 0    | 1    | 1        | -1       |
| 5              | null | null | null     | null     |
| 6              | null | null | null     | null     |
| 7              | 1    | 1    | 1        | -1       |
| 8              | 0    | 1    | 1        | -1       |
| 9              | 1    | 1    | 1        | -1       |
| 10             | null | null | null     | null     |

|    |      |      |      |      |
|----|------|------|------|------|
| 11 | null | null | null | null |
| 12 | null | null | null | null |

- Tahap ke – 6

Mengecek hasil dari perhitungan L dan H sama atau tidak, jika sama maka kembali ke tahap 1 dengan  $i++$ . Hasil pengecekan terhadap nilai L dan H ditunjukan pada tabel berikut.

Tabel 13.18 Nilai L dan H (Kondisi)

| Perulangan ke- | Hasil      | L    | H    | Target(y)<br>i | Target(y)<br>j |
|----------------|------------|------|------|----------------|----------------|
| 1              | SAMA       | 0    | 0    | 1              | 1              |
| 2              | TIDAK SAMA | 0    | 1    | 1              | -1             |
| 3              | SAMA       | 1    | 1    | 1              | -1             |
| 4              | TIDAK SAMA | 0    | 1    | 1              | -1             |
| 5              | null       | null | null | null           | null           |
| 6              | null       | null | null | null           | null           |
| 7              | SAMA       | 1    | 1    | 1              | -1             |
| 8              | SAMA       | 1    | 1    | 1              | 1              |
| 9              | SAMA       | 1    | 1    | 1              | 1              |
| 10             | null       | null | null | null           | null           |
| 11             | null       | null | null | null           | null           |
| 12             | null       | null | null | null           | null           |

- Tahap ke – 7

Menghitung nilai eta ( $\eta$ ) dengan persamaan (4.6) dan mengecek lokasi batas maksimal dari fungsi objektif. Contoh perhitungan  $\eta$  pada perulangan ke-4 untuk  $i = 4$  dan  $j = 5$  dengan menggunakan persamaan (4.6). Nilai  $K(x_i, x_j)$  ,  $K(x_i, x_i)$  ,  $K(x_j, x_j)$  didapatkan pada perhitungan menggunakan kernel RBF pada tabel sebelumnya.

$$\begin{aligned}\eta &= 2K(x_i, x_j) - K(x_i, x_i) - K(x_j, x_j) \\ &= 2 * 0,841979173 - 1,00 - 1,00 \\ &= \mathbf{-3,16041653663000E - 01}\end{aligned}$$

Berdasarkan contoh perhitungan di atas maka hasil perhitungan nilai *eta* pada masing-masing proses ditunjukkan pada Tabel berikut.

Tabel 13.19 Hasil perhitungan *eta*

| Perulangan ke- | Eta                   | i    | j    |
|----------------|-----------------------|------|------|
| 1              | null                  | 1    | 3    |
| 2              | -1,99999999998661E+00 | 2    | 6    |
| 3              | null                  | 3    | 6    |
| 4              | -3,16041653663000E-01 | 4    | 5    |
| 5              | null                  | null | null |
| 6              | null                  | null | null |
| 7              | null                  | null | null |
| 8              | null                  | null | null |
| 9              | null                  | null | null |
| 10             | null                  | null | null |
| 11             | null                  | null | null |
| 12             | null                  | null | null |

- Tahap ke – 8

Menghitung nilai alpha baru pada data ke-j dengan menggunakan persamaan (4.3). Contoh perhitungan *alpha baru* untuk data ke-j pada perulangan ke-4 untuk i = 4 dan j = 5. Nilai  $E_i$  dan  $E_j$  didapatkan dari perhitungan tahap 1 dan tahap 3 pada Tabel sebelumnya.

$$\alpha_j = \alpha_j - \frac{y_j(E_i - E_j)}{\eta}$$
$$= 0 - \frac{-1(-1,142844216 + 0,889859688)}{-3,16041653663000E - 01}$$

$$= 6,43175948426762$$

Berdasarkan contoh perhitungan di atas maka hasil perhitungan nilai *alpha\_baru* pada masing-masing proses ditunjukkan pada Tabel berikut.

Tabel 13.20 Nilai Alpha Baru data ke-*j*

| Perulangan ke- | alpha_baru ke- <i>j</i> | i    | j    |
|----------------|-------------------------|------|------|
| 1              | null                    | 1    | 3    |
| 2              | 1,00000000000066900E+00 | 2    | 6    |
| 3              | null                    | 3    | 6    |
| 4              | 6,4317594842676200E+00  | 4    | 5    |
| 5              | null                    | null | null |
| 6              | null                    | null | null |
| 7              | null                    | null | null |
| 8              | null                    | null | null |
| 9              | null                    | null | null |
| 10             | null                    | null | null |
| 11             | null                    | null | null |
| 12             | null                    | null | null |

- Tahap ke – 9

Menentukan batasan maksimum nilai alpha dengan memotong segmen garis selain batas maksimum. Hasil menentukan batasan maksimum nilai alpha pada masing-masing perulangan ditunjukkan pada Tabel berikut.

Tabel 13.21 Nilai Alpha data *j* pada segmen garis

| Perulangan ke- | alpha_baru ke- <i>j</i> | j    |
|----------------|-------------------------|------|
| 1              | null                    | 3    |
| 2              | 1                       | 6    |
| 3              | null                    | 6    |
| 4              | 1                       | 5    |
| 5              | null                    | null |
| 6              | null                    | null |
| 7              | null                    | null |
| 8              | null                    | null |
| 9              | null                    | null |
| 10             | null                    | null |
| 11             | null                    | null |

|    |      |      |
|----|------|------|
| 12 | null | null |
|----|------|------|

- Tahap ke – 10

Mengecek nilai absolut dari  $\alpha_{baru}[j] - \alpha_{lama}[j]$  lebih kecil dari nilai tol2 terpenuhi atau tidak, jika terpenuhi maka akan kembali ke tahap 1. Hasil dari mengecek nilai absolut pada masing-masing proses ditunjukan pada Tabel berikut.

Tabel 13.22 Kondisi nilai absolut

| Perulangan ke- | hasil | kondisi        |
|----------------|-------|----------------|
| 1              | null  | null           |
| 2              | 1     | Tidak Memenuhi |
| 3              | null  | null           |
| 4              | 1     | Tidak Memenuhi |
| 5              | null  | null           |
| 6              | null  | null           |
| 7              | null  | null           |
| 8              | null  | null           |
| 9              | null  | null           |
| 10             | null  | null           |
| 11             | null  | null           |
| 12             | null  | null           |

- Tahap ke – 11

Menghitung nilai alpha baru pada data ke-*i* menggunakan persamaan (4.8). Contoh perhitungan *alpha baru* data ke-*i* pada perulangan ke-4 untuk *i* = 4 dan *j* = 5. Nilai *aj* didapatkan dari perhitungan tahap ke-9.

$$\begin{aligned}\alpha_i &= \alpha_i + y_i * y_j (\alpha_j^{lama} - \alpha_j) \\ &= 0 + (1 * 1) * (0 - 1) \\ &= 1\end{aligned}$$

Berdasarkan perhitungan diatas didapatkan hasil  $\alpha_i$  pada masing-masing perulangan ditunjukan pada Tabel berikut.

Tabel 13.23 Nilai Alpha Baru Data ke-*i*

| Perulangan ke- | alpha_baru_ke-i | i    | j    |
|----------------|-----------------|------|------|
| 1              | null            | 1    | 3    |
| 2              | 1               | 2    | 6    |
| 3              | null            | 3    | 6    |
| 4              | 1               | 4    | 5    |
| 5              | null            | null | null |
| 6              | null            | null | null |
| 7              | null            | null | null |
| 8              | null            | null | null |
| 9              | null            | null | null |
| 10             | null            | null | null |
| 11             | null            | null | null |
| 12             | null            | null | null |

- Tahap ke – 12

Menghitung nilai bias ( $b_1$  dan  $b_2$ ) menggunakan persamaan (4.9) dan (4.10) dengan memenuhi beberapa kondisi antara lain:

- Kondisi 1: jika memenuhi  $0 < \text{alpha } i < C$  maka  $b_1$  digunakan.
- Kondisi 2: jika memenuhi kondisi  $0 < \text{alpha } j < C$  maka  $b_2$  digunakan.
- Kondisi 3: jika tidak memenuhi kondisi diatas maka nilai bias setengah dari total  $b_1$  dan  $b_2$ .

Contoh perhitungan *bias* data ke-*i* pada perulangan ke-4 untuk  $i = 4$  dan  $j = 5$ . Nilai  $E_i$  dan  $E_j$  didapatkan dari perhitungan tahap 1 dan tahap 3. Nilai  $K(x_i, x_j)$ ,  $K(x_i, x_i)$ ,  $K(x_j, x_j)$  didapatkan pada perhitungan menggunakan kernel RBF.

$$\begin{aligned} b_1 &= b - E_i - y_i(\alpha_i - \alpha_i^{lama})K(x_i, x_i) - y_j(\alpha_j - \alpha_j^{lama})K(x_i, x_j) \\ &= 3,92827E - 17 - (-1,142844216) - 1 * (1 - 0) * 1 - 1 * (1 - 0) \\ &\quad * 0,841979173 \\ &= \mathbf{0,984823389} \end{aligned}$$

$$\begin{aligned} b_2 &= b - E_i - y_i(\alpha_i - \alpha_i^{\text{lama}})K(x_i, x_j) - y_j(\alpha_j - \alpha_j^{\text{lama}})K(x_j, x_i) \\ &= 3,92827E - 17 - 0,889859688 - 1 * (1 - 0) * 0,841979173 - 1 \\ &\quad * (1 - 0) * 1 \\ &= \mathbf{-0,731838861} \end{aligned}$$

Tabel 13.24 Nilai b1 dan b2

| Perulangan ke- | bias_1      | bias_2       |
|----------------|-------------|--------------|
| 1              | null        | null         |
| 2              | 6,69273E-12 | -6,69265E-12 |
| 3              | null        | null         |
| 4              | 0,984823389 | -0,731838861 |
| 5              | null        | null         |
| 6              | null        | null         |
| 7              | null        | null         |
| 8              | null        | null         |
| 9              | null        | null         |
| 10             | null        | null         |
| 11             | null        | null         |
| 12             | null        | null         |

Setelah mendapatkan nilai pada bias\_1 dan bias\_2 maka akan dilakukan perhitungan untuk mengitung bias optimal (bias\_baru) sesuai dengan persamaan (4.11) pada tiap perulangan. Nilai  $\alpha_j$  dan  $\alpha_i$  masing-masing dapat dilihat pada tabel sebelumnya. Berikut contoh perhitngan bias\_baru pada perulangan ke-4. Diketahui  $\alpha_j = 1, \alpha_i = 1, C = 1$  sehingga bias baru pada perulangan ke-4 menggunakan kondisi ke-3.

$$\begin{aligned} b &= \frac{b_1 + b_2}{2} \\ &= \frac{0,984823389 + (-0,731838861)}{2} \\ &= \mathbf{0,126492264} \end{aligned}$$

Tabel 13.25 Nilai bias akhir

| Perulangan ke- | bias_baru   |
|----------------|-------------|
| 1              | null        |
| 2              | 3,92827E-17 |
| 3              | null        |
| 4              | 0,126492264 |
| 5              | null        |
| 6              | null        |
| 7              | null        |
| 8              | null        |
| 9              | null        |
| 10             | null        |
| 11             | null        |
| 12             | null        |

Setelah semua data latih tidak memenuhi kondisi KKT (tahap ke 1) maka didapatkan nilai optimal dari *bias* dan *alpha* pada masing-masing data latih. Tabel 13.26 menunjukan nilai *bias* dan *alpha* optimal dari proses pelatihan (Baik vs Buruk). Untuk mendapatkan nilai *alpha* pada tiap data latih dapat dilihat pada Tabel sebelumnya.

Tabel 13.26 Nilai Alpha dan Bias Pada Proses Pelatihan

|         |             |
|---------|-------------|
| Alpha 1 | 0           |
| Alpha 2 | 1           |
| Alpha 3 | 0           |
| Alpha 4 | 1           |
| Alpha 5 | 1           |
| Alpha 6 | 1           |
| bias    | 0,126492264 |

Berdasarkan nilai *alpha* dan *bias* dari proses pelatihan, maka didapatkan fungsi keputusan yang sesuai dengan persamaan *hyperplane* untuk pelatihan di atas adalah sebagai berikut:

$$f(x_d) = \text{sign}(\sum_{i=1}^n \alpha_i y_i K(x_i, x_d) + b)$$

$$\begin{aligned} f^1(x_{uji}) = & \text{sign}((0 * 1 * K(x_i, x_{uji})) + (1 * 1 * K(x_i, x_{uji}))) \\ & + (0 * 1 * K(x_i, x_{uji})) + (1 * 1 * K(x_i, x_{uji})) \\ & + (1 * -1 * K(x_i, x_{uji})) + (1 * -1 * K(x_i, x_{uji})) \\ & + (0,126492264)) \end{aligned}$$

Berikut contoh klasifikasi yang akan dilakukan pada sebuah data uji yang belum diketahui kelasnya. Tabel berikut merupakan data dari data uji yang akan digunakan dalam proses klasifikasi.

Tabel 13.27 Data Uji

| Data Citra Ke- | Fitur |     |     | Hasil Klasifikasi |
|----------------|-------|-----|-----|-------------------|
|                | R     | G   | B   |                   |
| x7             | 132   | 123 | 121 | ?                 |

Tahap awal proses testing adalah mencari nilai *dot product* dari data uji dengan data latih yang digunakan selama proses pelajaran menggunakan kernel RBF. Tabel berikut merupakan nilai *dot product* dari data uji dan data latih menggunakan kernel RBF.

Tabel 13.28 Hasil Hitung Kernel RBF Data Uji dan Data Training

| K(xi,xj) | X1          | X2          | ..... | X6          |
|----------|-------------|-------------|-------|-------------|
| X1       | 1           | 0,687976912 | ..... | 6.68E-06    |
| X2       | 0,687976912 | 1           | ..... | 6.69E-07    |
| X3       | 0,011855066 | 0,00108194  | ..... | 7.15E-02    |
| X4       | 1.05E-01    | 9.20E-03    | ..... | 0,142844308 |

|    |             |            |       |             |
|----|-------------|------------|-------|-------------|
| X5 | 3.28E-02    | 1.76E-03   | ..... | 0,11014033  |
| X6 | 6.68E-06    | 6.69E-07   | ..... | 1           |
| X7 | 0,003848776 | 0,00083007 | ..... | 0,001606014 |

Tahap selanjutnya memasukkan nilai *dot product* untuk data uji ke dalam fungsi *hyperplane* yang telah didapatkan pada proses pelatihan, berikut uraiannya:

$$\begin{aligned}f^1(x_{uji}) = & \text{sign}((0 * 1 * 0,003848776) + (1 * 1 * 0,00083007) \\& + (0 * 1 * 0,228778727) + (1 * 1 * 0,155984287) \\& + (1 * -1 * 0,082578989) \\& + (1 * -1 * 0,001606014) + (0,126492264))\end{aligned}$$

$$f^1(x_{uji}) = \text{sign}(0,1991216) = 1$$

Berdasarkan hasil dari fungsi *hyperplane* di atas bernilai **+1**, maka data uji tersebut diklasifikasikan ke dalam kelas **Baik**.

### 13.6.2 Sequential Training SVM

Metode SVM memiliki beberapa proses training, diantaranya adalah *Quadratic Programming* (QP), *Sequential Minimal Optimization* (SMO), dan *Sequential Training*. Untuk mencari *hyperplane* yang optimal dalam SVM dapat menggunakan QP, tetapi QP memerlukan waktu yang lama, algoritma yang cukup kompleks, dan rentan terhadap ketidakstabilan numerik.

SMO merupakan pengembangan dari QP, SMO lebih digunakan untuk menyelesaikan permasalahan optimasi yang kecil di setiap tahapnya, tetapi algoritma SMO cukup kompleks. Sedangkan *Sequential Training* memiliki algoritma yang lebih sederhana dan waktu yang diperlukan lebih cepat. Metode *Sequential Training* dikembangkan oleh Vijayakumar, dijelaskan dalam tahapan berikut (Vijayakumar, 1999).

1. Setelah melakukan perhitungan kernel, kemudian proses pertama yang dilakukan pada metode *Sequential Training* adalah initialisasi untuk parameter SVM,  $\alpha_i = 0$  dan parameter lain, misalnya  $\lambda = -0,000208$ ,  $\gamma = 0,259$ ,  $C = 20,8$  dan  $\varepsilon = 0,001$ .

Keterangan:

$\alpha_i$  = alfa / Lagrange Multiplier, untuk mencari support vector.

$\lambda$  = Koefisien *Lagrange Multiplier*.

$\gamma$  = Konstanta *Gamma*, untuk mengontrol kecepatan *learning*.

$C$  = Konstanta  $C$ , untuk membatasi nilai alfa pada saat proses *training*.

$\varepsilon$  = Epsilon, untuk ukuran *error* klasifikasi.

2. Menghitung matriks *Hessian* ( $D_{ij}$ ) untuk menampung matriks *kernel* yang digunakan. Fungsi untuk menghitung matriks *Hessian*:

$$D_{ij} = y_i y_j ( K(x_i, x_j) + \lambda^2 ) \quad (4.12)$$

Lakukan langkah 1 dan 2 untuk  $i, j = 1, 2, \dots, n$

Keterangan:

$x_i$  = Data ke- $i$ .

$x_j$  = Data ke- $j$ .

$y_i$  = Kelas data ke- $i$ .

$y_j$  = Kelas data ke- $j$ .

$n$  = Jumlah data.

$K(x_i, x_j)$  = Fungsi *kernel* yang digunakan.

3. Untuk setiap  $i, j = 1, 2, \dots, n$ , hitung dengan menggunakan persamaan berikut ini:

a)  $E_i = \sum_{j=1}^n \alpha_j D_{ij} \quad (4.13)$

Keterangan:

$\alpha_j$  = Alfa ke- $j$ .

$D_{ij}$  = Matriks *Hessian*.

$E_i$  = Error rate.

b)  $\delta\alpha_i = \min \{ \max [ \gamma (1 - E_i), -\alpha_i ], C - \alpha_i \} \quad (4.14)$

$\delta\alpha_i$  merupakan variabel tunggal, bukan bentuk perkalian  $\delta$  dan  $\alpha_i$ . Notasi  $\gamma$  merupakan *Learning Rate* untuk mengontrol kecepatan proses *learning*. Nilai konstanta untuk parameter  $\gamma$ , dapat dilihat pada inisialisasi parameter SVM.

$$\gamma = \frac{\text{konstanta}}{\text{Max } \{i\} D_{ii}} \quad (4.15)$$

$\text{Max}_{\{i\}} D_{ii}$  adalah nilai maximum yang diperoleh dari nilai diagonal pada matriks.

c)  $\alpha_i = \alpha_i + \delta \alpha_i \quad (4.16)$

Fungsi  $\delta\alpha_i$  adalah fungsi konvergensi untuk memantau perubahan funsi *Lagrange Multiplier*. Jika data *training* telah mencapai nilai konvergen ( $\max(|\delta\alpha_i|) < \varepsilon$ ), dan ketika maximum iterasi mencapai nilai yang ditentukan maka iterasi akan dihentikan.

- d) Proses diulangi sampai  $\alpha$  mencapai nilai konvergen. Konvergen dapat didefinisikan dari tingkat perubahan pada nilai  $\alpha$ .
- e) Selanjutnya nilai *Support Vector* (SV) dapat diketahui dengan  $SV = (\alpha_i > \text{ThresholdSV})$ . Nilai ThresholdSV didapatkan dari hasil beberapa percobaan, biasanya digunakan  $\text{ThresholdSV} \geq 0$ .

**Contoh perhitungan manual** menggunakan algoritma Sequential training SVM dalam proses pelatihan untuk mendapatkan nilai optimal dari  $\alpha$  dan bias pada kasus klasifikasi tingkat resiko penyakit stroke. Data latih yang digunakan dalam perhitungan ini menggunakan data sebanyak 18 data pasien yang terbagi ke dalam tiga kelas dari status resiko penyakit stroke yaitu 1 (normal), 2 (rentan), dan 3 (mengkhawatirkan). 18 data pasien untuk dijadikan *dataset* yang akan ditunjukkan pada Tabel berikut.

Tabel 13.29 *Dataset*

| No  | Umur | Kolesterol Total | HDL  | LDL   | Trigliserida | Kelas |
|-----|------|------------------|------|-------|--------------|-------|
| 1   | 44   | 174              | 36,5 | 123,3 | 71           | 1     |
| 2   | 60   | 169              | 36,9 | 110,1 | 110          | 1     |
| 3   | 23   | 196              | 40,3 | 137,1 | 93           | 1     |
| 4   | 41   | 213              | 47,4 | 143,2 | 112          | 1     |
| 5   | 40   | 253              | 46,5 | 179,3 | 136          | 1     |
| 6   | 48   | 197              | 30,5 | 120,5 | 79           | 1     |
| 119 | 55   | 255              | 37,8 | 184,6 | 163          | 2     |
| 120 | 42   | 191              | 40,1 | 117,1 | 169          | 2     |
| 121 | 42   | 242              | 42,5 | 162,5 | 185          | 2     |
| 122 | 25   | 210              | 40,1 | 134,1 | 179          | 2     |
| 123 | 35   | 197              | 39,6 | 119,6 | 189          | 2     |

Tabel 13.29 *Dataset*

| No  | Umur | Kolesterol Total | HDL  | LDL   | Trigliserida | Kelas |
|-----|------|------------------|------|-------|--------------|-------|
| 124 | 52   | 197              | 30,6 | 92,6  | 169          | 2     |
| 156 | 53   | 245              | 45,7 | 56,5  | 214          | 3     |
| 157 | 44   | 291              | 46,7 | 180,5 | 319          | 3     |
| 158 | 53   | 223              | 39,4 | 132,2 | 252          | 3     |
| 159 | 26   | 266              | 40,2 | 181,2 | 223          | 3     |
| 160 | 67   | 276              | 51,4 | 188,2 | 182          | 3     |
| 161 | 79   | 340              | 31,2 | 208,9 | 162          | 3     |

Seperti ditunjukkan pada tabel, nilai dari kolom F1 merupakan nilai dari fitur umur, kolom F2 menunjukkan nilai fitur kolesterol total, kolom F3 menunjukkan nilai fitur HDL, kolom F4 menunjukkan nilai fitur LDL, kolom F5 menunjukkan nilai fitur trigliserida, dan nilai kolom Kelas menunjukkan status resiko stroke. Untuk studi kasus ini dataset kemudian akan dibagi menjadi 3 *fold* dengan jumlah data yang sama banyaknya. Sebelum melakukan pembagian data terlebih dahulu dilakukan pengacakan. Pembagian data kedalam 3 *fold* ditunjukkan pada Tabel berikut.

Tabel 13.30 Pembagian *Dataset* ke dalam 3 *fold*

| No  | F1 | F2  | F3   | F4    | F5  | Kelas | fold   |
|-----|----|-----|------|-------|-----|-------|--------|
| 1   | 44 | 174 | 36,5 | 123,3 | 71  | 1     |        |
| 2   | 60 | 169 | 36,9 | 110,1 | 110 | 1     |        |
| 119 | 55 | 255 | 37,8 | 184,6 | 163 | 2     | fold 1 |
| 120 | 42 | 191 | 40,1 | 117,1 | 169 | 2     |        |
| 156 | 53 | 245 | 45,7 | 56,5  | 214 | 3     |        |

Tabel 13.30 Pembagian *Dataset* ke dalam 3 *fold*

| No  | F1 | F2  | F3   | F4    | F5  | Kelas | fold   |
|-----|----|-----|------|-------|-----|-------|--------|
| 157 | 44 | 291 | 46,7 | 180,5 | 319 | 3     |        |
| 3   | 23 | 196 | 40,3 | 137,1 | 93  | 1     | fold 2 |
| 4   | 41 | 213 | 47,4 | 143,2 | 112 | 1     |        |
| 121 | 42 | 242 | 42,5 | 162,5 | 185 | 2     |        |
| 122 | 25 | 210 | 40,1 | 134,1 | 179 | 2     |        |
| 158 | 53 | 223 | 39,4 | 132,2 | 252 | 3     |        |
| 159 | 26 | 266 | 40,2 | 181,2 | 223 | 3     |        |
| 5   | 40 | 253 | 46,5 | 179,3 | 136 | 1     | fold 3 |
| 6   | 48 | 197 | 30,5 | 120,5 | 79  | 1     |        |
| 123 | 35 | 197 | 39,6 | 119,6 | 189 | 2     |        |
| 124 | 52 | 157 | 30,6 | 92,6  | 169 | 2     |        |
| 160 | 67 | 276 | 51,4 | 188,2 | 182 | 3     |        |
| 161 | 79 | 340 | 31,2 | 208,9 | 162 | 3     |        |

Untuk studi kasus ini data uji yang digunakan adalah *fold 3* yang berisi 6 data, dan 12 data yang lain selain *fold 3* akan menjadi data latih.

Penjelasan dari diagram alir diatas akan dijabarkan sebagai berikut:

### 1. Inisialisasi nilai parameter

Inisialisasi nilai parameter SVM yang akan digunakan ditentukan oleh pengguna. Nilai parameter yang diset harus lebih dari nol. Untuk kasus ini parameter yang dilakukan inisialisasi adalah parameter *augmenting factor* ( $\lambda$ ), dan *gamma* ( $\gamma$ ), untuk nilai parameter *cost* ( $C$ ) dan varian ( $\sigma$ ) pada studi kasus ini digunakan nilai parameter yang didapat dari hasil pencarian ACO dari semut 2 dengan nilai  $C = 713,35$  dan nilai  $\sigma = 9,8294$ . Misal parameter yang telah di inisialisasi ditunjukkan pada Tabel berikut.

Tabel 13.31 Nilai Parameter

| $\lambda$ | $\gamma$ | $C$    | $\sigma$ |
|-----------|----------|--------|----------|
| 0,2       | 0,1      | 713,35 | 98,294   |

2. Menentukan kelas positif dan negatif dari data latih

Pada studi kasus ini , terdapat 3 kelas yang akan digunakan sebagai acuan untuk pengklasifikasian data, untuk itu digunakan metode *One-Against-All* untuk membantu memberi solusi permasalahan. Dalam proses *One-Against-All*, data latih yang digunakan perlu dilakukan pembagian antara kelas positif dan negatif. Untuk *level 1* data latih sebanyak 12 data dengan kelas 1 (normal) adalah sebagai kelas positif sedangkan data dengan kelas 2 (rentan) dan 3 (mengkhawatirkan) adalah sebagai kelas negatif. Penentuan kelas positif dan negatif untuk *level 1* ditunjukkan pada Tabel berikut.

Tabel 13.32 Tabel Penentuan Kelas Positif dan Negatif

| Data Latih Nomor | Kelas | $y$ |
|------------------|-------|-----|
| 1                | 1     | 1   |
| 2                | 1     | 1   |
| 119              | 2     | -1  |
| 120              | 2     | -1  |
| 156              | 3     | -1  |
| 157              | 3     | -1  |
| 3                | 1     | 1   |
| 4                | 1     | 1   |
| 121              | 2     | -1  |
| 122              | 2     | -1  |
| 158              | 3     | -1  |
| 159              | 3     | -1  |

### 3. Menghitung kernel RBF

Pada tahap ini akan dilakukan perhitungan *kernel gaussian* RBF yang dilakukan dengan menghitung semua data latih dengan Persamaan (4.17).

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \quad (4.17)$$

Variabel  $x$  dan  $y$  merupakan 2 data *training* yang akan dicari nilai *dot product* menggunakan fungsi kernel yang telah ditentukan. Nilai parameter varian menggunakan nilai yang telah didapatkan pada saat proses ACO. Untuk *One-Against-All* level 1, data latih yang digunakan sebanyak 12 data latih, maka *kernel gaussian* RBF yang dihasilkan berupa matriks  $12 \times 12$ . Untuk perhitungan baris pertama kolom pertama,  $K(1,1)$ :

$$\begin{aligned} K(1,1) &= \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \\ &= \exp\left(-\frac{(44 - 44)^2 + (174 - 174)^2 + (36,5 - 36,5)^2 + (123,3 - 123,3)^2 + (71 - 71)^2}{2 * (9,8294)^2}\right) \\ &= 1 \end{aligned}$$

Sehingga diperoleh matriks kernel seperti yang ditunjukkan dalam Tabel berikut.

Tabel 13.33 Tabel Matrik Kernel

|   | 1           | 2           | 3          | ... | 12         |
|---|-------------|-------------|------------|-----|------------|
| 1 | 1           | 3,6144E-05  | 3,2367E-43 | ... | 5,7189E-80 |
| 2 | 3,6144E-05  | 1           | 3,4042E-36 | ... | 1,4815E-64 |
| 3 | 3,2367E-43  | 3,4042E-36  | 1          | ... | 5,1046E-11 |
| 4 | 4,3746E-23  | 1,6878E-10  | 1,2062E-20 | ... | 9,8657E-30 |
| 5 | 2,0416E-68  | 9,2954E-45  | 7,9358E-44 | ... | 1,4843E-38 |
| 6 | 2,6073E-177 | 2,7847E-144 | 8,0214E-59 | ... | 1,1431E-23 |

|    |            |            |            |     |            |
|----|------------|------------|------------|-----|------------|
| 7  | 2,3594E-04 | 9,3497E-08 | 5,9877E-27 | ... | 4,1041E-54 |
| 8  | 4,2287E-09 | 1,3132E-08 | 4,8964E-15 | ... | 1,3418E-38 |
| 9  | 7,1696E-44 | 2,5768E-32 | 1,0121E-03 | ... | 1,2214E-06 |
| 10 | 5,8848E-31 | 2,8237E-19 | 1,2797E-13 | ... | 4,0974E-17 |
| 11 | 3,9294E-80 | 8,0492E-54 | 5,1314E-27 | ... | 8,2812E-14 |
| 12 | 5,7189E-80 | 1,4815E-64 | 5,1046E-11 | ... | 1          |

#### 4. Sequential learning

Pada proses pembelajaran algoritma *sequential* sesuai penelitian Vijayakumar, langkah pertama yang dilakukan adalah melakukan perhitungan matriks *Hessian* menggunakan nilai  $\lambda$  yang telah ditentukan sebelumnya, dengan rumus yang ditunjukkan pada Persamaan (4.12).

Untuk perhitungan kolom pertama kolom pertama:

$$\begin{aligned} D_{1,1} &= y_1 y_1 (K(x_1, x_1) + \lambda^2) \\ &= 1 * 1 (1 + 0,2^2) \\ &= 1,04 \end{aligned}$$

Demikian seterusnya sehingga didapatkan tabel matriks *Hessian* yang ditunjukkan pada Tabel berikut.

Tabel 13.34 Tabel Matrik *Hessian*

|   | 1     | 2     | 3     | 4     | ... | 12    |
|---|-------|-------|-------|-------|-----|-------|
| 1 | 1,04  | 0,04  | -0,04 | -0,04 | ... | -0,04 |
| 2 | 0,04  | 1,04  | -0,04 | -0,04 | ... | -0,04 |
| 3 | -0,04 | -0,04 | 1,04  | 0,04  | ... | 0,04  |
| 4 | -0,04 | -0,04 | 0,04  | 1,04  | ... | 0,04  |
| 5 | -0,04 | -0,04 | 0,04  | 0,04  | ... | 0,04  |
| 6 | -0,04 | -0,04 | 0,04  | 0,04  | ... | 0,04  |
| 7 | 0,04  | 0,04  | -0,04 | -0,04 | ... | -0,04 |

|    |       |       |       |       |     |       |
|----|-------|-------|-------|-------|-----|-------|
| 8  | 0,04  | 0,04  | -0,04 | -0,04 | ... | -0,04 |
| 9  | -0,04 | -0,04 | 0,04  | 0,04  | ... | 0,04  |
| 10 | -0,04 | -0,04 | 0,04  | 0,04  | ... | 0,04  |
| 11 | -0,04 | -0,04 | 0,04  | 0,04  | ... | 0,04  |
| 12 | -0,04 | -0,04 | 0,04  | 0,04  | ... | 1,04  |

Langkah kedua, inisialisasi nilai  $\alpha$  awal yaitu sama dengan 0. Lalu dilakukan iterasi menghitung nilai  $E_i$ ,  $\delta\alpha_i$ , dan memperbarui  $\alpha_i$  hingga iterasi yang diinginkan, untuk mendapatkan  $\alpha$  terbaru dari tiap data latih. Iterasi dalam studi kasus ini menggunakan 5 kali proses pembaharuan  $\alpha$ . Perhitungan nilai  $E_i$  dapat dinyatakan menggunakan Persamaan (4.13). Untuk perhitungan nilai  $E$  ke 1 pada iterasi 1:

$$\begin{aligned}
 E_1 &= \sum_{j=1}^{12} \alpha_j D_{1j} \\
 &= (0 * 1,04) + (0 * 0,04) + (0 * -0,04) + (0 * -0,04) + (0 * -0,04) + (0 * -0,04) \\
 &\quad + (0 * 0,04) + (0 * 0,04) + (0 * -0,04) + (0 * -0,04) + (0 * -0,04) + (0 * -0,04) \\
 &= 0
 \end{aligned}$$

Tabel hasil perhitungan nilai  $E$  ditunjukkan pada Tabel berikut.

Tabel 13.35 Tabel Nilai  $E$

| <b><math>E</math></b> | <b>iterasi 1</b> | <b>iterasi 2</b> | <b>iterasi 3</b> | ... | <b>iterasi 5</b> |
|-----------------------|------------------|------------------|------------------|-----|------------------|
| <b>1</b>              | 0                | 0,084027209      | 0,162018413      | ... | 0,300931167      |
| <b>2</b>              | 0                | 0,084003625      | 0,161975595      | ... | 0,300860678      |
| <b>3</b>              | 0                | 0,116101205      | 0,218211527      | ... | 0,387237635      |
| <b>4</b>              | 0                | 0,116462215      | 0,218855365      | ... | 0,388262276      |
| <b>5</b>              | 0                | 0,116            | 0,218030988      | ... | 0,386950164      |
| <b>6</b>              | 0                | 0,116            | 0,218030988      | ... | 0,386950164      |
| <b>7</b>              | 0                | 0,084434758      | 0,162758364      | ... | 0,302149377      |
| <b>8</b>              | 0                | 0,084411157      | 0,162715495      | ... | 0,302078739      |
| <b>9</b>              | 0                | 0,116102715      | 0,218214221      | ... | 0,387241923      |
| <b>10</b>             | 0                | 0,116463603      | 0,218857842      | ... | 0,388266222      |
| <b>11</b>             | 0                | 0,116            | 0,218030988      | ... | 0,386950164      |
| <b>12</b>             | 0                | 0,116000122      | 0,218031205      | ... | 0,386950511      |

Setelah diketahui keseluruhan  $E_i$ , maka dapat dicari  $\delta\alpha_i$  dengan nilai  $\gamma$  dan  $C$  yang sudah ditentukan, dengan rumus yang telah

ditunjukkan pada Persamaan (4.14). Untuk perhitungan nilai  $\delta\alpha_1$  ke 1 pada iterasi ke 1:

$$\begin{aligned}\delta\alpha_1 &= \min \left\{ \max [y(1 - E_1), -\alpha_1], C - \alpha_1 \right\} \\ &= \min \left\{ \max [0,1(1 - 0), -0], 713,35 - 0 \right\} \\ &= \min \{0,1, 713,35\} \\ &= 0,1\end{aligned}$$

Demikian seterusnya sehingga didapatkan sehingga didapatkan tabel nilai  $\delta\alpha_i$  yang ditunjukkan pada Tabel berikut.

Tabel 13.36 Tabel nilai  $\delta\alpha_i$

| $\delta\alpha_i$ | iterasi 1 | iterasi 2   | iterasi 3   | ... | iterasi 5   |
|------------------|-----------|-------------|-------------|-----|-------------|
| 1                | 0,1       | 0,091597279 | 0,083798159 | ... | 0,069906883 |
| 2                | 0,1       | 0,091599637 | 0,08380244  | ... | 0,069913932 |
| 3                | 0,1       | 0,088389879 | 0,078178847 | ... | 0,061276237 |
| 4                | 0,1       | 0,088353778 | 0,078114463 | ... | 0,061173772 |
| 5                | 0,1       | 0,0884      | 0,078196901 | ... | 0,061304984 |
| 6                | 0,1       | 0,0884      | 0,078196901 | ... | 0,061304984 |
| 7                | 0,1       | 0,091556524 | 0,083724164 | ... | 0,069785062 |
| 8                | 0,1       | 0,091558884 | 0,083728451 | ... | 0,069792126 |
| 9                | 0,1       | 0,088389728 | 0,078178578 | ... | 0,061275808 |
| 10               | 0,1       | 0,08835364  | 0,078114216 | ... | 0,061173378 |
| 11               | 0,1       | 0,0884      | 0,078196901 | ... | 0,061304984 |
| 12               | 0,1       | 0,088399988 | 0,078196879 | ... | 0,061304949 |

Sedangkan untuk memperbarui nilai  $\alpha$  dihitung dengan menggunakan Persamaan (4.16). Untuk perhitungan nilai  $\alpha$  ke 1 pada iterasi 1:

$$\begin{aligned}\alpha_1 &= \alpha_1 + \delta\alpha_1 \\ &= 0 + 0,1 = 0,1\end{aligned}$$

Berikut Tabel yang menunjukkan pembaruan  $\alpha$  dengan iterasi sebanyak 5 pada proses *multi class* level 1 menggunakan nilai  $\lambda = 0,2$  untuk perhitungan matriks *Hessian*, serta nilai  $y = 0,1$  dan nilai  $C = 713,35$  untuk proses pembaruan  $\alpha$ .

Tabel 13.37 Tabel Pembaruan Alpha

| $\alpha_i$ | iterasi 1 | iterasi 2   | iterasi 3   | ... | iterasi 5   |
|------------|-----------|-------------|-------------|-----|-------------|
| 1          | 0,1       | 0,191597279 | 0,275395438 | ... | 0,421879667 |

Tabel 13.37 Tabel Pembaruan *Alpha*

| $\alpha_i$ | iterasi 1 | iterasi 2   | iterasi 3   | ... | iterasi 5   |
|------------|-----------|-------------|-------------|-----|-------------|
| 2          | 0,1       | 0,191599637 | 0,275402078 | ... | 0,421899185 |
| 3          | 0,1       | 0,188389879 | 0,266568727 | ... | 0,397036732 |
| 4          | 0,1       | 0,188353778 | 0,266468242 | ... | 0,396747644 |
| 5          | 0,1       | 0,1884      | 0,266596901 | ... | 0,397117814 |
| 6          | 0,1       | 0,1884      | 0,266596901 | ... | 0,397117814 |
| 7          | 0,1       | 0,191556524 | 0,275280688 | ... | 0,421542376 |
| 8          | 0,1       | 0,191558884 | 0,275287335 | ... | 0,421561925 |
| 9          | 0,1       | 0,188389728 | 0,266568306 | ... | 0,397035522 |
| 10         | 0,1       | 0,18835364  | 0,266467855 | ... | 0,396746531 |
| 11         | 0,1       | 0,1884      | 0,266596901 | ... | 0,397117814 |
| 12         | 0,1       | 0,188399988 | 0,266596867 | ... | 0,397117716 |

5. Menghitung nilai bias *b*

Setelah dilakukan iterasi pembaruan nilai  $\alpha$  sebanyak 5 kali, kemudian dicari nilai  $\alpha$  terbesar dari masing-masing kelas positif dan negatif untuk digunakan saat penghitungan nilai bias. Pada studi kasus ini diketahui nilai  $\alpha$  terbesar pada kelas positif adalah nilai  $\alpha$  ke 2 dan nilai  $\alpha$  terbesar pada kelas negatif adalah nilai  $\alpha$  ke 5. Menghitung nilai bias dilakukan dengan Persamaan (4.18).

$$b = -\frac{1}{2} \sum_{i=1}^n \alpha_i y_i (K(x_i, x^+) + K(x_i, x^-)) \quad (4.18)$$

Dimana  $K(x_i, x^+)$  adalah nilai nilai kernel dari data kelas positif ke-*i* yang memiliki nilai  $\alpha$  terbesar pada kelas positif dan  $K(x_i, x^-)$  adalah nilai nilai kernel dari data kelas positif ke-*i* yang memiliki nilai  $\alpha$  terbesar pada kelas negatif. Untuk perhitungan nilai bias pada level 1:

$$\begin{aligned}
 b &= -\frac{1}{2} \sum_{i=1}^{12} \alpha_i y_i (K(x_i, x^+) + K(x_i, x^-)) \\
 &= -\frac{1}{2} \left[ \begin{array}{l} ((0,422 * 1 * (3,614E - 05 + 2,041E - 68)) + (0,422 * 1 * (1 + 9,2954E - 45))) \\ + (0,397 * -1 * (1,481E - 64 + 1,484E - 38)) + (0,397 * -1 * (1,6878E - 10 + 1,9908E - 20)) \\ + (0,397 * -1 * (9,2954E - 45 + 1)) + (0,397 * -1 * (2,7847E - 144 + 5,2925E - 65)) \\ + (0,421 * 1 * (9,3497E - 08 + 1,0213E - 55)) + (0,421 * 1 * (1,3132E - 08 + 1,2336E - 43)) \\ + (0,397 * -1 * (2,5768E - 32 + 3,4811E - 28)) + (0,397 * -1 * (2,8227E - 19 + 1,3402E - 21)) \\ + (0,397 * -1 * (8,0492E - 54 + 4,9925E - 18)) + (0,397 * -1 * (1,4815E - 64 + 1,4843E - 38)) \end{array} \right] \\
 &= -0,01239833
 \end{aligned}$$

## 6. Menghitung nilai fungsi $f(x)$

Setelah didapat nilai bias  $b$  kemudian mencari *hyperplane* dengan Persamaan (4.19) berikut.

$$f(x) = \text{sign} \left( \sum_{i=1}^n \alpha_i y_i K(x, x_i) + b \right) \quad (4.19)$$

Dengan menggunakan Persamaan (4.19) akan dihasilkan nilai yang nantinya digunakan untuk menentukan status resiko stroke dari data uji yang digunakan. Data uji yang digunakan dalam studi kasus ini ditunjukkan pada Tabel berikut.

Tabel 13.38 Tabel Data Uji yang Digunakan

| No  | Umur | Kolesterol Total | HDL  | LDL   | Trigliserida | Kelas |
|-----|------|------------------|------|-------|--------------|-------|
| 5   | 40   | 253              | 46,5 | 179,3 | 136          | 1     |
| 6   | 48   | 197              | 30,5 | 120,5 | 79           | 1     |
| 123 | 35   | 197              | 39,6 | 119,6 | 189          | 2     |
| 124 | 52   | 157              | 30,6 | 92,6  | 169          | 2     |
| 160 | 67   | 276              | 51,4 | 188,2 | 182          | 3     |
| 161 | 79   | 340              | 31,2 | 208,9 | 162          | 3     |

Pertama menghitung nilai kernel dari data uji terhadap masing-masing data latih yang digunakan dengan menggunakan Persamaan (4.17). Contoh perhitungan nilai kernel untuk data uji ke 1 terhadap data latih ke 1:

$$K(1,1) = \exp\left(\frac{-\|x - y\|^2}{2\sigma^2}\right)$$

$$= \exp\left(-\frac{(40-44)^2 + (253-174)^2 + (46,5-36,5)^2 + (179,3-123,3)^2 + (136-71)^2}{2 * (9,8294)^2}\right)$$

$$= 1,47487E - 31$$

Hasil perhitungan kernel data uji terhadap masing-masing data latih ditunjukkan pada Tabel berikut.

Tabel 13.39 Nilai Kernel Data Uji

| $K(x,y)$ | UJI 1       | UJI 2           | ... | UJI 6       |
|----------|-------------|-----------------|-----|-------------|
| 1        | 1,47487E-31 | 0,03409887<br>7 | ... | 1,4847E-100 |
| 2        | 5,67142E-30 | 2,62615E-05     | ... | 2,40036E-95 |
| 3        | 0,004108149 | 1,30785E-33     | ... | 1,09636E-19 |
| 4        | 1,30855E-20 | 2,51359E-19     | ... | 6,27989E-73 |
| 5        | 8,11143E-49 | 1,20167E-56     | ... | 2,80039E-81 |
| 6        | 2,80988E-79 | 9,2913E-159     | ... | 1,25644E-66 |
| 7        | 6,36595E-17 | 0,00207673<br>1 | ... | 7,49394E-77 |
| 8        | 1,50106E-08 | 1,16669E-05     | ... | 3,93288E-56 |
| 9        | 4,49275E-07 | 6,70989E-35     | ... | 1,05548E-31 |
| 10       | 3,15681E-14 | 2,15469E-25     | ... | 1,15095E-58 |
| 11       | 1,80234E-38 | 4,71297E-70     | ... | 1,37059E-64 |
| 12       | 1,17759E-18 | 1,30392E-67     | ... | 1,28808E-29 |

Kemudian menghitung nilai fungsi  $f(x)$  dengan Persamaan (4.19). Contoh perhitungan untuk nilai fungsi  $f(x)$  pada data uji ke 1:

$$\begin{aligned}
 f(x_5) &= \text{sign}\left(\sum_{i=1}^{12} \alpha_i y_i K(x_i, x_5) + b\right) \\
 &= \left( \begin{array}{l} ((0,4227 * 1 * 1,47487E - 31) + (0,422 * 1 * 5,67142E - 30) \\ + (0,397 * -1 * 0,004108149) + (0,397 * -1 * 1,30855E - 20) \\ + (0,397 * -1 * 8,11143E - 49) + (0,397 * -1 * 2,80988E - 79) \\ + (0,421 * 1 * 6,36595E - 17) + (0,421 * 1 * 1,50106E - 08) \\ + (0,397 * -1 * 4,49275E - 07) + (0,397 * -1 * 3,15681E - 14) \\ + (0,397 * -1 * 1,80234E - 38) + (0,397 * -1 * 1,17759E - 18) \end{array} \right) + -0,01239833 \\
 &= \text{sign}(-0,01402959) \\
 &= -1
 \end{aligned}$$

Hasil dari perhitungan nilai fungsi  $f(x)$  level 1 dari 6 data uji ditunjukkan pada Tabel berikut.

Tabel 13.40 Tabel Hasil Perhitungan Nilai Fungsi  $f(x)$  Level 1

| No Data Uji | $\sum \alpha_i y_i K(x_i, x_j) + b$ | $f(x)$ |
|-------------|-------------------------------------|--------|
| 5           | -0,01402959                         | -1     |
| 6           | 0,002878719                         | 1      |
| 123         | -0,063351219                        | -1     |
| 124         | -0,012415071                        | -1     |
| 160         | -0,013465423                        | -1     |
| 161         | -0,012398332                        | -1     |

## 7. Kembali ke langkah 2 untuk perhitungan level 2 One-Against-All

Setelah didapat nilai fungsi  $f(x)$  untuk perhitungan level 1 maka selanjutnya akan dilakukan perhitungan untuk level 2. Untuk level 2 data dengan kelas 1 tidak lagi digunakan sehingga data latih yang digunakan untuk pelatihan SVM sebanyak 8 data, dimana data dengan kelas 2 (rentan) sebagai kelas positif dan data dengan kelas 3 (mengkhawatirkan) sebagai kelas negatif. Tabel berikut menunjukkan pembagian kelas positif dan negatif untuk level 2.

Tabel 13.41 Tabel Pembagian Kelas Positif dan Negatif Level 2

| Data Latih Nomor | Kelas | y |
|------------------|-------|---|
| 119              | 2     | 1 |
| 120              | 2     | 1 |

Tabel 13.41 Tabel Pembagian Kelas Positif dan Negatif Level 2

| Data Latih Nomor | Kelas | y  |
|------------------|-------|----|
| 156              | 3     | -1 |
| 157              | 3     | -1 |
| 121              | 2     | 1  |
| 122              | 2     | 1  |
| 158              | 3     | -1 |
| 159              | 3     | -1 |

Kemudian menghitung kernel RBF dengan 8 data latih yang akan menghasilkan matriks kernel 8x8, dilanjutkan dengan melakukan *sequential learning*, menghitung bias, dan menghitung nilai fungsi  $f(x)$  untuk level 2.

Untuk hasil dari fungsi  $f(x)$  level 2 pada 6 data uji yang sama ditunjukkan pada Tabel berikut.

Tabel 13.42 Tabel Hasil Perhitungan Nilai Fungsi  $f(x)$  Level 2

| No Data Uji | $\sum \alpha_i y_i K(x, x_i) + b$ | $f(x)$ |
|-------------|-----------------------------------|--------|
| 5           | 0,001512222                       | 1      |
| 6           | -0,000169967                      | -1     |
| 123         | 0,052374101                       | 1      |
| 124         | -0,000152704                      | -1     |
| 160         | 0,000930431                       | 1      |
| 161         | -0,000169967                      | -1     |

Setelah diketahui hasil dari dua fungsi, kemudian hasilnya digabungkan untuk dilakukan prediksi kelas dari status resiko penyakit stroke yaitu 1 (normal), 2 (rentan), dan 3 (mengkhawatirkan). Hasil dari penggabungan dua fungsi ditunjukkan pada Tabel berikut.

Tabel 13.43 Tabel Hasil Penggabungan Dua Fungsi

| No Data Uji | f(x) level 1 | f(x) level 2 | Hasil               |
|-------------|--------------|--------------|---------------------|
| 5           | -1           | 1            | 2 (rentan)          |
| 6           | 1            | -1           | 1 (normal)          |
| 123         | -1           | 1            | 2 (rentan)          |
| 124         | -1           | -1           | 3 (mengkhawatirkan) |
| 160         | -1           | 1            | 2 (rentan)          |
| 161         | -1           | -1           | 3 (mengkhawatirkan) |

#### 8. Mencari nilai akurasi

Setelah didapatkan hasil dari klasifikasi kemudian dihitung nilai akurasi dari sistem. Mencari nilai akurasi dilakukan dengan melakukan perbandingan antara hasil prediksi data uji dari sistem dengan hasil prediksi dokter. Hasil dari perhitungan nilai akurasi ditunjukkan pada Tabel berikut.

Tabel 13.44 Tabel Perbandingan Hasil Sistem dengan Hasil Aktual

| No Data Uji | Hasil Sistem        | Hasil Dokter        |
|-------------|---------------------|---------------------|
| 5           | 2 (rentan)          | 1 (normal)          |
| 6           | 1 (normal)          | 1 (normal)          |
| 123         | 2 (rentan)          | 2 (rentan)          |
| 124         | 3 (mengkhawatirkan) | 2 (rentan)          |
| 160         | 2 (rentan)          | 3 (mengkhawatirkan) |

Tabel 13.44 Tabel Perbandingan Hasil Sistem dengan Hasil Aktual

| No Data Uji | Hasil Sistem        | Hasil Dokter        |
|-------------|---------------------|---------------------|
| 161         | 3 (mengkhawatirkan) | 3 (mengkhawatirkan) |

Dari tabel diatas dapat dihitung nilai akurasi dari data uji dengan menggunakan Persamaan:

$$\begin{aligned} Acc &= \frac{Benar}{Benar + Salah} \\ &= \frac{3}{3+3} \\ &= \frac{3}{6} \\ &= 0,5 \end{aligned}$$

#### Source Code 13.1 Sequential Training SVM

```
FnSVM_Squent_Non_Additive_Poly.m

%
% By: Imam Cholissodin | imamcs@ub.ac.id
% Cholissodin I., Setiawan B.D., 2013. Sentiment
% Analysis Dokumen E-Complaint Kampus
% Menggunakan Additive Selected Kernel SVM.
% Seminar Nasional Teknologi Informasi Dan
% Aplikasinya (SNATIA) .

function [w,b,index_support_vector_skenario, ...
 nilai_support_vector_skenario, ...

nilai_support_vector_skenario_dikali_kelas_skenario, ...
 skenario_train_pos_neg_index_skenario, ...
 kelas_data_skenario, ...

fitur_hist_pos_skenario,fitur_hist_neg_skenario]
= ...
```

```
FnSVM_Squent_Non_Additive_Poly(skenario_train_pos_index,...

skenario_train_neg_index,kelas_data_all,fitur_hist_pos_all,...

fitur_hist_neg_all,index_rasio,index_skenario,ratio_data_training)

% spesifikasi data train
jml_kelas_positif=length(skenario_train_pos_index);
jml_kelas_negatif=length(skenario_train_neg_index);

jml_data=jml_kelas_positif+jml_kelas_negatif;

% menggabungkan skenario_train_pos_index dan
skenario_train_neg_index
skenario_train_pos_neg_index=zeros(jml_data,1);
skenario_train_pos_neg_index(1:jml_kelas_positif,1)=...
 skenario_train_pos_index;
skenario_train_pos_neg_index(jml_kelas_positif+1:jml_data,1)=...
 skenario_train_neg_index;

skenario_train_pos_neg_index;
length(skenario_train_pos_neg_index);
skenario_train_pos_neg_index_skenario=skenario_train_pos_neg_index;

% parameter yang digunakan
y=zeros(jml_data,1);
y(1:jml_kelas_positif,1)=kelas_data_all(skenario_train_pos_index);
y(jml_kelas_positif+1:jml_data,1)=kelas_data_all(skenario_train_neg_index);

y_positif=find(y==1);
y_negatif=find(y==-1);

%y=kelas_data_all(skenario_train_pos_neg_index);
%y
%length(y)
```

```
kelas_data_skenario=y;

%y =
union(kelas_data_all(skenario_train_pos_index), .
..
% kelas_data_all(skenario_train_neg_index))'
fitur_hist_pos=fitur_hist_pos_all(skenario_train
_pos_neg_index);
fitur_hist_neg=fitur_hist_neg_all(skenario_train
_pos_neg_index);

fitur_hist_pos_skenario=fitur_hist_pos;
fitur_hist_neg_skenario=fitur_hist_neg;

x1_aksen=fitur_hist_pos;
x2_aksen=fitur_hist_neg;

 %[fitur_hist_pos fitur_hist_neg y]
%length(fitur_hist_pos)
%length(fitur_hist_neg)

%% svm train
 %% ploting data dan memberikan label berbeda
untuk setiap kelas
 %imageHandle = figure('Visible','off');
 %axesHdl = axes('parent', imageHandle);

 imageHandle = figure('Units', 'points',...
 'Position', [250 40 475 450]);
 axesHdl = axes('parent', imageHandle);
 hold on;

 % xlim([-1 2*max(max([fitur_hist_pos
 fitur_hist_neg]))])
 % ylim([-1 2*max(max([fitur_hist_pos
 fitur_hist_neg]))])

 %xlim([-2
 2*max(max(fitur_hist_pos,fitur_hist_neg))])
 %ylim([-2
 2*max(max(fitur_hist_pos,fitur_hist_neg))])

 %xlim([-1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_
 neg)))
 1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_
 neg))))])
```

```
%ylim([-
1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_
neg)))
1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_
neg))))])

 xlim([-1
1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_
neg))))])
 ylim([-1
1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_
neg))))])

 %xlim([-
1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_
neg)))
1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_
neg))))])
 %ylim([-
1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_
neg)))
1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_
neg))))])

 %hTitle = title (strcat('Visualisasi
Klasifikasi Dokumen E-Complaint SVM',...

%'Rasio=',num2str(ratio_data_training(index_rasi
o)), ' Skenario ke-',...
%num2str(index_skenario));
 hTitle = title (strcat('Training 2D SVM Non
Additive',...
 '
 Rasio=',num2str(ratio_data_training(index_rasio)
), ' Skenario Ke-',...
 num2str(index_skenario),' K(x,y)=(x.y)^2'));
 hXLabel = xlabel('Fitur 1 (X1) ') ;
 hYLabel = ylabel('Fitur 2 (X2) ') ;
 % hText = text(70, -70,...
 % sprintf('\\it{f(x) = sign(w.x + b)}'));

 % menggabungkan x1_akses dan x2_akses menjadi
 x_akses_positif & x_akses_negatif
 x_akses_positif=zeros(jml_kelas_positif,2);
 x_akses_positif(:,1)=x1_akses(y_positif);
 x_akses_positif(:,2)=x2_akses(y_positif);
```

```
x_aksen_negatif=zeros(jml_kelas_negatif,2);
x_aksen_negatif(:,1)=x1_aksen(y_negatif);
x_aksen_negatif(:,2)=x2_aksen(y_negatif);

%%
hData_positif = plot(x1_aksen(y_positif),
x2_aksen(y_positif),'+', 'LineWidth',2,'MarkerEdgeColor','g');
hData_negatif = plot(x1_aksen(y_negatif),
x2_aksen(y_negatif),'o', 'LineWidth',2,'MarkerEdgeColor','r');

x1=x1_aksen;
x2=x2_aksen;

% menggabungkan x1 dan x2 menjadi x
x_=zeros(jml_data,2);
x_(:,1)=x1;
x_(:,2)=x2;

% menampilkan vektor space
vektor_space_=zeros(jml_data,3);
vektor_space_(:,1) = x1;
vektor_space_(:,2) = x2;
vektor_space_(:,3) = y;

vektor_space_;

%% svmSequent
% inisialisasi nilai hi (sebagai nilai alfa)
hi=zeros(jml_data,1);

% hitung matrik Di (sebagai matrik hessian)
base matrix concept
tic
index_gen=(1:jml_data)';
index_i=padarray(index_gen,[0,jml_data-1],'replicate','post');
index_j=index_i';
yiyj=y(index_i).*y(index_j);

% menampilkan banyak dimensi
size(x_,2)

% x_fitur_ke_i=x_(:,1)
% x_fitur_ke_i=x_(:,2)
% x_fitur_ke_i=x_(:,3)
```

```
% % modified x_ asli dengan K1(x,y)=(x.y)^2
% % dari 2D ke 3D
% x_modif_=zeros(jml_data,3);
% x_modif_(:,1)=power(x_(:,1),2);
%
% x_modif_(:,2)=power(ones(jml_data,1).*2,0.5).*x_(:,1).*x_(:,2);
% x_modif_(:,3)=power(x_(:,2),2);
% %x_modif_
% %pause(10);

% menghitung hasil K1(x,y) per kolom fitur
% =====
% menggunakan kernel Linier K1(x,y)=x.y
for fitur_ke_i=1:size(x_,2)
 if(fitur_ke_i==3)
 x_fitur_ke_i=ones(jml_data,1);
 else
 x_fitur_ke_i=x_(:,fitur_ke_i);
 end
 phi_euler_i_phi_euler_j{fitur_ke_i}=...
 x_fitur_ke_i(index_i).*x_fitur_ke_i(index_j);
end

%size(phi_euler_i_phi_euler_j{1})
%pause(10);

% menggunakan kernel polynomial
K2(x,y)=(x.y)^2
%x_p=(x1^2 sqrt(2)*x1*x2 x2^2)'
%for fitur_ke_i=1:size(x_,2)

%phi_euler_i_phi_euler_j_poly1{fitur_ke_i}=...

%power(phi_euler_i_phi_euler_j{fitur_ke_i},2);
%end

% menggunakan kernel polynomial
K3(x,y)=(x.y + 1)^2
%for fitur_ke_i=1:size(x_,2)

%phi_euler_i_phi_euler_j_poly2{fitur_ke_i}=...

%power(phi_euler_i_phi_euler_j{fitur_ke_i}+1,2);
%end
```

```
% addtive kernel linier & poly 1
(K2(x,y)=(x.y)^2)

%
% ===== end menghitung K(x,y)
=====

% inisialisasi matrik hasil dot product per
dua data
% tanpa fungsi kernel

sum_phi_euler_i_phi_euler_j=zeros(jml_data,jml_d
ata);
%jml_data
for fitur_ke_i=1:size(x_,2)
 %for fitur_ke_i=1:size(x_,2)+1 % inject
dimensi ke-3
 sum_phi_euler_i_phi_euler_j=...
 sum_phi_euler_i_phi_euler_j...
 +phi_euler_i_phi_euler_j{fitur_ke_i};
end

%size(sum_phi_euler_i_phi_euler_j)

%pause(10)

% mendapatkan matrik kernel untuk
mendapatkan additive kernel
% menggunakan kernel Linier K1(x,y)=x.y
%k1_x_y=sum_phi_euler_i_phi_euler_j;

% kernel poly1
% menggunakan kernel polynomial
K2(x,y)=(x.y)^2
% x_p=(x1^2 sqrt(2)*x1*x2 x2^2)'
k2_x_y=power(sum_phi_euler_i_phi_euler_j,2);

% kernel poly2
% menggunakan kernel polynomial
K3(x,y)=(x.y + 1)^2

%k3_x_y=power(sum_phi_euler_i_phi_euler_j+c_2,2)
;
```

```
% mereplace sum_phi_euler_i_phi_euler_j
dengan additive kernel

%sum_phi_euler_i_phi_euler_j=k1_x_y+k2_x_y+k3_x_
y;

%if(mode_add==1) % mode tunggal
% sum_phi_euler_i_phi_euler_j=k1_x_y;
%elseif(mode_add==2)
 sum_phi_euler_i_phi_euler_j=k2_x_y; %
kernel polynomial
%elseif(mode_add==3)
% sum_phi_euler_i_phi_euler_j=k3_x_y;
%elseif(mode_add==4)
%
sum_phi_euler_i_phi_euler_j=k2_x_y+k3_x_y;
%end

size(sum_phi_euler_i_phi_euler_j)

% generate nilai matrik lamda
lamda = [0.0 0.1 0.5 1.0 2.0 5.0];

lamda_kuadrat=power(lamda(6)*ones(jml_data,jml_d
ata),2);

Dij=yiyj.* (sum_phi_euler_i_phi_euler_j+lamda_kua
drat);

%Dij;
%size(Dij)
toc

% penskalaan Dij
%Dij=Dij/power(10,6);

% % hitung matrik Di (sebagai matrik hessian)
base looping
% tic
% for i=1:jml_data
% for j=1:jml_data
%
Dij_base_loop(i,j)=y(i)*y(j)*(sum(x_(i,:).*x_(j,
:))+power(lamda(1),2));
% end
% end
```

```
%
% toc

% set nilai parameter
%learning_rate=0.19;
%learning_rate=0.00017444;

learning_rate=0.1/max(max(Dij.*eye(jml_data,jml_
data))); % base paper vijayakumar

%learning_rate=0.2/max(max(Dij.*eye(jml_data,jml_
data))); % base paper vijayakumar

%learning_rate=1.9/max(max(Dij.*eye(jml_data,jml_
data))); % base paper vijayakumar
%learning_rate=1.9/max(max(Dij))
%learning_rate=0.1/max(max(Dij)) % gamma
variable
C_param=1; % slack variable

epsilon = 1e-8;
iter=0;
stop_iter=1e-1;
while(iter<100000)
%while(~stop_iter & iter<150)
%while(stop_iter>epsilon)
% Hitung Ei
%Ei=sum(hi(index_j).*Dij,2);
Ei=(hi'*Dij)';

% Hitung Shi
Shi=min(max(learning_rate*(1-Ei),-
hi),C_param-hi);

% disp('learning_rate | Ei | 1-Ei |
learning_rate*(1-Ei) | -hi | Max')
% [learning_rate*ones(jml_data,1) Ei 1-Ei
learning_rate*(1-Ei) -hi max(learning_rate*(1-
Ei),-hi)]
% [max(learning_rate*(1-Ei),-hi) C_param-hi
min(max(learning_rate*(1-Ei),-hi),C_param-hi) hi
hi+Shi y]
% iter
%pause(1)

% Update hi (nilai alfa)
hi=hi+Shi;
```

```
iter=iter+1;
stop_iter=max(abs(Shi))
%stop_iter = (max(abs(Shi)) < epsilon);
end

[hi y]

hiy = hi.*y;
%index_support_vector = find(hi >
C_param/jml_data)
%index_support_vector = find(hi >
0.0001*C_param)
%index_support_vector = find(hi > 0.0008)
%index_support_vector = find(hi > 0.001)
index_support_vector = find(hi > 0)

byk_support_vector=length(index_support_vector)

index_support_vector_skenario=skenario_train_pos_n
eg_index(index_support_vector);

% nilai alfa dari support vector

nilai_support_vector_skenario=hi(index_support_v
ector);

nilai_support_vector_skenario_dikali_kelas_skena
rio=hiy(index_support_vector);

% mendapatkan nilai W menggunakan konsep
Linear Kernel
%for fitur_ke_i=1:size(x_,2)
% x_fitur_ke_i=x(:,fitur_ke_i);
%
w(fitur_ke_i)=sum(double(x_fitur_ke_i(index_supp
ort_vector)).*double(hiy(index_support_vector)))
;
%end

% menentukan matrik yi (nilai fitur)
yi_full=zeros(byk_support_vector,2);

yi_full(:,1)=fitur_hist_pos_all(index_support_ve
ctor_skenario);
```

```
yi_full(:,2)=fitur_hist_neg_all(index_support_ve
ctor_skenario);

 % mendapatkan nilai W menggunakan konsep Non
 Additive Kernel Poly
 % nilai 3 menyatakan dimensi tinggi dari hasil
 pemetaan data 2 dimensi
 w=zeros(3,1);
 for ii=1:byk_support_vector

 % data x masuk
 data_x=yi_full(ii,:);

 % hasil mapping data dengan (x.y)^2

 phi_1_x=zeros(3,1);
 phi_1_x(1,1)=power(data_x(1),2);

 phi_1_x(2,1)=power(2,0.5)*data_x(1)*data_x(2);
 phi_1_x(3,1)=power(data_x(2),2);

 %phi_x_additive=phi_1_x;

 w=w+(nilai_support_vector_skenario_dikali_kelas_
 skenario(ii).*phi_1_x)

 data_x=[];
 phi_1_x=[];
 end

 % mendapatkan nilai b
 index_x_positif=find(hiy>0);
 index_x_negatif=find(hiy<0);
 %hiy
 %index_x_positif=find(hiy>0.0008)
 %index_x_negatif=find(hiy<-0.0008)

 %b=-
 0.5*(sum(w.*x_(index_x_positif(1),:))+sum(w.*x_(
 index_x_negatif(1),:)))
 x_index_x_positif=x_(index_x_positif(1),:);
 % hasil mapping data dengan (x.y)^2
 % c_konstanta = 0
 phi_1_x=zeros(3,1);
 phi_1_x(1,1)=power(x_index_x_positif(1),2);
```

```
phi_1_x(2,1)=power(2,0.5)*x_index_x_positif(1)*x
_index_x_positif(2);
phi_1_x(3,1)=power(x_index_x_positif(2),2);

phi_x_index_x_positif=phi_1_x;

x_index_x_negatif=x_(index_x_negatif(1),:);
% hasil mapping data dengan (x.y)^2
phi_1_x=zeros(3,1);
phi_1_x(1,1)=power(x_index_x_negatif(1),2);

phi_1_x(2,1)=power(2,0.5)*x_index_x_negatif(1)*x
_index_x_negatif(2);
phi_1_x(3,1)=power(x_index_x_negatif(2),2);

phi_x_index_x_negatif=phi_1_x;

b=-
0.5*(sum(w.*phi_x_index_x_positif)+sum(w.*phi_x_
index_x_negatif))

phi_1_x=[];
phi_x_index_x_positif=[];
phi_x_index_x_negatif=[];

%b=4.5;

% % membuat garis hyperplane base svmSeq
% ii=1;
% i_batas_bawah=-
2*max(max(abs(fitur_hist_pos),abs(fitur_hist_neg
)));
%
i_batas_atas=2*max(max(abs(fitur_hist_pos),abs(f
itur_hist_neg)));
%
for i=i_batas_bawah:i_batas_atas
x_titik_svmSeq(ii)=i;
%
%
% cara 1
% y_hyperplane(ii)=(-b-
(w(1)*x_hyperplane(ii))/w(2);
%
%
% cara 2
% y_titik_svmSeq(ii)=((-w(1)*x_titik_svmSeq(ii))-b)/w(2);
%
```

```
% ii=ii+1;
%
% end

 % membuat garis hyperplane base Additive
Kernel
% disp('Membuat garis hyperplane base Additive
Kernel: ');
%
% x_titik_svmSeq=[];
% y_titik_svmSeq=[];
% counter_base_additive=0;
% for ii_x=0:0.1:100
% for ii_y=0:0.1:100
% x_candidat_plot_hyperplane=[ii_x ii_y];
% % menghitung fn_uji
%
% k2_x_y=power(sum(repmat(x_candidat_plot_hyperpla
ne,[byk_support_vector 1]).*yi_full,2),2);
%
k3_x_y=power(sum(repmat(x_candidat_plot_hyperpla
ne,[byk_support_vector 1]).*yi_full,2)+1,2);
%
% k_additive=k2_x_y+k3_x_y;
% %size(k_additive)
%
fn_uji_additive=sum(k_additive.*nilai_support_ve
ctor_skenario_dikali_kelas_skenario)+b;
%
% x_candidat_plot_hyperplane=[];
% k2_x_y=[];
% k3_x_y=[];
% k_additive=[];
%
% if(x_candidat_plot_hyperplane<0.5 &
x_candidat_plot_hyperplane>-0.5)
%
counter_base_additive=counter_base_additive+1;
%
x_titik_svmSeq(counter_base_additive)=ii_x;
%
y_titik_svmSeq(counter_base_additive)=ii_y;
%
fn_uji_additive=[];
%
end
%
end

%
size(x_titik_svmSeq)
size(y_titik_svmSeq)
```

```
% membuat garis hyperplane base Non Additive
Kernel Polynomial dengan rumus
% (cara lain) f(x)=w.p_x+b
% solving dengan rumus abc
ii=1;
i_batas_bawah=-
2*max(max(abs(fitur_hist_pos),abs(fitur_hist_neg
)));
i_batas_atas=2*max(max(abs(fitur_hist_pos),abs(f
itur_hist_neg)));
for i=i_batas_bawah:i_batas_atas
x_titik_svmSeq(ii)=i

% menentukan nilai a,b dan c dengan c_1=0
dan c_2=1 (static)
%a_abc=2*w(3)

%b_abc=(2*power(2,0.5)*w(2)*x_titik_svmSeq(ii))+(
power(2,0.5)*w(5))

%c_abc=(2*w(1)*power(x_titik_svmSeq(ii),2))+(pow
er(2,0.5)*w(4)*...
% x_titik_svmSeq(ii))+w(6)+b

% menentukan nilai a,b dan c dengan c_1=0
dan d=2(static)
% K(x,y)=(x.y)^2 kernel polynomial
a_abc=w(3);
b_abc=power(2,0.5)*w(2)*x_titik_svmSeq(ii)
c_abc=w(1)*power(x_titik_svmSeq(ii),2)+b
% x_titik_svmSeq(ii))+w(6)+b

% menentukan nilai a,b dan c dengan c_1 dan
c_2 dinamis
%a_abc=2*w(3)

%b_abc=(2*power(2,0.5)*w(2)*x_titik_svmSeq(ii))+...
%
((power(c_1,0.5)+power(c_2,0.5))*power(2,0.5)*w(
5))

%c_abc=(2*w(1)*power(x_titik_svmSeq(ii),2))+...
```

```
%
((power(c_1,0.5)+power(c_2,0.5))*power(2,0.5)*w(4)*...
% x_titik_svmSeq(ii))+((c_1+c_2)*w(6))+b

% cek nilai diskriminan
nilai_Diskriminan=power(power(b_abc,2)-
(4*a_abc*c_abc),0.5)

% cara 2
y_titik_svmSeq_temp1=(-
b_abc+(power(power(b_abc,2)-
(4*a_abc*c_abc),0.5)))/(2*a_abc);
y_titik_svmSeq_temp2=(-b_abc-
(power(power(b_abc,2)-
(4*a_abc*c_abc),0.5)))/(2*a_abc);
%x_titik_svmSeq(ii+1)=x_titik_svmSeq(ii);
%y_titik_svmSeq(ii+1)=(-b_abc-
(power(power(b_abc,2)-
(4*a_abc*c_abc),0.5)))/(2*a_abc)

if(y_titik_svmSeq_temp1>y_titik_svmSeq_temp2)
 y_titik_svmSeq(ii)=y_titik_svmSeq_temp1
else
 y_titik_svmSeq(ii)=y_titik_svmSeq_temp2
end

%ii=ii+2;
ii=ii+1;
end

[x_titik_svmSeq' y_titik_svmSeq']

hData_hyperplane_svmSeq=line(x_titik_svmSeq',
y_titik_svmSeq',...
 'LineWidth',2,'Color',[0 0 0]);

%pause(100)

%% End svmSequent

%hData_positif_support_vektor=[];
%hData_negatif_support_vektor=[];

index_support_vektor_positif=index_support_vecto
r(...
```

```
find(index_support_vector<(jml_kelas_positif+1))
);

index_support_vektor_negatif=index_support_vektor...
 find(index_support_vector>jml_kelas_positif));

x_aksesi_positif_support_vektor=x_(index_support_
vektor_positif,:);
x_aksesi_negatif_support_vektor=x_(index_support_
vektor_negatif,:);

hData_positif_support_vektor=...
plot(x_aksesi_positif_support_vektor...
(:,1),
x_aksesi_positif_support_vektor(:,2),'d','LineWidth',...
2);
hData_negatif_support_vektor=...
plot(x_aksesi_negatif_support_vektor...
(:,1),
x_aksesi_negatif_support_vektor(:,2),'.');

hLegend = legend(...
[hData_positif, hData_negatif,
hData_positif_support_vektor,
hData_negatif_support_vektor],...
'Kelas Positif (\it{x}^+)',...
'Kelas Negatif (\it{x}^-)',...
'Support Vector Dari Kelas Positif
(\it{SV}^+)',...
'Support Vector Dari Kelas Negatif (\it{SV}^-)
)', ...
'location', 'NorthWest');

set(gca, ...
'Box' , 'off' , ...
'TickDir' , 'out' , ...
'TickLength', [.02.02] , ...
'XMinorTick', 'on' , ...
'YMinorTick', 'on' , ...
'YGrid' , 'on' , ...
'XGrid' , 'on' , ...
'XColor' , [.3.3.3], ...
'YColor' , [.3.3.3], ...
'LineWidth' , 1);
```

```
%hText = text(10, 2, ...
%sprintf('halo...'));
%hText = text(10, 2, ...
%sprintf('\\it{f(x) = sign(w.x + b)}'));

xhText=10;
yhText=2;
hText = text(xhText, yhText+4, ...
sprintf('%s','{f(x)=sign(a*x^2 + b*x2 +
c)}'));
hText = text(xhText, yhText+2, ...
sprintf('%s %.6f %s %.6f %s','{a = ',...
double(w(3)),', b =
',double(power(2,0.5)*w(2)),',x1}');
if(b>=0)
hText = text(xhText, yhText, ...
sprintf('%s %.6f %s %.6f %s','{c =
',double(w(1)),',x1^2 + ',...
double(abs(b)),'}'));
else
hText = text(xhText, yhText, ...
sprintf('%s %.6f %s %.6f %s','{c =
',double(w(1)),',x1^2 - ',...
double(abs(b)),'}'));
end

% if(b<0)
% if(w(2)<0)
% hText = text(xhText, yhText, ...
% sprintf('%s %d %s %d %s %d
%{f(x)=sign(',...
% double(w(1)),',x1 -
} ',abs(double(w(2))),',x2 - ',abs(b),')'));
% else
% hText = text(xhText, yhText, ...
% sprintf('%s %d %s %d %s %d
%{f(x)=sign(',double(w(1)),',x1 +
} ',abs(double(w(2))),',x2 - ',abs(b),')'));
% end
% else
% if(w(2)<0)
% hText = text(xhText, yhText, ...
% sprintf('%s %d %s %d %s %d
%{f(x)=sign(',double(w(1)),',x1 -
} ',abs(double(w(2))),',x2 + ',abs(b),')'));
% else
% hText = text(xhText, yhText, ...
```

```
% sprintf('%d %d %s %d %s %d
%s', '{f(x)=sign(', double(w(1)), 'x1 +
}', abs(double(w(2))), 'x2 + ', abs(b), ')'));
% end
% end

set(gca ,...
 'FontName' , 'Helvetica');
set([hTitle, hXLabel, hYLabel, hText],...
 'FontName' , 'AvantGarde');
set([hLegend, gca] ,...
 'FontSize' , 8);
set([hXLabel, hYLabel, hText] ,...
 'FontSize' , 10);
set(hTitle ,...
 'FontSize' , 12 ,...
 'FontWeight' , 'bold');

% set(gca, ...
% 'Box' , 'off' ,...
% 'TickDir' , 'out' ,...
% 'TickLength' , [.02.02] ,...
% 'XMinorTick' , 'on' ,...
% 'YMinorTick' , 'on' ,...
% 'YGrid' , 'on' ,...
% 'XColor' , [.3.3.3] ,...
% 'YColor' , [.3.3.3] ,...
% 'LineWidth' , 1);

%
plot(x1_aksen(y_positif),x2_aksen(y_positif),'+' ,...
% x1_aksen(y_negatif),x2_aksen(y_negatif),'o')

%plot_svm=getframe();

nama_save_grafik=sprintf(strcat('Training 2D SVM
Non Additive',...
'
Rasio=',num2str(ratio_data_training(index_rasio)
), ' Skenario Ke-',...
 num2str(index_skenario),' K(x,y)=(x.y)^2'));

%
% menyimpan grafik svm
dialog outputPath = cd;
dialog.outputFilename = nama_save_grafik;
```

```
ext = '.png';
print(['-f',int2str(imageHandle)],'-
dpng',fullfile(dialog.outputPath,[dialog.outputF
ilename,ext]))
%imwrite(plot_svm,nama_save_grafik,'png')

%print -depsc2 Training_Dokument_Using_SVM.eps

hold off;

% menggambar tiga dimensi dari mapping data pada
kernel polynomial
% k(x,y)=(x.y)^2

% menghimpun titik data train semua fitur
% modified x_ asli dengan K1(x,y)=(x.y)^2
% dari 2D ke 3D
x_modif_=zeros(jml_data,3);
x_modif_(:,1)=power(x_(:,1),2);
x_modif_(:,2)=power(ones(jml_data,1).*2,0.5).*x_(:,1).*x_(:,2);
x_modif_(:,3)=power(x_(:,2),2);

X_3D=x_modif_(:,1);
Y_3D=x_modif_(:,2);
Z_3D=x_modif_(:,3);

imageHandle2 = figure('Units', 'points',...
 'Position', [250 40 475 450]);
axesHdl2 = axes('parent', imageHandle2);
hold on;
%S = repmat([1.75.5]*10,prod(size(X_3D)),1);
%C = repmat([1 2 3],prod(size(X_3D)),1);

%scatter3(X_3D(:,Y_3D(:,Z_3D(:,,'filled')),
view(-60,60)

%S = repmat([1.75.5]*10,prod(size(x)),1);
%C = repmat([1 2 3],prod(size(x)),1);
%scatter3(X(:,Y(:,Z(:,S(:,C(:,,'filled')),
view(-60,60)

xlim([-0.25*max(max(abs(x_modif_)))
1.25*max(max(abs(x_modif_))))])
ylim([-0.25*max(max(abs(x_modif_)))
1.25*max(max(abs(x_modif_))))])
```

```
zlim([-0.25*max(max(abs(x_modif_)))
1.25*max(max(abs(x_modif_))))]

%xlim([-1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_neg)))
1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_neg))))])
%ylim([-1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_neg)))
1.25*max(max(abs(fitur_hist_pos),abs(fitur_hist_neg))))])

%hTitle = title (strcat('Visualisasi Klasifikasi Dokumen E-Complaint SVM',...
 '%Rasio=',num2str(ratio_data_training(index_rasio)), ' Skenario ke-',...
 %num2str(index_skenario)));
hTitle = title (strcat('Training 3D SVM Non Additive',...
 '
Rasio=',num2str(ratio_data_training(index_rasio)), ' Skenario Ke-',...
 num2str(index_skenario),' K(x,y)=(x.y)^2'));
hXLabel = xlabel('Fitur 1 (X1^2)');
hYLabel = ylabel('Fitur 2 (sqrt(2)*X1*X2)');
hZLabel = zlabel('Fitur 3 (X2^2)');
%
hData_positif = plot(x1_aksen(y_positif),
x2_aksen(y_positif),'+', 'LineWidth',2,'MarkerEdgeColor','g');
hData_negatif = plot(x1_aksen(y_negatif),
x2_aksen(y_negatif),'o', 'LineWidth',2,'MarkerEdgeColor','r');

x_modif_hData_positif=zeros(length(x1_aksen(y_positif)),3);
x_modif_hData_positif(:,1)=power(x1_aksen(y_positif),2);
x_modif_hData_positif(:,2)=power(ones(length(x1_aksen(y_positif)),1).*2,0.5).*x1_aksen(y_positif).*x2_aksen(y_positif);
x_modif_hData_positif(:,3)=power(x2_aksen(y_positif),2);
```

```
x_modif_hData_negatif=zeros(length(x1_aksen(y_nega
tif)),3);
x_modif_hData_negatif(:,1)=power(x1_aksen(y_nega
tif),2);
x_modif_hData_negatif(:,2)=power(ones(length(x1_
aksen(y_negatif))),1).*2,0.5).*x1_aksen(y_negatif
).*x2_aksen(y_negatif);
x_modif_hData_negatif(:,3)=power(x2_aksen(y_nega
tif),2);

X_3D_hData_positif=x_modif_hData_positif(:,1);
Y_3D_hData_positif=x_modif_hData_positif(:,2);
Z_3D_hData_positif=x_modif_hData_positif(:,3);

X_3D_hData_negatif=x_modif_hData_negatif(:,1);
Y_3D_hData_negatif=x_modif_hData_negatif(:,2);
Z_3D_hData_negatif=x_modif_hData_negatif(:,3);

%hData_positif3d=scatter3(X_3D_hData_positif(:),
Y_3D_hData_positif(:),Z_3D_hData_positif(:),'fil
led','g','o','LineWidth',2,'MarkerEdgeColor','g'
), view(-60,60)
%hData_negatif3d=scatter3(X_3D_hData_negatif(:),
Y_3D_hData_negatif(:),Z_3D_hData_negatif(:),'fil
led','r','o','LineWidth',2,'MarkerEdgeColor','r'
), view(-60,60)

hData_positif3d=scatter3(X_3D_hData_positif(:),Y
_3D_hData_positif(:),Z_3D_hData_positif(:),'+','
LineWidth',2,'MarkerEdgeColor','g'), view(-
20,76)
hData_negatif3d=scatter3(X_3D_hData_negatif(:),Y
_3D_hData_negatif(:),Z_3D_hData_negatif(:),'o','
LineWidth',2,'MarkerEdgeColor','r'), view(-
20,76)
%
hData_positif3d=scatter3(X_3D_hData_positif(:),Y
_3D_hData_positif(:),Z_3D_hData_positif(:),'+','
LineWidth',2,'MarkerEdgeColor','g'), view(-
11,86)
%
hData_negatif3d=scatter3(X_3D_hData_negatif(:),Y
_3D_hData_negatif(:),Z_3D_hData_negatif(:),'o','
LineWidth',2,'MarkerEdgeColor','r'), view(-
11,86)
```

```
%hData_positif_support_vektor=[];
%hData_negatif_support_vektor [];

% hData_positif_support_vektor=...
% plot(x_akses_positif_support_vektor...
% (:,1),
x_akses_positif_support_vektor(:,2),'d','LineWidth',2);
% hData_negatif_support_vektor=...
% plot(x_akses_negatif_support_vektor...
% (:,1),
x_akses_negatif_support_vektor(:,2),'.');

x_modif_hData_positif_sv=zeros(length(x_akses_positif_support_vektor(:,1)),3);
x_modif_hData_positif_sv(:,1)=power(x_akses_positif_support_vektor(:,1),2);
x_modif_hData_positif_sv(:,2)=power(ones(length(x_akses_positif_support_vektor(:,1)),1).*2,0.5).*x_akses_positif_support_vektor(:,1).*x_akses_positif_support_vektor(:,2);
x_modif_hData_positif_sv(:,3)=power(x_akses_positif_support_vektor(:,2),2);

x_modif_hData_negatif_sv=zeros(length(x_akses_negatif_support_vektor(:,1)),3);
x_modif_hData_negatif_sv(:,1)=power(x1_akses(y_negatif),2);
x_modif_hData_negatif_sv(:,2)=power(ones(length(x_akses_negatif_support_vektor(:,1)),1).*2,0.5).*x_akses_negatif_support_vektor(:,1).*x_akses_negatif_support_vektor(:,2);
x_modif_hData_negatif_sv(:,3)=power(x_akses_negatif_support_vektor(:,2),2);

X_3D_hData_positif_sv=x_modif_hData_positif_sv(:,1);
Y_3D_hData_positif_sv=x_modif_hData_positif_sv(:,2);
Z_3D_hData_positif_sv=x_modif_hData_positif_sv(:,3);

X_3D_hData_negatif_sv=x_modif_hData_negatif_sv(:,1);
Y_3D_hData_negatif_sv=x_modif_hData_negatif_sv(:,2);
```

```
Z_3D_hData_negatif_sv=x_modif_hData_negatif_sv(:,
3);

hData_positif3d_sv=scatter3(X_3D_hData_positif_s
v(:,Y_3D_hData_positif_sv(:,Z_3D_hData_positif
_sv(:, 'd','LineWidth',2,'MarkerEdgeColor','b'),
view(-20,76)
hData_negatif3d_sv=scatter3(X_3D_hData_negatif_s
v(:,Y_3D_hData_negatif_sv(:,Z_3D_hData_negatif
_sv(:, '.', 'LineWidth',2,'MarkerEdgeColor','b'),
view(-20,76)
%
hData_positif3d_sv=scatter3(X_3D_hData_positif_s
v(:,Y_3D_hData_positif_sv(:,Z_3D_hData_positif
_sv(:, 'd','LineWidth',2,'MarkerEdgeColor','b'),
view(-11,86)
%
hData_negatif3d_sv=scatter3(X_3D_hData_negatif_s
v(:,Y_3D_hData_negatif_sv(:,Z_3D_hData_negatif
_sv(:, '.', 'LineWidth',2,'MarkerEdgeColor','b'),
view(-11,86)

% membuat hyperplane 3D berdasarkan nilai w
% generate nilai x_hyperplane3d
%x_hyperplane3d=0:3000;
%y_hyperplane3d=0:3000;
%z_hyperplane3d=-
((x_hyperplane3d.*w(1))+(b)+(y_hyperplane3d.*w(2
)))./w(3);
%hData_hyperplane_svmSeq=line(x_titik_svmSeq',
y_titik_svmSeq',...
% 'LineWidth',2,'Color',[0 0 0]);

[x_hyperplane3d,y_hyperplane3d] = meshgrid([-
1000:100:1000]);
z_hyperplane3d = ((-x_hyperplane3d.*w(1))-b-
(y_hyperplane3d.*w(2)))./w(3);
plot3(x_hyperplane3d,y_hyperplane3d,z_hyperplane
3d)
grid on

% [X,Y] = meshgrid([-2:0.1:2]);
% Z = X.*exp(-X.^2-Y.^2);
% plot3(X,Y,Z)
% grid on

% for i_gen=0:100
```

```
% c1_gen=i_gen*w(1)+b;
% for j_gen=0:100
% z_gen=(-c1_gen-(j_gen*w(2)))/w(3);
% plot3(i_gen,j_gen,z_gen);
% end
% end

%hData_hyperplane_svmSeq3d=plot3(x_hyperplane3d,
y_hyperplane3d,z_hyperplane3d)
%hData_hyperplane_svmSeq3d=scatter3(x_hyperplane
3d(:,y_hyperplane3d(:,z_hyperplane3d(:),'.','L
ineWidth',2), view(-60,60)
%grid on
hLegend = legend(...
 [hData_positif3d, hData_negatif3d,
hData_positif3d_sv, hData_negatif3d_sv],...
 'Kelas Positif (\it{x^+})' ,...
 'Kelas Negatif (\it{x^-})' ,...
 'Support Vector Dari Kelas Positif
(\it{SV^+})' ,...
 'Support Vector Dari Kelas Negatif (\it{SV^-
})' ,...
 'location', 'NorthWest');

set(gca, ...
 'Box' , 'off' ,...
 'TickDir' , 'out' ,...
 'TickLength', [.02.02] ,...
 'XMinorTick', 'on' ,...
 'YMinorTick', 'on' ,...
 'YGrid' , 'on' ,...
 'XGrid' , 'on' ,...
 'XColor' , [.3.3.3],...
 'YColor' , [.3.3.3],...
 'LineWidth' , 1);

hText=[];
xhText=-1000;
yhText=1000;
zhText=1000;

hText = text(xhText, yhText, zhText, ...
 sprintf('%s %.6f %s %.6f %s %.6f %s %.6f
%s', '{f(X,Y,Z)=sign((',double(w(1)),')X +
(' ,double(w(2)),')Y + (' ,double(w(3)),')Z +
(' ,double(b),'))}'));
```

```
% rumus_3d=sprintf('%s %.6f %s %.6f %s %.6f
%s', '{f_3d(x)=sign((',double(w(1)),')X +
(' ,double(w(2)),')Y + (' ,double(w(3)),')Z +
(' ,double(b),'))}');

%
% hFigure = figure('Color', 'w',...
Create a figure window
%
'MenuBar', 'none',...
%
'ToolBar', 'none');
%
hText = uicontrol('Parent', imageHandle2,...
%# Create a text object
%
'Style', 'text',...
%
'String', rumus_3d,...
%
'BackgroundColor', 'w',...
%
'ForegroundColor', 'r',...
%
'FontSize', 8,...
%
'FontWeight', 'bold');
%
set([hText imageHandle2], 'Pos', get(hText,
'Extent')); %# Adjust the sizes of the
%
% text and figure
%
imageData = getframe(imageHandle); %# Save the
figure as an image frame
%
delete(imageHandle);
%
textImage = imageData.cdata; %# Get the RGB
image of the text

%
surf([0 1; 0 1], [1 0; 1 0], [1 1; 0 0],...
%
'FaceColor', 'texturemap', 'CData',
textImage);
%
set(gca, 'Projection', 'perspective',
'CameraViewAngle', 45,...%
%
'CameraPosition', [0.5 -1 0.5], 'Visible',
'off');

%
hText = text(xhText, yhText+4, zhText,...%
%
sprintf('%s','{f(x)=sign(a*x2^2 + b*x2 +
c)}'));
%
hText = text(xhText, yhText+2, zhText,...%
%
sprintf('%s %.6f %s %.6f %s','{a = ',...
%
double(w(3)), ' , b =
',double(power(2,0.5)*w(2)), 'x1'}));
%
if(b>=0)
%
hText = text(xhText, yhText, zhText,...%
%
sprintf('%s %.6f %s %.6f %s','{c =
',double(w(1)), 'x1^2 + ',...
%
double(abs(b)), '}'));
%
else
```

```
% hText = text(xhText, yhText, zhText, ...
% sprintf('%s %.6f %s %.6f %s', '{c =
', double(w(1)), 'x1^2 - ', ...
% double(abs(b)), '}'));
% end

% if(b<0)
% if(w(2)<0)
% hText = text(xhText, yhText, ...
% sprintf('%s %d %s %d %s %d
%s', '{f(x)=sign(',
% double(w(1)), 'x1 -
} ', abs(double(w(2))), 'x2 - ', abs(b), ')'));
% else
% hText = text(xhText, yhText, ...
% sprintf('%s %d %s %d %s %d
%s', '{f(x)=sign(', double(w(1)), 'x1 +
} ', abs(double(w(2))), 'x2 - ', abs(b), ')'));
% end
% else
% if(w(2)<0)
% hText = text(xhText, yhText, ...
% sprintf('%s %d %s %d %s %d
%s', '{f(x)=sign(', double(w(1)), 'x1 -
} ', abs(double(w(2))), 'x2 + ', abs(b), ')'));
% else
% hText = text(xhText, yhText, ...
% sprintf('%s %d %s %d %s %d
%s', '{f(x)=sign(', double(w(1)), 'x1 +
} ', abs(double(w(2))), 'x2 + ', abs(b), ')'));
% end
% end

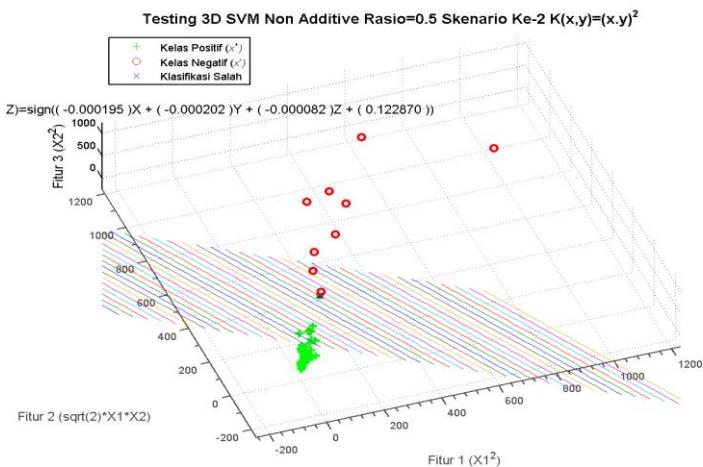
set(gca
 , ...
 'FontName' , 'Helvetica');
set([hTitle, hXLabel, hYLabel,hZLabel,
hText],...
 'FontName' , 'AvantGarde');
set([hLegend, gca]
 , ...
 'FontSize' , 8
);
set([hXLabel, hYLabel,hZLabel, hText] ,...
 'FontSize' , 10
);
set(hTitle
 , ...
 'FontSize' , 12
 , ...
 'FontWeight' , 'bold'
);
```

```
nama_save_grafik3d=sprintf(strcat('Training 3D
SVM Non Additive',...
'
Ratio=',num2str(ratio_data_training(index_rasio)
, ' Skenario Ke-',...
 num2str(index_skenario), ' K(x,y)=(x.y)^2'));

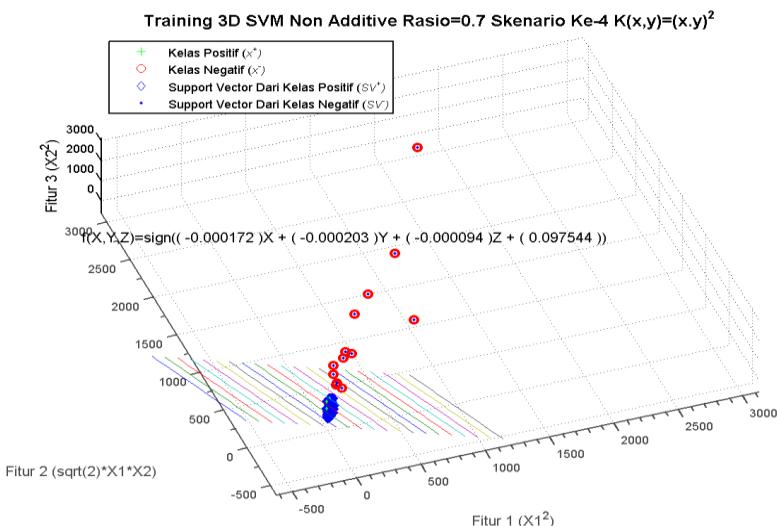
% menyimpan grafik svm
dialog.outputPath = cd;
dialog.outputFilename = nama_save_grafik3d;

ext = '.png';
print(['-f',int2str(imageHandle2)],'-
dpng',fullfile(dialog.outputPath,[dialog.outputF
ilename,ext]))
%imwrite(plot_svm,nama_save_grafik,'png')
%print -depsc2 Training_Dokument_Using_SVM.eps

hold off;
```



Gambar 13.9 Testing 3D  $K(x,y)=(x.y)^2$  Ke-1



Gambar 13.10 Testing 3D  $K(x,y)=(x.y)^2$  Ke-2

## 13.7 Klasifikasi SVM Multi Kelas (> 2 Kelas)

### 13.7.1 One Against All

Metode One-Against-All merupakan salah satu metode populer yang digunakan untuk mengurangi permasalahan klasifikasi multiclass menjadi satu set klasifikasi biner. Dengan menggunakan metode ini dibangun  $k$  buah model SVM dimana  $k$  merupakan banyaknya kelas. Misalkan terdapat permasalahan terhadap  $M$ -class dimana kita memiliki training sample  $\{x_1, y_1\}, \dots, \{x_n, y_n\}$ , dimana  $x$  adalah data dan  $y$  adalah kelas data. One against all membangun  $M$  biner SVM classifier, yang masing-masing memisahkan satu kelas dengan kelas lainnya. Setiap model pengklasifikasian ke- $i$  akan dilatih dengan keseluruhan data, untuk mencari solusi bidang pemisah terbaik dengan label positif, dan yang lain dengan label negative. Persamaan yang digunakan adalah sebagai berikut:

$$\min_{w^i, b^i, \xi^i} \frac{1}{2} (w^i)^T w^i + C \sum_t \xi^i \quad (5.1)$$

Dimana

$$(w^i)^T \phi(x_t) + b^i \geq 1 - \xi_t^i \rightarrow y_t = i,$$

$$(w^i)^T \phi(x_t) + b^i \geq 1 - \xi_t^i \rightarrow y_t = i,$$

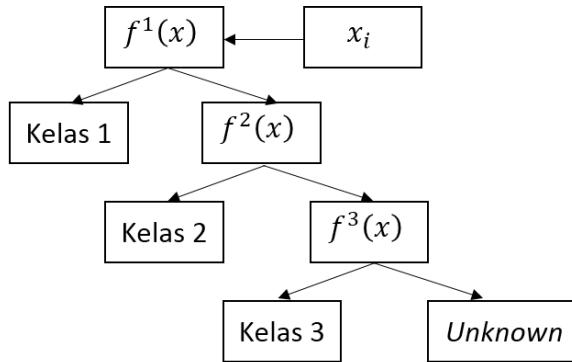
$$\xi_t^i \geq 0$$

Misalkan terdapat klasifikasi dengan 3 buah kelas. Maka akan dibangun 3 buah model SVM seperti pada Tabel berikut.

Tabel 13.45 Contoh 3 SVM Biner dengan Metode One-Against-All

| $y_i = 1$ | $y_i = -1$    | Hipotesis               |
|-----------|---------------|-------------------------|
| Kelas 1   | Bukan kelas 1 | $f^1(x) = (w^1)x + b^1$ |
| Kelas 2   | Bukan kelas 2 | $f^2(x) = (w^2)x + b^2$ |
| Kelas 3   | Bukan kelas 3 | $f^3(x) = (w^3)x + b^3$ |

Penggunaan model SVM dalam pengklasifikasian dapat dilihat pada Gambar 5.1.



Gambar 13.11 Klasifikasi dengan Metode One-Against-All

Sumber: Sembiring (2007)

### 13.7.2 One-Against-One

Metode ini membangun model klasifikasi biner berdasarkan rumus  $\frac{k(k-1)}{2}$  ( $k$  adalah jumlah kelas). Setiap model klasifikasi dilatih pada data dari dua kelas. Contohnya, terdapat permasalahan klasifikasi dengan 4 buah kelas, maka untuk pelatihan berdasarkan rumus  $\frac{k(k-1)}{2}$  akan dibangun 6 buah SVM biner seperti pada Tabel dan Persamaan yang digunakan adalah:

$$\min_{w^{ij}, b^{ij}, \xi_t^{ij}} \frac{1}{2} (w^{ij})^T w^{ij} + C \sum_t \xi_t^{ij} \quad (5.2)$$

Dimana

$$(w^{ij})^T \phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij} \rightarrow y_t = i,$$

$$(w^{ij})^T \phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij} \rightarrow y_t = i,$$

$$\xi_t^{ij} \geq 0$$

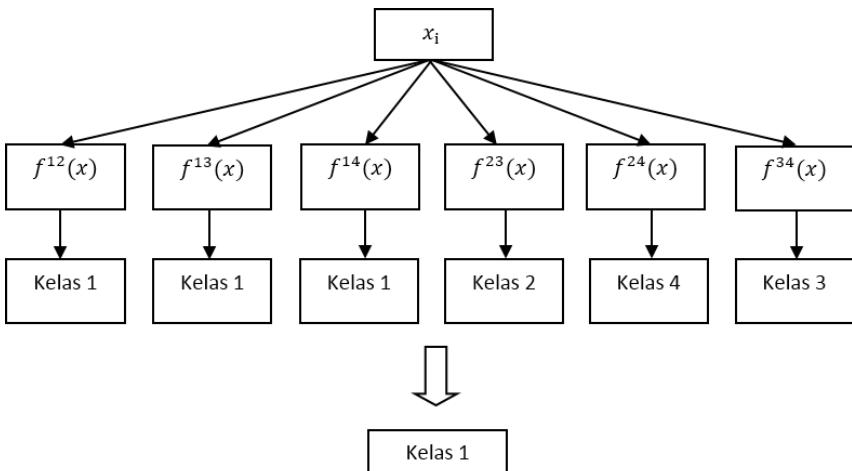
Tabel 13.46 Metode One-Against-One dengan 4 Kelas

| $y_i = 1$ | $y_i = -1$    | Hipotesis                        |
|-----------|---------------|----------------------------------|
| Kelas 1   | Bukan kelas 2 | $f^{12}(x) = (w^{12})x + b^{12}$ |
| Kelas 1   | Bukan kelas 3 | $f^{13}(x) = (w^{13})x + b^{13}$ |

| $y_i = 1$ | $y_i = -1$    | Hipotesis                        |
|-----------|---------------|----------------------------------|
| Kelas 1   | Bukan kelas 4 | $f^{14}(x) = (w^{14})x + b^{14}$ |
| Kelas 2   | Bukan kelas 3 | $f^{23}(x) = (w^{23})x + b^{23}$ |
| Kelas 2   | Bukan kelas 4 | $f^{24}(x) = (w^{24})x + b^{24}$ |
| Kelas 3   | Bukan kelas 4 | $f^{34}(x) = (w^{34})x + b^{34}$ |

Sumber: Sembiring (2007)

Penggunaan SVM dalam mengklasifikasikan data baru dapat dilihat pada Gambar 5.2.



Gambar 13.12 Klasifikasi One-Against-One untuk 4 Kelas

### 13.7.3 Binary Decision Tree SVM

Binary descision tree SVM merupakan metode yang digunakan untuk menyelesaikan permasalahan klasifikasi banyak kelas menggunakan pohon biner. Kelebihan dari metode ini terdapat dalam hal efektifitas komputasi dan tingkat akurasi yang tinggi. Prinsip dasar dari metode ini adalah membagi  $N$  kelas yang ada menjadi dua kelompok besar dan memisahkannya dengan menempatkannya ke anak kanan dan anak kiri dari pohon biner yang dibuat. Metode ini menggunakan pembagian kelas secara rekursif hingga didapatkan

pada setiap node hanya terdapat satu kelas yang merepresentasikan kategori tersebut. Kemudian proses training SVM dilakukan untuk menentukan kemana data akan dikelompokan menurut kelasnya. Ada banyak cara untuk membagi  $N$  kelas kedalam dua buah kelompok. Pemilihan cara yang tepat akan mempengaruhi kualitas dari pohon biner itu sendiri.

SVM-BDT diawali dengan memisahkan kelas yang ada kedalam dua group misalkan  $g_1$  dan  $g_2$ . Pengelompokan ini didapatkan dengan cara:

1. Menghitung *gravity center* atau titik pusat dari setiap kelas. Nilai titik pusat ini diperoleh dari rata-rata tiap fitur dari masing-masing kelas. Nilai *gravity center* dapat diperoleh dengan menggunakan persamaan berikut.

$$\overline{GC} c_i f_j = \frac{\sum_{m=1}^n x_m}{n} \quad (5.3)$$

Dimana:

$$\begin{aligned}\overline{GC} c_i f_j &= \text{Gravity center dari kelas ke-}i \text{ dan fitur ke-}j \\ n &= \text{Jumlah Kelas} \\ x_m &= \text{nilai data ke-}m \text{ pada kelas ke-}i \text{ dan fitur ke-}j\end{aligned}$$

2. Setelah diperoleh nilai *gravity center* maka akan dihitung jarak *Euclidian* dari setiap kelas terhadap kelas lainnya. Jarak *Euclidian* diperoleh dengan menggunakan persamaan:

$$d_{i,j} = \sqrt{\sum_{m=1}^n (\overline{f m}_i - \overline{f m}_j)^2} \quad (5.4)$$

Dimana:

$$\begin{aligned}d_{i,j} &= \text{Jarak antar Kelas } i \text{ dan kelas } j \\ \overline{f m}_i &= \text{rata-rata fitur ke-}m \text{ pada kelas ke-}i \\ \overline{f m}_j &= \text{rata-rata fitur ke-}m \text{ pada kelas ke-}j \\ n &= \text{jumlah fitur}\end{aligned}$$

3. Kemudian setelah dihitung jarak *Euclidian* dari setiap kelas, dua kelas yang memiliki jarak *Euclidian* terbesar masing-masing akan dipisahkan kedalam dua kelompok group menjadi  $g_1$  dan  $g_2$  misalkan kelas 1 dengan kelas 3 memiliki jarak *Euclidian* terbesar maka kelas 1 masuk kedalam  $g_1$  dan kelas 3 masuk kedalam  $g_2$ .
4. Setelah itu kelas yang memiliki jarak *Euclidian* terkecil dengan salah satu kelas dari dua kelas yang sudah dikelompokan sebelumnya maka kelas tersebut masuk

kedalam kelompok dengan jarak kelas terkecil. Misalkan kelas 2 memiliki jarak 0.2 dengan kelas 1 dan 0.5 dengan kelas 3 maka berdasarkan jarak *Euclidian* yang diperoleh kelas 2 masuk kedalam kelompok kelas 1 atau g1.

5. Proses akan terus berlanjut sampai semua kelas ditetapkan kedalam group yang sesuai. Berikut merupakan contoh ilustrasi SVM-BDT.

Misalkan terdapat dataset sebagai berikut:

Tabel 13.47 Dataset

| No | Fitur1 | Fitur2 | Fitur3 | Fitur4 | kelas |
|----|--------|--------|--------|--------|-------|
| 1  | 3      | 0      | 0      | 0      | 1     |
| 2  | 2      | 13     | 0      | 0      | 2     |
| 3  | 1      | 4      | 0      | 0      | 2     |
| 4  | 16     | 5      | 1      | 0      | 1     |
| 5  | 1      | 0      | 2      | 0      | 4     |
| 6  | 5      | 0      | 6      | 0      | 4     |
| 7  | 3      | 0      | 0      | 1      | 3     |
| 8  | 4      | 1      | 0      | 0      | 3     |

Dari contoh dataset pada Tabel 5.3, dataset terdiri dari 4 fitur dan 4 kelas. Nilai dari setiap fitur nantinya akan digunakan untuk menghitung *gravity center* dari setiap kelas.

### 1. Menghitung *gravity Center*

Berikut merupakan contoh perhitungan rata-rata *gravity center* untuk fitur 1 pada kelas 1.

$$\overline{GC\ c_1f_1} = \frac{3 + 16}{2} = 9.5$$

Dari hasil perhitungan diatas diperoleh nilai rata-rata *gravity center* untuk fitur 1 pada kelas 1 adalah 9.5. Hasil perhitungan rata-rata *gravity center* pada kelas 1 ditunjukan pada Tabel.

Tabel 13.48 *Gravity Center* Untuk Kelas 1

| Data Gravity Center Kelas 1 |        |        |        |        |       |
|-----------------------------|--------|--------|--------|--------|-------|
| No                          | Fitur1 | Fitur2 | Fitur3 | Fitur4 | Kelas |

|    |     |     |     |   |   |
|----|-----|-----|-----|---|---|
| 1  | 3   | 0   | 0   | 0 | 1 |
| 4  | 16  | 5   | 1   | 0 | 1 |
| GC | 9.5 | 2.5 | 0.5 | 0 |   |

Berdasarkan hasil perhitungan rata-rata *gravity center* dari setiap kelas maka diperoleh nilai rata-rata *gravity center* masing-masing kelas yang ditunjukkan pada Tabel.

Tabel 13.49 Data *Gravity Center* Masing-masing Kelas

| Fitur1 | Fitur2 | Fitur3 | Fitur4 | kelas |
|--------|--------|--------|--------|-------|
| 9.5    | 2.5    | 0.5    | 0      | 1     |
| 1.5    | 8.5    | 0      | 0      | 2     |
| 3.5    | 0.5    | 0      | 0.5    | 3     |
| 3      | 0      | 4      | 0      | 4     |

## 2. Menghitung *Euclidian Distance*

Setelah diperoleh nilai rata-rata *gravity center* dari setiap kelas maka langkah selanjutnya adalah menghitung *Euclidian distance* setiap kelas terhadap kelas lainnya. Contoh perhitungan jarak *Euclidian* berdasarkan data pada Tabel 5.5 adalah sebagai berikut:

$$d_{1,2} = \sqrt{(9.5 - 1.5)^2 + (2.5 - 8.5)^2 + (0.5 - 0)^2 + (0 - 0)^2}$$

$$d_{1,2} = 10.01249$$

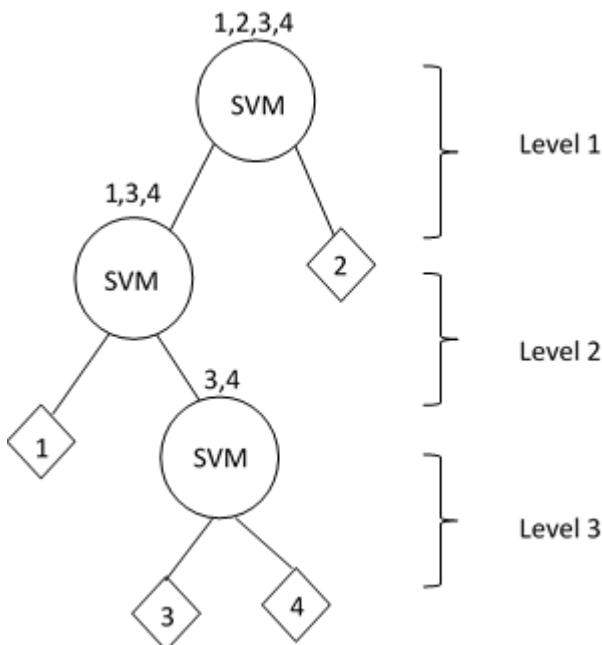
Hasil perhitungan diatas merupakan hasil perhitungan jarak *Euclidian* dari kelas 1 terhadap kelas 2. Hasil dari perhitungan jarak *Euclidian* seluruh kelas ditunjukkan pada Tabel.

Tabel 13.50 Jarak Euclidian dari Setiap Kelas

|    | c1       | c2       | c3       | c4       |
|----|----------|----------|----------|----------|
| c1 | 0        | 10.01249 | 6.363961 | 7.794229 |
| c2 | 10.01249 | 0        | 8.261356 | 9.513149 |
| c3 | 6.363961 | 8.261356 | 0        | 4.092676 |
| c4 | 7.794229 | 9.513149 | 4.092676 | 0        |

Setelah diperoleh jarak *Euclidian* dari setiap kelas maka langkah selanjutnya adalah membentuk pohon biner sesuai

dengan jarak *Euclidian* yang diperoleh. Ilustrasi Pohon Biner terhadap 4 kelas berdasarkan jarak *Euclidian* yang diperoleh ditunjukkan pada Gambar 5.3.



Gambar 13.13 Ilustrasi Pohon Biner

Berdasarkan jarak *Euclidian* yang diperoleh pada Tabel 5.6 jarak *Euclidian* terbesar adalah jarak antara c1 dengan c2 maka c2 dan c2 akan dipisahkan menjadi dua kelompok dimana c1 akan menjadi anak kiri dan c2 akan menjadi anak kanan. Kemudian langkah selanjutnya adalah melihat jarak dari kelas yang belum memiliki kelompok terhadap kelas yang sudah dipisahkan sebelumnya yaitu c1 dan c2. Untuk kelas c3 memiliki jarak paling dekat dengan c1 maka c3 masuk kedalam kelompok anak kiri bersama c1, untuk kelas c4 juga memiliki jarak terkecil dengan kelas c1 dibandingkan kelas c2 maka c4 masuk kedalam kelompok c2. Proses ini akan terus berlanjut secara rekursif sampai hanya ada satu kelas per kelompok yang mendefinisikan daun dari pohon keputusan. Setiap anak kiri akan dijadikan sebagai kelas positif dan anak kanan menjadi kelas negatif, sehingga pada level 1 akan didapatkan kelas positif {1,3,4} dan kelas negatif {2}. Pengelompokan ini nantinya akan digunakan kedalam *training SVM classifier*.

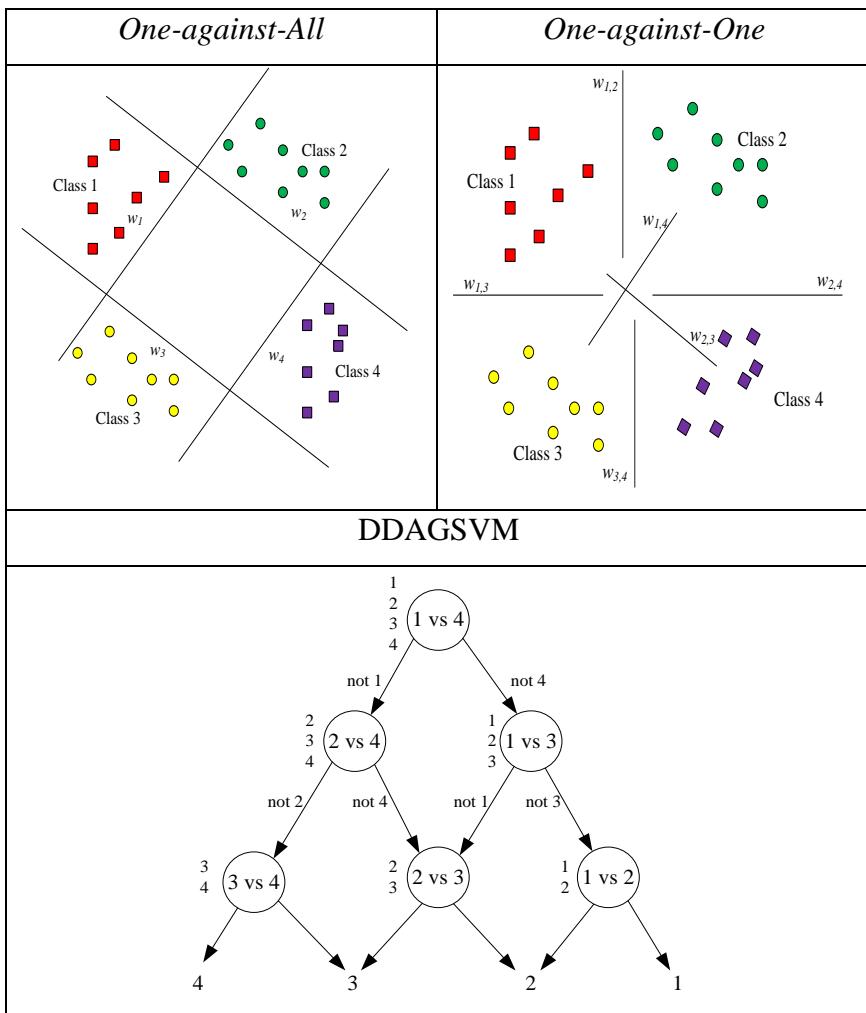
Pada *training SVM classifier* tahapan proses yang harus dilakukan adalah:

1. Menentukan algoritma training SVM, misalkan menggunakan algoritma sequential training SVM.
2. Lakukan training SVM untuk setiap level pada pohon biner. Berdasarkan Gambar 5.3 proses *training SVM* dilakukan sebanyak 3 kali karna pohon biner terdiri dari 3 level. Untuk training level 1 data yang digunakan adalah seluruh data yang masuk pada kelas positif dan negatif.

Untuk prosedur pengujian pada SVM-BDT tahapan-tahapan proses yang perlu dilakukan adalah Menghitung fungsi  $sign(f(x))$  pada setiap level pohon biner. Lakukan pengecekan terhadap nilai  $sign(f(x))$  pada setiap level jika bernilai positif maka masuk kelas +1 dan bergerak ke anak kiri, sebaliknya apabila bernilai negatif maka masuk kelas -1 dan bergerak ke anak kanan. Proses ini akan terus berlanjut secara rekursif sampai setiap data uji terkласifikasi kedalam salah satu kelas dari kelas yang tersedia.

### 13.7.4 Directed Acyclic Graph SVM (DAGSVM)

*Directed Acyclic Graph* (DAG) adalah graf yang berarah dan tidak siklik. Diperkenalkan oleh Platt, algoritma DAGSVM adalah pelatihan dengan *classifier N(N-1)/2* yang sama dengan *one-against-one*. Pada tahap pengenalan, algoritma bergantung pada *directed acyclic graph* untuk membuat keputusan. DAGSVM membuat model dari setiap pasangan kelas. Ketika sebuah model yang memisahkan kelas  $c_1$  dari kelas  $c_2$ , maka data uji termasuk ke dalam kelas  $c_1$ , dan seluruh model yang termasuk pada kelas  $c_2$  tidak akan diuji dengan data uji. Hal inilah yang membuat lebih cepat dalam melakukan pengenalan jika dibanding *one-against-one* dan *one-against-all* [MAD-09]. Gambar 5.4. menunjukkan ilustrasi klasifikasi *multi-class* dengan metode *one-against-all*, *one-against-one*, dan *Decision DAGSVM* (DDAGSVM).



Gambar 13.14 Ilustrasi klasifikasi dengan metode *One-against-One*, *One-against-All*, dan DDAGSVM

Pada *Decision DAG* (DDAG) terdapat sebuah *node* di *layer* paling atas yang disebut sebagai *root node*. Kemudian terdapat dua *node* di *layer* kedua, dan *layer* seterusnya di mana *layer* ke-*j* memiliki sebanyak *i* *node*. *Node* *i* pada *layer* *j* mengarahkan ke *node* *i* dan *i*+1 pada *layer* *j*+1, sehingga total ada sebanyak  $N(N-1)/2$  *node*. Setiap *node* mewakili satu *decision function*. Setiap *node* akan mengklasifikasikan data yang diolah ke rute kelas selanjutnya dengan menggunakan algoritma SVM. Hingga akhirnya didapatkan kelas sesungguhnya untuk data tersebut di *layer* terakhir.

Misalnya terdapat sejumlah N kelas. Metode DDAG bekerja dengan membentuk semua kombinasi pasangan dua kelas yakni sebanyak  $N(N-1)/2$  dimana  $N>2$ . Misalkan *hyperplane* optimal yang diperoleh dari kelas i dan kelas j adalah

$$f_{ij}(x) = \phi(x)w_{ij} + b_{ij} \quad (5.5)$$

keterangan:

$f_{ij}(x)$  = fungsi *hyperplane* optimal kelas i dan j.

$i = 1, 2, \dots, N-1$  dan  $j = i+1, i+2, \dots, N$  serta berlaku  $f_{ji}(x) = -f_{ij}(x)$ . Berikut adalah algoritma dari DDAG [TOM-12]:

1. Mendapatkan *hyperplane* optimal yakni persamaan (5.5) untuk semua kombinasi pasangan dua kelas.
2. Untuk data input  $x$  yang baru, maka dipilih sebarang pasangan dua kelas dan hitung nilai  $f_{ij}(x)$ . Jika  $f_{ij}(x) > 0$  maka kelas i ditandai sebagai kelas terpilih dan kelas j dihapus. Sebaliknya jika  $f_{ij}(x) < 0$  maka kelas j ditandai sebagai kelas yang terpilih dan kelas i dihapus.
3. Kelas yang terpilih dipasangkan dengan satu kelas yang dipilih secara acak dari keseluruhan kelas yang tersisa.
4. Langkah (2) dan (4) diulangi sampai menyisakan hanya satu kelas (kelas pemenang). Data input  $x$  diklasifikasikan sebagai anggota dari kelas pemenang.

Metode DDAG melakukan tahap *training* sebanyak  $(N-1)/2$  kali. Sedangkan pada tahap *testing* hanya dilakukan sebanyak  $(N-1)$  kali. Hal ini menunjukkan keunggulan metode DDAG karena waktu yang dibutuhkan lebih singkat dibanding metode lain.

## 13.8 SVM Untuk Kasus Regresi

### 13.8.1 Tentang Support Vector Regression (SVR)

Konsep SVR didasarkan pada *risk minimization*, yaitu untuk mengestimasi suatu fungsi dengan cara meminimalkan batas dari *generalization error*, sehingga SVR mampu mengatasi *overfitting*. Fungsi regresi dari metode SVR adalah sebagai berikut (Sethu Vijayakumar & Si Wu, 1999).

#### ALGORITMA SEKUENSIAL TRAINING SVR:

1. Inisialisasi parameter SVR yang digunakan
2. Rumus memperoleh Matrik Hessian

$$R_{ij} = (K(x_i, x_j) + \lambda^2) \quad (6.1)$$

3. Untuk tiap training point lakukan:

$$E_i = y_i - \sum_{j=1}^l (\alpha_j^* - \alpha_j) R_{ij} \quad (6.2)$$

$$\delta\alpha_i^* = \min \left\{ \max \left\{ \gamma(E_i - \varepsilon), -\alpha_i^* \right\}, C - \alpha_i^* \right\} \quad (6.3)$$

$$\delta\alpha_i = \min \left\{ \max \left\{ \gamma(-E_i - \varepsilon), -\alpha_i \right\}, C - \alpha_i \right\} \quad (6.4)$$

$$\alpha_i^* = \alpha_i^* + \delta\alpha_i^* \quad (6.5)$$

$$\alpha_i = \alpha_i + \delta\alpha_i \quad (6.6)$$

4. Kembali pada langkah ketiga, sampai pada kondisi iterasi maksimum atau

$$\max(|\delta\alpha_i|) < \varepsilon \quad \max(|\delta\alpha_i^*|) < \varepsilon$$

5. Dengan menggunakan fungsi peramalan sebagai berikut:

$$f(x) = \sum_{i=1}^l (\alpha_i^* - \alpha_i)(K(x_i, x) + \lambda^2) \quad (6.7)$$

#### Metode Kernel

$$k(x, x_i) = \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right) \quad (6.8)$$

Misal nilai  $\sigma = 0.7$

(Rajkumar et al., 2013) (Javed et al., 2009)

## Normalisasi Data

Persamaan yang digunakan untuk normalisasi data adalah (S Gopal et al., 2015) seperti yang ditunjukkan pada persamaan.

$$x' = \frac{(x - x_{\min})}{x_{\max} - x_{\min}} \quad (6.9)$$

Dimana:

$x'$  = Hasil normalisasi data

$x$  = Nilai data yang akan dinormalisasi

$x_{\max}$  = Nilai maksimum dari dataset yang digunakan

$x_{\min}$  = Nilai minimum dari dataset yang digunakan

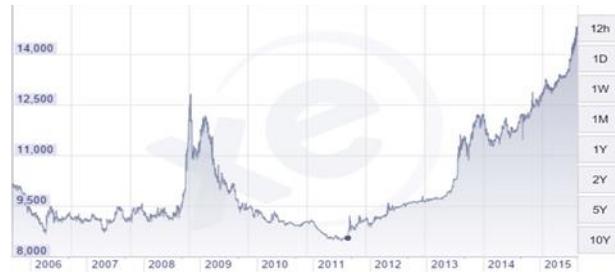
## Nilai Evaluasi

Persamaan yang digunakan untuk normalisasi data adalah (Swanson, 2010) seperti yang ditunjukkan pada persamaan.

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{\hat{y}_i - y_i}{y_i} \times 100 \right| \quad (6.10)$$

## 13.8.2 Komponen SVR (Training dan Testing)

Berikut merupakan contoh proses training dan testing terhadap kasus peramalan nilai kurs. Diketahui Grafik Fluktuasi Nilai Tukar IDR terhadap USD Periode 2006-2015 pada Gambar berikut.



Gambar 13.15 Nilai Tukar IDR terhadap USD Periode 2006-2015

Grafik diatas adalah pergerakan nilai tukar mulai dari tahun 2006 hingga 2015 (Nilai grafik tahunan). Dengan titik terkuat terjadi pada tahun 2011 dan titik terlemah terjadi pada tahun 2015. Berikut diketahui Data Nilai Tukar IDR Terhadap USD Juli 2015 yang ditunjukkan oleh Tabel berikut.

Tabel 13.51 Data Nilai Tukar IDR Terhadap USD Juli 2015

| TANGGAL | NILAI TUKAR |
|---------|-------------|
| 42190   | 13338       |
| 42191   | 13356       |
| 42192   | 13332       |
| 42193   | 13331       |
| 42194   | 13337       |
| 42195   | 13316       |
| 42196   | 13316       |
| 42197   | 13316       |
| 42198   | 13353       |
| 42199   | 13304       |
| 42200   | 13304       |
| 42201   | 13309       |

dan dataset dengan 4 fitur yang ditunjukkan oleh Tabel berikut.

Tabel 13.52 Dataset dengan 4 fitur

| No | Tgl/Bln/Thn  | X1    | X2    | X3    | X4    | Y     |
|----|--------------|-------|-------|-------|-------|-------|
| 1  | 9 Juli 2015  | 13338 | 13356 | 13332 | 13331 | 13337 |
| 2  | 10 Juli 2015 | 13356 | 13332 | 13331 | 13337 | 13316 |
| 3  | 11 Juli 2015 | 13332 | 13331 | 13337 | 13316 | 13316 |
| 4  | 12 Juli 2015 | 13331 | 13337 | 13316 | 13316 | 13316 |
| 5  | 13 Juli 2015 | 13337 | 13316 | 13316 | 13316 | 13353 |
| 6  | 14 Juli 2015 | 13313 | 13346 | 13347 | 13304 | 13304 |
| 7  | 15 Juli 2015 | 13346 | 13347 | 13304 | 13304 | 13304 |
| 8  | 16 Juli 2015 | 13347 | 13304 | 13304 | 13304 | 13309 |

Data Latih      Data Uji

### Proses Normalisasi

Setelah ditentukan data latih dan data uji maka selanjutnya adalah melakukan proses normalisasi data, dimana:

$x'$  = Hasil normalisasi data

$x$  = Nilai data yang akan dinormalisasi

$x_{max}$  = Nilai maksimum dari keseluruhan data, misal mulai dari tahun 2006 – 2015

$x_{min}$  = Nilai minimum dari keseluruhan data, misal mulai dari tahun 2006 – 2015

Misal didapatkan,  $x_{min} = 9634$  dan  $x_{max} = 14728$

Dengan menggunakan persamaan 6.9 maka diperoleh hasil data normalisasi untuk data 1 fitur  $x_1$  adalah sebagai berikut:

$$x' = \frac{(x - x_{\min})}{x_{\max} - x_{\min}}$$

$$x_1' = \frac{(13338 - 9634)}{14728 - 9634} = 0.727129$$

Hasil normalisasi dari data latih ditunjukkan oleh Tabel 13.53 dan hasil normalisasi data uji ditunjukkan oleh Tabel 13.54.

Tabel 13.53 Hasil Normalisasi Data Latih

| No | Tgl/Bln/Thn  | X1       | X2       | X3       | X4       | Y        |
|----|--------------|----------|----------|----------|----------|----------|
| 1  | 9 Juli 2015  | 0.727130 | 0.730664 | 0.725952 | 0.725756 | 0.726934 |
| 2  | 10 Juli 2015 | 0.730664 | 0.725952 | 0.725756 | 0.726934 | 0.722811 |
| 3  | 11 Juli 2015 | 0.725952 | 0.725756 | 0.726934 | 0.722811 | 0.722811 |
| 4  | 12 Juli 2015 | 0.725756 | 0.726934 | 0.722811 | 0.722811 | 0.722811 |
| 5  | 13 Juli 2015 | 0.726934 | 0.722811 | 0.722811 | 0.722811 | 0.730075 |

Tabel 13.54 Hasil Normalisasi Data Uji

| No | Tgl/Bln/Thn  | X1       | X2       | X3       | X4       | Y        |
|----|--------------|----------|----------|----------|----------|----------|
| 1  | 14 Juli 2015 | 0.722811 | 0.722811 | 0.722811 | 0.730075 | 0.722222 |
| 2  | 15 Juli 2015 | 0.722811 | 0.722811 | 0.730075 | 0.722222 | 0.728700 |
| 3  | 16 Juli 2015 | 0.722811 | 0.730075 | 0.722222 | 0.728700 | 0.728897 |

### Proses Perhitungan Jarak Data

Setelah dilakukan proses normalisasi maka langkah selanjutnya adalah menghitung jarak dari setiap data. Berikut merupakan contoh perhitungan jarak data 1 terhadap data 2.

Tabel 13.55 Data 1 dan 2

| No | Tgl/Bln/Thn  | X1         | X2         | X3         | X4         |
|----|--------------|------------|------------|------------|------------|
| 1  | 9 Juli 2015  | 0.72712996 | 0.73066353 | 0.7259521  | 0.72575579 |
| 2  | 10 Juli 2015 | 0.73066353 | 0.7259521  | 0.72575579 | 0.72693365 |

$$\begin{aligned}x_{1,2} = \| x_1 - x_2 \|^2 &= (0.7271299 - 0.7306635)^2 + (0.7306635 - 0.7259521)^2 \\&+ (0.7259521 - 0.7257557)^2 + (0.7257557 - 0.7269336)^2 \\&= 3.61095 \times 10^{-5}\end{aligned}$$

Keterangan:

$X_1, X_2, X_3, X_4 \rightarrow$  menyatakan fitur data atau pola-pola data, sedangkan  $x_1$  dan  $x_2$  menyatakan data ke-1 dan data ke-2.

Hasil perhitungan jarak dari setiap data latih ditunjukkan Tabel 13.56.

Tabel 13.56 Jarak Data Latih

| Data ke-<br><i>i</i> | 1                        | 2                        | ... | 5                        |
|----------------------|--------------------------|--------------------------|-----|--------------------------|
| 1                    | 0                        | $3.61095 \times 10^{-5}$ | ... | $8.02348 \times 10^{-5}$ |
| 2                    | $3.61095 \times 10^{-5}$ | 0                        | ... | $4.94435 \times 10^{-5}$ |
| 3                    | $3.51075 \times 10^{-5}$ | $4.06184 \times 10^{-5}$ | ... | $2.66293 \times 10^{-5}$ |
| 4                    | $3.43368 \times 10^{-5}$ | $5.07152 \times 10^{-5}$ | ... | $1.83823 \times 10^{-5}$ |
| 5                    | $8.02348 \times 10^{-5}$ | $4.94435 \times 10^{-5}$ | ... | 0                        |

### Perhitungan Matriks Hessian

Setelah diperoleh jarak dari setiap data langkah selanjutnya adalah menghitung matrik hessian. berikut merupakan contoh perhitungan matrik hessian untuk data 1 terhadap data 2.

$$[R]_{1,2} = K(x_1, x_2) + \lambda^2 = \exp\left(-\frac{3.61095 \times 10^{-5}}{2 \times (0.7)^2}\right) + 4.32^2 = 19.662363154$$

Hasil perhitungan matrik hessian untuk seluruh data ditunjukkan pada Tabel 13.57.

Tabel 13.57 Matrik Hessian

| R <sub>ij</sub> | 1          | 2          | 3          | 4          | 5          |
|-----------------|------------|------------|------------|------------|------------|
| 1               | 19.6624    | 19.6623632 | 19.6623642 | 19.662365  | 19.6623181 |
| 2               | 19.6623632 | 19.6624    | 19.6623586 | 19.6623483 | 19.6623495 |
| 3               | 19.6623642 | 19.6623586 | 19.6624    | 19.6623812 | 19.6623728 |
| 4               | 19.662365  | 19.6623483 | 19.6623812 | 19.6624    | 19.6623812 |
| 5               | 19.6623181 | 19.6623495 | 19.6623728 | 19.6623812 | 19.6624    |

### Hitung Nilai Error, $\delta\alpha_i^*$ dan $\delta\alpha_i$ , serta $\alpha_i$ dan $\alpha_i^*$

Setelah diperoleh nilai matrik hessian maka selanjutnya dihitung nilai Error. berikut merupakan contoh perhitungan nilai  $E_i$  untuk data 1.

$$E_i = y_i - \sum'_{j=1} (\alpha_j^* - \alpha_j) R_{ij}$$

$$\begin{aligned}E_1 &= 0.72693 - ((0.0073646 - 0) * 19.6624 - (0.0072136 - 0) * 19.6623 \\&\quad - (0.0072136 - 0) * 19.6623 - (0.0072136 - 0) * 19.6623 \\&\quad - (0.0075296 - 0) * 19.6623 \\&= 0.00954858\end{aligned}$$

Hasil perhitungan  $E_i$  dari seluruh data ditunjukkan pada Tabel 13.58.

Tabel 13.58 Nilai  $E_i$

| Data | $E_i$       |
|------|-------------|
| 1    | 0.009548583 |
| 2    | 0.005426086 |
| 3    | 0.005426086 |
| 4    | 0.005426086 |
| 5    | 0.012689533 |

Setelah diperoleh nilai  $E_i$  maka dilakukan perhitungan  $\delta\alpha_i^*$  dan  $\delta\alpha_i$  dengan persamaan 6.3 dan 6.4. Berikut contoh perhitungan nilai  $\delta\alpha_i^*$  dan  $\delta\alpha_i$  untuk data 1.

$$\begin{aligned}\delta\alpha_i^* &= \min \left\{ \max \left\{ \gamma(E_i - \varepsilon), -\alpha_i^* \right\}, C - \alpha_i^* \right\} \\ \delta\alpha_1^* &= (\min(\max(0.00406 * (0.0095485 - 0.0004)\right.\end{aligned}$$

$$\left., 0.0073646), 100 - 0.0073646))\right)$$

$$= 0.00003722\ 2650829$$

$$\delta\alpha_i = \min \left\{ \max \left\{ \gamma(-E_i - \varepsilon), -\alpha_i \right\}, C - \alpha_i \right\}$$

$$\delta\alpha_1 = (\min(\max(0.00406 * (0.0095485 - 0.0004), -0), 100 - 0)) = 0$$

Hasil perhitungan  $\delta\alpha_i^*$  dan  $\delta\alpha_i$  untuk seluruh data latih ditunjukkan oleh Tabel 13.59.

Tabel 13.59 Nilai  $\delta\alpha_i^*$  dan  $\delta\alpha_i$

| Data | $\delta\alpha_i^*$ | $\delta\alpha_i$ |
|------|--------------------|------------------|
| 1    | 3.7223E-05         | 0                |
| 2    | 2.045E-05          | 0                |
| 3    | 2.045E-05          | 0                |
| 4    | 2.045E-05          | 0                |
| 5    | 5.0002E-05         | 0                |

Setelah diperoleh nilai  $\delta\alpha_i^*$  dan  $\delta\alpha_i$  maka dilakukan perhitungan terhadap  $\alpha_i$  dan  $\alpha_i^*$ . Berikut merupakan contoh perhitungan  $\alpha_i$  dan  $\alpha_i^*$  untuk data 1.

$$\alpha_i^* = \alpha_i^* + \delta\alpha_i^*$$

$$\alpha_1^* = 0.0073646102 + 0.0000372226 = 0.0074018329$$

$$\alpha_i = \alpha_i + \delta\alpha_i$$

$$\alpha_1 = 0 + 0 = 0$$

Hasil perhitungan  $\alpha_i$  dan  $\alpha_i^*$  untuk seluruh data latih ditunjukkan oleh Tabel 13.60.

Tabel 13.60 Hitung nilai alpha

| Data | $\alpha_i^*$      | $\alpha_i$ |
|------|-------------------|------------|
| 1    | <b>0.00740183</b> | 0          |
| 2    | <b>0.0072341</b>  | 0          |
| 3    | <b>0.0072341</b>  | 0          |
| 4    | <b>0.0072341</b>  | 0          |
| 5    | <b>0.00752963</b> | 0          |

### Testing Data Latih Dan Data Uji

Dengan menggunakan fungsi peramalan diperoleh nilai  $f(x)$  untuk data latih ke 1 adalah sebagai berikut:

$$f(x) = \sum_{i=1}^l (\alpha_i^* - \alpha_i)(K(x_i, x) + \lambda^2)$$

$$f(x) = ((0.00740183290 - (0)) \times 19.6624)$$

$$+ ((0.00723410171 - (0)) \times 19.662363)$$

$$+ ((0.00723410171 - (0)) \times 19.662364)$$

$$+ ((0.00723410171 - (0)) \times 19.662364)$$

$$+ ((0.00752962809 - (0)) \times 19.662318)$$

$$= 0.720306367977400$$

Hasil perhitungan nilai  $f(x)$  untuk seluruh data latih pada Tabel 13.61.

Tabel 13.61 Nilai  $f(x)$  Data Latih

| Data Latih ke- <i>i</i> | Nilai Aktual Kurs | F(x)       |
|-------------------------|-------------------|------------|
| 1                       | 0.726933647       | 0.72030637 |
| 2                       | 0.72281115        | 0.72030644 |
| 3                       | 0.72281115        | 0.72030686 |
| 4                       | 0.72281115        | 0.72030685 |
| 5                       | 0.730074598       | 0.72030646 |

Hasil perhitungan nilai  $f(x)$  untuk seluruh data uji pada Tabel 13.62.

Tabel 13.62 Nilai  $f(x)$  Data Uji

| Data Uji ke- <i>i</i> | Nilai Aktual Kurs | F(x)       |
|-----------------------|-------------------|------------|
| 1                     | 0.722222222       | 0.7203045  |
| 2                     | 0.728700432       | 0.72030434 |
| 3                     | 0.728896741       | 0.72030458 |

### Denormalisasi Peramalan

$$x' = \frac{(x - x_{\min})}{x_{\max} - x_{\min}}$$

$$x = x'(x_{\max} - x_{\min}) + x_{\min}$$

$$x = (0.720306367977 \times (14728 - 9634)) + 9634 = 13303.24057$$

Hasil denormalisasi untuk seluruh data latih pada Tabel 13.63.

Tabel 13.63 Hasil Denormalisasi Data Latih

| Data Latih ke- <i>i</i> | F(x)        | F(x) Denormalisasi | Nilai Aktual |
|-------------------------|-------------|--------------------|--------------|
| 1                       | 0.720306368 | 13303.24057        | 13337        |
| 2                       | 0.720306437 | 13303.24309        | 13316        |
| 3                       | 0.720306858 | 13303.24396        | 13316        |
| 4                       | 0.720306853 | 13303.24388        | 13316        |
| 5                       | 0.72030646  | 13303.2422         | 13353        |

Hasil denormalisasi untuk seluruh data latih pada Tabel 6.14.

Tabel 13.64 Hasil Denormalisasi Data Uji

| Data Uji ke- <i>i</i> | F(x)        | F(x) Denormalisasi | Nilai Aktual |
|-----------------------|-------------|--------------------|--------------|
| 1                     | 0.720304504 | 13303.23114        | 13304        |
| 2                     | 0.720304342 | 13303.23032        | 13304        |
| 3                     | 0.720304581 | 13303.23153        | 13309        |

## MAPE

Menghitung Mape dengan menggunakan persamaan 6.10, sehingga diperoleh nilai Mape sebagai berikut:

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{\hat{y}_i - y_i}{y_i} \right| \times 100 |$$

$$\begin{aligned} MAPE_{\text{data latih}} &= \frac{1}{5} \left( \left| \frac{13303.24057 - 13337}{13337} \right| \times 100 \right) + \left| \frac{13303.24309 - 13316}{13316} \right| \times 100 \\ &+ \left| \frac{13303.24396 - 13316}{13316} \right| \times 100 + \left| \frac{1303.24388 - 13316}{13316} \right| \times 100 \\ &+ \left| \frac{1333.2422 - 13353}{13353} \right| \times 100 \\ &= 0.182630337 \end{aligned}$$

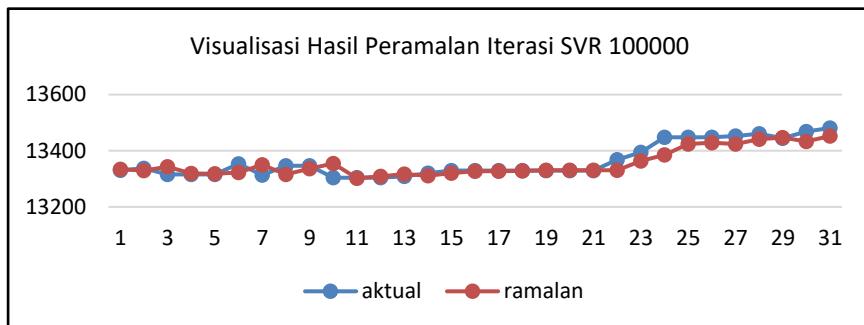
Hasil perhitungan nilai Mape untuk data latih dan data uji dapat dilihat pada Tabel 13.65.

Tabel 13.65 Nilai MAPE

| NILAI | Data Latih  | Data Uji    |
|-------|-------------|-------------|
|       | 0.253126116 | 0.073378327 |
|       | 0.095801356 | 0.320468153 |
|       | 0.095794862 | 0.327927367 |
|       | 0.095795468 |             |
|       | 0.372633884 |             |
|       | MAPE        | 0.182630337 |

### Contoh Grafik Hasil Peramalan Nilai Kurs

Mape = 0.127551, berikut contoh grafik hasil peramalan nilai kurs yang ditunjukan oleh Gambar 13.16.



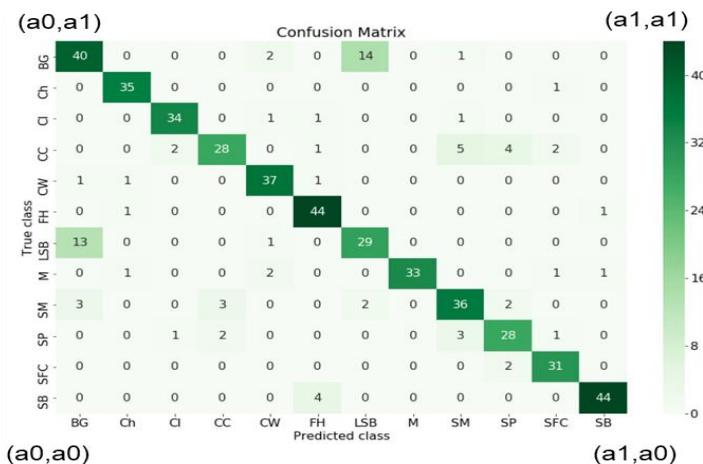
Gambar 13.16 Visualisasi Hasil Peramalan Iterasi SVR 100000

## 13.9 Pengukuran Nilai Performa Klasifikasi

### 13.9.1 Esensi Confusion Matrix (CM)

Kita dapat melihat dari CM untuk semua kelas yang hasil label prediksi berbeda dari label sebenarnya, yaitu pada bagian selain diagonal [(a1,a0) dan (a0,a1)] pada segitiga atas [(a1,a0), (a0,a1) dan (a1,a1)] yang dapat mudah diidentifikasi nilainya dengan nilai Recall (dimana Recall itu sama dengan Sensitivity) dan pada segitiga bawah [(a1,a0), (a0,a1) dan (a0,a0)] yang dapat mudah diidentifikasi nilainya dengan nilai Presisi atau dengan nilai Spesificity. Kelemahan Presisi dan Recall jika dipasangkan sebagai nilai evaluasi pendukung, yaitu karena hanya melihat dari salah satu kelas saja, yaitu pada bagian True Positive (TP) sebagai pembilangnya. Di mana Recall dilihat dari sudut pandang True Class, sedangkan Presisi dari sudut pandang Predicted Class. Keduanya tanpa memperhatikan True Negative (TN). Jadi sebaiknya untuk evaluasi pendukung menggunakan nilai Sensitivity (dengan pembilang TP) dan Spesificity (dengan pembilang TN) yang keduanya dibagi dengan total masing-masing True Class-nya. Sehingga dengan temuan nilai presisi dan recall tersebut kita dapat mengambil langkah-langkah untuk memperbaiki algoritmanya atau meng-im-

prove sebagian langkah pada algoritmanya. Sehingga mampu meningkatkan nilai evaluasi utama, yaitu dapat menggunakan Akurasi, ROC maupun F-Score.



Di mana, (a0,a0) , (a1,a0), (a0,a1) dan (a1,a1) hanya sbg tambahan utk mempermudah pemahaman. Kenapa sebaiknya pengukuran performa algoritma klasifikasi secara standar atau default atau wajib minimal menggunakan Akurasi, bukan ROC atau F-Score, dengan pertimbangan berikut:

| Kelas Aktual (Actual Class) | Kelas Prediksi (Sistem/Predicted/ Classified) |                       |
|-----------------------------|-----------------------------------------------|-----------------------|
|                             | Kelas 1                                       | Kelas 2               |
| Kelas 1                     | TP(C <sub>1,1</sub> )                         | FN(C <sub>1,2</sub> ) |
| Kelas 2                     | FP(C <sub>2,1</sub> )                         | TN(C <sub>2,2</sub> ) |

- Karena nilai Akurasi menggunakan semua komponen pada Confusion Matrix (CM), sehingga lebih lengkap dalam mengevaluasi hasil suatu algoritma, artinya cara evaluasinya lebih menyeluruh.
- Akurasi tidak mungkin nilainya pernah “undefined” sedangkan ROC dan F-Score nilainya terkadang bisa “undefined” (disebut undefined, krn pembaginya bernilai nol), karena pembaginya berupa (FN+TP+FP+TN), berikut contohnya sebagai bukti, misal didapat CM berikut, dari suatu hasil uji data test sebanyak 15 data.

| Kelas Aktual<br>(Actual Class) | Kelas Prediksi (Sistem/<br>Predicted/ Classified) |                   |
|--------------------------------|---------------------------------------------------|-------------------|
|                                | Kelas 1                                           | Kelas 2           |
| Kelas 1                        | $TP(C_{1,1}) = 0$                                 | $FN(C_{1,2}) = 0$ |
| Kelas 2                        | $FP(C_{2,1}) = 10$                                | $TN(C_{2,2}) = 5$ |

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{0+5}{0+5+10+0} = \frac{5}{15} = 0,33$$

$$Precision(P) = \frac{TP}{FP+TP} = \frac{0}{10+0} = 0$$

$$Recall(R) = \frac{TP}{FN+TP} = \frac{0}{0+0} = (undefined)$$

1/0 hasilnya adalah infinity, tetapi 0/0 bukan infinity, ini disebut dengan undefined (NaN). Sehingga F-Score juga undefined.

$$F = \frac{2 \times P \times R}{P+R} = (undefined)$$

Detail penjelasan kenapa Area Under ROC Curve (AUC), F-Score terkadang terdapat nilai “undefined” (zero denominator in ROC and Precision-Recall, artinya terkadang terdapat pembagi nol) dalam perhitungannya:

- ROC didalamnya mengandung nilai “Sensitivity (Sensitivity itu rumusnya sama dgn Recall) & Specificity”, yang artinya ROC membungkusnya menjadi 1 nilai, yaitu Area Under ROC atau luas wilayah di bawah ROC Curve. Di mana “Sensitivity & Specificity” terkadang nilainya “undefined”.
- F-Score didalamnya mengandung nilai “Precision & Recall”, lalu nanti membungkusnya menjadi 1 nilai.

Solusinya (Data Science/DICE Group at UPB, 2017):

- Dalam beberapa kasus yang jarang terjadi, perhitungan Precision atau Recall dapat menyebabkan pembagian dengan 0. Mengenai presisi dengan pembagi nol, ini dapat terjadi jika TP = 0 dan FP = 0. Sedangkan mengenai recall dengan pembagi nol, ini dapat terjadi jika TP = 0 dan FN = 0.
- Untuk kasus khusus ini, ditetapkan bahwa jika positif sebenarnya (TP), positif palsu (FP) dan negatif palsu (FN) semuanya bernilai 0, maka nilai precision, recall dan F1-measure langsung diset sama dengan 1. Hal tersebut dapat terjadi

jika semua hasil klasifikasi data ujinya semuanya masuk pada negatif sebenarnya (TN). Jika dibuat dalam bentuk persamaan menjadi berikut:

$$Precision(P) = \begin{cases} 1, & \text{jika } TP = 0, FP = 0, FN = 0 \\ 0, & \text{jika } TP = 0, FN = 0 \\ \frac{TP}{TP+FP}, & \text{lainnya} \end{cases}$$

$$Recall(R) = \begin{cases} 1, & \text{jika } TP = 0, FP = 0, FN = 0 \\ 0, & \text{jika } TP = 0, FP = 0 \\ \frac{TP}{FN+TP}, & \text{lainnya} \end{cases}$$

- Ini mungkin terjadi dalam kasus di mana pada ground truth atau gold standard-nya berisi data dengan label kelas sebenarnya atau kelas aktualnya tanpa anotasi/keterangan atau label apa pun atau masuk pada label yang lain (misal masuk pada kelas yang lain, yaitu semua masuk pada cell TN). Dan sistem sebagai annotator untuk memprediksi hasil label kelas dari data uji, tidak mengembalikan anotasi dengan tepat (artinya sistem menghasilkan label yang masuk pada kelas lainnya, yaitu masuk pada cell TN).
- Jika  $TP = 0$  dan salah satu dari dua penghitung lainnya lebih besar dari 0 ( $FN > 0$  atau  $FP > 0$ ), maka nilai precision, recall dan F1-measure langsung diset sama dengan 0 (ini masih bisa untuk dihitung seperti biasa).

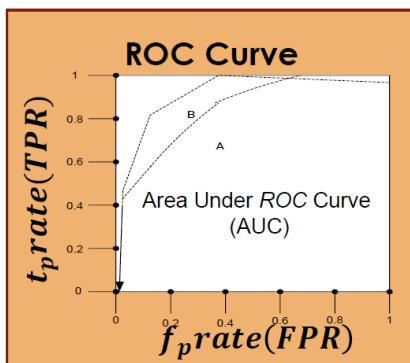
Solusi lain untuk nilai evaluasi tertentu seperti Presisi, Recall, F-Score, Area Under ROC Curve (AUC) agar terbebas dari kondisi “undefined”, yaitu menggunakan nilai smoothing ( $k$ , misal nilai konstanta  $k = 1$ ), seperti pada laplacian smoothing pada penghitungan peluang likelihood pada Naive Bayes (Prabhakar Raghavan & Christopher Manning), atau seperti pada penghitungan nilai MAPE (Berretti, Thampi, dan Srivastava, 2015):

$$Precision(P) = \frac{TP+k}{FP+TP+k}$$

$$Recall(R) = \frac{TP+k}{FN+TP+k}$$

$$F = \frac{2 \times P \times R}{P+R}$$

AUC mengukur kinerja algoritme classifier (misal, untuk klasifikasi dokumen, dll.) dengan menghitung luas daerah di bawah kurva ROC dari beberapa percobaan dari sampel yang dipilih secara acak dari populasi masing-masing kelas yang ada, semakin besar nilai Area Under the ROC Curve (AUC), semakin bagus hasil klasifikasinya. Karena AUC bagian dari daerah berbentuk persegi, maka nilainya akan selalu antara 0,0 dan 1,0. Tabel Kategori Pengklasifikasi Berdasarkan Nilai AUC (Mohanty at all, 2011).



| Nilai AUC   | Diklasifikasikan sebagai |
|-------------|--------------------------|
| 0,90 – 1,00 | Excellent / Sangat baik  |
| 0,80 – 0,90 | Good / Baik              |
| 0,70 – 0,80 | Fair / Sama              |
| 0,60 – 0,70 | Poor / Rendah            |
| 0,50 – 0,60 | Fail / Gagal             |

Jika pada ROC, digunakan rumus f<sub>p</sub>rate(FPR), t<sub>n</sub>rate(TNR), t<sub>p</sub>rate(TPR) seperti berikut.

$$f_p rate(FPR) = \frac{FP}{FP+TN+k} = 1 - TNR$$

$$t_n rate(TNR) = Specificity = \frac{TN+k}{FP+TN+k}$$

$$t_p rate(TPR) = Sensitivity = \frac{TP+k}{FN+TP+k}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Hasil usulan rumus-rumus di atas, yaitu dengan menggunakan parameter smoothing k, akan membuat nilai evaluasi seperti Presisi, Recall, F-Score, ROC terbebas dari kondisi “undefined dan dapat menjamin jika nilai akurasi = 1, maka nilai F = 1, dan juga AUC = 1. Kedua pendekatan solusi tersebut dapat digunakan untuk menghitung Microaveraging & Macroaveraging untuk Presisi, Recall dan F-Score. mikroAVG menunjukkan kinerja dari sekumpulan semua data di dalam dataset, sementara ukuran makroAVG menunjukkan kinerja rata-rata per dokumen. Berikut contohnya (misal menggunakan solusi ke-1):

- Contoh ke-1 (per Data, langsung di hitung TP, FP, FN, P, R, dan F1):

|                 | TP | FP | FN | P   | R   | F1  |
|-----------------|----|----|----|-----|-----|-----|
| Data 1          | 0  | 0  | 1  | 0   | 0   | 0   |
| Data 2          | 0  | 0  | 1  | 0   | 0   | 0   |
| Data 3          | 0  | 0  | 0  | 1   | 1   | 1   |
| Sums<br>(micro) | 0  | 0  | 2  | 0   | 0   | 0   |
| Avg<br>(macro)  | -  | -  | -  | 1/3 | 1/3 | 1/3 |

Kesimpulan:

Dapat dilihat bahwa ukuran mikro semuanya 0, dan ukuran makro 1/3.

- Contoh ke-2 (per kelas, “kelas tertentu” dgn label yes/+ vs “selain kelas tertentu tersebut” dgn label no/-):

| Kelas 1 |   | truth |     |
|---------|---|-------|-----|
|         |   | +     | -   |
| call    | + | 10    | 10  |
|         | - | 10    | 970 |

| Kelas 2 |   | truth |     |
|---------|---|-------|-----|
|         |   | +     | -   |
| call    | + | 90    | 10  |
|         | - | 10    | 890 |

| Gabungan Kelas 1 & 2 |   | truth |      |
|----------------------|---|-------|------|
|                      |   | +     | -    |
| call                 | + | 100   | 20   |
|                      | - | 20    | 1860 |

Keterangan:

“Truth” adalah kelas aktual dan “call” adalah hasil keputusan dari sistem (*classifier*).

Contoh Perhitungan Macroaveraging (evaluasi kinerja rata-rata tiap Data):

- $P = \text{Avg}[(10/(10 + 10)) + (90/(90 + 10))] = [(10/(10 + 10)) + (90/(90 + 10))] / 2$
- $R = \text{Avg}[10/(10 + 10) + 90/(90 + 10)]$
- $F = 2PR/(P+R)$

Contoh Perhitungan Microaveraging (evaluasi kinerja seluruh Data atau Dataset)

- $P = 100/(100 + 20)$
- $R = 100/(100 + 20)$
- $F = 2PR/(P+R)$

### 13.9.2 Rumus Evaluasi untuk $n > 2$ Kelas

Berikut untuk Rumus Evaluasi dengan Confusion Matrix/Tabel Kontingensi dengan  $n$  Kelas (Putri, et al., 2015) & (Manliguez, 2016). Misal Banyak Data Uji dimasukkan dalam variable (ALL).

| Kelas Aktual<br>( <i>Actual Class</i> ) | Kelas Prediksi (Sistem/<br><i>Predicted/ Classified</i> ) |           |    |           |
|-----------------------------------------|-----------------------------------------------------------|-----------|----|-----------|
|                                         | Kelas 1                                                   | Kelas 2   | .. | Kelas $n$ |
| Kelas 1                                 | $C_{1,1}$                                                 | $C_{1,2}$ |    | $C_{1,n}$ |
| Kelas 2                                 | $C_{2,1}$                                                 | $C_{2,2}$ |    | $C_{2,n}$ |
| ..                                      |                                                           |           |    | ..        |
| Kelas $n$                               | $C_{n,1}$                                                 | $C_{n,2}$ | .. | $C_{n,n}$ |

$TP_i = C_{ii}$  , di mana  $i$  merupakan indeks kelas.  $TP_{all} = \sum_{i=1}^n C_{ii}$  , di mana  $n$  merupakan banyaknya kelas.

$TN_i = \sum_{j=1, j \neq i}^n C_{jj}$  , di mana  $j$  merupakan indeks kelas.

$TN_{all} = \sum_{i=1}^n TN_i = n \sum_{i=1}^n C_{ii} - \sum_{i=1}^n C_{ii} = (n - 1)TP_{all}$

$FP_i = \sum_{\substack{j=1 \\ j \neq i}}^n C_{ji}$ , di mana pada  $C$ , variabel  $j$  merupakan indeks kelas yang dijalankan sbg baris,  $i$  indeks kelas yang dijalankan sbg kolom.

$$FP_{all} = \sum_{i=1}^n FP_i = \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n C_{ij} - \sum_{i=1}^n C_{ii} = ALL - TP_{all}$$

$FN_i = \sum_{\substack{j=1 \\ j \neq i}}^n C_{ij}$ , di mana pada  $C$ , variabel  $j$  merupakan indeks kelas yang dijalankan sbg kolom,  $i$  indeks kelas yang dijalankan sbg baris.

$$FN_{all} = \sum_{i=1}^n FN_i = \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n C_{ij} - \sum_{i=1}^n C_{ii} = ALL - TP_{all}$$

Jadi  $FP_{all} = FN_{all}$

$$Accuracy = \frac{TP_{all}}{ALL} \cong \frac{\text{Sum}(Distinct C_{ij} \text{ dari } (TP_{all}, TN_{all}))}{\text{Sum}(Distinct C_{ij} \text{ dari } (TP_{all}, TN_{all}, FP_{all}, FN_{all}))}$$

Dari rumus *Accuracy* di atas, maka untuk menghitung lainnya berdasarkan pendekatan rumus di atas jika dijadikan sebagai hipotesis, maka akan didapat sebagai berikut.

$$Precision (P) = \frac{\text{Sum}(Distinct C_{ij} \text{ dari } (TP_{all}))}{\text{Sum}(Distinct C_{ij} \text{ dari } (TP_{all}, FP_{all}))} = \frac{TP_{all}}{ALL}$$

$$Recall (R) = \frac{\text{Sum}(Distinct C_{ij} \text{ dari } (TP_{all}))}{\text{Sum}(Distinct C_{ij} \text{ dari } (TP_{all}, FN_{all}))} = \frac{TP_{all}}{ALL}$$

Maka,

$$F = \frac{2PR}{P+R} = P = R$$

Dari uraian di atas, jika  $n > 2$  kelas, maka nilai evaluasi dapat cukup menggunakan *Accuracy*. Alternatifnya jika ingin tetap menggunakan transformasi dari  $n > 2$  menjadi 2 kelas, dapat menggunakan kaidah Macroaveraging dan Microaveraging seperti penjelasan sebelum-sebelumnya.

## BAB 14 Algoritma Regresi

### 14.1 Regresi Linier

Regresi merupakan suatu teknik untuk membuat dan mendapatkan model persamaan sehingga dapat digunakan untuk menebak atau mengetahui hasil prediksi nilai  $Y$  (satu target atau multi-target) dari data masukan  $X$  (dengan dimensi tertentu). Sedangkan Prediksi berbeda dengan klasifikasi (dalam machine learning, klasifikasi dianggap sebagai salah satu jenis dari prediksi), di mana klasifikasi digunakan untuk memprediksi label kelas/kategori yang sifatnya diskrit, sedangkan regresi memprediksi nilai  $Y$  yang bersifat kontinyu. Least Squares Error adalah bagaimana cara kita mendapatkan Persamaan Regresi ( $\hat{y} = ax + b + \varepsilon$ ) sehingga dapat meminimalkan jumlah kuadrat error antara titik data aktual ( $y$ ) dengan nilai titik data hasil estimasi ( $\hat{y}$ ) atau dibaca “ $y$  topi”, konsep ini disebut sum of squared errors (SSE).

Tabel 14.1 Contoh Dataset untuk Regresi Linier Sederhana

| No | $X$ , misal menyatakan lama waktu belajar | $Y$ , misal menyatakan banyaknya materi yang dipahami |
|----|-------------------------------------------|-------------------------------------------------------|
| 1  | 2                                         | 4                                                     |
| 2  | 3                                         | 5                                                     |
| 3  | 5                                         | 7                                                     |
| 4  | 7                                         | 10                                                    |
| 5  | 9                                         | 15                                                    |

$$SSE = \sum_{i=1}^n (\varepsilon_i)^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - ax_i - b)^2$$

Minimalkan SSE atau jumlah kuadrat error data ( $\varepsilon^2$ ) dengan Turunan terhadap  $b$  seperti berikut.

$$\nabla_b SSE = 0$$

$$\nabla_b SSE = \nabla_b \sum_{i=1}^n (y_i - ax_i - b)^2 = 0$$

$$= \sum_{i=1}^n 2(y_i - ax_i - b)(-1) = 0$$

Kedua sisi diberikan perlakuan yang sama, misal sama-sama kedua sisi tersebut dibagi  $-2n$ .

$$\frac{1}{n} \sum_{i=1}^n (y_i - ax_i - b) = 0$$

$$\frac{1}{n} \sum_{i=1}^n y_i - \frac{1}{n} \sum_{i=1}^n ax_i - \frac{1}{n} \sum_{i=1}^n b = 0$$

$$\frac{1}{n} \sum_{i=1}^n y_i - \frac{a}{n} \sum_{i=1}^n x_i - b = 0$$

$$b = \frac{1}{n} \sum_{i=1}^n y_i - \frac{a}{n} \sum_{i=1}^n x_i$$

$$b = \bar{y} - a\bar{x}$$

Kemudian lakukan Turunan terhadap  $a$ , tetapi sebelumnya substitusi dulu menjadi seperti berikut

$$\nabla_a SSE = \nabla_a \sum_{i=1}^n (y_i - ax_i - b)^2 = 0$$

$$= \nabla_a \sum_{i=1}^n (y_i - ax_i - (\bar{y} - a\bar{x}))^2 = 0$$

$$= \nabla_a \sum_{i=1}^n (y_i - a(x_i - \bar{x}) - \bar{y})^2 = 0$$

$$\sum_{i=1}^n 2(y_i - a(x_i - \bar{x}) - \bar{y})(-(x_i - \bar{x})) = 0$$

Kedua sisi diberikan perlakuan yang sama, misal sama-sama kedua sisi tersebut dibagi  $-2$ .

$$\sum_{i=1}^n (y_i - a(x_i - \bar{x}) - \bar{y})(x_i - \bar{x}) = 0$$

$$\sum_{i=1}^n ((y_i - \bar{y})(x_i - \bar{x}) - a(x_i - \bar{x})^2) = 0$$

$$\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x}) - a \sum_{i=1}^n (x_i - \bar{x})^2 = 0$$

$$a = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

Berdasarkan pembuktian dengan pendekatan turunan di atas untuk mendapatkan nilai-nilai  $a, b$  pada Regresi Linier sederhana, yaitu 2 Dimensi, maka dengan cara dan kaidah yang sama melalui turunan di atas, akan dengan sangat mudah untuk memecahkan kasus pada  $n$  Dimensi, yaitu pada Regresi Linier Berganda. Dengan peubah  $(x_1, x_2, \dots, x_{n-1})$ , yaitu sebanyak  $n-1$ , dan banyaknya koefisien betha ( $\beta$ ) sebanyak

$n$ , karena dimulai dari betha nol ( $\beta_0$ ) sampai ( $\beta_{n-1}$ ). Jika pada 2D  $b = \beta_0$  dan  $a = \beta_1$ , sehingga dapat ditulis juga  $\hat{y} = \beta_0 + \beta_1 x$ . Maka Ketika pada  $n$  dimensi dapat dituliskan  $\hat{y}_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \dots + \beta_{n-1} x_{i,n-1} + \varepsilon$ . Di mana  $i$  menyatakan indek untuk datanya.

Pada Regresi kompleks, yaitu khusus pada bentuk  $k$  Dimensi untuk  $X$  dan untuk  $Y$  juga memiliki  $i$  Dimensi, yang artinya pada  $X$  sebagai input atau variable independen juga ada banyak parameter yang dilibatkan yaitu sebanyak ( $k$ ) dan  $Y$  sebagai output atau target atau variable dependen juga ada banyak dimensi yaitu sebanyak ( $i$ ). Sehingga untuk menyederhanakan bentuk model atau Persamaan dapat menggunakan bentuk penulisan dalam persamaan matriks  $Y$ ,  $X$ , dan matrik  $\beta$  berikut. Di mana nantinya Persamaan ini Ketika sudah diturunkan terhadap  $\beta$  dapat digunakan sebagai dasar untuk memahami dengan mudah Algoritma Extreme Learning Machine (ELM) yang diusulkan oleh Prof. Huang.

$$Y = X\beta$$

$$\varepsilon = Y - X\beta$$

Minimalkan nilai error data dari nilai skalar ( $\varepsilon^T \varepsilon$ ) dengan Turunan terhadap  $\beta$  seperti berikut.

$$\begin{aligned}\min_{\beta} (\varepsilon^T \varepsilon) &= \min_{\beta} (Y - X\beta)^T (Y - X\beta) \\&= \min_{\beta} (Y^T Y - Y^T X\beta - (X\beta)^T Y + (X\beta)^T (X\beta)) \\&= \min_{\beta} (Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T (X\beta)) \\&= \min_{\beta} (Y^T (Y - X\beta) - \beta^T X^T (Y - X\beta))\end{aligned}$$

Makna dari ( $\nabla_{\beta} (\varepsilon^T \varepsilon)$ ) adalah seperti kalimat sebelumnya yaitu “Minimalkan nilai error data dari nilai skalar ( $\varepsilon^T \varepsilon$ ) dengan Turunan terhadap  $\beta$ ”. Berikut langkah-langkah untuk turunannya.

$$\begin{aligned}\nabla_{\beta} (\varepsilon^T \varepsilon) &= \nabla_{\beta} (Y^T (Y - X\beta) - \beta^T X^T (Y - X\beta)) = 0 \\&= \nabla_{\beta} (Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T (X\beta)) = 0 \\&= \nabla_{\beta} (Y^T Y) - \nabla_{\beta} (Y^T X\beta) - \nabla_{\beta} (\beta^T X^T Y) + \nabla_{\beta} (\beta^T X^T (X\beta)) = 0 \\&= 0 - (Y^T X)^T - (Y^T X)^T + 2X^T X\beta = 0\end{aligned}$$

$$\begin{aligned} -2(Y^T X)^T + 2X^T X\beta &= 0 \\ -(Y^T X)^T + X^T X\beta &= 0 \\ X^T X\beta &= (Y^T X)^T \\ X^T X\beta &= X^T Y \\ (X^T X)^{-1}(X^T X)\beta &= (X^T X)^{-1}X^T Y \\ \beta &= (X^T X)^{-1}X^T Y \end{aligned}$$

Sehingga ketika digunakan untuk melakukan prediksi data uji ( $X_{test}$ ) untuk menghasilkan  $Y_{test}$ , maka dapat digunakan bentuk *Generative modelling* sebagai berikut, yang secara *default* atau jika dikembangkan lagi bisa juga untuk regresi Non-Linier.

$$\hat{Y} = X\hat{\beta}$$

$$Y_{test} = X_{test} \left( (X_{train}^T X_{train})^{-1} X_{train}^T Y_{train} \right)$$



Done.:D

Tabel 14.2 Contoh Dataset untuk Regresi kompleks dalam Matriks

| No | $X$ , misal menyatakan lama waktu belajar dalam 3 hari aktif, hari ke-1 ( $x_1$ ),..., hari ke-3 ( $x_3$ ) |       |       | $Y$ , misal menyatakan banyaknya materi yang dipahami ( $y_1$ ), dan peringkat di kelas ( $y_2$ ) |       |
|----|------------------------------------------------------------------------------------------------------------|-------|-------|---------------------------------------------------------------------------------------------------|-------|
|    | $x_1$                                                                                                      | $x_2$ | $x_3$ | $y_1$                                                                                             | $y_2$ |
| 1  | 2                                                                                                          | 3     | 1     | 4                                                                                                 | 5     |
| 2  | 3                                                                                                          | 3     | 5     | 5                                                                                                 | 4     |
| 3  | 5                                                                                                          | 6     | 4     | 7                                                                                                 | 3     |
| 4  | 7                                                                                                          | 8     | 10    | 10                                                                                                | 2     |
| 5  | 9                                                                                                          | 10    | 12    | 15                                                                                                | 1     |

Kemudian pada Kode Program Regresi Linier 2 Dimensi maupun yang  $n$  Dimensi, yaitu Regresi Linier Berganda, selain menggunakan Teknik Gradient Descent dasar, juga dapat menggunakan teknik optimasi Gradient Descent yang lebih *powerful*, seperti optimizer = ["Stochastic Gradient Descent (SGD)", "Annealed Gradient Descent (SGDAnn)", "SGDMomentum", "Adaptive Moment Estimation (Adam)", "Adam Annealing (AdamAnn)"] atau lainnya. Di mana kode untuk beberapa varian optimizer tersebut sudah ada pada Source Code pada kasus “Prototype Style Transfer dengan CNN Net. (*Build from Scratch*) like ResNet, VGG, Inception, etc”.

#### Source Code 14.1 Regresi Linier 2D

```
#Regresi Linier

% Solusi dengan Gradient Descent 2D:
alpha=0.00001; % learning rate
dataset=data;
[m,dim]=size(dataset);

% inisialisasi b0 dan b1
b0=0.0; b1=0.0;
byk_iter=iterMax;
for j=1:byk_iter
 sum_error=0.0;
 for i=1:m x=dataset(i,1);
 y=dataset(i,2);
 y_topi=b0 + b1*x;
 error=(y_topi-y);
 sum_error=sum_error+(error^2);

 %update nilai b0 dan b1
 b0=b0-alpha*error;
 b1=b1-alpha*error*x;
 end
end

% =====

figure;
plot(dataset(:,1),dataset(:,2), 'o')
hold on
```

```
%plot garis regresi
xx=min(dataset(:,1)):0.001:max(dataset(:,1));
line(xx,b0+b1.*xx,'Color','red',...
'LineStyle','--')

hold off
```

### Source Code 14.2 Regresi Linier $n$ Dimensi

```
#Regresi Linier n Dimensi

% Solusi dengan Gradient Descent n Dimensi:
alpha=0.00001; % learning rate
dataset=data;
[m,dim]=size(dataset);

% inisialisasi b0, b1,..., bdim-1, dimana dim
% adalah byk dimensi fitur dan target
for i=1:dim
 bTemp(i) = 0.0;
 % bTemp(1) -> b0, bTemp(2) -> b1,
 % bTemp(3) -> b2 end

byk_iter=iterMax
for j=1:byk_iter
 sum_error=0.0;
 for i=1:m
 instance=dataset(i,:);
 y_topi=bTemp(1);

 % dengan for
 %for ii=2:dim
 % y_topi =...
 % y_topi + bTemp(ii)*instance(ii-
 % 1);
 %end

 % tanpa for
 y_topi=...
 y_topi+...
 sum(bTemp(2:dim).*instance(1:dim-1));
 error=(y_topi-instance(dim));
 sum_error=sum_error+(error^2);
 bTemp(1) = bTemp(1)-alpha*error;
```

```
% dengan for
%for ii=2:dim
% bTemp(ii) = bTemp(ii)-
% alpha*error*instance(ii-1);
%end

% tanpa for
bTemp(2:dim) =...
bTemp(2:dim)-...
alpha*error*instance(1:dim-1);
end
end
for i=1:dim
disp(strcat('b',num2str(i-1), ' = ',...
num2str(bTemp(i))));
end
```

Sedangkan kode untuk Regresi Kompleks, disajikan menggunakan *library* untuk operasi matriksnya dari numpy. Tetapi jika Anda mahasiswa Informatika atau memang dibutuhkan keahlian, penguasaan dan ketrampilan dalam pemrograman, sebaiknya mengubah Sebagian dari kode library dari operasi matriks tersebut dengan *koding from scratch*.

#### Source Code 14.3 Koding Regresi Kompleks

```
#Koding Regresi Kompleks

Koding ini Berdasarkan Penggunaan Turunan
Matrik utk Generative Modelling
import numpy as np

Ket. X
misal menyatakan berapa jam waktu belajar dlm
3 hari,
Kolom ke-1 adalah hari ke-1 (x1),..., Kolom ke-
3 adalah hari ke-3 (x3)

Mengisi data pada Matriks X
X = np.array([
 [2., 3., 1.],
 [3., 3., 5.],
```

```
[5., 6., 4.],
[7., 8., 10.],
[9., 10., 12.]
])

Dan Y
Kolom ke-1 menyatakan banyaknya materi yg
berhasil dipahami
Kolom ke-2 menyatakan rangking di kelas
Mengisi data pada Matriks Y
Y = np.array([
 [4., 1.],
 [5., 2.],
 [7., 3.],
 [10., 4.],
 [15., 5.]
])

Menampilkan X dan Y
print("Menampilkan X sbg Input:")
print(X)
print()
print("Menampilkan Y sbg Output:")
print(Y)

#-----
Hitung Matrik Beta (B)
B = np.linalg.inv(np.transpose(X).dot(X)).dot(np.transpose(X)).dot(Y)
print("Hitung Matrik Beta (B)")
print(B)
B[:,0]=np.clip(B[:,0], 1, 15)
B[:,1]=np.clip(B[:,1], 1, 10)

print()
print("Hitung Matrik Beta (B) dgn clip")
print(B)

#-----
Memasukkan data uji (Xuji)
Xuji = np.array([
 [2., 3., 1.],
 [3., 3., 5.]
])
print(Xuji)
```

```
#-----

Menghitung Hasil Output Ytopi dari inputan
Xuji
Kolom ke-1 menyatakan banyaknya materi yg ber-
hasil dipahami
Kolom ke-2 menyatakan rangking di kelas
Ytopi = Xuji.dot(B)

Agar angkanya menjadi bulat, maka dibulatkan
ke atas
Ytopi = np.ceil(Ytopi)
[[4. 1.]
[5. 2.]]

Artinya hasilnya, hasil tersebut tidak selalu
sama persis dengan data aslinya,
karena menggunakan pendekatan regresi, yang
kemungkinan besar terkadang ada
nilai error yang rendah maupun error yang
tinggi

print(Ytopi)
```

Link kode program lengkapnya:

[https://drive.google.com/file/d/1yYeOQPLjmXbC3qEUwSDKyLg\\_iTWWWG1X/view?usp=sharing](https://drive.google.com/file/d/1yYeOQPLjmXbC3qEUwSDKyLg_iTWWWG1X/view?usp=sharing)



## 14.2 Regresi Logistik

Regresi logistik *merupakan* bentuk regresi yang digunakan ketika outputnya berupa nilai biner dan juga memungkinkan nilai target prediksinya dalam interval suatu nilai peluang, yaitu [0;1], dengan inputnya berupa data yang bisa sangat variatif (nominal, ordinal, interval atau tingkat rasio atau lainnya) dan dengan dimensi berapapun. Berikut Persamaan dari regresi logistic.

$$y = b_0 - b_1 x$$

Terdapat suatu fungsi sigmoid, yaitu

$$p = \frac{e^y}{e^y + 1}$$

Sedemikian hingga, pada persamaan y dapat diubah menjadi berikut bentuk logaritma natural.

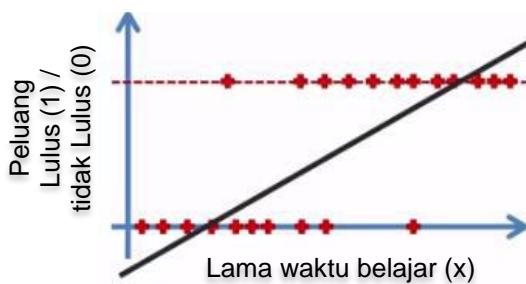
$$\ln\left(\frac{p}{1-p}\right) = b_0 - b_1 x$$

Di mana *Odd ratio* =  $\left(\frac{p}{1-p}\right)$

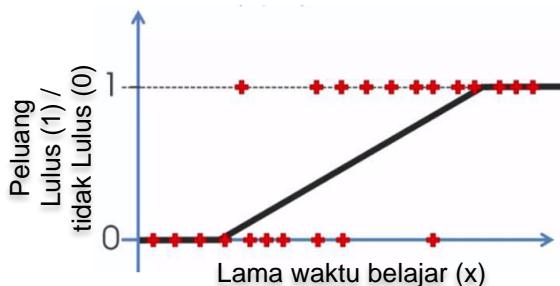
Untuk lebih memudahkan pemahaman, perhatikan dataset berikut, yang jika digunakan pendekatan regresi linier vs regresi logistik.

| No | Lama waktu belajar | Hasil<br>(1=Lulus, 0 =tdkLulus) |
|----|--------------------|---------------------------------|
| 1  | 29                 | 0                               |
| 2  | 15                 | 0                               |
| 3  | 33                 | 1                               |
| 4  | 28                 | 1                               |
| 5  | 39                 | 1                               |
| 6  | 44                 | 1                               |
| 7  | 31                 | 1                               |
| 8  | 19                 | 0                               |
| 9  | 9                  | 1                               |
| 10 | 24                 | 0                               |
| 11 | 32                 | 0                               |
| 12 | 31                 | 0                               |
| 13 | 37                 | 1                               |
| 14 | 35                 | 1                               |

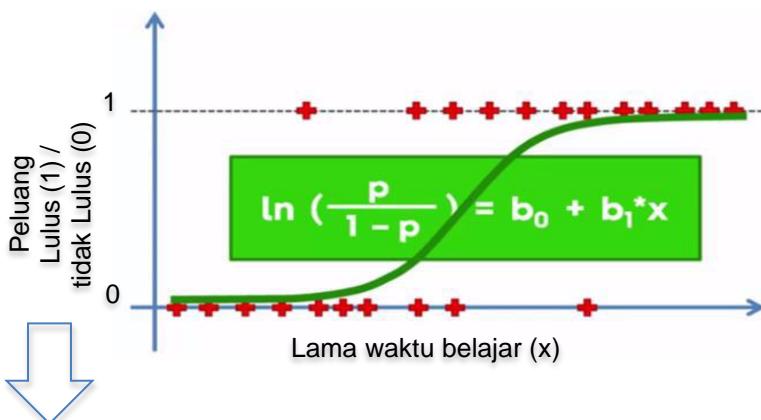
### Hasil Regresi Linier



### Hasil Regresi Linier dengan fungsi tangga.



Dan berikut hasil dari regresi logistik yang terlihat lebih natural dan representative dengan sebaran titik-titik datanya. Di mana trade-off berada ditengah-tengah diantara garis kurva lengkung dari regresi logistik.



P asimtot terhadap 0 dan 1,  
 $P = (0;1)$  atau  $0 < P < 1$

Berikut kode Regresi Logistik dengan Teknik Gradient Descent dasar. Di mana,  $b0$  merupakan intercept,  $b1$  adalah slope, dan alpha ( $\alpha$ ) adalah *learning rate*.

#### Source Code 14.4 Koding Regresi Logistik

```
#Koding Regresi Logistik

%Solusidengan Gradient Descent:
alpha=0.00001; % learning rate
dataset=data;
[m,dim]=size(dataset);

% =====
% y = b0 + b1*x
% inisialisasib0 danb1
b0=0.0;
b1=0.0;
byk_iter=iterMax;
for j=1:byk_iter
 sum_error=0.0;
 for i=1:m
 x=dataset(i,1);
 y=dataset(i,2);
 %[x y]
 y_topi_init=b0 + b1*x;
 y_topi=exp(y_topi_init)/(exp(y_topi_init)...
 + 1);
 error=(y_topi-y);
 sum_error=sum_error+(error^2);

 %update nilaib0 danb1
 b0=b0-alpha*error;
 b1=b1-alpha*error*x;
 end
end
```

## 14.3 Pengukuran Nilai Performa Regresi

Setiap hasil dari proses prediksi (biasanya untuk istilah perkiraan jangka pendek) atau peramalan (biasanya untuk istilah perkiraan jangka panjang) diperlukan proses evaluasi untuk mengetahui performa dari model peramalan yang diterapkan. Pada bagian ini digunakan beberapa dari rumus evaluasi dari keilmuan statistik standar, yaitu mulai dari Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), dan Correlation Coefficient (CC), meskipun masih banyak lagi lainnya.

### 14.3.1 MAE

Mean Absolute Error (MAE) merupakan rata-rata dari perbedaan antara nilai peramalan dengan nilai aktual dalam bentuk absolut. Range nilai dari MAE berkisar dari 0 sampai tak hingga, karena signifikansi dari error bergantung pada nilai aktualnya (Berretti, Thampi, dan Srivastava, 2015). Semakin kecil nilai dari MAE maka semakin baik model peramalan yang diterapkan.  $P_i$  menyatakan nilai hasil prediksi,  $A_i$  menyatakan nilai yang sebenarnya,  $n$  merupakan banyak data yang diuji, dan  $i$  menyatakan indeks data yang diuji. Persamaan untuk menghitung nilai MAE ditunjukkan dalam Persamaan berikut.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |P_i - A_i|$$

### 14.3.2 RMSE

Root Mean Square Error (RMSE) merupakan salah satu standar pengukuran error yang paling banyak digunakan dalam evaluasi hasil peramalan. RMSE didapat dari nilai rata-rata pangkat perbandingan antara nilai yang diprediksi dengan nilai yang sebenarnya. Semakin kecil nilai dari RMSE maka semakin akurat hasil peramalan yang dilakukan (Brooks dan Tsolacos, 2010). Persamaan untuk menghitung nilai RMSE ditunjukkan dalam Persamaan berikut.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (P_i - A_i)^2}$$

### 14.3.3 CC

*Correlation Coefficient* (CC) merupakan nilai yang mengukur derajat hubungan antara dua variabel dan nilainya berkisar antara -1,00 sampai 1,00. Semakin kuat hubungan antar variabel maka nilai CC semakin mendekati angka -1,00 atau +1,00. Sedangkan semakin lemah hubungan antar variable maka nilai CC semakin mendekati angka 0,00. Nilai korelasi yang positif menunjukkan hubungan langsung antar dua variabel, dimana peningkatan pada satu variabel akan menyebabkan peningkatan pada variabel lain dan penurunan pada satu variabel akan menyebabkan penurunan pula pada lainnya. Nilai korelasi yang negatif mengindikasikan hubungan invers atau negatif antar dua variabel, dimana peningkatan pada satu variabel akan menyebabkan penurunan pada variabel lain dan sebaliknya (Jackson, 2011). Persamaan untuk menghitung nilai CC ditunjukkan dalam Persamaan berikut.

$$CC = \frac{(n \sum_{i=1}^n P_i A_i) - (\sum_{i=1}^n P_i)(\sum_{i=1}^n A_i)}{\sqrt{(n \sum_{i=1}^n P_i^2 - (\sum_{i=1}^n P_i)^2)(n \sum_{i=1}^n A_i^2 - (\sum_{i=1}^n A_i)^2)}}$$

### 14.3.4 MAPE

Mean Absolute Percentage Error (MAPE) merupakan rata-rata nilai absolut kesalahan peramalan dalam bentuk persentase (Berretti, Thampi, dan Srivastava, 2015). Semakin kecil nilai dari MAPE maka semakin akurat model peramalan yang diterapkan. Persamaan untuk menghitung nilai MAPE standar ditunjukkan dalam Persamaan berikut.

$$MAPE = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{P_i - A_i}{A_i} \right|$$

Sedangkan untuk MAPE yang khusus berikut, intervalnya dikondisikan seperti nilai evaluasi *Accuracy*, yaitu [0%;100%] atau dapat ditulis dengan bentuk  $0\% < MAPE < 100\%$ . Berikut detail persamaannya.

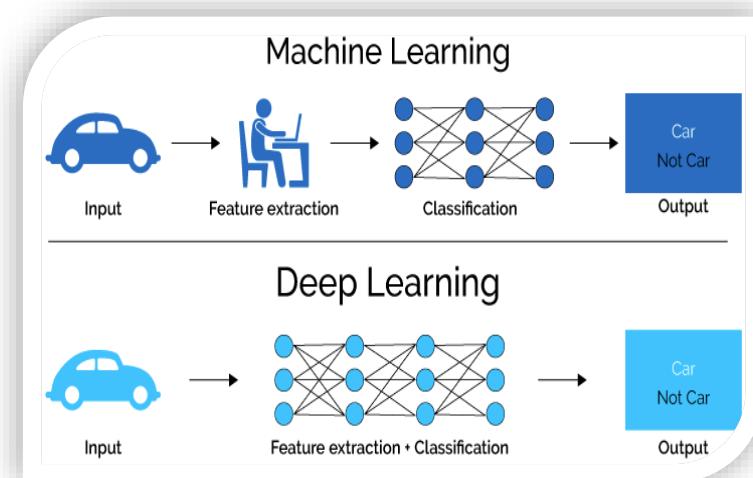
$$MAPE = \frac{1}{n} \sum_{i=1}^n Error_i$$
$$Error_i = \begin{cases} 100\%, & jika \left( \left| \frac{(P_i+c)-(A_i+c)}{(A_i+c)} \right| \times 100\% \right) > 100\% \\ \left| \frac{(P_i+c)-(A_i+c)}{(A_i+c)} \right| \times 100\%, & lainnya \end{cases}$$

Di mana  $c$  dapat diisikan dengan suatu niali konstanta yang kecil, misal  $10^{-1}$  atau lainnya. Nilai  $c$  ini sangat dibutuhkan untuk suatu kondisi ketika terdapat nilai  $A_i = 0$ .

# BAB 15 Machine Learning & Deep Learning

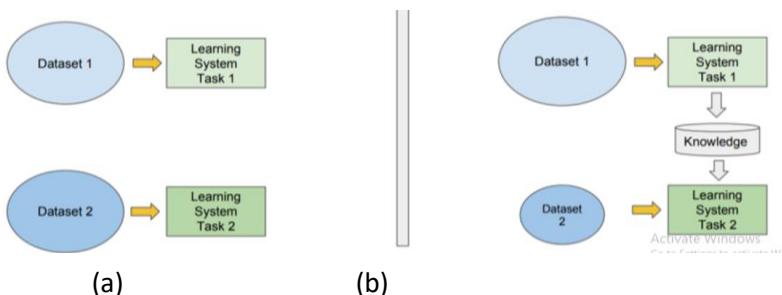
## 15.1 Tradisional Machine Learning vs Transfer Learning

Tradisional Machine Learning memiliki ciri khas, yaitu pembelajarannya lebih terlihat terisolasi (*isolated*) dan hanya dilakukan satu kali pembelajaran atau pembelajarannya tunggal (*single task learning*). Selain itu, hasil pengetahuan pembelajaran tidak untuk disimpan atau diakumulasi untuk pembelajaran estafet atau berkelanjutan, dan juga dapat dikatakan bahwa pembelajaran yang dilakukan tanpa mempertimbangkan pengetahuan dari hasil pembelajaran sebelumnya. Artinya misalkan jika ada penambahan data baru, maka hasil pembelajaran sebelumnya tidak dapat digunakan untuk pembelajaran saat ini, sehingga harus mengulang lagi pembelajaran dari awal dengan melibatkan data lama dan data baru yang membuat komputasinya lebih lambat atau berat, dengan kata lain bahwa Tradisional Machine Learning tidak dapat melakukan *update* pengetahuan dari pembelajaran yang sebelumnya untuk pembelajaran berikutnya.



Gambar 15.1 Ilustrasi Machine Learning dan Deep Learning

Sedangkan Transfer Learning yang ada pada Machine Learning modern seperti Deep Learning, adalah kebalikan dari Tradisional Machine Learning. Di mana keunggulannya adalah dapat melakukan *update* pengetahuan dari hasil pembelajaran sebelumnya, lalu proses pembelajaran bisa lebih cepat karena dapat melibatkan penambahan data secara bertahap dalam ukuran lebih kecil atau fleksibel sesuai kemampuan komputer sehingga terlihat membutuhkan lebih sedikit data pelatihan, serta lebih akurat karena dapat melakukan ekstraksi fitur secara mandiri dan detail serta lebih mudah untuk melibatkan lebih banyak data (Big Data).



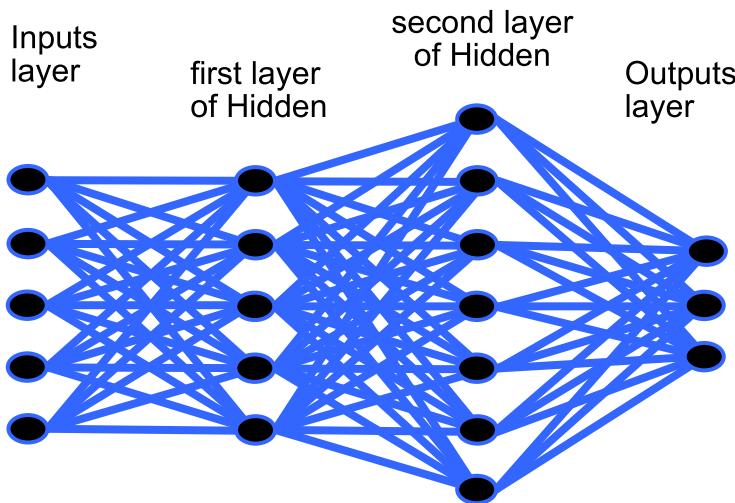
Gambar 15.2 (a) Tradisional Machine Learning, (b) Transfer Learning

## 15.2 Welcome to Deep Learning

Deep Learning (DL) / Pembelajaran Mendalam/ Belajar Secara Mendalam adalah salah satu cabang dari Machine Learning yang terdiri dari algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. DL sangat baik untuk diterapkan pada supervised learning, unsupervised learning dan semi-supervised learning maupun untuk reinforcement learning dalam berbagai aplikasi seperti pengenalan citra, suara, klasifikasi teks, dan sebagainya.

Model pada DL pada dasarnya dibangun berdasarkan Jaringan saraf tiruan (Neural Network), yang risetnya sudah berlangsung sejak era 80an namun baru-baru ini kembali bangkit dengan adanya komputer yang semakin cepat apalagi ditambah dengan adanya teknologi pada Big Data, misal pada hadoop dan spark berbasis multi node cluster, maupun pemrosesan secara paralel berbasis GPU, etc.

Jika suatu jaringan memiliki lebih dari 3 layer, maka jaringan tersebut termasuk Deep Network.



Gambar 15.3 Ilustrasi arsitektur pada Deep Learning

Model dari arsitektur pada Gambar 15.3 tersebut dapat dibuat dalam bentuk persamaan. Sebagai dasar pemahaman untuk uraian dari hidden layer yang banyak, dalam pembuktian untuk mendapatkan persamaan estimasinya dapat dihitung dari arsitektur pada Gambar 15.4 menggunakan arsitektur algoritma Extreme Learning Machine (ELM) yang nantinya dapat dikembangkan pembuktiannya menjadi Deep Extreme Learning Machine (Deep ELM). Berikut uraian penurunan untuk pembuktian dan penentuan persamaan dengan metode least square estimator (LSE) untuk mendapatkan beta estimator dari Artificial Neural Network (ANN) dari jenis bagian algoritmanya yang memiliki nama ELM.

$$H\beta = Y$$

$$H\beta + \varepsilon = Y$$

$$\varepsilon = Y - H\beta$$

$$\begin{aligned} \min_{\beta} (\varepsilon^T \varepsilon) &= \min_{\beta} (Y - H\beta)^T (Y - H\beta) \\ &= \min_{\beta} (Y^T Y - Y^T H\beta - (H\beta)^T Y + (H\beta)^T (H\beta)) \end{aligned}$$

$$\begin{aligned}
 &= \min_{\beta} (Y^T Y - Y^T H\beta - \beta^T H^T Y + \beta^T H^T (H\beta)) \\
 &= \min_{\beta} (Y^T (Y - H\beta) - \beta^T H^T (Y - H\beta)) \\
 \frac{\partial}{\partial \beta} (\varepsilon^T \varepsilon) &= \frac{\partial}{\partial \beta} (Y^T (Y - H\beta) - \beta^T H^T (Y - H\beta)) = 0 \\
 &= \frac{\partial}{\partial \beta} (Y^T Y - Y^T H\beta - \beta^T H^T Y - \beta^T H^T H\beta) = 0 \\
 &= \frac{\partial}{\partial \beta} (Y^T Y) - \frac{\partial}{\partial \beta} Y^T H\beta - \frac{\partial}{\partial \beta} \beta^T H^T Y + \frac{\partial}{\partial \beta} \beta^T H^T H\beta = 0 \\
 &= 0 - (Y^T H)^T - (Y^T H)^T + 2H^T H\beta = 0
 \end{aligned} \tag{1}$$

Hasil turunan dari  $\beta$  berdasarkan Persamaan (1) dapat disederhanakan lebih terperinci lagi. Di mana variabel dari  $\beta$  diubah menjadi  $\hat{\beta}$  sebagai nilai estimatornya yang secara langsung dapat dihitung dengan hanya mengoperasikan matrik  $H$  dengan  $Y$  seperti pada Persamaan (2).

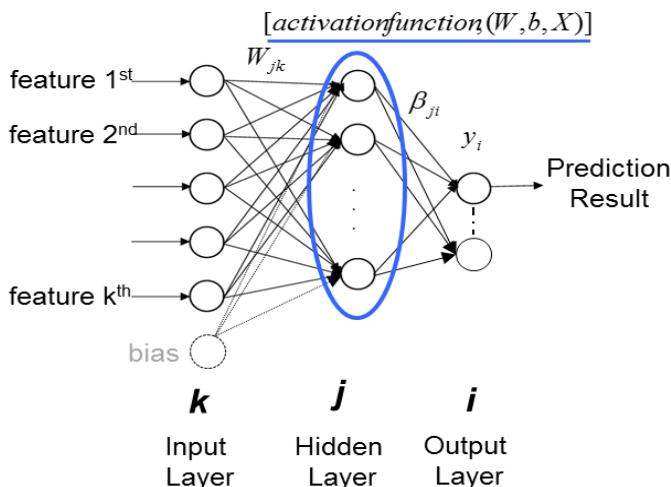
$$\begin{aligned}
 &-H^T Y - H^T Y + 2H^T H\hat{\beta} = 0 \\
 &-2H^T Y + 2H^T H\hat{\beta} = 0 \\
 &-H^T Y + H^T H\hat{\beta} = 0 \\
 &H^T H\hat{\beta} = H^T Y \\
 &(H^T H)^{-1}(H^T H)\hat{\beta} = (H^T H)^{-1}H^T Y \\
 &\hat{\beta} = (H^T H)^{-1}H^T Y
 \end{aligned} \tag{2}$$

Pada ELM, Persamaan (2) dapat dituliskan menjadi bentuk yang lebih sederhana, yaitu pada Persamaan (3). Kemudian pada proses trainingnya dengan Persamaan (4), sedangkan proses testingnya dengan Persamaan (5).

$$\hat{\beta} = H^+ \cdot Y \tag{3}$$

$$Y = \left( H = \frac{1}{(1 + \exp(-(\mathbf{X}_{train} \cdot W^T + ones(N_{train}, 1) \cdot b)))} \right) \left( (H)^T \cdot (H)^{-1} \cdot H^T \right) Y \tag{4}$$

$$\hat{Y} = \left( \frac{1}{(1 + \exp(-(\mathbf{X}_{test} \cdot W^T + ones(N_{test}, 1) \cdot b)))} \right) \left( (H)^T \cdot (H)^{-1} \cdot H^T \right) Y \tag{5}$$



Gambar 15.4 Ilustrasi arsitektur pada Non-Deep ELM

Beberapa varian dari arsitektur Deep Learning (DL) yang telah banyak digunakan pada berbagai bidang.

- Deep neural networks, seperti jaringan syaraf tiruan (JST) backpropagation dengan banyak layer tersembunyi.
- Convolutional neural networks (CNN)
- Deep belief networks (DBN)
- Recurrent neural networks (RNN)
- Restricted Boltzmann Machine (RBM)
- Deep Reinforcement Learning (DRL)
- Deep Q Learning (DQL)
- Hierarchical Temporal Memory (HTM)
- Stacked Denoising Autoencoders (SDA)
- Dan lainnya

Semua arsitektur di atas sudah banyak digunakan pada kasus-kasus di computer vision, automatic speech recognition, natural language processing, audio recognition dan bioinformatics. DL merupakan *rebranding* (kemasan baru) dari neural networks. Hal yang identik dari DL adalah adanya unsupervised/supervised features. Nanti akan dibahas lebih lanjut pada bab-bab selanjutnya.

## 15.3 Prinsip Kerja Deep Learning

Deep Learning merupakan pembelajaran yang berbasis pada fitur yang berbentuk hirarki, yang mana bentuk fitur hirarki tersebut dapat diskalakan dalam ukuran tertentu yang dapat disesuaikan dengan kasus yang diproses. Sehingga membuat algoritma Deep Learning mampu untuk melakukan ekstraksi fitur otomatis dari data mentah secara detail, karena proses ekstraksi yang dilakukan menggunakan struktur eksplorasi, dimana fitur-fitur yang dieksplorasi tersebut tidak mungkin dapat dilihat secara kasat mata. Hal ini dikarenakan distribusi pembeda kelas data biasanya terlalu dalam, sehingga dari fitur level tinggi harus ditransformasi terlebih dahulu ke fitur yang paling rendah yang dapat mudah dipahami oleh mesin pembelajaran. Artinya jika fitur level rendah saja dapat mudah diidentifikasi oleh Deep Learning, apalagi yang fitur level tinggi. Perpaduan inilah yang menjadikan Deep Learning mampu menghasilkan representasi fitur yang baik dan optimal. Penggalian fitur secara eksplorasi ini mencoba untuk mengolah suatu *space* atau *region* pada posisi tertentu sampai dicoba untuk digali dan difilter sampai sedalam-dalamnya. Sedangkan jika secara eksplorasi, Deep Learning berusaha untuk mengumpulkan target spae atau region sebanyak-banyaknya atau seluas-luasnya dengan teknik yang bisa sama atau berbeda pada saat melakukan eksplorasi. Berikut adalah prinsip kerja dari algoritma Deep Learning.

- Buat arsitektur yang tepat, terutama pada jenis metode Deep Learning yang digunakan sesuai dengan kebutuhan dari kompleksitas kasus yang akan diolah dan diselesaikan. Arsitektur tersebut terdiri dari layer input, hidden (darknet/layer gelap/tersembunyi), dan output.
- Siapkan data atau item apa saja yang diperlukan ketika proses pelatihan dan pengujian Deep Learning.
- Pertimbangkan untuk tempat implementasinya (misal di lokal atau menggunakan cloud) dan kode programnya (membuat kode program *from scratch*, menggabungkan dengan membuat sebagian secara *scratch* dan menggunakan sebagian kode dari library, atau menggunakan *full library*).
- Lakukan pengujian yang sesuai dengan standar dari algoritma Deep Learning, mulai dari parameter, sampai ke pengujian bentuk arsitekturnya. Misal pengujinya secara *waterfall* (sekuensial tanpa loop) atau secara *recycle* (sekuensial dengan

loop tertentu). Di mana pengujian *recycle* ini mengkondisikan pengujianya diulang-ulang sampai beberapa kali siklus (perulangan), misal terdapat 3 macam pengujian, pertama menguji parameter A, lalu hasil nilai dari parameter A yang terbaik akan digunakan untuk pengujian parameter B, lalu hasil pengujian nilai dari parameter A dan B yang terbaik akan digunakan untuk pengujian parameter C, lalu hasil pengujian nilai dari parameter B dan C yang terbaik akan digunakan untuk pengujian parameter A (loop ke-1 dan seterusnya sampai iterasi tertentu), dan sebaiknya iterasi tersebut dihentikan ketika nilai paramter A, B dan C sudah stabil serta nilai akurasinya sudah stabil pada nilai yang paling tinggi.

- Pikirkan bagaimana mengoptimasi lagi algoritma Deep Learning dari hasil yang didapatkan saat ini.

## 15.4 How to Build Core Engine Deep Learning

Pada saat proses implementasi sekaligus untuk menentukan cara membuat kode programnya, Anda dapat menggunakan beberapa teknik, misalnya tempat implementasinya (misal di lokal atau menggunakan cloud) dan kode programnya (membuat kode program from scratch, menggabungkan dengan membuat sebagian secara scratch dan menggunakan sebagian kode dari library, atau menggunakan full library) atau dengan cara lainnya. Berikut beberapa toolboxes atau kumpulan library terkait Deep Learning yang dapat digunakan pada komputasi secara non-distribusi atau dengan terdistribusi (menggunakan prinsip kerja pengolahan Big Data) maupun paralel (dengan GPU) dengan berbagai macam bahasa pemrograman, misal python, java, c/c++/c#, golang dan lainnya.

- Torch
- Keras
- TensorFlow
- Scikit-Learn
- Theano
- Caffe
- CNTK
- Deeplearning4j
- dan lainnya

## 15.5 Tugas Kelompok

1. Jelaskan perbedaan algoritma machine learning non-deep dengan deep learning (DL), dan sebutkan macam-macam varian dari arsitektur DL yang telah banyak digunakan pada berbagai bidang, beserta karakteristiknya?

2. Berdasarkan Persamaan (1), jelaskan bagian berikut

$$\frac{\partial}{\partial \beta} (\varepsilon^T \varepsilon) = \frac{\partial}{\partial \beta} (Y^T (Y - H\beta) - \beta^T H^T (Y - H\beta)) = 0$$

3. Lakukan instalasi, minimal software berikut yang dibutuhkan untuk implementasi Deep Learning pada laptop/ PC Anda:

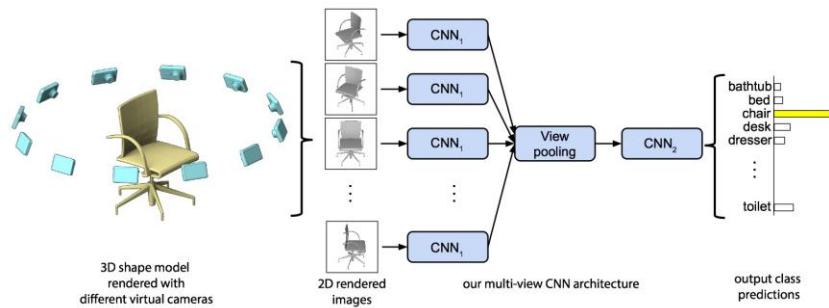
- Anaconda + TensorFlow + pyTorch/Keras/Scikit-learn/etc

4. Jelaskan secara singkat prinsip-prinsip kerja dari algoritma Deep Learning!

5. Jelaskan perbedaan konsep Non-Distributed Programming, Distributed Programming dan GPU Programming pada algoritma Deep Learning!

## BAB 16 Algoritma Convolutional Neural Networks (CNN)

Convolutional Neural Network (Convnets/ CNN) sangat mirip dengan jaringan saraf tiruan yang standar yang dapat divisualisasikan sebagai kumpulan neuron atau node atau unit yang disusun sebagai graf asiklik (graf yang tanpa adanya loop di dalamnya). CNN memiliki ciri khas yaitu dapat lapisan tersembunyi yang hanya terhubung ke subset neuron di lapisan sebelumnya. Karena konektivitas tersebut, CNN dapat mempelajari fitur secara implisit. Arsitektur CNN menghasilkan ekstraksi fitur hirarki yaitu filter yang dilatih untuk tujuan spesifik, misal pada lapisan pertama biasanya difokuskan pada identifikasi tepian atau fluktuasi warna, kemudian lapisan kedua biasanya lebih ke identifikasi bentuk, dan filter lapisan berikutnya biasanya lebih diarahkan untuk mempelajari bagian-bagian parsialisasi dari objek, baik yang terlihat sedikit atau sebagian maupun yang terlihat cukup banyak serta lapisan terakhir digunakan untuk mengidentifikasi objek.

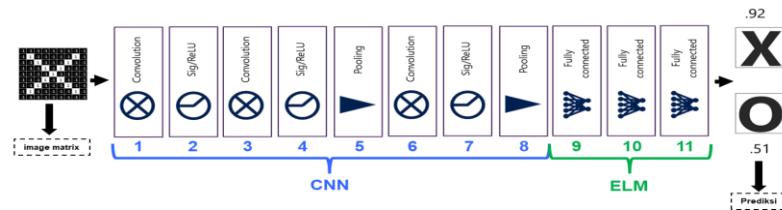


Gambar 16.1 Ilustrasi Ekstrasi Fitur CNN pada Objek 3D ke 2D

### 16.1 Struktur Algoritma CNN

Pada Gambar 16.2, Set Map untuk Network Arsitektur Pre-trained atau model CNN-nya bisa menggunakan ResNet, Visual Geometry Group (VGG), Inception atau lainnya. Pada manualisasi yang kami gunakan, Map tersebut kami buat sederhana dengan tetap memenuhi kaidah CNN pada Deep Learning, yaitu dengan struktur CNN-ELM atau Simplified Deep Learning CNN – Extreme Learning Machines (SDL-

ELM) terdiri dari lapisan input, lapisan output dan beberapa lapisan tersembunyi diatur sebagai lapisan konvolusi satu kesatuan, diikuti oleh lapisan penggabungan (*pooling layer*). Jumlah konvolusi dan *pooling layer*, tergantung pada kerumitan kasus. Lapisan konvolusi terdiri dari beberapa kelompok fitur dan pooling layer terdiri dari ringkasan beberapa kelompok fitur. Berikut adalah langkah-langkah detail dari CNN-ELM.



Gambar 16.2 Map contoh CNN-ELM

1. Membuat map SDL-ELM yang relevan (hal ini didesain oleh pengguna) dengan mengkombinasikan *Convolution*, *Sig/ReLU*, *Pooling*, dan *Fully Connected process*, seperti pada Gambar 16.2.
2. Tentukan nilai Parameter.
  - a. Untuk proses normalisasi dari nilai fitur, yaitu:  
 $\text{maxActual (mac)} = 300$ ;  $\text{minActual (mic)} = 0$ ;  
 $\text{maxNorm (mao)} = 1$ ;  $\text{minNorm (mio)} = 0$ ;
  - b. Untuk proses convolution. Set, misal dengan tiga macam filters:  
yang mana, 1<sup>st</sup> (conv11): average filter, 2<sup>nd</sup> (conv12): max filter, dan 3<sup>rd</sup> (conv13): std filter, std (standard deviation).  
 $\text{numFilter} = 3$ ; dan, % banyaknya *padding* ( $k$ ), filter matrix size ( $k \times k$ ), misal  $k = 3$ ;
  - c. Untuk proses pooling:  
yang mana, % *filter matrix size* [*windows\_size* x *windows\_size*], misal  
 $\text{windows\_size}=2$ ;
3. Proses Pelatihan
  - a. Preprocessing  
[*numData*,...  
*numFeature*,*target*,*norm*]=FnPreProses('datatrainClassify',..  
..  
*mac*, *mic*, *mao*, *mio*);

- 1. Load data training, get numData dan numFeature.
- 2. Create "image matrix" untuk tiap data awal (hanya nilai fiturnya saja yang diambil) dari dataset, yaitu menggunakan Repmat technique.
- 3. Normalization dari semua "image matrix" data.  

$$\text{norm}\{i\} = (((a\{i\}-\text{mic}) ./ (\text{mac}-\text{mic})) * (\text{mao}-\text{mio})) + \text{mio};$$
 yang mana  $a\{i\}$  adalah tiap element matrix data  $i$ -th, dan  $\text{norm}\{i\}$  mendefinisikan sebagai suatu matrix dengan ukuran [numFeature x numFeature], seperti berikut

image matrix 1<sup>st</sup> =

|     |     |     |    |
|-----|-----|-----|----|
| 253 | 131 | 259 | 95 |
| 253 | 131 | 259 | 95 |
| 253 | 131 | 259 | 95 |
| 253 | 131 | 259 | 95 |

norm image matrix 1<sup>st</sup> =

|        |        |        |        |
|--------|--------|--------|--------|
| 0.8433 | 0.4367 | 0.8633 | 0.3167 |
| 0.8433 | 0.4367 | 0.8633 | 0.3167 |
| 0.8433 | 0.4367 | 0.8633 | 0.3167 |
| 0.8433 | 0.4367 | 0.8633 | 0.3167 |

- b. Feature Abstraction dengan **CNN** (berdasar Gambar 16.2).

- 1. Convolution Init.

$hC = \text{FnConvDL}(\text{norm}, \text{numData}, k);$

jika  $k=3$ , maka expand edge norm image matrix (padding) dengan zero value sebanyak  $\text{pad\_size} = (k-1)/2 = (3-1)/2=1$ , dimana  $k$  adalah bilangan ganjil  $\geq 3$ . Berikut contohnya,

$$\text{padding}(\text{norm image matrix } 1^{\text{st}}) = \left( \begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right)$$

Dimana ukuran dari green box adalah  $[k \times k]$

Persamaan dan Ilustrasi dari filter 1<sup>st</sup>: average filter

$$hC\{1\}_{[4 \times 4]} = \frac{1}{k \cdot k} \sum_{i=1}^k \sum_{j=1}^k f(i, j) \quad (6)$$

$$hC\{1\}_{[4 \times 4]} = \frac{1}{4} \left( \begin{array}{cccc} 1 & 2 & 3 & 4 \\ 2 & \dots & \dots & \dots \\ 3 & \dots & \dots & \dots \\ 4 & \dots & \dots & \dots \end{array} \right)$$

Persamaan dan Ilustrasi hasil dari filter 2<sup>nd</sup>: max filter

$$hC\{1\}_{[4 \times 4]} = \text{Max} \left( \bigcup_{i=1}^k \bigcup_{j=1}^k f(i, j) \right) \quad (7)$$

$$hC\{2\}\{1\}_{[4 \times 4]} = \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} \left( \begin{array}{cccc} 0.8433 & ..... & ..... & ..... \\ ..... & ..... & ..... & ..... \\ ..... & ..... & ..... & ..... \\ ..... & ..... & ..... & ..... \end{array} \right)$$

Persamaan dan Ilustrasi hasil dari filter 3<sup>rd</sup>: std filter

$$hC\{1\}\{..\}_{[4 \times 4]} = STD\left(\bigcup_{i=1}^k \bigcup_{j=1}^k f(i, j)\right) \quad (8)$$

$$hC\{3\}\{1\}_{[4 \times 4]} = \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} \left( \begin{array}{cccc} 0.3667 & ..... & ..... & ..... \\ ..... & ..... & ..... & ..... \\ ..... & ..... & ..... & ..... \\ ..... & ..... & ..... & ..... \end{array} \right)$$

- 2. **Sigmoid/ReLU**

`hA=FnSigDL(hC,numFilter,numData);`

Misal menggunakan activation function sigmoid:

$$hA\{1\}\{1,1\} = 1/(1+\exp(-hC\{1\}\{1,1\})) \quad (9)$$

- 3. Convolution In.

`hC=FnConvInDL(hA,numData,k,numFilter);`

- 4. **Sigmoid/ReLU**

`hA=FnSigDL(hC,numFilter,numData);`

- 5. Pooling

`hP=FnPooDL(hA,windows_size,numFilter,numData);`

Menghitung besarnya nilai pad, dimana ml, nl masing-masing adalah banyaknya rows dan column dari  $hA\{1\}\{1\}$ .

`padX=(ceil(nl/windows_size)*windows_size)-nl;`

`padY=(ceil(ml/windows_size)*windows_size)-ml;`

`mpoolI=sqrt((ml+padY)*(nl+padX)/windows_size^2);`

`npooll = mpoolI;`

jika  $padX > 0$  atau  $padY > 0$ , maka padding  $hA\{1\}\{1\}$ ,  $padX$  expand tepian setelah kolom terakhir dari matrix  $hA\{1\}\{1\}$ ,  $padY$  expand tepian setelah baris terakhir dari matrix  $hA\{1\}\{1\}$ , eg `padX = 2, padY = 2`

$$\text{padding}(hA\{1\}\{1\}) = \begin{pmatrix} 0.4574 & 0.4360 & 0.4357 & 0.4552 & 0 & 0 \\ 0.4376 & 0.4065 & 0.4059 & 0.4340 & 0 & 0 \\ 0.4376 & 0.4065 & 0.4059 & 0.4340 & 0 & 0 \\ 0.4574 & 0.4360 & 0.4357 & 0.4552 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

dimana ukuran dari black box adalah [windows\_size x windows\_size]

$$hP\{1\}\{1\}_{[\text{mpool} \times \text{npool}]} = \begin{pmatrix} 0.4574 & \dots \\ \dots & \dots \end{pmatrix}$$

- 6. Convolution In.  
 $hC = \text{FnConvInDL}(hP, \text{numData}, k, \text{numFilter});$
- 7. Sigmoid/ReLU  
 $hA = \text{FnSigDL}(hC, \text{numFilter}, \text{numData});$
- 8. Pooling  
 jika ukuran ( $hA[i][j]$ ) = [2 x 2], maka set windows\_size = 1  
 $hP = \text{FnPoolDL}(hA, \text{windows\_size}, \text{numFilter}, \text{numData});$
- c. Fully Connected dengan ELM (berdasar Gambar 16.2).
  - 9. Fully connected 1<sup>st</sup>  
 Misal, num\_neuron\_hidden\_layer=5;  
 $[hFC11, W11, Bias11, Beta11] = \text{FnELMtrainClassify}(hP, \text{target}, \dots, \text{num_neuron_hidden_layer}, \text{numData}, \text{numFilter});$

Di bawah ini adalah ilustrasi bagaimana mendapatkan  $X(1,:)$  sebagai data pertama untuk dimasukkan pada Fully connected 1<sup>st</sup>,

$$X(1,:) \text{ merger from } hP\{1\}\{1\}, hP\{2\}\{1\}, \text{ and } hP\{3\}\{1\}$$

$$hP\{1\}\{1\} = \begin{pmatrix} 0.4495 & 0.4495 \\ 0.4495 & 0.4495 \end{pmatrix} \quad hP\{2\}\{1\} = \begin{pmatrix} 0.3950 & 0.3950 \\ 0.3950 & 0.3950 \end{pmatrix} \quad hP\{3\}\{1\} = \begin{pmatrix} 0.4351 & 0.4351 \\ 0.4351 & 0.4351 \end{pmatrix}$$

For example, the element is given in the form of a variable:

$$hP\{1\}\{1\} = \begin{pmatrix} a1 & c1 \\ b1 & d1 \end{pmatrix} \quad hP\{2\}\{1\} = \begin{pmatrix} a2 & c2 \\ b2 & d2 \end{pmatrix} \quad hP\{3\}\{1\} = \begin{pmatrix} a3 & c3 \\ b3 & d3 \end{pmatrix}$$

Here's how to generate X (1, :)

$$X(1,:) = [a1 \ b1 \ c1 \ d1 \ a2 \ b2 \ c2 \ d2 \ a3 \ b3 \ c3 \ d3]$$

$$X(1,:) = [0.4495 \ 0.4495 \ 0.4495 \ 0.4495 \ 0.3950 \ 0.3950 \ 0.3950 \ 0.3950 \ 0.4351 \ 0.4351 \ 0.4351 \ 0.4351]$$



$$X(8,:) = \dots$$

- 10. Fully connected 2<sup>nd</sup>

- Eg, num\_neuron\_hidden\_layer=7;  
[hFC12,W12,Bias12,Beta12]=FnELMtrainClassify(hP,targ  
et,...  
    num\_neuron\_hidden\_layer,numData,numFilter);
- 11. Fully connected 3<sup>rd</sup>  
Eg, num\_neuron\_hidden\_layer=4;  
[hFC13,W13,Bias13,Beta13]=FnELMtrainClassify(hP,targ  
et,...  
    num\_neuron\_hidden\_layer,numData,numFilter);
4. Proses Pengujian
- a. Preprocessing  
[numData2,...  
numFeature2,target2,norm2]=FnPreProses('dataatestClassif  
y.xlsx',...  
mac, mao, mao, mao);
  - b. Feature Abstraction dengan **CNN** (berdasar Gambar 16.2).
    - 1. Convolution Init.  
hC2=FnConvDL(norm2,numData2,k);
    - 2. **Sigmoid/ReLU**  
hA2=FnSigDL(hC2,numFilter,numData2);
    - 3. Convolution In.  
hC2=FnConvInDL(hA2,numData2,k,numFilter);
    - 4. **Sigmoid/ReLU**  
hA2=FnSigDL(hC2,numFilter,numData2);
    - 5. Pooling  
hP2=FnPoolDL(hA2,windows\_size,numFilter,numData2)  
;
    - 6. Convolution In.  
hC2=FnConvInDL(hP2,numData2,k,numFilter);
    - 7. **Sigmoid/ReLU**  
hA2=FnSigDL(hC2,numFilter,numData2);
    - 8. Pooling  
jika ukuran (hA2{i}{j}) = [2 x 2], maka set windows\_size =  
1  
hP2=FnPoolDL(hA2,windows\_size,numFilter,numData2)  
;
  - c. Fully Connected dengan **ELM** (berdasar Gambar 16.2).
    - 9. Fully connected 1<sup>st</sup>  
[Akurasi1,kelasPrediksi1,Ytest\_predict1]=...  
FnELMtestClassify(hP2,target2,...

W11,Bias11,Beta11,numData2,numFilter);  
o 10. Fully connected 2<sup>nd</sup>  
[Akurasi2,kelasPrediksi2, Ytest\_predict2]=...  
FnELMtestClassify(hP2,target2,...  
W12,Bias12,Beta12,numData2,numFilter);  
o 11. Fully connected 3<sup>rd</sup>  
[Akurasi3,kelasPrediksi3,Ytest\_predict3]=...  
FnELMtestClassify(hP2,target2,...  
W13,Bias13,Beta13,numData2,numFilter);  
d. Voting untuk mendapatkan hasil final  
% proses voting dari beberapa "Fully connected" testing  
ELM dengan membandingkan kelas aktual (target) dengan  
kelas prediksi  
CompareKelas=[target2' kelasPrediksi1 kelasPrediksi2  
kelasPrediksi3];  
  
% hitung frekuensi atau kemunculan untuk bahan voting  
AllKelasPrediksi= CompareKelas(:,2:end)  
kelasPrediksiVoting= mode(AllKelasPrediksi)';  
num\_data\_test = numData2;  
  
% hitung akurasi final dari hasil voting  
nBenar=numel(find(target2'-kelasPrediksiVoting==0));  
akurasi=(nBenar/num\_data\_test)\*100

Jadi, langkah terakhir dari algoritma SDLCNN-ELM untuk proses mendapatkan nilai evaluasi dalam bentuk nilai MAD/MAPE/etc untuk kasus regresi, Silhouette Index/etc untuk kasus Clustering dan Akurasi/etc untuk kasus klasifikasi serta kasus lainnya.

## 16.2 Studi Kasus: CNN untuk Menelesaikan Kasus Prediksi maupun Klasifikasi

Dalam memahami lebih detail tentang algoritma CNN, berikut diberikan contoh penghitungan yang dapat digunakan untuk model prediksi maupun klasifikasi misal dengan data time series dari curah hujan yang mencakup wilayah tertentu saja, misal pada wilayah Karangploso Kabupaten Malang, 8 tahun dari tahun 2004-2011, dalam rentang waktu bulanan seperti pada Tabel 16.1.

Tabel 16.1 Data Curah Hujan untuk Prediksi

| No | Data         | X <sub>1</sub> | X <sub>2</sub> | X <sub>3</sub> | X <sub>4</sub> | Y   |
|----|--------------|----------------|----------------|----------------|----------------|-----|
| 1  | Januari 2004 | 253            | 131            | 259            | 95             | 263 |
| 2  | Januari 2005 | 131            | 259            | 95             | 263            | 50  |
| 3  | Januari 2006 | 259            | 95             | 263            | 50             | 154 |
| 4  | Januari 2007 | 95             | 263            | 50             | 154            | 73  |
| 5  | Januari 2008 | 263            | 50             | 154            | 73             | 141 |
| 6  | Januari 2009 | 50             | 154            | 73             | 141            | 146 |
| 7  | Januari 2010 | 154            | 73             | 141            | 146            | 87  |
| 8  | Januari 2011 | 73             | 141            | 146            | 87             | 139 |

Sedangkan jika Tabel 16.1 diarahkan untuk klasifikasi, maka nilai Y harus menjadi label kelas. Pada manualisasi disini akan diberikan contoh, yaitu dengan memisalkan dataset yang digunakan adalah sebagai berikut, pada Tabel 16.2.

Tabel 16.2 Data Curah Hujan untuk Klasifikasi

| No | X <sub>1</sub> | X <sub>2</sub> | X <sub>3</sub> | X <sub>4</sub> | Kelas |
|----|----------------|----------------|----------------|----------------|-------|
| 1  | 253            | 131            | 259            | 95             | 1     |
| 2  | 131            | 259            | 95             | 263            | 1     |
| 3  | 259            | 95             | 263            | 50             | 1     |
| 4  | 95             | 263            | 50             | 154            | 1     |
| 5  | 263            | 50             | 154            | 73             | 2     |
| 6  | 50             | 154            | 73             | 141            | 2     |
| 7  | 154            | 73             | 141            | 146            | 3     |
| 8  | 73             | 141            | 146            | 87             | 3     |

Hal yang pertama kali harus dilakukan sebelum melakukan proses training dan testing adalah dengan set nilai dari parameter-parameter yang digunakan. Penentuan nilai parameter ini dapat didasarkan dari kondisi data maupun dari pertimbangan dari hasil penelitian-penelitian sebelumnya.

- Untuk proses normalisasi nilai dari fitur

```
max = 300;
min = 0;
max2 = 1;
min2 = 0;
```

- Untuk proses Convolution. Set, misal dengan 3 macam filter

```
% ke-1 (conv11) : average filter
% ke-2 (conv12) : max filter
% ke-3 (conv13) : std filter, std (standard deviasi)
bykFilter = 3;

% banyaknya padding, ukuran matrik filter (k x k) pada
proses convolution
k = 3;
```

- Untuk proses Pooling

```
% sebagai filter [windows_size x windows_size]
% pada proses pooling
windows_size=2;
```

## 16.2.1 Proses Training

Berikut beberapa langkah-langkah untuk melakukan proses training pada algoritma CNN.

- pre-Proses data training, berikut kodingnya

```
[bykData,byk_fitur,target,norm]=FnPreProses('datatrainClas
sify.xlsx',...
max, min, max2, min2);
byk_kelas=numel(unique(target)) = 3
dimana target = kolom "kelas" pada Tabel di bawah.
```

berikut detail prosesnya:

1. Baca file data train, get bykData dan byk\_fitur

| No | Fitur ke-1 | Fitur ke-2 | Fitur ke-3 | Fitur ke-4 | kelas |
|----|------------|------------|------------|------------|-------|
| 1  | 253        | 131        | 259        | 95         | 1     |
| 2  | 131        | 259        | 95         | 263        | 1     |
| 3  | 259        | 95         | 263        | 50         | 1     |
| 4  | 95         | 263        | 50         | 154        | 1     |
| 5  | 263        | 50         | 154        | 73         | 2     |
| 6  | 50         | 154        | 73         | 141        | 2     |
| 7  | 154        | 73         | 141        | 146        | 3     |
| 8  | 73         | 141        | 146        | 87         | 3     |

bykData = 8 dan byk\_fitur = 4

2. Membentuk dataset sebagai matriks square [byk\_fitur x byk\_fitur] menggunakan konsep replika matriks  
`repmat(data{i},[byk_fitur 1])`, misal data ke-1

| No | Fitur ke-1 | Fitur ke-2 | Fitur ke-3 | Fitur ke-4 |
|----|------------|------------|------------|------------|
| 1  | 253        | 131        | 259        | 95         |

↓  
Dikonversi  
menjadi matrik  
persegi berikut

data ke-1 =

|     |     |     |    |
|-----|-----|-----|----|
| 253 | 131 | 259 | 95 |
| 253 | 131 | 259 | 95 |
| 253 | 131 | 259 | 95 |
| 253 | 131 | 259 | 95 |

↓  
↓

data ke-8 =

|    |     |     |    |
|----|-----|-----|----|
| 73 | 141 | 146 | 87 |
| 73 | 141 | 146 | 87 |
| 73 | 141 | 146 | 87 |
| 73 | 141 | 146 | 87 |

3. Menghitung hasil normalisasi dengan rumus (norm)  
`norm{i}=(((a{i}-min)./(max-min))*(max2-min2))+min2;`

data ke-1 =

|     |     |     |    |
|-----|-----|-----|----|
| 253 | 131 | 259 | 95 |
| 253 | 131 | 259 | 95 |
| 253 | 131 | 259 | 95 |
| 253 | 131 | 259 | 95 |

↓  
↓  
↓

norm data ke-1 = 
$$\begin{pmatrix} 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \end{pmatrix}$$

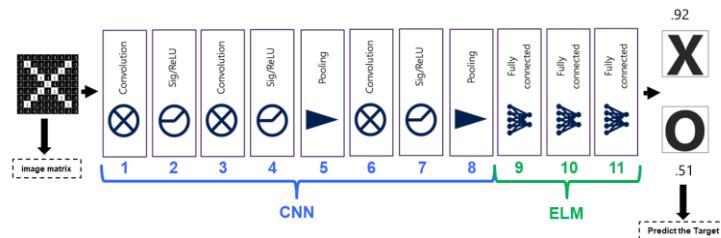


data ke-8 = .....

dimana `a{i}` menyatakan akan mengambil tiap elemen pada matrik data ke-i. Dan `norm{i}` menyatakan matrik dengan ukuran [byk\_fitur x byk\_fitur].

byk\_kelas = 3

- Masuk ke Proses misal dgn **CNN** dan **ELM** berikut:



### 1. Convolution di awal

```
hC=FnConvDL(norm,bykData,k);
```

berikut detail prosesnya: (misal dengan 3 macam filter)

Untuk Filter ke-1: average filter (norm data ke-1 sampai ke-8, bykData=8, k=3)

$$\text{norm data ke-1} = \begin{pmatrix} 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \end{pmatrix}$$

↓  
norm data ke-8 = ....

Ketika  $k=3$ , maka lakukan penambahan tepian matrik (padding) sebanyak  $(k-1)/2 = 1$ , dengan nilai  $\text{pad\_size} = (3-1)/2=1$ , dimana nilai  $k$  harus bilangan ganjil  $\geq 3$ .

Untuk Filter ke-1: **average filter** (norm data ke-1 sampai ke-8, bykData=8, k=3)

$$\text{norm data ke-1} = \begin{pmatrix} 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \end{pmatrix}$$

Untuk menentukan matrik  $hC\{1\}\{1\}$  pada baris 1 kolom 1.

padding (norm data ke-1) =

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Ketika  $k=3$ , maka lakukan penambahan tepian matrik (padding) sebanyak  $(k-1)/2 = 1$ , dengan nilai  $\text{pad\_size} = (3-1)/2=1$ , dimana nilai  $k$  harus bilangan ganjil  $\geq 3$ .

hasil Convolution (hC):

$$hC\{1\}\{1\}_{1,1} = \text{average}(\boxed{\phantom{00}}) = 0.2844$$

Ukuran filter kotak hijau =  $[k \times k]$

$$hC\{1\}\{1\}_{4 \times 4} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 0.2844 & \dots & \dots \\ 3 & \dots & \dots & \dots \\ 4 & \dots & \dots & \dots \end{pmatrix}$$

Filter kotak hijau tersebut akan dijalankan secara bertahap ke region di sebelahnya, misal dengan langkah pergeseran sebanyak 1 pixel. Besarnya pergeseran ini dapat juga menggunakan lebih dari 1 pixel.

Hasil pergeseran (proses konvolusi) berikutnya adalah sebagai berikut

Untuk Filter ke-1: **average filter** (norm data ke-1 sampai ke-8, bykData=8, k=3)

$$\text{norm data ke-1} = \begin{bmatrix} 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \end{bmatrix}$$

Ketika  $k=3$ , maka lakukan penambahan tepian matriks (padding) sebanyak  $(k-1)/2 = 1$ , dengan nilai pad\_size =  $(3-1)/2=1$ , dimana nilai  $k$  harus bilangan ganjil  $\geq 3$ .

Untuk menentukan matrik  $hC\{1\}\{1\}$  pada baris 1 kolom 2.

$$\text{padding (norm data ke-1)} = \begin{array}{c} 1,2 \\ \left( \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 \\ 0 & 0.8433 & 0.4367 & 0.8633 \\ 0 & 0.8433 & 0.4367 & 0.8633 \\ 0 & 0.8433 & 0.4367 & 0.8633 \\ 0 & 0 & 0 & 0 \end{array} \right) \end{array}$$

Menambahkan tepian matriks norm sebanyak pad\_size=1, dengan nilai 0.

hasil Convolution (hC):

$$hC\{1\}\{1\}_{1,2} = \text{average}(\boxed{\phantom{00}}) = 0.4763$$

$$hC\{1\}\{1\}_{4 \times 4} = \begin{array}{c} 1 \quad 2 \quad 3 \quad 4 \\ \left( \begin{array}{cccc} 0.2844 & 0.4763 & \dots & \dots \\ 0.4267 & 0.7144 & 0.5389 & 0.3933 \\ \dots & \dots & \dots & \dots \\ 0.2844 & 0.4763 & 0.3593 & 0.2622 \end{array} \right) \end{array}$$

Sampai pada pergeseran terakhir, seperti berikut

$$\text{norm data ke-1} = \begin{bmatrix} 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \end{bmatrix}$$

Ketika  $k=3$ , maka lakukan penambahan tepian matriks (padding) sebanyak  $(k-1)/2 = 1$ , dengan nilai pad\_size =  $(3-1)/2=1$ , dimana nilai  $k$  harus bilangan ganjil  $\geq 3$ .

Untuk menentukan matrik  $hC\{1\}\{1\}$  pada baris 4 kolom 4.

$$\text{padding (norm data ke-1)} = \begin{array}{c} 4,4 \\ \left( \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 \\ 0 & 0.8433 & 0.4367 & 0.8633 \\ 0 & 0.8433 & 0.4367 & 0.8633 \\ 0 & 0.8433 & 0.4367 & 0.8633 \\ 0 & 0 & 0 & 0 \end{array} \right) \end{array}$$

Menambahkan tepian matriks norm sebanyak pad\_size=1, dengan nilai 0.

hasil Convolution (hC):

$$hC\{1\}\{1\}_{4,4} = \text{average}(\boxed{\phantom{00}}) = 0.2622$$

$$hC\{1\}\{1\}_{4 \times 4} = \begin{array}{c} 1 \quad 2 \quad 3 \quad 4 \\ \left( \begin{array}{cccc} 0.2844 & 0.4763 & 0.3593 & 0.2622 \\ 0.4267 & 0.7144 & 0.5389 & 0.3933 \\ 0.4267 & 0.7144 & 0.5389 & 0.3933 \\ 0.2844 & 0.4763 & 0.3593 & 0.2622 \end{array} \right) \end{array}$$

Lakukan sampai norm data ke-8

Kemudian lanjut untuk filter ke-2, pergeserannya juga sama:

Untuk Filter ke-2: **max filter** (norm data ke-1 sampai ke-8, bykData=8, k=3)

$$\text{norm data ke-1} = \begin{bmatrix} 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \end{bmatrix}$$

Ketika  $k=3$ , maka lakukan penambahan tepian matriks (padding) sebanyak  $(k-1)/2 = 1$ , dengan nilai pad\_size =  $(3-1)/2=1$ , dimana nilai  $k$  harus bilangan ganjil  $\geq 3$ .

Untuk menentukan matrik  $hC\{2\}\{1\}$  pada baris 1 kolom 1.

$$\text{padding (norm data ke-1)} = \begin{array}{c} 1,1 \\ \left( \begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right) \end{array}$$

Menambahkan tepian matriks norm sebanyak pad\_size=1, dengan nilai 0.

hasil Convolution (hC):

$$hC\{2\}\{1\}_{1,1} = \text{max}(\boxed{\phantom{00}}) = 0.8433$$

$$hC\{2\}\{1\}_{4 \times 4} = \begin{array}{c} 1 \quad 2 \quad 3 \quad 4 \\ \left( \begin{array}{cccc} 0.8433 & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ 0.8433 & \dots & \dots & \dots \end{array} \right) \end{array}$$

Sampai pada pergeseran terakhir, seperti berikut

$$\text{norm data ke-1} = \begin{pmatrix} 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \end{pmatrix}$$

Ketika  $k=3$ , maka lakukan penambahan tepian matrik (padding) sebanyak  $(k-1)/2 = 1$ , dengan nilai pad\_size =  $(3-1)/2=1$ , dimana nilai k harus bilangan ganjil  $\geq 3$ .

Untuk menentukan matrik  $hC(2)\{1\}$  pada baris 4 kolom 4.

**padding (norm data ke-1)** =  $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Menambahkan tepian matrik norm sebanyak pad\_size=1, dengan nilai 0.

hasil Convolution (hC):

$hC(2)\{1\}_{4,4} = \text{std}(\boxed{\phantom{0}}) = 0.2622$

$hC(2)\{8\}_{[4 \times 4]} = \dots$

Lakukan sampai norm data ke-8

Kemudian lanjut untuk filter ke-3, pergeserannya juga sama:

Untuk Filter ke-2: **std filter** (norm data ke-1 sampai ke-8, bykData=8, k=3)

$$\text{norm data ke-1} = \begin{pmatrix} 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \end{pmatrix}$$

Ketika  $k=3$ , maka lakukan penambahan tepian matrik (padding) sebanyak  $(k-1)/2 = 1$ , dengan nilai pad size =  $(3-1)/2=1$ , dimana nilai k harus bilangan ganjil  $\geq 3$ .

Untuk menentukan matrik  $hC(3)\{1\}$  pada baris 1 kolom 1.

**padding (norm data ke-1)** =  $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Menambahkan tepian matrik norm sebanyak pad\_size=1, dengan nilai 0.

hasil Convolution (hC):

$hC(3)\{1\}_{1,1} = \text{std}(\boxed{\phantom{0}}) = 0.3667$

$hC(3)\{1\}_{[4 \times 4]} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0.3667 & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix}$

Sampai pada pergeseran terakhir, seperti berikut

$$\text{norm data ke-1} = \begin{pmatrix} 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \end{pmatrix}$$

Ketika  $k=3$ , maka lakukan penambahan tepian matrik (padding) sebanyak  $(k-1)/2 = 1$ , dengan nilai pad size =  $(3-1)/2=1$ , dimana nilai k harus bilangan ganjil  $\geq 3$ .

Untuk menentukan matrik  $hC(3)\{1\}$  pada baris 4 kolom 4.

**padding (norm data ke-1)** =  $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0.8433 & 0.4367 & 0.8633 & 0.3167 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Menambahkan tepian matrik norm sebanyak pad\_size=1, dengan nilai 0.

hasil Convolution (hC):

$hC(3)\{1\}_{4,4} = \text{std}(\boxed{\phantom{0}}) = 0.2622$

$hC(3)\{8\}_{[4 \times 4]} = \dots$

Lakukan sampai norm data ke-8

## 2. Aktivasi dengan Sigmoid/ReLU/lainnya

```
hA=FnSigDL(hC,bykFilter,bykData);
```

berikut detail prosesnya:

Lakukan Aktivasi dengan parameter (hC, bykFilter=3, bykData=8) mulai dari

$$\begin{matrix} hC\{1\}\{1\}, hC\{1\}\{2\}, \dots, hC\{1\}\{8\} \\ hC\{2\}\{1\}, hC\{2\}\{2\}, \dots, hC\{2\}\{8\} \\ hC\{3\}\{1\}, hC\{3\}\{2\}, \dots, hC\{3\}\{8\} \end{matrix} \xrightarrow{\quad} \begin{matrix} hA\{1\}\{1\}, hA\{1\}\{2\}, \dots, hA\{1\}\{8\} \\ hA\{2\}\{1\}, hA\{2\}\{2\}, \dots, hA\{2\}\{8\} \\ hA\{3\}\{1\}, hA\{3\}\{2\}, \dots, hA\{3\}\{8\} \end{matrix}$$

Contoh hasil perhitungan Aktivasi (hA)

$$hA\{1\}\{1,1\} = 1/(1+\exp(hC\{1\}\{1,1\})) = 1/(1+\exp(0.2844)) = 0.4294$$

$$hC\{1\}\{1\} = \begin{pmatrix} 0.2844 & 0.4763 & 0.3593 & 0.2622 \\ 0.4267 & 0.7144 & 0.5389 & 0.3933 \\ 0.4267 & 0.7144 & 0.5389 & 0.3933 \\ 0.2844 & 0.4763 & 0.3593 & 0.2622 \end{pmatrix} \quad hA\{1\}\{1\} = \begin{pmatrix} 0.4294 & 0.3831 & 0.4111 & 0.4348 \\ 0.3949 & 0.3286 & 0.3684 & 0.4029 \\ 0.3949 & 0.3286 & 0.3684 & 0.4029 \\ 0.4294 & 0.3831 & 0.4111 & 0.4348 \end{pmatrix}$$

↓                                                   ↓

$$hC\{3\}\{8\} = \dots \quad hA\{3\}\{8\} = \dots$$

## 3. Convolution In (di tengah proses)

```
hC=FnConvInDL(hA,bykData,k,bykFilter);
```

berikut detail prosesnya:

Lakukan Conv. In dengan parameter (hA, bykData=8, k=3, bykFilter=3) mulai dari

$$\begin{matrix} hA\{1\}\{1\}, hA\{1\}\{2\}, \dots, hA\{1\}\{8\} \\ hA\{2\}\{1\}, hA\{2\}\{2\}, \dots, hA\{2\}\{8\} \\ hA\{3\}\{1\}, hA\{3\}\{2\}, \dots, hA\{3\}\{8\} \end{matrix} \xrightarrow{\quad} \begin{matrix} hC\{1\}\{1\}, hC\{1\}\{2\}, \dots, hC\{1\}\{8\} \\ hC\{2\}\{1\}, hC\{2\}\{2\}, \dots, hC\{2\}\{8\} \\ hC\{3\}\{1\}, hC\{3\}\{2\}, \dots, hC\{3\}\{8\} \end{matrix}$$

Proses perhitungan “3. Conv. In” sama dengan proses pada “1. Convolution di awal”.

## 4. Aktivasi dengan Sigmoid/ReLU/lainnya

```
hA=FnSigDL(hC,bykFilter,bykData);
```

berikut detail prosesnya:

Lakukan Aktivasi dengan parameter (hC, bykFilter=3, bykData=8) mulai dari

$$\begin{matrix} hC\{1\}\{1\}, hC\{1\}\{2\}, \dots, hC\{1\}\{8\} \\ hC\{2\}\{1\}, hC\{2\}\{2\}, \dots, hC\{2\}\{8\} \\ hC\{3\}\{1\}, hC\{3\}\{2\}, \dots, hC\{3\}\{8\} \end{matrix} \xrightarrow{\quad} \begin{matrix} hA\{1\}\{1\}, hA\{1\}\{2\}, \dots, hA\{1\}\{8\} \\ hA\{2\}\{1\}, hA\{2\}\{2\}, \dots, hA\{2\}\{8\} \\ hA\{3\}\{1\}, hA\{3\}\{2\}, \dots, hA\{3\}\{8\} \end{matrix}$$

Proses perhitungan “4. Aktivasi” sama dengan proses pada “2. Aktivasi”.

## 5. Pooling (jika windows\_size (ws)=2)

```
hP=FnPoolDL(hA,windows_size,bykFilter,bykData);
```

berikut detail prosesnya:

Pooling dengan parameter (hA, windows\_size=2, bykFilter=3, bykData=8) mulai dari

$$\begin{array}{l} hA\{1\}\{1\}, hA\{1\}\{2\}, \dots, hA\{1\}\{8\} \\ hA\{2\}\{1\}, hA\{2\}\{2\}, \dots, hA\{2\}\{8\} \\ hA\{3\}\{1\}, hA\{3\}\{2\}, \dots, hA\{3\}\{8\} \end{array} \rightarrow \begin{array}{l} hP\{1\}\{1\}, hP\{1\}\{2\}, \dots, hP\{1\}\{8\} \\ hP\{2\}\{1\}, hP\{2\}\{2\}, \dots, hP\{2\}\{8\} \\ hP\{3\}\{1\}, hP\{3\}\{2\}, \dots, hP\{3\}\{8\} \end{array}$$

Contoh hasil perhitungan Pooling (hP):

```
% hitung pad yang dibutuhkan (ml = byk baris hA\{1\}\{1\} = 4, nl = byk kolom hA\{1\}\{1\} = 4)
padX=(ceil(nl/windows_size)*windows_size)-nl = 0
padY=(ceil(ml/windows_size)*windows_size)-ml = 0
mpooll=sqrt((ml+padY)*(nl+padX)/windows_size^2) = 2, npooll=mpooll = 2
```

$$\begin{array}{l} hP\{1\}\{1\}_{1,1} = \max(\boxed{\phantom{000}}) = 0.4574 \\ hA\{1\}\{1\} = \begin{pmatrix} 0.4574 & 0.4360 & 0.4357 & 0.4552 \\ 0.4376 & 0.4065 & 0.4059 & 0.4340 \\ 0.4376 & 0.4065 & 0.4059 & 0.4340 \\ 0.4574 & 0.4360 & 0.4357 & 0.4552 \end{pmatrix} \\ \downarrow \\ hA\{3\}\{8\} = \dots \end{array} \quad \begin{array}{l} hP\{1\}\{1\}_{[mpool \times npool]} = \begin{pmatrix} 0.4574 & \dots \\ \dots & \dots \end{pmatrix} \\ \downarrow \\ hP\{3\}\{8\} = \dots \end{array}$$

Di mana ukuran filter kotak hijau = [ws x ws]

Sampai pada pergeseran terakhir, seperti berikut

$$\begin{array}{l} hA\{1\}\{1\}, hA\{1\}\{2\}, \dots, hA\{1\}\{8\} \\ hA\{2\}\{1\}, hA\{2\}\{2\}, \dots, hA\{2\}\{8\} \\ hA\{3\}\{1\}, hA\{3\}\{2\}, \dots, hA\{3\}\{8\} \end{array} \rightarrow \begin{array}{l} hP\{1\}\{1\}, hP\{1\}\{2\}, \dots, hP\{1\}\{8\} \\ hP\{2\}\{1\}, hP\{2\}\{2\}, \dots, hP\{2\}\{8\} \\ hP\{3\}\{1\}, hP\{3\}\{2\}, \dots, hP\{3\}\{8\} \end{array}$$

Contoh hasil perhitungan Pooling (hP):

```
% hitung pad yang dibutuhkan (ml = byk baris hA\{1\}\{1\} = 4, nl = byk kolom hA\{1\}\{1\} = 4)
padX=(ceil(nl/windows_size)*windows_size)-nl = 0
padY=(ceil(ml/windows_size)*windows_size)-ml = 0
mpooll=sqrt((ml+padY)*(nl+padX)/windows_size^2) = 2, npooll=mpooll = 2
```

$$\begin{array}{l} hP\{1\}\{1\}_{2,2} = \max(\boxed{\phantom{000}}) = 0.4552 \\ hA\{1\}\{1\} = \begin{pmatrix} 0.4574 & 0.4360 & 0.4357 & 0.4552 \\ 0.4376 & 0.4065 & 0.4059 & 0.4340 \\ 0.4376 & 0.4065 & 0.4059 & 0.4340 \\ 0.4574 & 0.4360 & 0.4357 & 0.4552 \end{pmatrix} \\ \downarrow \\ hA\{3\}\{8\} = \dots \end{array} \quad \begin{array}{l} hP\{1\}\{1\}_{[mpool \times npool]} = \begin{pmatrix} 0.4574 & 0.4552 \\ 0.4574 & 0.4552 \end{pmatrix} \\ \downarrow \\ hP\{3\}\{8\} = \dots \end{array}$$

## 6. Convolution In (di tengah proses)

```
hC=FnConvInDL(hP,bykData,k,bykFilter);
```

berikut detail prosesnya:

Lakukan Conv. In dengan parameter (hP, bykData=8, k=3, bykFilter=3) mulai dari

$$\begin{array}{l} hP\{1\}\{1\}, hP\{1\}\{2\}, \dots, hP\{1\}\{8\} \\ hP\{2\}\{1\}, hP\{2\}\{2\}, \dots, hP\{2\}\{8\} \\ hP\{3\}\{1\}, hP\{3\}\{2\}, \dots, hP\{3\}\{8\} \end{array} \rightarrow \begin{array}{l} hC\{1\}\{1\}, hC\{1\}\{2\}, \dots, hC\{1\}\{8\} \\ hC\{2\}\{1\}, hC\{2\}\{2\}, \dots, hC\{2\}\{8\} \\ hC\{3\}\{1\}, hC\{3\}\{2\}, \dots, hC\{3\}\{8\} \end{array}$$

Proses perhitungan "6. Conv. In" sama dengan proses pada "3. Conv. In".

## 7. Aktivasi dengan Sigmoid/ReLU/lainnya

```
hA=FnSigDL(hC,bykFilter,bykData);
```

berikut detail prosesnya:

Lakukan Aktivasi dengan parameter (hC, bykFilter=3, bykData=8) mulai dari

$$\begin{array}{l} hC\{1\}\{1\}, hC\{1\}\{2\}, \dots, hC\{1\}\{8\} \\ hC\{2\}\{1\}, hC\{2\}\{2\}, \dots, hC\{2\}\{8\} \\ hC\{3\}\{1\}, hC\{3\}\{2\}, \dots, hC\{3\}\{8\} \end{array} \rightarrow \begin{array}{l} hA\{1\}\{1\}, hA\{1\}\{2\}, \dots, hA\{1\}\{8\} \\ hA\{2\}\{1\}, hA\{2\}\{2\}, \dots, hA\{2\}\{8\} \\ hA\{3\}\{1\}, hA\{3\}\{2\}, \dots, hA\{3\}\{8\} \end{array}$$

Proses perhitungan "7. Aktivasi" sama dengan proses pada "2. Aktivasi".

## 8. Pooling (jika windows\_size (ws)=1)

```
hP=FnPoolDL(hA,windows_size,bykFilter,bykData);
```

berikut detail prosesnya:

Pooling dengan parameter (hA, windows\_size=1, bykFilter=3, bykData=8) mulai dari

$$\begin{array}{l} hA\{1\}\{1\}, hA\{1\}\{2\}, \dots, hA\{1\}\{8\} \\ hA\{2\}\{1\}, hA\{2\}\{2\}, \dots, hA\{2\}\{8\} \\ hA\{3\}\{1\}, hA\{3\}\{2\}, \dots, hA\{3\}\{8\} \end{array} \rightarrow \begin{array}{l} hP\{1\}\{1\}, hP\{1\}\{2\}, \dots, hP\{1\}\{8\} \\ hP\{2\}\{1\}, hP\{2\}\{2\}, \dots, hP\{2\}\{8\} \\ hP\{3\}\{1\}, hP\{3\}\{2\}, \dots, hP\{3\}\{8\} \end{array}$$

Contoh hasil perhitungan Pooling (hP):

% hitung pad yang dibutuhkan (ml = byk baris hA\{1\}\{1\} = 2, nl = byk kolom hA\{1\}\{1\} = 2)  
 padX=(ceil(nl/windows\_size)\*windows\_size)-nl = 0  
 padY=(ceil(ml/windows\_size)\*windows\_size)-ml = 0  
 mpool=sqrt((ml+padY)\*(nl+padX)/windows\_size^2) = 2, npool=mpool = 2

$$hP\{1\}\{1\}_{1,1} = \max(\boxed{\phantom{00}}) = 0.4495$$

$$hA\{1\}\{1\} = \begin{pmatrix} 0.4495 & 0.4495 \\ 0.4495 & 0.4495 \end{pmatrix}$$

$$hA\{3\}\{8\} = \dots$$

$$hP\{1\}\{1\}_{[mpool \times npool]} = \begin{pmatrix} 0.4495 & \dots \\ \dots & \dots \end{pmatrix}$$

$$hP\{3\}\{8\} = \dots$$

Di mana ukuran filter kotak hijau = [ws x ws]

Sampai pada pergeseran terakhir, seperti berikut

$$\begin{array}{l} hA\{1\}\{1\}, hA\{1\}\{2\}, \dots, hA\{1\}\{8\} \\ hA\{2\}\{1\}, hA\{2\}\{2\}, \dots, hA\{2\}\{8\} \\ hA\{3\}\{1\}, hA\{3\}\{2\}, \dots, hA\{3\}\{8\} \end{array} \rightarrow \begin{array}{l} hP\{1\}\{1\}, hP\{1\}\{2\}, \dots, hP\{1\}\{8\} \\ hP\{2\}\{1\}, hP\{2\}\{2\}, \dots, hP\{2\}\{8\} \\ hP\{3\}\{1\}, hP\{3\}\{2\}, \dots, hP\{3\}\{8\} \end{array}$$

Contoh hasil perhitungan Pooling (hP):

% hitung pad yang dibutuhkan (ml = byk baris hA\{1\}\{1\} = 2, nl = byk kolom hA\{1\}\{1\} = 2)  
 padX=(ceil(nl/windows\_size)\*windows\_size)-nl = 0  
 padY=(ceil(ml/windows\_size)\*windows\_size)-ml = 0  
 mpool=sqrt((ml+padY)\*(nl+padX)/windows\_size^2) = 2, npool=mpool = 2

$$hP\{1\}\{1\}_{2,2} = \max(\boxed{\phantom{00}}) = 0.4495$$

$$hA\{1\}\{1\} = \begin{pmatrix} 0.4495 & 0.4495 \\ 0.4495 & 0.4495 \end{pmatrix}$$

$$hA\{3\}\{8\} = \dots$$

$$hP\{1\}\{1\}_{[mpool \times npool]} = \begin{pmatrix} 0.4495 & 0.4495 \\ 0.4495 & 0.4495 \end{pmatrix}$$

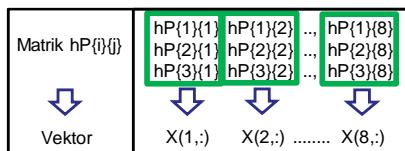
$$hP\{3\}\{8\} = \dots$$

## 9. Fully connected ke-1, misal dengan train ELM

```
byk_neuron_hidden_layer=5;
[hFC11,W11,Bias11,Beta11]=FnELMtrainClassify(hP,target,byk_neuron_hidden_layer, bykData,bykFilter);
```

berikut detail prosesnya:

train ELM dengan parameter (hP, target, byk\_neuron\_hidden\_layer=5, bykData=8,bykFilter=3) mulai dari  
 9.1. Konversi  $hP\{i\}\{j\}$  ( $hP\{i\}\{j\}$  disini adalah matrik yang memiliki ukuran baris ( $mhP$ ) dan ukuran kolom ( $nhP$ )) menjadi vektor (yang dimaksud vektor disini adalah matrik dengan ukuran 1 baris dan memiliki beberapa kolom). Dimana  $i = \{1,2,3\}$ ,  $j = \{1,2,3,\dots,8\}$



Matrik X memiliki ukuran [ $bykData \times bykFilter * mhP * nhP$ ], dari kumpulan vektor, mulai dari  $X(1,:)$ ,  $X(2,:)$ , ...,  $X(8,:)$ . Dimana  $X(1,:)$  artinya data train ke-1 (hasil final dari proses CNN) yang berukuran [ $1 \times bykFilter * mhP * nhP$ ]

membentuk  $X(1,:)$  berikut caranya:

$X(1,:)$  merupakan gabungan dari  $hP\{1\}\{1\}$ ,  $hP\{2\}\{1\}$ , dan  $hP\{3\}\{1\}$

$$hP\{1\}\{1\} = \begin{pmatrix} 0.4495 & 0.4495 \\ 0.4495 & 0.4495 \end{pmatrix} \quad hP\{2\}\{1\} = \begin{pmatrix} 0.3950 & 0.3950 \\ 0.3950 & 0.3950 \end{pmatrix} \quad hP\{3\}\{1\} = \begin{pmatrix} 0.4351 & 0.4351 \\ 0.4351 & 0.4351 \end{pmatrix}$$

Misal, elemennya diberikan dalam bentuk variabel:

$$hP\{1\}\{1\} = \begin{pmatrix} a1 & c1 \\ b1 & d1 \end{pmatrix} \quad hP\{2\}\{1\} = \begin{pmatrix} a2 & c2 \\ b2 & d2 \end{pmatrix} \quad hP\{3\}\{1\} = \begin{pmatrix} a3 & c3 \\ b3 & d3 \end{pmatrix}$$

Cara menyusunnya:

$$X(1,:) = [a1 \ b1 \ c1 \ d1 \ a2 \ b2 \ c2 \ d2 \ a3 \ b3 \ c3 \ d3] \\ X(1,:) = [0.4495 \ 0.4495 \ 0.4495 \ 0.4495 \ 0.3950 \ 0.3950 \ 0.3950 \ 0.3950 \ 0.4351 \ 0.4351 \ 0.4351 \ 0.4351]$$

↓

$$X(8,:) = \dots$$

hasil X yang lengkap (proses 9.1. selesai):

X merupakan gabungan dari  $X(1,:)$ ,  $X(2,:)$ , sampai  $X(8,:)$ .

$$X = \begin{pmatrix} X(1,:) \\ X(2,:) \\ X(3,:) \\ X(4,:) \\ X(5,:) \\ X(6,:) \\ X(7,:) \\ X(8,:) \end{pmatrix} = \begin{pmatrix} 0.4495 & 0.4495 & 0.4495 & 0.4495 & 0.3950 & 0.3950 & 0.3950 & 0.3950 & 0.4351 & 0.4351 & 0.4351 & 0.4351 \\ 0.4495 & 0.4495 & 0.4495 & 0.4495 & 0.3948 & 0.3948 & 0.3948 & 0.3948 & 0.4350 & 0.4350 & 0.4350 & 0.4350 \\ 0.4495 & 0.4495 & 0.4495 & 0.4495 & 0.3948 & 0.3948 & 0.3948 & 0.3948 & 0.4350 & 0.4350 & 0.4350 & 0.4350 \\ 0.4498 & 0.4498 & 0.4498 & 0.4498 & 0.3948 & 0.3948 & 0.3948 & 0.3948 & 0.4348 & 0.4348 & 0.4348 & 0.4348 \\ 0.4499 & 0.4499 & 0.4499 & 0.4499 & 0.3948 & 0.3948 & 0.3948 & 0.3948 & 0.4352 & 0.4352 & 0.4352 & 0.4352 \\ 0.4500 & 0.4500 & 0.4500 & 0.4500 & 0.3955 & 0.3955 & 0.3955 & 0.3955 & 0.4348 & 0.4348 & 0.4348 & 0.4348 \\ 0.4500 & 0.4500 & 0.4500 & 0.4500 & 0.3955 & 0.3955 & 0.3955 & 0.3955 & 0.4350 & 0.4350 & 0.4350 & 0.4350 \\ 0.4499 & 0.4499 & 0.4499 & 0.4499 & 0.3999 & 0.3999 & 0.3999 & 0.3999 & 0.4350 & 0.4350 & 0.4350 & 0.4350 \end{pmatrix}$$

berikut detail prosesnya (proses 9.2. selesai):

train ELM dengan parameter (hP, **target**, byk\_neuron\_hidden\_layer=5, bykData=8,bykFilter=3) kemudian  
 9.2. Konversi **target** disini adalah kelas (1 Dimensi atau 1 baris 1 kolom) dari data training menjadi vektor (yang dimaksud vektor disini adalah matrik dengan ukuran 1 baris dan memiliki beberapa kolom sebanyak kelas yang ada), tampung dalam variabel Y.  
 Pada "Proses Training (1 dari 3)" terdapat tabel yang memuat kolom **target** atau **kelas**:  
 karena byk\_kelas = 3, maka ukuran kolom Y = 3.

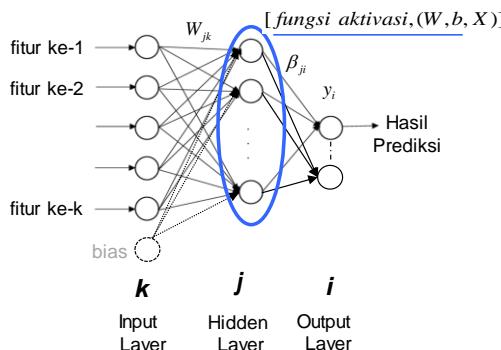
| No | Fitur ke-1 | Fitur ke-2 | Fitur ke-3 | Fitur ke-4 | <b>kelas</b> | Hasil konversi <b>target</b> (Y) |
|----|------------|------------|------------|------------|--------------|----------------------------------|
| 1  | 253        | 131        | 259        | 95         | 1            | 1 -1 -1                          |
| 2  | 131        | 259        | 95         | 263        | 1            | 1 -1 -1                          |
| 3  | 259        | 95         | 263        | 50         | 1            | 1 -1 -1                          |
| 4  | 95         | 263        | 50         | 154        | 1            | 1 -1 -1                          |
| 5  | 263        | 50         | 154        | 73         | 2            | -1 1 -1                          |
| 6  | 50         | 154        | 73         | 141        | 2            | -1 1 -1                          |
| 7  | 154        | 73         | 141        | 146        | 3            | -1 -1 1                          |
| 8  | 73         | 141        | 146        | 87         | 3            | -1 -1 1                          |

Pada kolom **kelas**, konversi kelas 1, Y-nya menjadi [1 -1 -1], kelas 2, Y-nya menjadi [-1 1 -1], dan kelas 3, Y-nya menjadi [-1 -1 1]  
 Angka 1 menyatakan kolom kelas aktual, kolom lainnya -1.

9.3. Lakukan proses train ELM dengan X sebagai data masukan dan Y sebagai target

$$X = \begin{pmatrix} 0.4495 & 0.4495 & 0.4495 & 0.4495 & 0.3950 & 0.3950 & 0.3950 & 0.3950 & 0.4351 & 0.4351 & 0.4351 \\ 0.4495 & 0.4495 & 0.4495 & 0.4495 & 0.3948 & 0.3948 & 0.3948 & 0.3948 & 0.4350 & 0.4350 & 0.4350 \\ 0.4496 & 0.4496 & 0.4496 & 0.4496 & 0.3948 & 0.3948 & 0.3948 & 0.3948 & 0.4349 & 0.4349 & 0.4349 \\ 0.4498 & 0.4498 & 0.4498 & 0.4498 & 0.3948 & 0.3948 & 0.3948 & 0.3948 & 0.4349 & 0.4349 & 0.4349 \\ 0.4499 & 0.4499 & 0.4499 & 0.4499 & 0.3948 & 0.3948 & 0.3948 & 0.3948 & 0.4349 & 0.4349 & 0.4349 \\ 0.4500 & 0.4500 & 0.4500 & 0.4500 & 0.3995 & 0.3995 & 0.3995 & 0.3995 & 0.4350 & 0.4350 & 0.4350 \\ 0.4499 & 0.4499 & 0.4499 & 0.4499 & 0.3995 & 0.3995 & 0.3995 & 0.3995 & 0.4350 & 0.4350 & 0.4350 \\ 0.4499 & 0.4499 & 0.4499 & 0.4499 & 0.3999 & 0.3999 & 0.3999 & 0.3999 & 0.4350 & 0.4350 & 0.4350 \end{pmatrix} \quad Y = \begin{pmatrix} 1 & -1 & -1 \\ 1 & -1 & -1 \\ 1 & -1 & -1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \\ -1 & -1 & 1 \end{pmatrix}$$

Diketahui (anggap variabel diproses 9.3. ini variabel local),  
 $N_{\text{train}}$  : Banyaknya data training = bykData = 8  
 $k$ : Banyaknya dimensi input layer = bykFilter\*mhP\*nHP = 3\*2\*2 = 12  
 $i$ : Banyaknya dimensi output layer = byk\_kelas = 3  
 $j$ : Banyaknya dimensi hidden layer = byk\_neuron\_hidden\_layer = 5

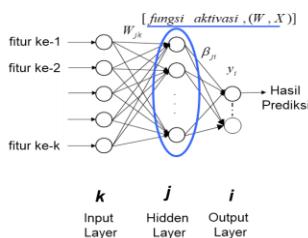


### Langkah-langkah training algoritma ELM tanpa bias:

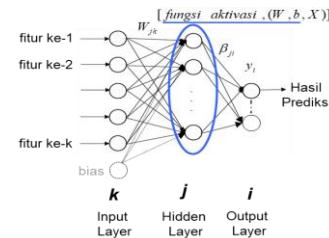
1. Buat random  $W_{jk}$  sebagai bobot masukan.
2. Hitung matrik keluaran hidden layer  $H = 1/(1+\exp(-X \cdot W^T))$
3. Hitung output weight  $\hat{\beta} = H^+ \cdot Y$  dimana  $H^* = (H^T \cdot H)^{-1} \cdot H^T$
4. Hitung  $\hat{Y} = H \cdot \hat{\beta}$

### Langkah-langkah testing algoritma ELM tanpa bias:

1. Diketahui  $W_{jk}$ , dan  $\hat{\beta}$
2. Hitung  $H = 1/(1+\exp(-X_{test} \cdot W^T))$
3. Hitung Hitung  $\hat{Y} = H \cdot \hat{\beta}$
4. Hitung nilai evaluasi, misal dengan MAPE atau akurasi, atau lainnya



(tanpa bias)



(dengan bias)

### Langkah-langkah training algoritma ELM dengan bias:

1. Buat random  $W_{jk}$  sebagai bobot masukan dan nilai bias  $b$  jika ada. Ukuran matrik bias  $b$  adalah  $[1 \times j]$
2. Hitung matrik keluaran hidden layer  $H = 1/(1+\exp(-(X \cdot W^T + ones(N_{train}, 1) * b)))$ . Dan  $N_{train}$  adalah banyaknya data training.
3. Hitung output weight  $\hat{\beta} = H^+ \cdot Y$  dimana  $H^* = (H^T \cdot H)^{-1} \cdot H^T$
4. Hitung  $\hat{Y} = H \cdot \hat{\beta}$

### Langkah-langkah testing algoritma ELM dengan bias:

1. Diketahui  $W_{jk}$ , nilai bias  $b$  jika ada dan  $\hat{\beta}$
2. Hitung  $H = 1/(1+\exp(-(X_{test} \cdot W^T + ones(N_{test}, 1) * b)))$
3. Hitung Hitung  $\hat{Y} = H \cdot \hat{\beta}$
4. Hitung nilai evaluasi, misal dengan MAPE atau akurasi, atau lainnya

berikut detail proses 9.3 (lanjutan ke-1):

1. Men-generate bobot input ( $W_{j \times k}$ ) antara input layer dan hidden layer. Ukuran W adalah  $[j \times k]$ , yang nilainya di-generate antara  $[-0.5, 0.5]$ .

$$W = \begin{pmatrix} 0.0304 & -0.0287 & -0.0985 & -0.0100 & -0.0613 & 0.1180 & -0.0674 & -0.2246 & 0.0408 & 0.1016 & -0.0919 & 0.0467 \\ 0.3433 & -0.0395 & -0.1786 & -0.3792 & 0.2078 & 0.0021 & -0.0730 & 0.2434 & -0.3307 & 0.1074 & 0.0976 & -0.3729 \\ -0.2847 & 0.4425 & -0.1212 & 0.3601 & 0.0296 & -0.0146 & -0.4707 & -0.3832 & 0.4301 & -0.0311 & -0.0743 & -0.4185 \\ 0.2167 & 0.1283 & 0.2190 & 0.4472 & 0.1524 & -0.3899 & 0.0104 & 0.2693 & -0.2918 & 0.0963 & -0.3782 & -0.2105 \\ 0.3627 & 0.3064 & -0.3069 & 0.1309 & 0.2782 & 0.4165 & 0.4637 & 0.2657 & -0.4802 & 0.2235 & -0.1500 & 0.1768 \end{pmatrix}$$

2. Men-generate bias ( $b$ ), dengan ukuran  $[1 \times j]$  yang nilainya di-generate antara  $[0.0, 1.0]$ . Dan Hitung matrik inisialisasi output hidden layer ( $H_{init} = X.W^T$ ), dengan ukuran [bykData  $\times$   $j$ ]

$$b = \begin{pmatrix} 0.6388 & 0.0110 & 0.9773 & 0.3855 & 0.3997 \end{pmatrix}$$

$$H_{init} = \begin{pmatrix} -0.0987 & -0.1809 & -0.5176 & 0.1300 & 0.6841 \\ -0.0986 & -0.1809 & -0.5174 & 0.1301 & 0.6839 \\ -0.0986 & -0.1809 & -0.5175 & 0.1302 & 0.6840 \\ -0.0987 & -0.1809 & -0.5175 & 0.1305 & 0.6841 \\ -0.0986 & -0.1811 & -0.5176 & 0.1303 & 0.6841 \\ -0.0998 & -0.1792 & -0.5215 & 0.1309 & 0.6909 \\ -0.0998 & -0.1792 & -0.5215 & 0.1307 & 0.6908 \\ -0.0999 & -0.1791 & -0.5218 & 0.1307 & 0.6913 \end{pmatrix}$$

3. Hitung matrik  $H = 1/(1+exp(-(X.W^T + ones(N_{train}, 1)*b)))$ , dengan ukuran [bykData  $\times$   $j$ ]

$$ones(N_{train}, 1)*b = \begin{pmatrix} 0.6388 & 0.0110 & 0.9773 & 0.3855 & 0.3997 \\ 0.6388 & 0.0110 & 0.9773 & 0.3855 & 0.3997 \\ 0.6388 & 0.0110 & 0.9773 & 0.3855 & 0.3997 \\ 0.6388 & 0.0110 & 0.9773 & 0.3855 & 0.3997 \\ 0.6388 & 0.0110 & 0.9773 & 0.3855 & 0.3997 \\ 0.6388 & 0.0110 & 0.9773 & 0.3855 & 0.3997 \\ 0.6388 & 0.0110 & 0.9773 & 0.3855 & 0.3997 \\ 0.6388 & 0.0110 & 0.9773 & 0.3855 & 0.3997 \end{pmatrix}$$

$$H = \begin{pmatrix} 0.6318 & 0.4576 & 0.6129 & 0.6261 & 0.7472 \\ 0.6318 & 0.4576 & 0.6130 & 0.6261 & 0.7472 \\ 0.6318 & 0.4576 & 0.6130 & 0.6262 & 0.7472 \\ 0.6318 & 0.4576 & 0.6130 & 0.6262 & 0.7472 \\ 0.6318 & 0.4576 & 0.6129 & 0.6262 & 0.7472 \\ 0.6316 & 0.4580 & 0.6120 & 0.6263 & 0.7485 \\ 0.6316 & 0.4580 & 0.6120 & 0.6263 & 0.7485 \\ 0.6316 & 0.4581 & 0.6119 & 0.6263 & 0.7486 \end{pmatrix}$$

4. Hitung matrik  $H^+ = (H^T.H)^{-1}.H^T$  dengan ukuran  $[j \times$  bykData]

$$H^+ = \begin{pmatrix} -3.0027e+06 & 2.1281e+06 & 1.1699e+06 & -23230 & 2.0661e+05 & -2.8233e+06 & -5.1876e+06 & 7.5323e+06 \\ -9.3392e+05 & 6.6579e+05 & 3.7222e+05 & -760.55 & 44773 & -8.7272e+05 & -1.6117e+06 & 2.3363e+06 \\ 2.3713e+06 & -1.6778e+06 & -9.2515e+05 & 14126 & -1.5941e+05 & 2.2239e+06 & 4.0922e+06 & -5.9332e+06 \\ -3.8559e+05 & 2.6609e+05 & 1.4998e+05 & 3748.5 & 26117 & -3.5285e+05 & 6.5754e+05 & 9.5004e+05 \\ 1.4889e+06 & -1.0539e+06 & -5.84e+05 & 5380.3 & -93247 & 1.3932e+06 & 2.5678e+06 & -3.7242e+06 \end{pmatrix}$$

berikut detail proses 9.3 (lanjutan ke-2):

5. Hitung beta topi atau hasil nilai beta yang diestimasi, dengan ukuran [ $j \times i$ ].

$$\hat{\beta} = H^+ Y = \begin{pmatrix} 5.4406e+05 & -5.2334e+06 & 4.6893e+06 \\ 2.0666e+05 & -1.6559e+06 & 1.4492e+06 \\ -4.3485e+05 & 4.1289e+06 & -3.6941e+06 \\ 68473 & -6.5347e+05 & 5.85e+05 \\ -2.8728e+05 & 2.6e+06 & -2.3127e+06 \end{pmatrix}$$

6. Hitung  $\hat{Y}$  topi atau  $\hat{Y}$  prediksi, dgn ukuran [bykData  $\times$  i].

$$\hat{Y} = H \cdot \hat{\beta} = \begin{pmatrix} 0.7889 & -1.0673 & -0.74441 \\ 1.2042 & -1.4288 & -0.7982 \\ 1.2159 & -1.0882 & -1.1505 \\ 0.74778 & -0.31104 & -1.4595 \\ -0.88156 & 0.80262 & -0.94384 \\ -0.88716 & -0.058267 & -0.077316 \\ -0.94071 & -0.27665 & 0.19461 \\ -1.0997 & -0.75165 & 0.82864 \end{pmatrix}$$

dari hasil proses 9.3. maka dapat di-set return sebagai berikut:

hFC11 =  $\hat{Y}$

W11 =  $W$

Bias11 = *bias*

Beta11 =  $\hat{\beta}$

“9. Fully connected ke-1, misal dengan train ELM” selesai.

10. Fully connected ke-2, misal dengan train ELM

```
byk_neuron_hidden_layer=7;
[hFC12,W12,Bias12,Beta12]=FnELMtrainClassify(hP,target,
byk_neuron_hidden_layer, bykData,bykFilter);
```

Lakukan “10. Fully connected ke-2” seperti pada “9. Fully connected ke-1”, dari hasil proses 10.3. maka dapat di-set return sebagai berikut:

hFC12 =  $\hat{Y}$

W12 =  $W$

Bias12 = *bias*

Beta12 =  $\hat{\beta}$

“10. Fully connected ke-2, misal dengan train ELM”  
selesai.

### 11. Fully connected ke-3, misal dengan train ELM

```
byk_neuron_hidden_layer=4;
[hFC13,W13,Bias13,Beta13]=FnELMtrainClassify(hP,target,
byk_neuron_hidden_layer, bykData,bykFilter);
```

Lakukan “11. Fully connected ke-3” seperti pada “9. Fully connected ke-1”, dari hasil proses 11.3. maka dapat di-set return sebagai berikut:

```
hFC13= \hat{Y}
W13= W
Bias13= bias
Beta13= $\hat{\beta}$
```

“11. Fully connected ke-3, misal dengan train ELM”  
selesai.

## 16.2.2 Proses Testing

Berikut beberapa langkah-langkah untuk melakukan proses testing pada algoritma CNN.

- pre-Proses data testing, berikut kodingnya

```
[bykData2,byk_fitur2,target2,norm2]=FnPreProses('datatestC
lassify.xlsx',...
max, min, max2, min2);
```

dimana target2 = kolom “kelas” pada Tabel di bawah.

berikut detail prosesnya:

1. Baca file data test, get bykData dan byk\_fitur

| No | Fitur ke-1 | Fitur ke-2 | Fitur ke-3 | Fitur ke-4 | kelas |
|----|------------|------------|------------|------------|-------|
| 1  | 253        | 131        | 259        | 95         | 1     |
| 2  | 131        | 259        | 95         | 263        | 1     |
| 3  | 263        | 50         | 154        | 73         | 2     |
| 4  | 50         | 154        | 73         | 141        | 2     |
| 5  | 154        | 73         | 141        | 146        | 3     |
| 6  | 73         | 141        | 146        | 87         | 3     |

bykData2 = 6 dan byk\_fitur2 = 4

2. Membentuk dataset sebagai matriks square [byk\_fitur x byk\_fitur] menggunakan konsep replika matriks  
`repmat(data{i},[byk_fitur 1])`, misal data ke-1

| No | Fitur ke-1 | Fitur ke-2 | Fitur ke-3 | Fitur ke-4 |
|----|------------|------------|------------|------------|
| 1  | 253        | 131        | 259        | 95         |



Dikonversi menjadi matrik persegi berikut

|             |     |     |     |    |
|-------------|-----|-----|-----|----|
| data ke-1 = | 253 | 131 | 259 | 95 |
|             | 253 | 131 | 259 | 95 |
|             | 253 | 131 | 259 | 95 |
|             | 253 | 131 | 259 | 95 |

↓

|             |    |     |     |    |
|-------------|----|-----|-----|----|
| data ke-6 = | 73 | 141 | 146 | 87 |
|             | 73 | 141 | 146 | 87 |
|             | 73 | 141 | 146 | 87 |
|             | 73 | 141 | 146 | 87 |

3. Menghitung hasil normalisasi dengan rumus (norm)  
`norm{i}=(((a{i}-min)./(max-min))*(max2-min2))+min2;`

|             |     |     |     |    |
|-------------|-----|-----|-----|----|
| data ke-1 = | 253 | 131 | 259 | 95 |
|             | 253 | 131 | 259 | 95 |
|             | 253 | 131 | 259 | 95 |
|             | 253 | 131 | 259 | 95 |

↓

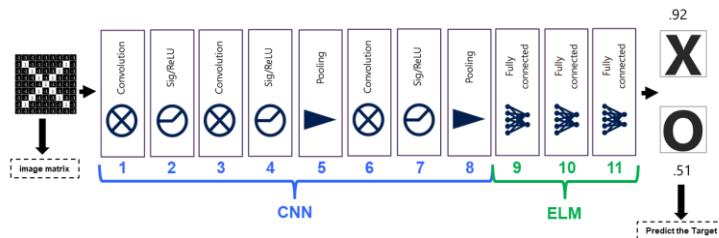
$$\text{norm2 data ke-1} = \begin{pmatrix} 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \\ 0.8433 & 0.4367 & 0.8633 & 0.3167 \end{pmatrix}$$

↓

data ke-6 = .....

dimana `a{i}` menyatakan akan mengambil tiap elemen pada matrik data ke-i. Dan `norm{i}` menyatakan matrik dengan

- Masuk ke Proses misal dgn **CNN** dan **ELM** berikut:



### 1. Convolution di awal

```
hC2=FnConvDL(norm2,bykData2,k);
Lakukan Conv. dengan parameter (norm2, bykData2 = 6, k = 3). Prosesnya (misal dengan 3 macam filter), lakukan seperti pada Proses Training "1. Convolution di awal".
```

### 2. Aktivasi dengan Sigmoid/ReLU/lainnya

```
hA2=FnSigDL(hC2,bykFilter,bykData2);
```

berikut detail prosesnya:

```
Lakukan Aktivasi dengan parameter (hC2, bykFilter=3, bykData2=6). Prosesnya lakukan seperti pada Proses Training "2. Aktivasi dengan Sigmoid/ReLU/lainnya"
```

### 3. Convolution In (di tengah proses)

```
hC2=FnConvInDL(hA2,bykData2,k,bykFilter);
```

berikut detail prosesnya:

```
Lakukan Conv. In dengan parameter (hA2, bykData2=6, k=3, bykFilter=3). Proses perhitungan "3. Conv. In" sama dengan proses pada "1. Convolution di awal".
```

### 4. Aktivasi dengan Sigmoid/ReLU/lainnya

```
hA2=FnSigDL(hC2,bykFilter,bykData2);
```

berikut detail prosesnya:

```
Lakukan Aktivasi dengan parameter (hC2, bykFilter=3, bykData2=6). Proses perhitungan "4. Aktivasi" sama dengan proses pada "2. Aktivasi".
```

## 5. Pooling

```
hP2=FnPoolDL(hA2,windows_size,bykFilter,bykData2);
```

berikut detail prosesnya:

Pooling dengan parameter (hA2, windows\_size=2, bykFilter=3, bykData2=6). Proses perhitungan “5. Pooling” sama dengan proses pada “5. Pooling” pada proses Training.

## 6. Convolution In (di tengah proses)

```
hC2=FnConvInDL(hP2,bykData2,k,bykFilter);
```

berikut detail prosesnya:

Lakukan Conv. In dengan parameter (hP2, bykData2=6, k=3, bykFilter=3). Proses perhitungan “6. Conv. In” sama dengan proses pada “1. Convolution di awal”.

## 7. Aktivasi dengan Sigmoid/ReLU/lainnya

```
hA2=FnSigDL(hC2,bykFilter,bykData2);
```

berikut detail prosesnya:

Lakukan Aktivasi dengan parameter (hC2, bykFilter=3, bykData2=6). Proses perhitungan “7. Aktivasi” sama dengan proses pada “2. Aktivasi”.

## 8. Pooling (jika windows\_size (ws)=1)

```
hP2=FnPoolDL(hA2,windows_size,bykFilter,bykData2);
```

berikut detail prosesnya:

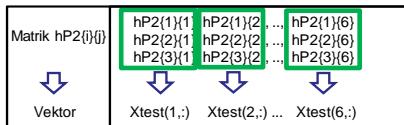
Pooling dengan parameter (hA2, windows\_size=1, bykFilter=3, bykData2=6). Proses perhitungan “8. Pooling” sama dengan proses pada “5. Pooling” pada proses Training.

## 9. Fully connected ke-1, misal dengan train ELM

```
[Akurasi1,kelasPrediksi1,Ytest_predict1]=...
FnELMtestClassify(hP2,target2,...
W11,Bias11,Beta11,bykData2,bykFilter);
```

berikut detail prosesnya:

testing ELM dengan parameter ( $hP2$ , $target2$ ,...  
 $W11$ , $Bias11$ , $Beta11$ , $bykData2=6$ , $bykFilter=3$ ) mulai dari  
9.1. Konversi  $hP2\{i\}\{j\}$  ( $hP2\{i\}\{j\}$  disini adalah matrik  
yang memiliki ukuran baris ( $mhP2$ ) dan ukuran kolom ( $nhP2$ ))  
menjadi vektor (yang dimaksud vektor disini adalah matrik  
dengan ukuran 1 baris dan memiliki beberapa kolom). Dimana  
 $i = \{1,2,3\}$ ,  $j = \{1,2,3,\dots,6\}$



Matrik  $Xtest$  memiliki ukuran [ $bykData2 \times bykFilter * mhP2 * nhP2$ ], dari kumpulan vektor, mulai dari  $Xtest(1,:)$ ,  $Xtest(2,:)$ , ...,  $Xtest(6,:)$ . Dimana  $Xtest(1,:)$  artinya data test ke-1 (hasil final dari proses CNN) yang berukuran [1 x  $bykFilter * mhP2 * nhP2$ ]

membentuk  $Xtest(1,:)$  berikut caranya:

$Xtest(1,:)$  merupakan gabungan dari  $hP2\{1\}\{1\}$ ,  $hP2\{2\}\{1\}$ , dan  $hP2\{3\}\{1\}$

$$hP2\{1\}\{1\} = \begin{pmatrix} 0.4495 & 0.4495 \\ 0.4495 & 0.4495 \end{pmatrix} \quad hP2\{2\}\{1\} = \begin{pmatrix} 0.3949 & 0.3949 \\ 0.3949 & 0.3949 \end{pmatrix} \quad hP2\{3\}\{1\} = \begin{pmatrix} 0.4351 & 0.4351 \\ 0.4351 & 0.4351 \end{pmatrix}$$

Misal, elemennya diberikan dalam bentuk variabel:

$$hP2\{1\}\{1\} = \begin{pmatrix} a1 & c1 \\ b1 & d1 \end{pmatrix} \quad hP2\{2\}\{1\} = \begin{pmatrix} a2 & c2 \\ b2 & d2 \end{pmatrix} \quad hP2\{3\}\{1\} = \begin{pmatrix} a3 & c3 \\ b3 & d3 \end{pmatrix}$$

Cara menyusunnya:

$$Xtest(1,:) = [a1 \ b1 \ c1 \ d1 \ a2 \ b2 \ c2 \ d2 \ a3 \ b3 \ c3 \ d3] \\ Xtest(1,:) = [0.4495 \ 0.4495 \ 0.4495 \ 0.4495 \ 0.3950 \ 0.3950 \\ \ 0.3950 \ 0.3950 \ 0.4351 \ 0.4351 \ 0.4351 \ 0.4351]$$

↓

$Xtest(6,:)$  = ...

hasil  $Xtest$  yang lengkap (proses 9.1. selesai):

$Xtest$  merupakan gabungan dari  $Xtest(1,:)$ ,  $Xtest(2,:)$ , sampai  $Xtest(6,:)$ .

$$Xtest = \begin{pmatrix} Xtest(1,:) \\ Xtest(2,:) \\ Xtest(3,:) \\ Xtest(4,:) \\ Xtest(5,:) \\ Xtest(6,:) \end{pmatrix} = \begin{pmatrix} 0.44947 & 0.44947 & 0.44947 & 0.44947 & 0.39499 & 0.39499 & 0.39499 & 0.39499 & 0.4351 & 0.4351 & 0.4351 \\ 0.44948 & 0.44948 & 0.44948 & 0.44948 & 0.39483 & 0.39483 & 0.39483 & 0.39483 & 0.4351 & 0.4351 & 0.4351 \\ 0.44986 & 0.44986 & 0.44986 & 0.44986 & 0.39483 & 0.39483 & 0.39483 & 0.39483 & 0.4351 & 0.4351 & 0.4351 \\ 0.44987 & 0.44987 & 0.44987 & 0.44987 & 0.39502 & 0.39502 & 0.39502 & 0.39502 & 0.43496 & 0.43496 & 0.43496 \\ 0.44989 & 0.44989 & 0.44989 & 0.44989 & 0.39552 & 0.39552 & 0.39552 & 0.39552 & 0.43496 & 0.43496 & 0.43496 \\ 0.44993 & 0.44993 & 0.44993 & 0.44993 & 0.39988 & 0.39988 & 0.39988 & 0.39988 & 0.43496 & 0.43496 & 0.43496 \end{pmatrix}$$

berikut detail prosesnya (proses 9.2. selesai):

testing ELM dengan parameter (hP2,target2,...)

W11,Bias11,Beta11,bykData2=6,bykFilter=3) kemudian

9.2. Konversi **target2** disini adalah kelas (1 Dimensi atau 1 baris 1 kolom) dari data testing menjadi vektor (vektor ini adalah matrik dgn ukuran 1 baris dan memiliki beberapa kolom sebanyak kelas yang ada seperti pada data training), tempung dalam variabel Ytest.

Pada "Proses Testing (1 dari 3)" terdapat tabel yang memuat kolom **target2** atau **kelas**:

karena `byk_kelas` pada data training = 3, maka ukuran kolom `Ytest` = 3.

| No | Fitur ke-1 | Fitur ke-2 | Fitur ke-3 | Fitur ke-4 | kelas |
|----|------------|------------|------------|------------|-------|
| 1  | 253        | 131        | 259        | 95         | 1     |
| 2  | 131        | 259        | 95         | 263        | 1     |
| 3  | 263        | 50         | 154        | 73         | 2     |
| 4  | 50         | 154        | 73         | 141        | 2     |
| 5  | 154        | 73         | 141        | 146        | 3     |
| 6  | 73         | 141        | 146        | 87         | 3     |

| Konversi target2 (Ytest) |    |    |
|--------------------------|----|----|
| 1                        | -1 | -1 |
| 1                        | -1 | -1 |
| -1                       | 1  | -1 |
| -1                       | 1  | -1 |
| -1                       | -1 | 1  |
| -1                       | -1 | 1  |

Konversi **kelas** 1, Ytest-nya menjadi [1 -1 -1], **kelas** 2 menjadi [-1 1 -1], dan **kelas** 3 menjadi [-1 -1 1]. Pada Ytest, angka 1 menyatakan kolom kelas aktual, kolom lainnya -1.

9.3. Lakukan proses testing ELM (dengan bias) dgn Xtest sebagai data masukan dan Ytest sebagai target

$$\text{Xtest} = \begin{pmatrix} 0.44947 & 0.44947 & 0.44947 & 0.34949 & 0.34949 & 0.34949 & 0.4351 & 0.4351 & 0.4351 \\ 0.44947 & 0.44947 & 0.44947 & 0.34949 & 0.34949 & 0.34949 & 0.4351 & 0.4351 & 0.4351 \\ 0.44947 & 0.44947 & 0.44947 & 0.34949 & 0.34949 & 0.34949 & 0.4351 & 0.4351 & 0.4351 \\ 0.44948 & 0.44948 & 0.44948 & 0.34949 & 0.34949 & 0.34949 & 0.4351 & 0.4351 & 0.4351 \\ 0.44949 & 0.44949 & 0.44949 & 0.34949 & 0.34949 & 0.34949 & 0.4351 & 0.4351 & 0.4351 \\ 0.45001 & 0.45001 & 0.45001 & 0.39952 & 0.39952 & 0.39952 & 0.43482 & 0.43482 & 0.43482 \\ 0.44948 & 0.44949 & 0.44950 & 0.34949 & 0.34950 & 0.34950 & 0.43483 & 0.43483 & 0.43483 \\ 0.44949 & 0.44950 & 0.44951 & 0.34949 & 0.34950 & 0.34950 & 0.43484 & 0.43484 & 0.43484 \\ 0.44950 & 0.44951 & 0.44952 & 0.34949 & 0.34950 & 0.34950 & 0.43485 & 0.43485 & 0.43485 \\ 0.44951 & 0.44952 & 0.44953 & 0.34949 & 0.34950 & 0.34950 & 0.43486 & 0.43486 & 0.43486 \end{pmatrix}$$

Diketahui (anggap variabel diproses 9.3. ini variabel local),

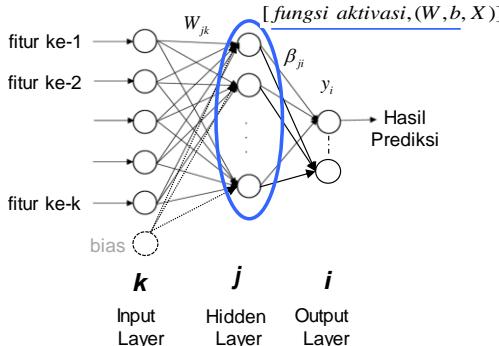
$N_{test}$  : Banyak data testing = bykData = 6

k: Banyak dimensi input layer = bvkFilter\*mhP2\*nhp2 = 3\*2\*2 = 12

i: Banyak dimensi output layer = byk kelas =

t. banyak dimensi output layer = byk\_kelus = 3 (Sama dg n di pre-training, atau get dari ukuran kolom Beta11)

j: Banyak dimensi hidden layer = (get dari ukuran baris Beta1 atau get dari ukuran baris W11)



Load bobot input ( $W_j \times k$ ) antara input layer dan hidden layer. Ukuran  $W$  adalah  $[j \times k]$ ,  $W = W11$

$$W = \begin{pmatrix} 0.0304 & -0.0287 & -0.0985 & -0.0100 & -0.0613 & 0.1180 & -0.0674 & -0.2246 & 0.0408 & 0.1016 & -0.0919 & 0.0467 \\ 0.3433 & -0.0985 & -0.1788 & -0.3792 & 0.2078 & 0.0021 & -0.0730 & 0.2434 & -0.3307 & 0.1074 & 0.0976 & -0.3729 \\ -0.2847 & 0.4425 & -0.1212 & -0.3601 & 0.0296 & -0.0146 & -0.4707 & -0.3832 & 0.4301 & -0.0311 & -0.0743 & -0.4185 \\ 0.2167 & 0.1283 & 0.2190 & 0.4472 & 0.1524 & -0.3899 & 0.0104 & 0.2693 & -0.2918 & 0.0963 & -0.3782 & -0.2105 \\ 0.3627 & 0.3064 & -0.3069 & 0.1309 & 0.2782 & 0.4165 & 0.4637 & 0.2657 & -0.4802 & 0.2235 & -0.1500 & 0.1768 \end{pmatrix}$$

Load bias ( $b$ ), dengan ukuran  $[1 \times j]$ , bias = Bias11.

$$b = \begin{pmatrix} 0.6388 & 0.0110 & 0.9773 & 0.3855 & 0.3997 \end{pmatrix}$$

Load Hitung  $\hat{\beta}$  beta topi, dengan ukuran  $[j \times i]$ ,  $\hat{\beta} = \text{Beta11}$

$$\hat{\beta} = \begin{pmatrix} 5.4406e+05 & -5.2334e+06 & 4.6893e+06 \\ 2.0666e+05 & -1.6559e+06 & 1.4492e+06 \\ -4.3485e+05 & 4.1289e+06 & -3.6941e+06 \\ 68473 & -6.5347e+05 & 5.85e+05 \\ -2.8728e+05 & 2.6e+06 & -2.3127e+06 \end{pmatrix}$$

1. Hitung matrik inisialisasi output hidden layer ( $H_{init} = X_{test} \cdot W^T$ ), dengan ukuran  $[N_{test} \times j]$

$$H_{init} = \begin{pmatrix} -0.098653 & -0.18089 & -0.51758 & 0.12997 & 0.68411 \\ -0.098626 & -0.1809 & -0.51743 & 0.13005 & 0.68391 \\ -0.098649 & -0.18109 & -0.51757 & 0.1303 & 0.68406 \\ -0.098603 & -0.17917 & -0.52152 & 0.13092 & 0.69089 \\ -0.099777 & -0.17921 & -0.52149 & 0.13069 & 0.6908 \\ -0.099866 & -0.17908 & -0.52181 & 0.13075 & 0.69133 \end{pmatrix}$$

2. Hitung matrik  $H = 1/(1+\exp(-(X_{test} \cdot W^T + \text{ones}(N_{test}, 1) * b)))$ , dengan ukuran  $[N_{test} \times j]$

$$\text{ones}(N_{test}, 1) * b = \begin{pmatrix} 0.6388 & 0.0110 & 0.9773 & 0.3855 & 0.3997 \\ 0.6388 & 0.0110 & 0.9773 & 0.3855 & 0.3997 \\ 0.6388 & 0.0110 & 0.9773 & 0.3865 & 0.3997 \\ 0.6388 & 0.0110 & 0.9773 & 0.3865 & 0.3997 \\ 0.6388 & 0.0110 & 0.9773 & 0.3865 & 0.3997 \\ 0.6388 & 0.0110 & 0.9773 & 0.3865 & 0.3997 \end{pmatrix}$$

$$H = \begin{pmatrix} 0.63185 & 0.45763 & 0.61295 & 0.62609 & 0.74721 \\ 0.63185 & 0.45763 & 0.61298 & 0.62611 & 0.74718 \\ 0.63185 & 0.45758 & 0.61295 & 0.62616 & 0.7472 \\ 0.63158 & 0.45806 & 0.61201 & 0.62631 & 0.74849 \\ 0.63159 & 0.45805 & 0.61202 & 0.62626 & 0.74848 \\ 0.63156 & 0.45808 & 0.61194 & 0.62627 & 0.74858 \end{pmatrix}$$

3. Hitung  $Y$  topi atau  $Y$  prediksi

$$\hat{Y} = H \cdot \hat{\beta}$$

$$\hat{Y} = \begin{pmatrix} \text{Index ke-1} & \text{Index ke-2} & \text{Index ke-3} \\ 5.4028 & -59.6 & -3.2066 \\ 5.8183 & -59.964 & -3.261 \\ 3.7322 & -57.73 & -3.4043 \\ 3.7226 & -58.534 & -2.5263 \\ 3.6691 & -58.752 & -2.2546 \\ 3.5097 & -59.223 & -1.6194 \end{pmatrix}$$

| Kelas Aktual dari $X_{test}$ | Kelas Prediksi (KP) = Indek dari Maks ( $Y$ prediksi) |
|------------------------------|-------------------------------------------------------|
| 1                            | 1                                                     |
| 1                            | 1                                                     |
| 2                            | 1                                                     |
| 2                            | 1                                                     |
| 3                            | 1                                                     |
| 3                            | 1                                                     |

$$Akurasi(AK) = \frac{2}{N_{test}} \times 100\% = 33.33\%$$

dari hasil proses 9.3. maka dapat di-set return sebagai berikut:

$\text{Ytest\_predict1} = \hat{Y}$

kelasPrediksi1= KP

Akurasi1= AK

“9. Fully connected ke-1, misal dengan testing ELM”  
selesai.

10. Fully connected ke-2, misal dengan testing ELM

```
[Akurasi2, kelasPrediksi2, Ytest_predict2]=...
FnELMtestClassify(hP2,target2,...
W12,Bias12,Beta12,bykData2,bykFilter);
```

Lakukan “10. Fully connected ke-2” seperti pada “9. Fully connected ke-1” dengan testing ELM, dari hasil proses 10.3. maka dapat di-set return sebagai berikut:

$\text{Ytest\_predict2} = \hat{Y}$

kelasPrediksi2= KP

Akurasi2= AK

“10. Fully connected ke-2, misal dengan testing ELM”  
selesai.

11. Fully connected ke-3, misal dengan testing ELM

```
[Akurasi3, kelasPrediksi3, Ytest_predict3]=...
FnELMtestClassify(hP2,target2,...
W13,Bias13,Beta13,bykData2,bykFilter);
```

Lakukan “11. Fully connected ke-3” seperti pada “9. Fully connected ke-1” dengan testing ELM, dari hasil proses 11.3. maka dapat di-set return sebagai berikut:

$\text{Ytest\_predict3} = \hat{Y}$

kelasPrediksi3= KP

Akurasi3= AK

“11. Fully connected ke-3, misal dengan testing ELM”  
selesai.

### Proses voting dari beberapa "Fully connected" testing ELM

Misal didapat sebagai berikut:

$$\text{kelasPrediksi1} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \text{kelasPrediksi2} = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 3 \\ 3 \end{pmatrix} \quad \text{kelasPrediksi3} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 3 \\ 3 \end{pmatrix}$$

Final kelasPrediksi = mode  
(mode sbg hasil voting)

$$\downarrow \quad \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 1 & 3 \\ 1 & 3 & 3 \\ 1 & 3 & 3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \\ 3 \\ 3 \end{pmatrix}$$

Kelas yg paling banyak muncul tiap baris,  
lalu bandingkan dengan **kelas Aktual**  
untuk menghitung Akurasi.

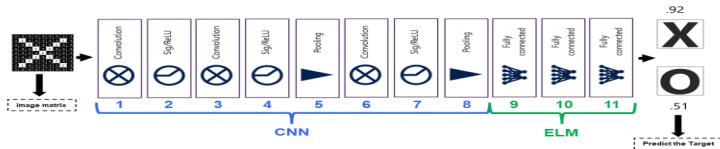
$$\text{Akurasi} = \frac{5}{N_{test} = 6} \times 100\% = 83.33\%$$

| Kelas Aktual dari Xtest |  |
|-------------------------|--|
| 1                       |  |
| 1                       |  |
| 2                       |  |
| 2                       |  |
| 3                       |  |
| 3                       |  |

Selesai. 😊

## 16.3 Tugas Kelompok

1. Jelaskan cara kerja dari algoritma CNN berdasarkan Map berikut.



2. Berdasarkan contoh pada manualisasi, lakukan proses perhitungan dari dataset yang ada pada manualisasi dengan algoritma full connected-nya sebagai berikut (pilih salah satu).

- a. NB
- b. KNN
- c. SVM (pilih salah satu, dari OAO/OAA/DT/DAG)
- d. C4.5/C5.0

3. Berdasarkan soal no.1, Jika pada saat melakukan convolution menggunakan filter kernel seperti berikut:

|   |   |   |
|---|---|---|
| 1 |   | 1 |
|   | 1 |   |
| 1 |   | 1 |

Filter kernal di atas memiliki makna apa!

4. Lakukan proses perhitungan dari data Curah Hujan untuk Prediksi berikut dengan algoritma CNN base ELM.

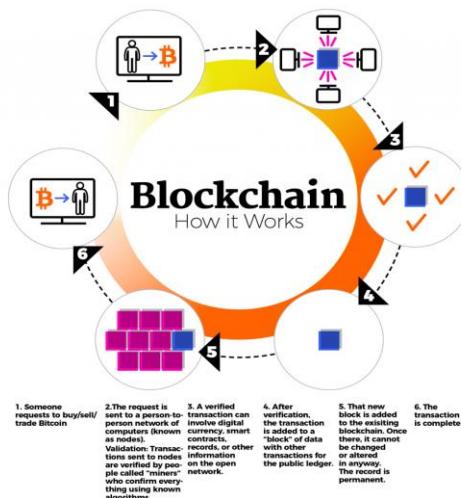
| No | Waktu        | X <sub>1</sub> | X <sub>2</sub> | X <sub>3</sub> | X <sub>4</sub> | Y   |
|----|--------------|----------------|----------------|----------------|----------------|-----|
| 1  | Januari 2004 | 253            | 131            | 259            | 95             | 263 |
| 2  | Januari 2005 | 131            | 259            | 95             | 263            | 50  |
| 3  | Januari 2006 | 259            | 95             | 263            | 50             | 154 |
| 4  | Januari 2007 | 95             | 263            | 50             | 154            | 73  |
| 5  | Januari 2008 | 263            | 50             | 154            | 73             | 141 |
| 6  | Januari 2009 | 50             | 154            | 73             | 141            | 146 |
| 7  | Januari 2010 | 154            | 73             | 141            | 146            | 87  |
| 8  | Januari 2011 | 73             | 141            | 146            | 87             | 139 |

- a. Proses Training (dengan data dari Januari 2004 sampai 2011)  
b. Proses Testing (dengan data dari Januari 2004 sampai 2011) dan Visualisasikan hasilnya untuk dibandingkan dengan data aktualnya.  
c. Proses Testing (untuk menentukan hasil prediksi pada Januari 2012, sampai 2020)  
d. Proses Testing (untuk menentukan hasil prediksi pada Januari 2021, sampai 2100) dan Visualisasikan hasilnya.
5. Berdasarkan no.2, lakukan proses perhitungan dari data Curah Hujan untuk Prediksi berikut dengan algoritma CNN base SVR.
- a. Proses Training (dengan data dari Januari 2004 sampai 2011)  
b. Proses Testing (dengan data dari Januari 2004 sampai 2011) dan Visualisasikan hasilnya untuk dibandingkan dengan data aktualnya.  
c. Proses Testing (untuk menentukan hasil prediksi pada Januari 2012, sampai 2020)  
d. Proses Testing (untuk menentukan hasil prediksi pada Januari 2021, sampai 2100) dan Visualisasikan hasilnya.

# BAB 17 Prototype Deep Blockchain

## 17.1 Pengantar

Dalam website IBM disebutkan, bahwa sebuah whitepaper berjudul "Bitcoin: A Peer-to-Peer Electronic Cash System" yang dipublikasikan oleh Satoshi Nakamoto (2008) menyebutkan solusi menransfer pembayaran sebagai alternatif yang sangat baik dan aman tanpa keterlibatan otoritas pusat (seperti bank) dengan menggabungkan beberapa teknik kriptografi dan jaringan peer-to-peer, atau disebut dengan Cryptocurrency bernama Bitcoin. Seiring berkembangnya bitcoin kemudian, mulai saat dikombinasikan dengan sistem penyimpanan terdistribusi atau dikenal dengan istilah "Blockchain" yang penerapannya jauh lebih luas, yaitu bukan hanya untuk transaksi keuangan dalam bentuk transfer pembayaran atau cryptocurrency. Sejak itu, blockchain telah mulai digunakan oleh industri dan telah menjadi fundamental teknologi dibalik cryptocurrency digital seperti Bitcoin, teknologi komputasi terdistribusi seperti Ethereum, dan open-source framework seperti Hyperledger Fabric, di mana IBM Blockchain Platform dibangun [Kansal, S., 2020]. Saat ini blockchain sudah banyak diterapkan misal pada tabungan dengan underlying emas, ada juga yang menggunakan pada supply chain [Dean, M., 2018].



Gambar 17.1 Blockchain dan Bagaimana Cara Kerjanya

Blockchain adalah cara menyimpan data digital yang dapat berupa jenis data apa saja (termasuk bitcoin, properti dari supply chain, dan tujuan umum lainnya), misal pada bitcoin, propertinya berupa log transaksi dari satu akun ke akun lainnya yang semua datanya disimpan dalam bentuk blok taut atau dirantai menggunakan hash kriptografi seperti Elliptic Curve Cryptosystem (ECC) [Cholissodin, I., 2007][Taleb, N., 2019], di mana dalam implementasinya dapat menggunakan Multiple Elliptic Curves Digital Signature Algorithm (MECDSA) [Bi, W., 2018], atau lainnya, dengan beberapa hard constraints berikut:

- Blok tidak dapat dimodifikasi setelah ditambahkan, dengan kata lain, hanya bisa ditambahkan (*append only*).
- Ada aturan khusus untuk menambahkan data ke dalamnya.
- Arsitekturnya didistribusikan.

Berikut manfaat dari constraints tersebut.

- *Immutability* data terjamin, yaitu isi data bersifat permanen dan tidak dapat dimodifikasi (hanya dapat ditambahkan) serta memiliki ketahanan yang tinggi.
- Tidak ada titik kontrol tunggal dan juga tidak ada titik kegagalan tunggal, artinya semuanya terdesentralisasi antar node yang terdistribusi dan bersanad jelas.
- Mudah untuk melakukan verifikasi untuk mengaudit jejak dari urutan data yang telah ditambahkan.

## 17.2 Studi Kasus: Blockchain Pada Isi Informasi

Isi informasi ini bisa sangat luas, misal untuk informasi *tracking* status seseorang terkait Covid-19 atau lainnya dalam bidang e-Health, terkait *tracking* barang dalam bidang e-Trade, dan sebagainya. Sebagai ilustrasi untuk memudahkan pemahaman terkait blockchain, pada study kasusnya misal diterapkan blockchain pada content informasi yang diposting yang memungkinkan untuk saling dishare dengan struktur data pada blockchain, misal yang terdiri dari tiga elemen,

yaitu Content, Author, Timestamp, yang mana content informasi tersebut tidak akan dapat berubah atau bersifat permanen melalui aplikasi tertentu, misal dalam bentuk web sederhana dengan langkah-langkah berikut:

### 1. Store transactions into blocks

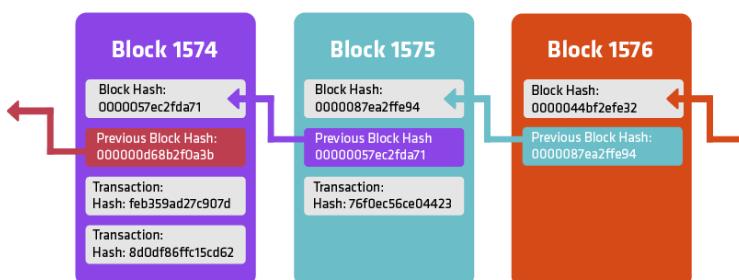
Misal struktur data dalam blockchain digunakan format JSON. Dan dalam ilustrasi ini, term "data" sering diganti dengan istilah "transaksi" yang dikemas ke dalam block. Block dapat berisi satu atau banyak transaksi. Karena sangat memungkinkan ada beberapa block, maka setiap block harus memiliki ID yang unik.

### 2. Add digital fingerprints to the blocks

Menambahkan semacam sidit jari atau semacam tanda tangan digital untuk mencegah segala bentuk gangguan atau serangan pada data yang disimpan di dalam blok, dideteksi menggunakan fungsi hash kriptografis. Di mana fungsi hash adalah fungsi yang mengambil data dari berbagai ukuran dan menghasilkan data dengan ukuran tetap. Upaya yang diperlukan untuk mengetahui hash dari input (mudah) sedangkan mencari tahu input dari hash (hampir tidak mungkin). Ada berbagai fungsi hash yang populer, misal menggunakan fungsi hashing SHA-256.

### 3. Chain the blocks

Blockchain merupakan kumpulan blok, dan setiap blok dalam koleksi setara dengan array. Hal tersebut tidak cukup untuk keamanan, maka dibutuhkan cara untuk memastikan bahwa setiap perubahan di blok sebelumnya akan membatalkan seluruh rantai (invalidates the entire chain).



Gambar 17.2 Blockchain - previous\_hash dan hash block

Misal pada bitcoin, yaitu dengan menciptakan ketergantungan di antara blok-blok secara berturut-turut (*chaining*), yaitu memasukkan hash dari blok sebelumnya dalam blok saat ini atau yang disebut dengan *previous\_hash*. Jika setiap block saling ketergantungan, bagaimana dengan block pertama (block genesis). Caranya dapat menambahkan *previous\_hash* nya ke dalam kelas Block dan menerapkannya pada struktur kelas Blockchain pada pembangkitan block awalnya (block genesis). Jadi setiap block, berdasarkan Gambar 17.2 memiliki *previous\_hash* dan hash dari block itu sendiri, di mana *previous\_hash* disebut juga *parent\_hash*. Jika konten dari salah satu blok sebelumnya berubah, maka hashnya pun akan berubah yang akan menyebabkan ketidakcocokan dengan *previous\_hash* di blok berikutnya. Pada akhirnya, seluruh rantai yang mengikuti blok yang diganti menjadi tidak valid, dan satu-satunya cara untuk memperbaikinya adalah dengan menghitung ulang seluruh rantai (recompute the entire chain).

#### 4. Implement a proof of work algorithm

Namun terdapat satu masalah. Jika seseorang penyerang mengubah blok sebelumnya, hash dari semua blok yang mengikuti dapat dihitung ulang dengan mudah untuk membuat blockchain yang seolah-olah valid. Untuk mencegah hal ini, kita dapat memanfaatkan semacam asimetri *concept* atau dengan ditambahkan kendala tertentu dalam fungsi hash untuk membuat menghitung hash menjadi sulit (hampir tidak mungkin) dan acak. Misal dalam hal ini, ditambahkan kendala, dengan mensyaratkan hash harus mulai dengan "n nol diawali" di mana n dapat berupa bilangan bulat positif.

Tanpa mengubah data blok, hash tidak akan berubah, dan tentu saja kita tidak ingin mengubah data yang ada. Misal kita akan menambahkan beberapa data dummy yang dapat kami ubah, dan pada block ditambahkan field baru yang disebut nonce. Nonce adalah angka yang bisa terus menerus diubah sampai mendapatkan hash yang memenuhi constraints. Nonce yang memenuhi constraints berfungsi sebagai "proof of work" bahwa beberapa perhitungan telah dilakukan. Teknik ini adalah versi sederhana dari algoritma Hashcash yang digunakan dalam Bitcoin. Besarnya jumlah nol yang disimpan pada variabel "difficulty", maka akan sangat menen-

tukan tingkat kesulitan dari algoritma "proof of work" (semakin besar jumlah nol, semakin sulit untuk mengetahui nilai *nonce* yang memenuhi *constraints*).

Hal tersebut menunjukkan, karena asimetri, maka "proof of work" menjadi sulit untuk dihitung tetapi sangat mudah untuk memverifikasi setelah Anda mengetahui nonce (Anda hanya perlu menjalankan fungsi hash lagi).

#### 5. Add blocks to the chain

Untuk menambahkan block ke dalam chain, pertama harus diverifikasi bahwa:

- Data belum dirusak ("proof of work" yang diberikan sudah "bernilai benar" atau "correct").
- Urutan transaksi dipertahankan (field `previous_hash` dari block yang akan ditambahkan akan diambilkan dari hash block terakhir dalam chain yang ada).

Transaksi yang masuk awalnya akan disimpan sebagai kumpulan transaksi yang belum dikonfirmasi (pending transactions to the blockchain). Proses menempatkan transaksi yang belum dikonfirmasi dalam block dan masih dalam dalam proses menghitung "proof of work" dikenal sebagai "**Mining of blocks**". Jika sekali saja nilai *once* telah memenuhi kondisi *constraints*, maka dapat dikatakan bahwa sebuah blok telah ditambang (mined) dan dapat dimasukkan ke dalam blockchain, dan sebaliknya. Pada sebagian besar cryptocurrency (termasuk Bitcoin), penambang (miners) dapat diberikan beberapa award cryptocurrency sebagai hadiah (reward) atas usaha mereka menggunakan sumber daya komputasi yang ada dari mereka untuk menghitung "proof of work".

#### 6. Create interfaces

Membuat antarmuka untuk blockchain node untuk kemudahan berinteraksi dengan aplikasi yang dibangun, misal menggunakan microframework web dengan Flask dari Python untuk membuat REST API, atau dengan framework lainnya.

#### 7. Establish consensus & decentralization

Kita diminta untuk menetapkan *consensus* (kesepakatan bersama antar stakeholder dalam jaringan) dan melakukan *decentralization*

untuk komputasi terdistribusinya menggunakan Interoperability (sistem untuk melakukan pertukaran dan penggunaan informasi antar berbagai unit, di mana unit-unit tersebut dalam bentuk node-node ) seperti pada Gambar 17.3. Di mana jika blockchain yang digunakan masih hanya berjalan di satu komputer, meskipun telah menautkan block disertai field hash dan telah menyelesaikan "proof of work", maka dinilai sistem penyelesaian tersebut masih belum dapat dipercayai 100% karena masih dari satu entitas (yaitu masih hanya dari satu mesin atau satu komputer). Maka dari itu, dibutuhkan pendistribusian data ke beberapa node untuk proses "Mining" yang lebih luas. Jadi, untuk beralih dari satu node ke jaringan peer-to-peer, maka dibuat mekanisme untuk membuat node baru dengan mengetahui peer lain dalam jaringan (komputasi terdistribusi).



Gambar 17.3 Consensus & Decentralization based Interoperability

Untuk mendaftar menjadi node yang ada di jaringan blockchain, yaitu dengan mengikuti hal-hal berikut:

- Meminta node jarak jauh (remote node) untuk menambahkan new peer ke daftar peers yang dikenal.
- Menginisialisasi blockchain dari new node dengan remote node.
- Menyinkronkan ulang blockchain dengan jaringan jika node tidak terhubung.

Namun, terkadang terdapat masalah ketika menggunakan banyak node. Permasalahan tersebut bisa karena manipulasi yang disengaja atau alasan yang tidak disengaja (seperti latensi

jaringan), sehingga membuat salinan rantai (copy of chains) beberapa node dapat berbeda. Dalam hal ini, node perlu menyepakati beberapa versi rantai untuk menjaga integritas seluruh sistem (perlu suatu algoritma konsensus). Algoritma konsensus sederhana bisa untuk menyepakati valid chain terpanjang ketika node chain yang berpartisipasi terlihat berbeda dalam jaringan, yaitu nampak terlihat menyimpang. Konsensus ini wajib disetujui oleh semua node yang sudah terverifikasi sebelumnya dalam jaringan. Alasan di balik pendekatan ini adalah bahwa chain terpanjang adalah perkiraan yang baik dari jumlah pekerjaan yang paling banyak dilakukan (mengingat bahwa "proof of work" memang sulit untuk dihitung).

Selanjutnya, perlu dikembangkan cara agar setiap node dapat mengumumkan ke jaringan ketika ia telah berhasil menambang block (*mined a block*) sehingga semua rekan dalam jaringan tersebut dapat memperbarui blockchain mereka dan beralih untuk menambang (*mine*) transaksi lainnya. Node lain dapat dengan mudah memverifikasi "proof of work" dan menambahkan blok yang ditambang (*mined block*) ke rantai masing-masing (ingat bahwa verifikasi itu mudah sekali setelah *nonce* diketahui).

#### 8. Build the application

Build aplikasi sehingga dapat terhubung ke sebuah node atau banyak node di jaringan blockchain untuk mengambil data dan juga mengirimkan data baru.

#### 9. Run the application ( clone dari [https://github.com/satwik-kansal/python\\_blockchain\\_app](https://github.com/satwik-kansal/python_blockchain_app) )

Link kode simulasi blockchain lainnya:

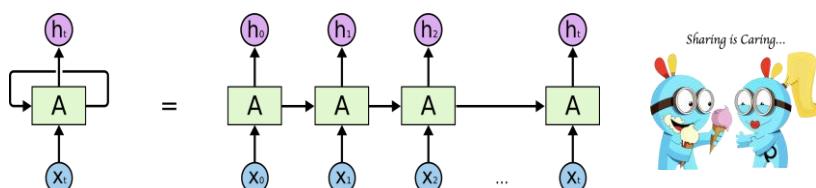
[https://drive.google.com/drive/folders/1\\_KI4mVA1wjx0PXm-PeBDwLepEtvA-we\\_?usp=sharing](https://drive.google.com/drive/folders/1_KI4mVA1wjx0PXm-PeBDwLepEtvA-we_?usp=sharing)



# BAB 18 Recurrent Neural Networks (RNN) Native dan Modified Long Short-Term Memory (MLSTM) Berbasis ELM

## 18.1 Pengantar

Algoritme Recurrent Neural Network atau RNN pada penelitian yang dilakukan oleh Rehman et al. (2014) memiliki beberapa kelebihan diantaranya yaitu dapat menghilangkan koneksi yang berlebihan, waktu yang efektif. Algoritme Recurrent Neural Network dapat dikombinasikan dengan algoritme lainnya misalnya Extreme Learning Machine. Kombinasi ini menghasilkan algoritme RELMNN atau Recurrent Extreme Learning Machine Neural Network. RELMNN menambahkan mekanisme recurrent dalam jaringan ELM, dengan satu atau lebih output yang dikembalikan ke sistem sebagai input baru. RELMNN dibangun dengan tujuan menangani dataset yang sekuensial berdasarkan waktu (*time series*), kembalian nilai output ke sistem akan meningkatkan kemampuan jaringan dalam melatih dan beradaptasi (Ertugrul, 2016). RNN memiliki "memori" atau "remembering" sebagai ciri khas, sehingga dapat dikatakan sangat berbeda dengan jaringan saraf lainnya dan berbagi parameter yang memungkinkan dapat dibuat berbagai arsitektur RNN yang berbeda. Memori tersebut digunakan untuk mengingat semua informasi tentang, apa yang telah dihitung (*sharing output*) pada keadaan sebelumnya, untuk digunakan pada proses pada waktu berikutnya, dengan parameter yang sama untuk digunakan sebagai masukan pada semua node pada input layer atau hidden layer untuk menghasilkan keluaran target sebagai output.

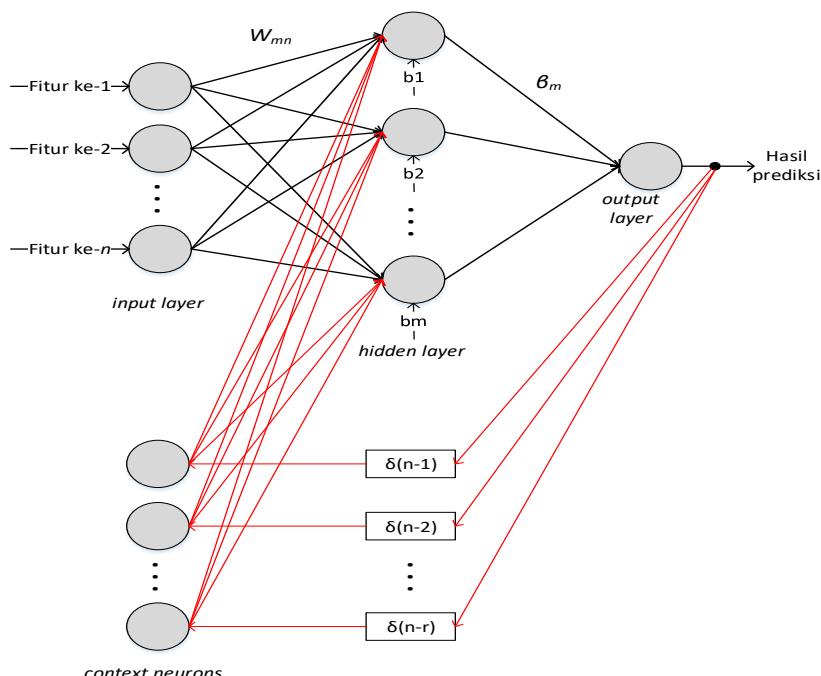


Gambar 18.1 Memori RNN untuk Sharing hasil ke Next Proses

RNN cenderung mengalami penumpukan besarnya gradien yang mengakibatkan saling berbenturan antar nilai-nilai gradien tersebut yang mengakibatkan masalah, yakni nilai-nilai gradien menjadi semakin tidak jelas atau malah menghilangkan nilai akumulasinya. Untuk mengatasi hal tersebut, maka beberapa penelitian mengembangkan RNN dengan dilengkapi Long Short-Term Memory (LSTM) dan lainnya.

## 18.2 Struktur Algoritma RNN

Algoritme RELMNN memiliki arsitektur yang sedikit berbeda dibanding arsitektur jaringan ELM. Perbedaannya terdapat pada mekanisme recurrent yang disimpan pada context neurons yang merupakan neuron dengan fungsi untuk menyimpan nilai delayed output. Neuron tersebut dijalankan seperti neuron input tambahan. Arsitektur RELMNN terdapat pada Gambar, di mana arsitektur jaringan yang digunakan dalam satu data, sedangkan proses dalam algoritme RELMNN menggunakan keseluruhan data dalam satu iterasi.



Gambar 18.2 Jaringan Algoritme Recurrent Extreme Learning Machine Neural Network

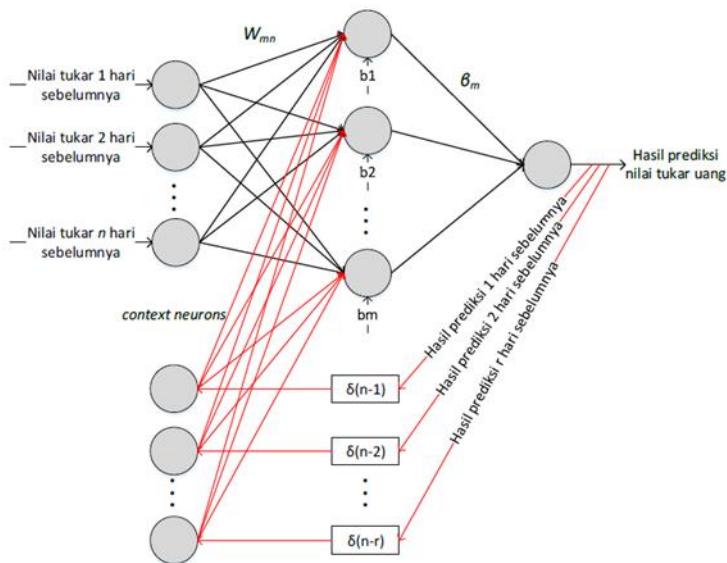
Sumber: Ertugrul (2016)

## 18.3 Studi Kasus: RNN untuk Prediksi Nilai Tukar Dollar ke Rupiah

Sistem prediksi menggunakan masukan berupa data nilai tukar Rupiah Indonesia terhadap Dolar Amerika Serikat dan parameter lain yaitu: jumlah fitur, jumlah hidden neuron, jumlah context neuron, nilai input bobot yang didapatkan secara random dalam range [-1,1] dan nilai bias yang didapatkan secara random dalam range [0,1]. Dalam pengerjaannya, metode RELMNN terbagi menjadi dua bagian yaitu proses training dan proses testing. Namun, sebelum melakukan proses tersebut, perlu dilakukan proses normalisasi data pada range [0.1,0.9]. Setelah itu, proses training dilakukan dengan menggunakan data latih, sampai seluruh tahapan proses training selesai dilakukan dan didapatkan nilai bobot, bias, matriks delay, dan matriks bobot keluaran yang dibutuhkan pada proses selanjutnya. Selanjutnya, proses testing dilakukan dengan menggunakan data uji, sampai didapatkan hasil prediksi dan didapatkan nilai evaluasi dengan perhitungan MAPE. Tahapan terakhir yaitu melakukan denormalisasi hasil prediksi, untuk mendapatkan hasil prediksi berupa nilai tukar uang.

Tabel 18.1 Data nilai tukar Rupiah terhadap Dolar Amerika Serikat

| Data ke- <i>i</i> | Tanggal     | Fitur 1  | Fitur 2  | Fitur 3  | Nilai Tukar |
|-------------------|-------------|----------|----------|----------|-------------|
| 1                 | 24 Feb. '17 | 13360    | 13342    | 13311    | 13340       |
| 2                 | 25 Feb. '17 | 13342    | 13311    | 13340    | 13330       |
| 3                 | 26 Feb. '17 | 13311    | 13340    | 13330    | 13330       |
| 4                 | 27 Feb. '17 | 13340    | 13330    | 13330    | 13355       |
| 5                 | 28 Feb. '17 | 13330    | 13330    | 13355    | 13346       |
| 6                 | 1 Mar. '17  | 13330    | 13355    | 13346    | 13340       |
| 7                 | 2 Mar. '17  | 13355    | 13346    | 13340    | 13383       |
| 8                 | 3 Mar. '17  | 13346    | 13340    | 13383    | 13373       |
| 9                 | 4 Mar. '17  | 13340    | 13383    | 13373    | 13323       |
| 10                | 5 Mar. '17  | 13383    | 13373    | 13323    | 13323       |
| 11                | 6 Mar. '17  | 13373    | 13323    | 13323    | 13322.75    |
| 12                | 7 Mar. '17  | 13323    | 13323    | 13322.75 | 13337.75    |
| 13                | 8 Mar. '17  | 13323    | 13322.75 | 13337.75 | 13407       |
| 14                | 9 Mar. '17  | 13322.75 | 13337.75 | 13407    | 13394       |
| 15                | 10 Mar. '17 | 13337.75 | 13407    | 13394    | 13363       |



Gambar 18.3 Arsitektur Jaringan RNN untuk Prediksi Nilai Tukar

Berikut adalah langkah-langkah perhitungan manualisasi RELMNN, yaitu dimulai dengan,

1. Menentukan jumlah fitur, *hidden neuron*, dan *context neuron*. Dalam manualisasi ini, inisialisasi jumlah fitur ( $n$ ) yang digunakan sebanyak 3, sedangkan jumlah *hidden neuron* ( $m$ ) sebanyak 2, dan jumlah *context neuron* ( $r$ ) sebanyak 2. Sedangkan datasets dengan jumlah 15 data dibagi menjadi 2 bagian yaitu untuk data training sebanyak 10 data dan data testing sebanyak 5 data.
2. Lakukan Normalisasi Data, misal berikut hasilnya.

| Data ke- <i>i</i> | X1       | X2       | X3       | T        |
|-------------------|----------|----------|----------|----------|
| 1                 | 0,508333 | 0,358333 | 0,100000 | 0,341667 |
| 2                 | 0,358333 | 0,100000 | 0,341667 | 0,258333 |
| 3                 | 0,100000 | 0,341667 | 0,258333 | 0,258333 |
| 4                 | 0,341667 | 0,258333 | 0,258333 | 0,466667 |
| 5                 | 0,258333 | 0,258333 | 0,466667 | 0,391667 |
| 6                 | 0,258333 | 0,466667 | 0,391667 | 0,341667 |
| 7                 | 0,466667 | 0,391667 | 0,341667 | 0,700000 |
| 8                 | 0,391667 | 0,341667 | 0,700000 | 0,616667 |
| 9                 | 0,341667 | 0,700000 | 0,616667 | 0,200000 |
| 10                | 0,700000 | 0,616667 | 0,200000 | 0,200000 |

|    |          |          |          |          |
|----|----------|----------|----------|----------|
| 11 | 0,616667 | 0,200000 | 0,200000 | 0,197917 |
| 12 | 0,200000 | 0,200000 | 0,197917 | 0,322917 |
| 13 | 0,200000 | 0,197917 | 0,322917 | 0,900000 |
| 14 | 0,197917 | 0,322917 | 0,900000 | 0,791667 |
| 15 | 0,322917 | 0,900000 | 0,791667 | 0,533333 |

3. Masuk ke Proses Training, lihat pada sub bab berikutnya.

### 18.3.1 Proses Training

Adapun sampel data nilai tukar Rupiah Indonesia terhadap Dolar Amerika Serikat yang digunakan pada manualisasi ini adalah data dengan parameter teknikal yang berjumlah 15 record dengan 3 fitur.

1. Inisialisasi matriks delay  $\delta$  dengan Persamaan  $\delta_{tr} = T(t-(n+r)+n)$ .

Berikut contoh menghitung nilai delay  $\delta$ .

$$\delta_{1,1} = T(1-(3+1)+3)$$

$$\delta_{1,1} = T(0) = 0$$

$$\delta_{2,1} = T(2-(3+1)+3)$$

$$\delta_{2,1} = T(1) = 0,341667$$

Berikut hasil inisialisasi matriks delay  $\delta$  dimana  $t$  adalah urutan data,  $r$  adalah urutan *context neuron*.

| Data ke-t | r1       | r2       |
|-----------|----------|----------|
| 1         | 0,000000 | 0,000000 |
| 2         | 0,341667 | 0,000000 |
| 3         | 0,258333 | 0,341667 |
| 4         | 0,258333 | 0,258333 |
| 5         | 0,466667 | 0,258333 |
| 6         | 0,391667 | 0,466667 |
| 7         | 0,341667 | 0,391667 |
| 8         | 0,700000 | 0,341667 |
| 9         | 0,616667 | 0,700000 |
| 10        | 0,200000 | 0,616667 |

2. Membuat nilai random untuk matriks  $W'_{m(n+r)}$  sebagai bobot masukan (input weight) dengan range [-1;1], dalam bentuk array ukuran  $m$  (jumlah hidden neuron)  $\times n+r$  (jumlah fitur + context neurons). Kemudian buat nilai random untuk matriks bias  $b$  dengan range [0;1] dalam ukuran  $1 \times$  (jumlah hidden neuron).

## 2.1. Membuat nilai random untuk matriks bobot

| M\N+R | 1         | 2        | 3         |
|-------|-----------|----------|-----------|
| 1     | -0,956910 | 0,000000 | 0,476625  |
| 2     | -0,863870 | 0,000000 | -0,415650 |

Lanjutan

| M\N+R | 4        | 5         |
|-------|----------|-----------|
| 1     | 0,291437 | 0,922641  |
| 2     | 0,517960 | -0,431670 |

## 2.2. Membuat nilai random untuk bias.

|   | 1        | 2        |
|---|----------|----------|
| 1 | 0,488982 | 0,286403 |

3. Menghitung nilai matriks keluaran pada hidden layer dengan menggunakan

$$H' = \frac{1}{1 + \exp(-( [x_{train}, \delta] W'^T + b(ones(i_{train}, 1), :)))}$$

Keterangan:

- $H'$  : matriks keluaran hidden layer pada proses recurrent  
 $x_{train}$  : matriks input yang telah di normalisasi  
 $\delta$  : matriks delay  
 $[x_{train}, \delta]$  : matriks gabungan dari matriks  $x_{train}$  dan  $\delta$   
 $W'^T$  : matriks transpose dari bobot pada proses recurrent  
 $i_{train}$  : jumlah data latih  
 $b$  : matriks bias

- 3.1. Menggabungkan matriks data latih yang hanya berisi fitur ( $x$ ) dengan matriks delay, dengan cara menambahkan kolom pada matriks data latih dengan matriks delay.

| No | x1     | x2     | x3     | d1     | d2     |
|----|--------|--------|--------|--------|--------|
| 1  | 0,5083 | 0,3583 | 0,1000 | 0,0000 | 0,0000 |
| 2  | 0,3583 | 0,1000 | 0,3417 | 0,3417 | 0,0000 |
| .. |        |        |        |        |        |
| 10 | 0,7000 | 0,6167 | 0,2000 | 0,2000 | 0,6167 |

- 3.2. Memperbanyak matriks bias sebanyak jumlah data latih, dengan cara memperbanyak baris matriks bias sejumlah data latih dengan nilai yang sama.

|    | 1      | 2       |
|----|--------|---------|
| 1  | 0,4889 | 0,28640 |
| 2  | 0,4889 | 0,28640 |
| .. |        |         |
| 10 | 0,4889 | 0,28640 |

- 3.3. Menghitung perkalian matriks gabungan dengan transpose matriks bobot.

$$[x_{train}, \delta] W'^T$$

|    | 1       | 2       |
|----|---------|---------|
| 1  | -0,1652 | -0,5806 |
| 2  | -0,0041 | -0,3025 |
| .. |         |         |
| 10 | 0,5236  | -1,0223 |

- 3.4. Menghitung penjumlahan matriks hasil perkalian dengan matriks bias.

$$([x_{train}, \delta] W'^T + b(ones(i_{train}, 1), :))$$

|    | 1      | 2        |
|----|--------|----------|
| 1  | 0,3238 | -0,2942  |
| 2  | 0,4849 | -0,0161  |
| .. |        |          |
| 10 | 1,0125 | -0,73593 |

- 3.5. Menghitung fungsi aktivasi dari hasil penjumlahan untuk mendapatkan matriks keluaran pada hidden layer

$$H' = \frac{1}{1 + \exp\left(-([x_{train}, \delta]W'^T + b(ones(i_{train}, 1), :))\right)}$$

| $H'$ | 1      | 2      |
|------|--------|--------|
| 1    | 0,5802 | 0,4270 |
| 2    | 0,6189 | 0,4960 |
| ..   |        |        |
| 10   | 0,7335 | 0,3239 |

4. Menghitung  $\hat{\beta}'$  sebagai bobot keluaran setelah proses recurrent dengan menggunakan Persamaan berikut

$$\hat{\beta}' = H'^+ T$$

dimana  $H'^+ = (H'^T H')^{-1} H'^T$  atau disebut dengan matriks Moore-Penrose Pseudo Invers.

Keterangan:

$\hat{\beta}'$  : matriks bobot keluaran setelah proses *recurrent*

$H'^+$  : matriks Moore-Penrose Pseudo Invers pada proses *recurrent*

$T$  : matriks target

$H'$  : matriks keluaran hidden layer pada proses *recurrent*

- 4.1. Menghitung perkalian antara transpose dari matriks keluaran hidden layer dengan matriks keluaran hidden layer

| $H'^T H'$ | 1      | 2      |
|-----------|--------|--------|
| 1         | 5,3468 | 3,1704 |
| 2         | 3,1704 | 1,9375 |

4.2. Menghitung matriks invers dari hasil perhitungan  $H'^T H'$ .

| $(H'^T H')^{-1}$ | 1      | 2      |
|------------------|--------|--------|
| 1                | 5,3468 | 3,1704 |
| 2                | 3,1704 | 1,9375 |

4.3. Menghitung matriks Moore-Penrose Pseudo Invers dengan mengalikan matriks invers dengan matriks transpose dari  $H'$ .  
 $H'^+ = (H'^T H')^{-1} H'^T$

|   | 1       | 2       | .. | 10      |
|---|---------|---------|----|---------|
| 1 | -0,7456 | -1,2131 | .. | 1,2810  |
| 2 | 1,4405  | 2,2411  | .. | -1,9290 |

4.4. Menghitung  $\hat{\beta}'$  sebagai bobot keluaran

| $\hat{\beta}'$ | 1      |
|----------------|--------|
| 1              | 0,1291 |
| 2              | 0,6461 |

5. Menghitung hasil prediksi menggunakan Persamaan berikut  
 $\hat{Y}' = H' \hat{\beta}'$

| $\hat{Y}'$ | 1        |
|------------|----------|
| 1          | 0,350771 |
| 2          | 0,400339 |
| 3          | 0,418879 |
| 4          | 0,385277 |
| 5          | 0,406606 |
| 6          | 0,388423 |
| 7          | 0,358998 |
| 8          | 0,387013 |
| 9          | 0,363681 |
| 10         | 0,303961 |

### 18.3.2 Proses Testing

Berikut beberapa tahapan untuk proses testing dari sampel data nilai tukar Rupiah Indonesia terhadap Dolar Amerika Serikat yang digunakan pada penelitian ini adalah data dengan parameter teknikal yang berjumlah 5 record dengan 3 fitur.

1. *Load  $W'_{mn}$ ,  $b$ , dan  $\hat{\beta}'$  dari proses training.*
2. Inisialisasi matriks delay  $\delta'$  dengan Persamaan  $\delta_{tr} = T(t-(n+r)+n)$ .

Berikut contoh menghitung nilai delay  $\delta$ .

$$\delta_{11,1} = T(11-(3+1)+3)$$

$$\delta_{11,1} = T(10) = 0,2000$$

Berikut hasil inisialisasi matriks delay  $\delta'$  dimana  $t$  adalah urutan data,  $r$  adalah urutan *context neuron*.

| Data ke-t | r1       | r2       |
|-----------|----------|----------|
| 11        | 0,200000 | 0,200000 |
| 12        | 0,197917 | 0,200000 |
| 13        | 0,322917 | 0,197917 |
| 14        | 0,900000 | 0,322917 |
| 15        | 0,791667 | 0,900000 |

3. Menghitung nilai matriks keluaran pada hidden layer dengan menggunakan Persamaan berikut.

$$H' = \frac{1}{1 + \exp\left(-([x_{test}, \delta'] W'^T + b(\text{ones}(i_{test}, 1), :))\right)}$$

Keterangan:

- $H'$  : matriks keluaran hidden layer pada proses recurrent  
 $x_{test}$  : matriks input yang telah di normalisasi  
 $\delta'$  : matriks delay  
 $[x_{test}, \delta']$  : matriks gabungan dari matriks  $x_{test}$  dan  $\delta'$   
 $W'^T$  : matriks transpose dari bobot pada proses recurrent  
 $i_{test}$  : jumlah data uji  
 $b$  : matriks bias

3.1. Menggabungkan matriks data uji yang hanya berisi fitur ( $x$ ) dengan matriks delay, dengan cara menambahkan kolom pada matriks data uji dengan matriks delay.

| No | x1     | x2     | x3     | d1     | d2     |
|----|--------|--------|--------|--------|--------|
| 1  | 0,6167 | 0,2000 | 0,2000 | 0,2000 | 0,2000 |
| 2  | 0,2000 | 0,2000 | 0,1979 | 0,1979 | 0,2000 |
| .. |        |        |        |        |        |
| 5  | 0,3229 | 0,9000 | 0,7917 | 0,7917 | 0,9000 |

3.2. Memperbanyak matriks bias sebanyak jumlah data uji, dengan cara memperbanyak baris matriks bias sejumlah data uji dengan nilai yang sama.

|    | 1      | 2       |
|----|--------|---------|
| 1  | 0,4889 | 0,28640 |
| 2  | 0,4889 | 0,28640 |
| .. |        |         |
| 5  | 0,4889 | 0,28640 |

3.3. Menghitung perkalian matriks gabungan dengan transpose matriks bobot.

$$[x_{test}, \delta'] W'^T$$

|    | 1       | 2       |
|----|---------|---------|
| 1  | -0,0993 | -0,6543 |
| 2  | 0,2979  | -0,2946 |
| .. |         |         |
| 5  | 1,8166  | -0,8373 |

3.4. Menghitung penjumlahan matriks hasil perkalian dengan matriks bias.

$$([x_{test}, \delta'] W'^T + b(ones(i_{test}, 1), :))$$

|    | 1      | 2       |
|----|--------|---------|
| 1  | 0,3897 | -0,3679 |
| 2  | 0,7868 | -0,0082 |
| .. |        |         |
| 5  | 2,3055 | -0,5509 |

3.5. Menghitung fungsi aktivasi dari hasil penjumlahan untuk mendapatkan matriks keluaran pada hidden layer

$$H' = \frac{1}{1 + \exp\left(-([x_{test}, \delta'] W'^T + b(ones(i_{test}, 1), :))\right)}$$

| $H'$ | 1      | 2      |
|------|--------|--------|
| 1    | 0,5962 | 0,4090 |
| 2    | 0,6871 | 0,4980 |
| ..   |        |        |
| 5    | 0,9093 | 0,3657 |

4. Menghitung hasil prediksi menggunakan Persamaan berikut  
 $\hat{Y}' = H' \hat{\beta}'$

| $\hat{Y}'$ | 1        |
|------------|----------|
| 1          | 0,341241 |
| 2          | 0,410422 |
| 3          | 0,415248 |
| 4          | 0,425723 |
| 5          | 0,353637 |

5. Denormalisasi hasil prediksi dengan menggunakan Persamaan berikut,

$$denorm = \frac{norm - 0,1}{0,8} (max - min) + min$$

Di mana  $norm = [0,1;0,8]$  atau diubah sesuai kebutuhan dan harus konsisten mulai dari proses awal normalisasi data.

Keterangan:

- $norm$  : nilai data setelah proses normalisasi  
 $min$  : atau nilai dari hasil proses data yang sudah dinormalisasi  
 $max$  : nilai minimum dari seluruh data asli  
 $denorm$  : atau disesuaikan dengan kondisi data  
          : nilai maksimum dari seluruh data asli  
          : atau disesuaikan dengan kondisi data  
          : nilai data setelah proses denormalisasi

Nilai max dan min merupakan nilai yang didapatkan pada proses normalisasi

| $\widehat{Y}'$ | 1            |
|----------------|--------------|
| 1              | 13339,948909 |
| 2              | 13348,250631 |
| 3              | 13348,829808 |
| 4              | 13350,086772 |
| 5              | 13341,436448 |

## 6. Menghitung nilai evaluasi dengan menggunakan MAPE

$$MAPE = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{P_i - A_i}{A_i} \right|$$

$$MAPE = 0,2262 \%$$

Source Code 18.1 Koding Recurrent Extreme Learning Machine Neural Network (RELMNN)

```
#Koding RELMNN

=====
By: Imam Cholissodin | imamcs@ub.ac.id
Dosen Fakultas Ilmu Komputer (Filkom)
Universitas Brawijaya (UB)
Tgl 26 October 2021
Semoga Bermanfaat. Aamiin :D
=====
#
11th ICGT 2021 dan lainnya
#
from google.colab import drive
drive.mount('/content/drive')

Buat Folder
import os
os.chdir("/content/drive/My Drive")
path = "Publikasi GreenTech 2021/Koding LSTM utk Manualisasi"
path = "#Penelitian&Pengmas 2021/Publikasi GreenTech 2021/Ko-
ding LSTM utk Manualisasi"
path hasil = "#Penelitian&Pengmas 2021/Publikasi GreenTech 2021/Ko-
ding LSTM utk Manualisasi/HasilTrainNTest"
if not os.path.exists(path):
 os.makedirs(path)
if not os.path.exists(path_hasil):
 os.makedirs(path_hasil)
os.chdir("Publikasi GreenTech 2021/Koding LSTM utk Manualisasi")
os.chdir("#Penelitian&Pengmas 2021/Publikasi GreenTech 2021/Ko-
ding LSTM utk Manualisasi")

!pwd
!ls -l -a --block-size=K

RNN Native menggunakan ELM
base buku ajar AI, Machine learning & Deep Learning
Link https://www.researchgate.net/publica-
tion/348003841_Buku_Ajar_AI_Machine_Learning_Deep_Learning_(page_343 - 353)
```

```
RNN Native menggunakan ELM pada prediksi data nilai tukar Rupiah (IDR) terhadap Dollar Amerika (US)

download file data nilai_tukar_idr_us.xlsx
!gdown --id 1nYH0cuNTkSkqjzFIda5gdYOPtzkb73K
https://docs.google.com/spreadsheets/d/1nYH0cuNTkSkqjzFIda5gdYOPtzkb73K/edit?usp=sharing&ouid=117226089223813103435&rtpof=true&sd=true

RNN Native menggunakan ELM pada prediksi data nilai tukar Rupiah (IDR) terhadap Dollar Amerika (US)
Load data
import pandas as pd
import numpy as np
import time
file_name = 'data nilai tukar idr us.xlsx'
df_idr_us = pd.read_excel(file_name, sheet_name='Sheet1')
display(df_idr_us)

def myrandfloat(mbaris,nkolom,lower,upper):
 #BatasRANDplusOne=10000
 BatasRANDplusOne=max(10000,2*np.math.ceil(upper))
 #mbaris=1
 #nkolom=1
 Rand_Sample=np.random.randint(BatasRANDplusOne,size=(mbaris,nkolom))
 min_Rand_Sample = 0
 max_Rand_Sample = BatasRANDplusOne - 1
 upper_boundary= upper
 lower_boundary= lower
 normalize_Rand_Sample_minMax=((Rand_Sample-min_Rand_Sample)/(max_Rand_Sample-min_Rand_Sample))* (upper_boundary-lower_boundary)
 #print(normalize_Rand_Sample_minMax)
 #x = normalize_Rand_Sample_minMax
 return normalize_Rand_Sample_minMax

def fn_sigmoid(x):
 return 1. / (1. + np.e**(-x))

def fn_tanh(x):
 return (np.e**x-np.e**(-x)) / (np.e**x+np.e**(-x))

def fn_tanh(x):
return (np.exp(2*x) - 1) / (np.exp(2*x) + 1)

def fn_tanh(x):
return np.tanh(x)

def fn_relu(x):
 return np.maximum(0, x)

def create_matrix_delay(isTimeSeriesData, cara, r, Ntrain, Ntest, Ytrain, Ytest, Target):
 s_tr_train = np.zeros([Ntrain,r])
 s_tr_test = np.zeros([Ntest,r])
 byk_data = Ntrain + Ntest
 if isTimeSeriesData == False:
 if(cara==1):
 # isTimeSeriesData = False | cara ke-1

 for i in range(r):
 s_tr_train [r-((r-1)-i)::,i,None] = Ytrain[:Ntrain-(r-((r-1)-i)),:]
 # print("s_tr_train: \n",s_tr_train)

 for i in range(r):
 s_tr_test[r-((r-1)-i)::,i,None] = Ytest[:Ntest-(r-((r-1)-i)),:]
 # print("s_tr_test: \n",s_tr_test)

 return s_tr_train, s_tr_test
 if(cara==2):
 # isTimeSeriesData = False | cara ke-2

 s_tr_train_cara2 = np.zeros([Ntrain,r])
 for tt in range(Ntrain):
 for rr in range(r):
 if (tt > rr):
 # (tt - (n + rr) + n) - 1 = (tt-rr) - 1, jika indeksnya dimulai dari 0
 s_tr_train_cara2[tt,rr] = Ytrain.flatten()[((tt-rr) - 1)]
```

```
 else:
 s_tr_train_cara2[tt,rr] = 0.
 # print("\n s_tr_train_cara2: \n",s_tr_train_cara2)

 # print("\n")

 s_tr_test_cara2 = np.zeros([Ntest,r])
 for tt in range(Ntest):
 for rr in range(r):
 if (tt > rr):
 # (tt - (n + rr) + n) - 1 = (tt-rr) - 1, jika indeksnya dimulai dari 0
 s_tr_test_cara2[tt,rr] = Ytest.flatten()[((tt-rr) - 1)]
 else:
 s_tr_test_cara2[tt,rr] = 0.
 # print("\n s_tr_test_cara2: \n",s_tr_test_cara2)
 return s_tr_train_cara2, s_tr_test_cara2

if isTimeSeriesData == True:
 if(cara==1):
 # isTimeSeriesData = True | cara ke-1
 # s_tr_train = np.zeros([Ntrain,r])
 for i in range(r):
 s_tr_train [r-((r-1)-i)::,i,None] = Target[:Ntrain-(r-((r-1)-i)),:]
 # print("s_tr_train: \n",s_tr_train)

 # print("\n")

 # s_tr_test = np.zeros([Ntest,r])
 for i in range(r):
 a_a = r
 b_b = -1
 # mencari suku ke-n dgn kaidah deret aritmatika
 un_un = (a_a-1) + ((i+1)-1)*(b_b)
 s_tr_test [:,i,None] = Target[Ntrain-r+un_un:Ntrain-r+Ntest+un_un,:]

 # print("s_tr_test: \n",s_tr_test)

 return s_tr_train, s_tr_test
 if(cara==2):
 # isTimeSeriesData = True | cara ke-2
 s_tr_train_n_test_cara2 = np.zeros([byk_data,r])
 for tt in range(byk_data):
 for rr in range(r):
 if (tt > rr):
 # (tt - (n + rr) + n) - 1 = (tt-rr) - 1, jika indeksnya dimulai dari 0
 s_tr_train_n_test_cara2[tt,rr] = Target.flatten()[((tt-rr) - 1)]
 else:
 s_tr_train_n_test_cara2[tt,rr] = 0.
 # print("\n s_tr_train_n_test_cara2: \n",s_tr_train_n_test_cara2)

 # s_tr_train = np.zeros([Ntrain,r])
 # s_tr_test = np.zeros([Ntest,r])
 s_tr_train = s_tr_train_n_test_cara2[:Ntrain,:]
 s_tr_test = s_tr_train_n_test_cara2[Ntrain:,:]
 return s_tr_train, s_tr_test

def get_MAPE_0_100(aktual,predict):
 banyak_fitur_target = aktual.shape[1]
 # Rü-
 mus MAPE yang digunakan, jika data aktualnya ada yang nol atau tidak ada yg nol
 # dan untuk memastikan MAPE pada interval = [0%;100%] --> dengan kondisi ini
 # Ref: (Berretti,Thampi,danSrivastava,2015) dalam
 # Hapsari,KD.,Cholissodin,I.,Santoso,E.,2016
 konstanta_smoothing = 0.00000001
 c = konstanta_smoothing
 mape_init = np.abs(((aktual+c) - (predict+c)) / (aktual+c))*100
 mape_norm = np.sum(np.where(mape_init>100, 100, mape_init))/(len(predict)*banyak_fitur_target)
 return mape_norm

convert df to array np
arr_df_idr_us = df_idr_us.iloc[:,2:].values
print(arr_df_idr_us)
print(arr_df_idr_us.shape)

import numpy as np

set nilai parameter RNN native
inisialisasi jumlah fitur (n) yang digunakan sebanyak 3,
sedangkan jumlah hidden neuron (m) sebanyak 2,
```

```
dan jumlah context neuron (r) sebanyak 2.
Sedangkan datasets dengan jumlah 15 data dibagi menjadi 2 bagian,
untuk data training sebanyak 10 data dan data testing sebanyak 5 data.

n = 3
m = 2
r = 2
byk_data = arr_df_idr_us.shape[0]
Ntrain = 10
Ntest = byk_data - Ntrain
print(Ntest)

Normalisasi data
get_min = np.min(arr_df_idr_us)
get_max = np.max(arr_df_idr_us)
lower_boundary,upper_boundary = 0.1,0.9
data_norm = (((arr_df_idr_us-get_min)/(get_max-get_min))*(upper_boundary-
lower_boundary))+lower_boundary
print('data norm: \n',data_norm)

Xtrain_feature,Xtrain = data_norm[:Ntrain,:-1],data_norm[:Ntrain,:]
print('\n Xtrain feature: \n',Xtrain_feature)
print('\n Xtrain: \n',Xtrain)

Xtest_feature,Xtest = data_norm[Ntrain:,:-1],data_norm[Ntrain,:]

Ytrain, Ytest = data_norm[:Ntrain,-1], data_norm[Ntrain:,-1]
Ytrain,Ytest = Ytrain[...,None],Ytest[...,None]
print('\n Ytrain: \n',Ytrain)
print('\n Ytest: \n',Ytest)

Ytrain_non_norm, Ytest_non_norm = arr_df_idr_us[:Ntrain,-
1], arr_df_idr_us[Ntrain:,-1]
Ytrain_non_norm,Ytest_non_norm = Ytrain_non_norm[...,None],Ytest_non_norm[...,None]

proses training

#
1. inisialisasi matriks delay (s)
dengan persamaan s_tr = T[t-(n+r)+n] atau s_tr = T[t-r - 1], jika indeks dimulai dari 0
di mana t menyatakan urutan data dan r menyatakan urutan context neuron
#
Target = data_norm[:, -1] # ukurannya (15,)
Target = Target[...,None] # agar matriknya menjadi ukuran (15,1)
print(Target)
print(Target.shape)

isTimeSeriesData=True # dapat diisikan True/False
cara=2 # dapat disikan 1 atau 2
s_tr_train,s_tr_test = create_matrix_delay(isTimeSe-
riesData, cara, r, Ntrain, Ntest, Ytrain, Ytest, Target)
print("\n s_tr_train: \n", s_tr_train)
print()
print("\n s_tr_test : \n", s_tr_test)

2.1 Random nilai matrik W (bobot input)
W = myrandfloat(m,n+r,lower=-1,upper=1)
print("\n W : \n", W)

2.2 Random nilai matrik bias
bias = myrandfloat(1,m,lower=0,upper=1)
print("\n bias : \n", bias)

#
3. Menghitung nilai Htrain, Hplus
#
3.1 Menggabungkan data latih dengan matriks delay
Xtrain_merge_s = np.hstack((Xtrain_feature,s_tr_train))

is_singular_matrix = True
while(is_singular_matrix):
 # 2.1 Random nilai matrik W (bobot input dgn ordo banyak_hidden_neu-
 ron(m), banyak_fitur(n)+recurrent(r))
 # bobot = np.random.rand
 W = myrandfloat(m,n+r,lower=-1,upper=1)
 bobot_input = W
 # print("\n W : \n", W)
```

```
2.2 Random nilai matrik bias
bias = myrandfloat(1,m,lower=0,upper=1).flatten()
print("\n bias : \n", bias)

hitung matrik h train dengan fungsi aktivasi jenis sigmoid
h_train = fn.sigmoid(np.dot(Xtrain_merge_s, np.transpose(bobot_input)) + bias*np.ones(Ntrain,1))

print('\n h_train cara1: \n',h_train)
print('\n h_train: \n',h_train)

time.sleep(100)

cek matrik singular
cek_matrik = np.dot(np.transpose(h_train), h_train)
det_cek_matrik = np.linalg.det(cek_matrik)
if det_cek_matrik != 0:
 #proceed

 #if np.linalg.cond(cek_matrik) < 1/sys.float_info.epsilon:
 # i = np.linalg.inv(cek_matrik)
 # is_singular_matrix = False
 else:
 is_singular_matrix = True

h_plus = np.dot(np.linalg.inv(cek_matrik), np.transpose(h_train))

print("\n h_plus: \n", h_plus)

4. Beta_topi
output weight disebut juga sebagai Beta_topi atau bobot_output
Beta_topi = np.dot(h_plus, Ytrain)
bobot_output = Beta_topi
print("\n Beta_topi: \n", Beta_topi)

Coba testing dengan data Xtrain
h_test = 1 / \
(1 + np.exp(-(np.dot(data_testing, np.transpose(bobot_input)) + bias)))
h_test = fn.sigmoid(np.dot(Xtrain_merge_s, np.transpose(bobot_input)) + bias*np.ones(Ntrain,1))
Ytrain_predict = np.dot(h_test, bobot_output)
Ytrain_predict_denorm = (Ytrain_predict * (get_max - get_min) + get_min)
print("\n Ytrain_predict_denorm: \n", Ytrain_predict_denorm)

proses testing
#
1. load Xtest, bobot_input, bias dan bobot_output
#
2. get matrik delay data test (s_tr_test)
print("\n s_tr_test: \n", s_tr_test)

3. hitung matrik h test dengan fungsi aktivasi jenis sigmoid
Xtest_merge_s = np.hstack((Xtest_feature,s_tr_test))
h_test = fn.sigmoid(np.dot(Xtest_merge_s, np.transpose(bobot_input)) + bias*np.ones(Ntest,1))
print('\n h_test: \n',h_test)

4. Coba testing dengan data Xtest
Ytest_predict = np.dot(h_test, bobot_output)
Ytest_predict_denorm = (Ytest_predict * (get_max - get_min) + get_min)
print("\n Ytest_predict: \n", Ytest_predict_denorm)

Hitung MAPE
aktual = Ytest_non_norm # non_norm atau denorm bermakna sama
predict = Ytest_predict_denorm
mape = get_MAPE_0_100(aktual,predict)
print('\n MAPE dgn interval [0% ; 100%] = ', mape.round(2),'%')
```

## Output:

```
Downloading...
From:
https://drive.google.com/uc?id=1nYH0cuNTkSkqjzFIda5gdYOPTzb3hK
To: /content/data_nilai_tukar_idr_us.xlsx
100% 10.4k/10.4k [00:00<00:00, 21.8MB/s]

Data ke-i Tanggal Fitur 1 Fitur 2 Fitur 3 Nilai Tukar
0 1 24 Feb. '17 13360.00 13342.00 13311.00 13340.00
1 2 25 Feb. '17 13342.00 13311.00 13340.00 13330.00
2 3 26 Feb. '17 13311.00 13340.00 13330.00 13330.00
3 4 27 Feb. '17 13340.00 13330.00 13330.00 13355.00
4 5 28 Feb. '17 13330.00 13330.00 13355.00 13346.00
5 6 1 Mar. '17 13330.00 13355.00 13346.00 13340.00
6 7 2 Mar. '17 13355.00 13346.00 13340.00 13383.00
7 8 3 Mar. '17 13346.00 13340.00 13383.00 13373.00
8 9 4 Mar. '17 13340.00 13383.00 13373.00 13323.00
9 10 5 Mar. '17 13383.00 13373.00 13323.00 13323.00
10 11 6 Mar. '17 13373.00 13323.00 13323.00 13322.75
11 12 7 Mar. '17 13323.00 13322.75 13322.75 13337.75
12 13 8 Mar. '17 13323.00 13337.75 13407.00 13407.00
13 14 9 Mar. '17 13322.75 13337.75 13407.00 13394.00
14 15 10 Mar. '17 13337.75 13407.00 13394.00 13363.00

[[13360. 13342. 13311. 13340.]
 [13342. 13311. 13340. 13330.]
 [13311. 13340. 13330. 13330.]
 [13340. 13330. 13330. 13355.]
 [13330. 13330. 13355. 13346.]
 [13330. 13355. 13346. 13340.]
 [13355. 13346. 13340. 13383.]
 [13346. 13340. 13383. 13373.]
 [13340. 13383. 13373. 13323.]
 [13383. 13373. 13323. 13323.]
 [13373. 13323. 13323. 13322.75]
 [13323. 13323. 13322.75 13337.75]
 [13323. 13322.75 13337.75 13407.]
 [13322.75 13337.75 13407. 13394.]
 [13337.75 13407. 13394. 13363.]]
(15, 4)

data norm:
[[0.50833333 0.35833333 0.1 0.34166667]
 [0.35833333 0.1 0.34166667 0.25833333]
 [0.1 0.34166667 0.25833333 0.25833333]
 [0.34166667 0.25833333 0.25833333 0.46666667]
 [0.25833333 0.25833333 0.46666667 0.39166667]
 [0.25833333 0.46666667 0.39166667 0.34166667]
 [0.46666667 0.39166667 0.34166667 0.7]
 [0.39166667 0.34166667 0.7 0.61666667]
 [0.34166667 0.7 0.61666667 0.2]
 [0.7 0.61666667 0.2 0.2]
 [0.61666667 0.2 0.2 0.19791667]
 [0.2 0.2 0.19791667 0.32291667]
 [0.2 0.19791667 0.32291667 0.9]
 [0.19791667 0.32291667 0.9 0.79166667]
 [0.32291667 0.9 0.79166667 0.53333333]]

s_tr_train:
[[0. 0.]
 [0.34166667 0.]
 [0.25833333 0.34166667]
 [0.25833333 0.25833333]
 [0.46666667 0.25833333]
 [0.39166667 0.46666667]
 [0.34166667 0.39166667]
 [0.7 0.34166667]
 [0.61666667 0.7]
 [0.2 0.61666667]]

h train:
```

```
[[0.46723374 0.6957836]
[0.54249669 0.73176439]
[0.5421593 0.67888335]
[0.49883759 0.72217405]
[0.52863213 0.7415195]
[0.48744992 0.73316695]
[0.44845568 0.75899815]
[0.49208466 0.79783589]
[0.435771 0.78450666]
[0.34927924 0.77854586]]

h_plus:
[[0.40596132 1.4392629 2.10519083 0.69636217 1.0405285 0.33095099
-0.77007776 -0.39978102 -1.34581154 -2.98338932]
[-0.13468952 -0.79162618 -1.22877136 -0.31636907 -0.5338452 -0.0797598
0.63185289 0.40112816 1.00613166 2.05649007]]

Beta_topi:
[[0.24851202]
[0.35024925]]

Ytrain_predict_denorm:
[[13345.54184522]
[13348.54722054]
[13346.76110306]
[13347.18317392]
[13348.54445681]
[13347.28112071]
[13347.21937538]
[13349.56611618]
[13347.77445134]
[13345.51057883]]

s_tr_test:
[[0.2 0.2]
[0.19791667 0.2]
[0.32291667 0.19791667]
[0.9 0.32291667]
[0.79166667 0.9]]

h_test:
[[0.44039833 0.7586745]
[0.54082157 0.68446607]
[0.54693951 0.70633886]
[0.55147522 0.79656391]
[0.41272701 0.80823568]]

Ytest_predict:
[[13347.01626782]
[13346.9169015]
[13347.79830758]
[13350.94023686]
[13348.02255021]]]

MAPE dgn interval [0% ; 100%] = 0.23 %
```

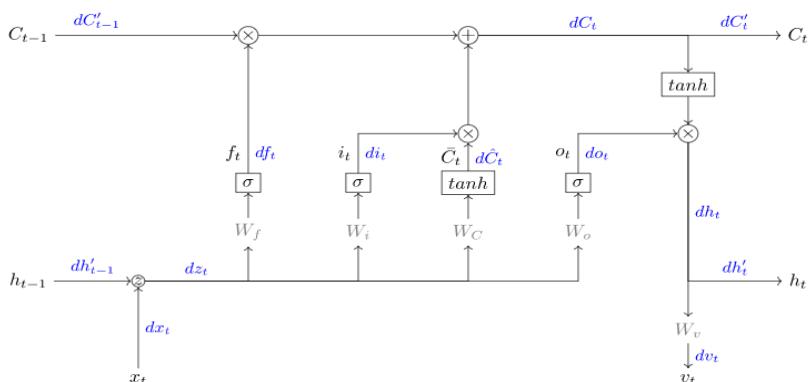
Link kode program lengkapnya:

[https://colab.research.google.com/drive/1Lwy\\_EahfZISvfBKT3RhDg9\\_iIPnA76Ve?usp=sharing](https://colab.research.google.com/drive/1Lwy_EahfZISvfBKT3RhDg9_iIPnA76Ve?usp=sharing)



## 18.4 Struktur RNN Berbasis Modified Long Short-Term Memory (MLSTM) Menggunakan ELM

Selain arsitektur sel LSTM (LSTM cell) yang juga merupakan bagian dari RNN. Terdapat juga Gated Recurrent Unit (GRU) yang juga merupakan variasi dari sel LSTM yang menggunakan mekanisme gate dan parameter yang lebih sedikit, sehingga pada prosesnya lebih sedikit membutuhkan memori dan proses eksekusi lebih cepat daripada LSTM, tetapi dari segi hasil LSTM lebih akurat dengan proses eksekusi yang tidak lebih cepat dari pada GRU (Lendave, V. 2021). Pada LSTM terdapat 3 gates, yaitu Input Gate, Forget Gate, dan Output Gate, tetapi dari penelitian (Jozefowicz, et al. 2015) di dalam LSTM tidak mempertimbangkan variabel bias yang membuat kinerjanya kurang optimal. Pada GRU terdapat 2 gates, yaitu Update gate dan Reset gate.



Bentuk operasi dari proses forward:

$$\begin{aligned} z &= [h_{t-1}, x_t] \\ f_t &= \sigma(W_f \cdot z + b_f) \\ .. \end{aligned}$$

$$\hat{y}_t = softmax(v_t)$$

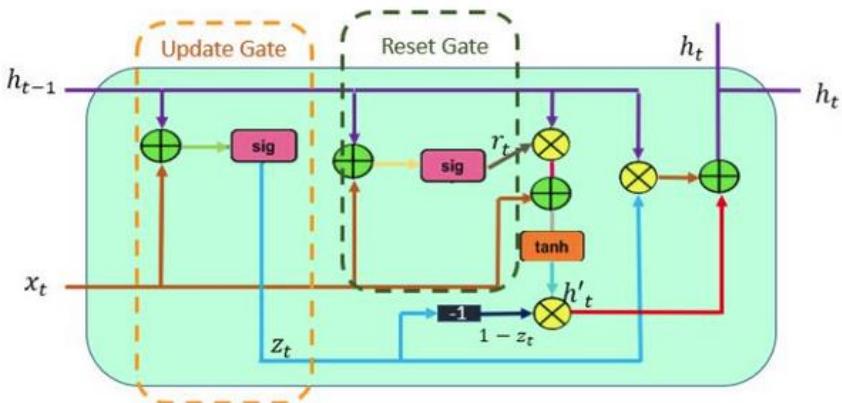
```
forward dengan python
z = np.row_stack((h_prev, x))
f = sigmoid(np.dot(p.W_f.v, z) + p.b_f.v)
i = sigmoid(np.dot(p.W_i.v, z) + p.b_i.v)
C_bar = tanh(np.dot(p.W_C.v, z) + p.b_C.v)

C = f * C_prev + i * C_bar
o = sigmoid(np.dot(p.W_o.v, z) + p.b_o.v)
h = o * tanh(C)

v = np.dot(p.W_v.v, h) + p.b_v.v
y = np.exp(v) / np.sum(np.exp(v)) #softmax
```

Gambar 18.4 Alur kerja dan Snippet kode program di proses *forward* Algoritma Long Short-Term Memory (LSTM)

Sumber: (Jayasiri, V. 2017)



Gambar 18.5 Alur kerja Gated Recurrent Unit (GRU)

Sumber: (Lendave, V. 2021)

Berdasarkan permasalahan bias, kemudian juga terkait masalah harus menggunakan banyak parameter serta skema gate yang juga sangat banyak pilihan bentuk kemungkinan arsitekturnya (yang seharusnya dapat digunakan algoritma semacam Particle Swarm Optimization (PSO) atau Algoritma Genetika untuk mencari arsitektur yang terbaik dari nilai evaluasi yang optimal) dan terlalu banyak proses yang seharusnya dapat diringkas dan disederhanakan, dan lama waktu proses training baik pada LSTM konvensional yang termasuk di dalamnya juga ada GRU, maka diusulkan “Algoritme Modified Long Short-Term Memory (MLSTM)” menggunakan ELM yang hanya dengan satu iterasi saat proses training, dapat mengatasi hal berikut:

- Tidak adanya variabel bias pada LSTM dan GRU konvensional yang membuat nilai hasilnya kurang optimal, di mana bias ini berperan sebagai nilai *soft augmented* atau nilai konstanta penambah untuk menggeser secara teliti dan dinamis sesuai nilai random yang didapatkan untuk lebih mempermudah bobot dari input maupun output ELM dalam mencapai target yang tepat.
- Simplified* penggunaan banyak gate seperti pada LSTM maupun GRU, dimodifikasi (*modified*) menjadi cukup dengan mono gate (karena susunan arsitekturnya masih banyak kemungkinan yang belum dicoba, sehingga dinilai tidak optimal), yaitu dengan selain membuat sekuens sebanyak  $n$  fitur, juga ada sekuens sebanyak 1

fitur, 2 fitur, ..., n-1 fitur, sampai n fitur dengan konfigurasi sederhana untuk mengatasi permasalahan RRN Native terkait “Long-Term Dependencies” dan sekaligus LSTM konvensional yaitu menanamkan kunci sukses proses *training* untuk proses *testing* dengan memastikan setiap pola fitur yang dikombinasi dengan fitur recurrent (seperti penambahan nilai fitur dengan konstanta, misal dengan nilai [-1 -1 .. -1] baik pada “Pattern Feature atau Fitur Pola” maupun “Recurrent Feature atau fitur Recurrent” yang nanti ketika sudah dikonversi menggunakan teknik one hot encode setiap nilai -1 akan menjadi [0 0 .. 0] sepanjang token yang ada, jika datanya berupa teks), harus ada pada proses pelatihan (*training*). Kemudian juga diberikan proses untuk menjalankan metode `create_matrix_delay(s)` yang didalamnya argumennya mempertimbangkan “`isTimeSeriesData = False`, dapat diisikan juga dengan nilai `True`”. Karena ada beberapa konfigurasi tersebut, maka algoritma ini disebut dengan MLSTM. Berikut ini merupakan contoh fitur pola, fitur recurrent dan juga target sebagai bagian persiapan proses training MLSTM, dari data teks, misal diketahui ( misal dengan  $n = 3$ ,  $r = 2$ , dan  $m = \text{int}(\text{np.ceil}((2/3)*( (n+r)*\text{len\_unik\_term} )))$  ),

- **semua\_token\_dataInput** (data teks “KataBijak\_new.txt”):  
['tujuan' 'hidup' 'kita' 'adalah' 'menjadi' 'bahagia' 'new\_line'  
'kesabaran' 'adalah' 'obat' 'terbaik' 'untuk' 'segala' 'kesulitan'  
'new\_line' 'siapapun' 'bisa' 'jadi' 'apapun']  
**len\_sekuens\_semua\_token\_dataInput:** 19  
**integer\_encoding\_semua\_token\_dataInput:**  
[15 4 8 0 9 2 10 6 0 11 14 16 12 7 10 13 3 5 1]
- **get\_unik\_term\_terurut\_alphabet:**  
['adalah' 'apapun' 'bahagia' 'bisa' 'hidup' 'jadi' 'kesabaran'  
'kesulitan' 'kita' 'menjadi' 'new\_line' 'obat' 'segala' 'siapapun'  
'terbaik' 'tujuan' 'untuk']  
**len\_unik\_term:** 17  
**integer\_encoding (dari get\_unik\_term\_terurut\_alphabet):**  
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16]

| <b>Pattern Feature<br/>(Fitur Pola)</b> | <b>Recurrent Feature<br/>(Fitur Recurrent)</b> | <b>Target</b>     |
|-----------------------------------------|------------------------------------------------|-------------------|
| Xtrain_feature:<br>[-1. -1. 15.]        | s_tr_train:<br>array([[-1., -1.],              | Ytrain:<br>[[ 4.] |
| [-1. -1. 4.]                            | [ 4., -1.],                                    | [ 8.]             |
| [-1. -1. 8.]                            | [ 8., 4.],                                     | [ 0.]             |
| [-1. -1. 0.]                            | [ 0., 8.],                                     | [ 9.]             |
| [-1. -1. 9.]                            | [ 9., 0.],                                     | [ 2.]             |
| [-1. -1. 2.]                            | [ 2., 9.],                                     | [10.]             |
| [-1. -1. 10.]                           | [10., 2.],                                     | [ 6.]             |
| [-1. -1. 6.]                            | [ 6., 10.],                                    | [ 0.]             |
| [-1. -1. 0.]                            | [ 0., 6.],                                     | [11.]             |
| [-1. -1. 11.]                           | [11., 0.],                                     | [14.]             |
| [-1. -1. 14.]                           | [14., 11.],                                    | [16.]             |
| [-1. -1. 16.]                           | [16., 14.],                                    | [12.]             |
| [-1. -1. 12.]                           | [12., 16.],                                    | [ 7.]             |
| [-1. -1. 7.]                            | [ 7., 12.],                                    | [10.]             |
| [-1. -1. 10.]                           | [10., 7.],                                     | [13.]             |
| [-1. -1. 13.]                           | [13., 10.],                                    | [ 3.]             |
| [-1. -1. 3.]                            | [ 3., 13.],                                    | [ 5.]             |
| [-1. -1. 5.]                            | [ 5., 3.],                                     | [ 1.]             |
| [-1. 15. 4.]                            | [ 1., 5.],                                     | [ 8.]             |
| [-1. 4. 8.]                             | [ 8., 1.],                                     | [ 0.]             |
| [-1. 8. 0.]                             | [ 0., 8.],                                     | [ 9.]             |
| [-1. 0. 9.]                             | [ 9., 0.],                                     | [ 2.]             |
| [-1. 9. 2.]                             | [ 2., 9.],                                     | [10.]             |
| [-1. 2. 10.]                            | [10., 2.],                                     | [ 6.]             |
| [-1. 10. 6.]                            | [ 6., 10.],                                    | [ 0.]             |
| [-1. 6. 0.]                             | [ 0., 6.],                                     | [11.]             |
| [-1. 0. 11.]                            | [11., 0.],                                     | [14.]             |
| [-1. 11. 14.]                           | [14., 11.],                                    | [16.]             |
| [-1. 14. 16.]                           | [16., 14.],                                    | [12.]             |
| [-1. 16. 12.]                           | [12., 16.],                                    | [ 7.]             |
| [-1. 12. 7.]                            | [ 7., 12.],                                    | [10.]             |
| [-1. 7. 10.]                            | [10., 7.],                                     | [13.]             |
| [-1. 10. 13.]                           | [13., 10.],                                    | [ 3.]             |
| [-1. 13. 3.]                            | [ 3., 13.],                                    | [ 5.]             |
| [-1. 3. 5.]                             | [ 5., 3.],                                     | [ 1.]             |
| [15. 4. 8.]                             | [ 1., 5.],                                     | [ 0.]             |
| [ 4. 8. 0.]                             | [ 0., 1.],                                     | [ 9.]             |
| [ 8. 0. 9.]                             | [ 9., 0.],                                     | [ 2.]             |

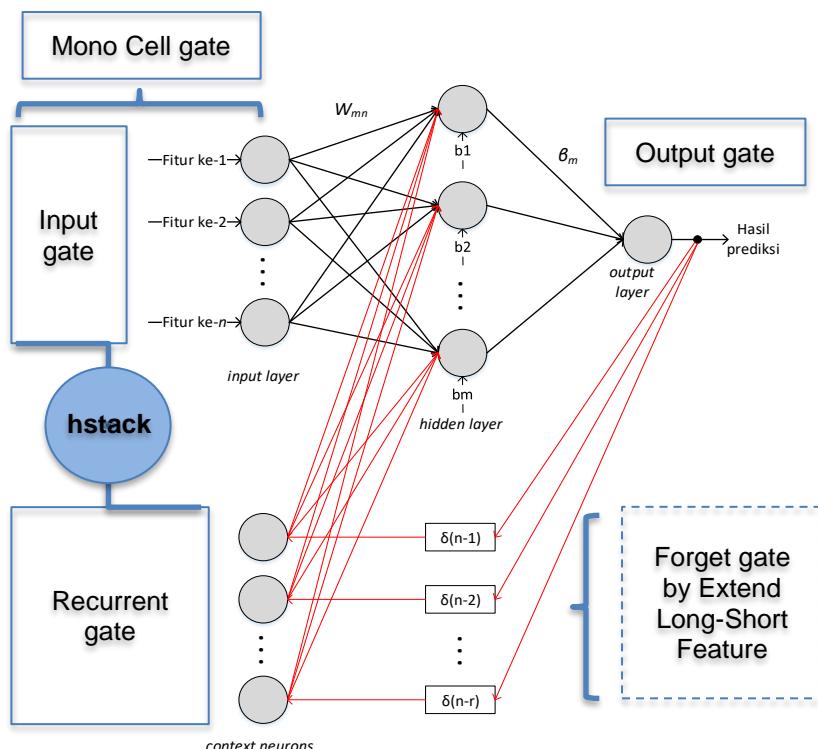
|               |             |        |
|---------------|-------------|--------|
| [ 0. 9. 2.]   | [ 2., 9.],  | [10.]  |
| [ 9. 2. 10.]  | [10., 2.],  | [ 6.]  |
| [ 2. 10. 6.]  | [ 6., 10.], | [ 0.]  |
| [10. 6. 0.]   | [ 0., 6.],  | [11.]  |
| [ 6. 0. 11.]  | [11., 0.],  | [14.]  |
| [ 0. 11. 14.] | [14., 11.], | [16.]  |
| [11. 14. 16.] | [16., 14.], | [12.]  |
| [14. 16. 12.] | [12., 16.], | [ 7.]  |
| [16. 12. 7.]  | [ 7., 12.], | [10.]  |
| [12. 7. 10.]  | [10., 7.],  | [13.]  |
| [ 7. 10. 13.] | [13., 10.], | [ 3.]  |
| [10. 13. 3.]  | [ 3., 13.], | [ 5.]  |
| [13. 3. 5.])  | [ 5., 3.])] | [ 1.]) |

- c. Basis penggunaan ELM yang mengadopsi sebagian proses pada RNNELM yang dikembangkan untuk MLSTM untuk penyederhaan dari proses pelatihan LSTM yang cukup lama sebanding dengan proses pelatihan algoritma Artificial Neural Network (ANN) jenis Backpropagation, yang ternyata dapat disederhanakan dengan ELM.
- d. Pada proses testing, MLSTM pada proses awal digunakan untuk fokus pada penentuan fitur recurrent sebanyak ( $r$ ), kemudian setelah proses tersebut terpenuhi (yaitu untuk “pola fitur” maupun “recurrent fitur” sudah mengandung nilai integer “-1”) maka dengan model MLSTM yang sama, dapat digunakan untuk proses prediksi sepanjang apapun token yang dibutuhkan.

Berikut adalah bentuk penulisan potongan kode program secara singkat dari MLSTM:

- $W = \text{myrandfloat}(m, (n+r)*\text{len\_unik\_term}, \text{lower}=-1, \text{upper}=1)$
- bias ( $b$ ) =  $\text{myrandfloat}(1, m, \text{lower}=0, \text{upper}=1).\text{flatten}()$
- $h_{\text{train}} = \text{fn\_sigmoid}(\text{np.dot}(\text{full\_conn\_onehot\_Xtrain\_merge\_s}, \text{np.transpose(bobot\_input)}) + \text{bias} * \text{np.ones}((Ntrain, 1)))$
- $h_{\text{plus}} = \text{np.dot}(\text{np.linalg.inv}(cek\_matrik), \text{np.transpose}(h_{\text{train}}))$
- $\text{Beta\_topi} = \text{np.dot}(h_{\text{plus}}, Ytrain\_onehot\_encode)$
- $h_{\text{test}} = \text{fn\_sigmoid}(\text{np.dot}(\text{full\_conn\_onehot\_Xtest\_merge\_s}, \text{np.transpose}(W)) + \text{bias} * \text{np.ones}((Ntest, 1)))$
- $Ytest\_predict = \text{np.dot}(h_{\text{test}}, \text{Beta\_topi})$
- $\text{onehot\_binary\_arr\_Ytest\_predict}, \text{lokasi\_maks\_as\_int\_encode} = \text{y\_predict\_to\_onehot\_binary}(Ytest\_predict)$
- $\text{arr\_token\_Ytest\_predict} = \text{onehot\_binary\_to\_arr\_token}(onehot\_binary\_arr\_Ytest\_predict, \text{get\_unik\_term\_terurut\_alphabet})$

## 18.5 Studi Kasus: RNN berbasis Simplified/ Modified LSTM (MLSTM) menggunakan ELM pada prediksi data Teks untuk Pembuatan Karya Sastra (dengan Extend Long-Short Feature pada Mono Cell Gate)



Gambar 18.6 Jaringan Algoritme Modified Long Short-Term Memory (MLSTM)

Langkah manualisasi pada algoritma Modified Long Short-Term Memory (MLSTM) Berbasis ELM, hampir sama dengan RNNELM, yang membedakan hanya pada pre prosesing atau proses awalnya, dan adanya pemberian label kelompok matrik delay sebagai mono gate yang di dalamnya terdapat hasil proses “Extend Long-Short Feature pada fitur pola maupun pada fitur Recurrent” tetapi masih dalam satu

varibel matrik delay (s). Berikut beberapa pembagian Extend Long-Short Feature-nya:

a. Pada Proses Training

- Pada fitur pola (PF), jika banyak fitur ( $n$ ) = 3, maka “fitur pola” berisi array seperti pada tabel sebelumnya pada kolom “Pattern Feature (Fitur Pola)”, yaitu dengan selain membuat sekuens sebanyak  $n$  fitur, juga ada sekuens sebanyak 1 fitur, 2 fitur, ...,  $n-1$  fitur, sampai  $n$  fitur, dengan ketentuan
  - Untuk 1 fitur, salah satu contoh interger encode [15.] menjadi array [-1. -1. 15.], di mana “-1” dalam array tersebut yaitu sepanjang nilai  $n-1$ .
  - Untuk 2 fitur, salah satu contoh interger encode [15. 4.] menjadi array [-1. 15. 4.], di mana “-1” dalam array tersebut yaitu sepanjang nilai  $n-2$ , sampai, ..
  - Untuk  $n$  fitur, salah satu contoh interger encode [15. 4. 8.] tetap menjadi array [15. 4. 8.], di mana “-1” dalam array tersebut yaitu sepanjang nilai  $n-n$  atau tidak ada sama sekali.
- Pada fitur recurrent (RF), jika banyak recurrent ( $r$ ) = 2, maka berikut susunan lengkap dari hasil hstack PF dengan RF, dan semua susunan ini harus ada:
  - Dari setiap anggota PF atau tiap baris array-nya akan dilakukan hstack dengan array [-1., -1.], jika  $r = 5$ , maka dengan array [-1., -1., -1., -1., -1.] atau sepanjang dari nilai  $r$ -nya.
  - Dari setiap anggota PF atau tiap baris array-nya akan dilakukan hstack dengan array dengan ketentuan berikut:

```
def create_matrix_delay_roll_mono_gate(r, Ntrain, Ntest, Ytrain, Ytest, Target):
 # jika para create_matrix_delay_roll, maka pasti isTimeSeriesData == False
 s_tr_train = np.zeros([Ntrain,r])
 s_tr_test = np.zeros([Ntest,r])
 # diset -1, agar nanti ketika di fn_onehot_encoding_multi_d nilai fiturnya menjadi nol semua,
 for i in range(r):
 s_tr_train[:,i,None] = np.roll(Ytrain,(i+1))
 s_tr_train_final = np.vstack((~np.ones([Ntrain,r]),s_tr_train))
 for i in range(r):
 s_tr_test[:,i,None] = np.roll(Ytest,(i+1))
 s_tr_test_final = np.vstack((~np.ones([Ntest,r]),s_tr_test))
 return s_tr_train_final, s_tr_test_final
```

b. Pada Proses Testing (jika ingin memprediksi sampai  $i_k$  sekuens, misal  $i_k = 7$ )

- Pada fitur pola (PF), jika banyak fitur ( $n$ ) = 3, maka “fitur pola” berisi data test.
  - Jika data test terdiri dari 1 fitur, misal berupa interger encode [15.], maka akan dikonversi menjadi array [-1. -1. 15.], di mana “-1” dalam array tersebut yaitu sepanjang nilai  $n-1$ .

- Jika data test terdiri dari 2 fitur, misal berupa interger encode [15. 4.], maka akan dikonversi menjadi array [-1. 15. 4.], di mana “-1” dalam array tersebut yaitu sepanjang nilai n-2.
- Jika data test terdiri dari  $\geq n$  fitur, misal berupa interger encode [15. 4. 8. 0. 9.], maka diambil n fitur terakhir yaitu menjadi array [8. 0. 9.], di mana “-1” dalam array tersebut tidak ada sama sekali.
- Pada fitur recurrent (RF), jika banyak recurrent ( $r$ ) = 2, maka berikut ketentuan dari hasil hstack PF dengan RF:
  - Jika data test terdiri dari 1 fitur, misal dari array [-1. -1. 15.] yang juga merupakan fitur pola (PF) dilakukan hstack dengan array [-1., -1.], sepanjang dari nilai  $r$ -nya. Misal hasil ketika ***i\_k = 0***, dari variabel pada sebagian kode program python yaitu “lokasi\_maks\_as\_int\_encode (int\_encode)” berdasar bagian kode program

```
onehot_binary_arr_Ytest_predict, lokasi_maks_as_int_encode = \
y_predict_to_onehot_binary(Ytest_predict_Xtest_by_token)
```

Misal didapat int\_encode = 4., maka array PF di-update jadi [-1. 15. 4.] dan dilakukan hstack dengan array [-1., -1.].

Misal hasil ketika ***i\_k=1***, Misal didapat int\_encode = 8., maka array PF di-update jadi [15. 4. 8.] dan dilakukan hstack dengan array [-1., -1.].

Misal hasil ketika ***i\_k=2***, Misal didapat int\_encode = 0., maka array PF di-update jadi [4. 8. 0.] dan dilakukan hstack dengan array [-1., -1.].

Misal hasil ketika ***i\_k=3***, Misal didapat int\_encode = 9., maka array PF di-update jadi [8. 0. 9.] dan dilakukan hstack dengan array [9. 0.], tidak lagi menggunakan array [-1., -1.], karena di dalam PF yang sudah tidak mengandung nilai elemen “-1” sudah menghasilkan int\_encode sebanyak  $r$ , yaitu [9. 0.] dari proses prediksinya sebelumnya.

Misal hasil ketika ***i\_k=4***, Misal didapat int\_encode = 2., maka array PF di-update jadi [0. 9. 2.] dan dilakukan hstack dengan array [2. 9.], tidak lagi menggunakan array [-1., -1.], karena di dalam PF yang sudah tidak mengandung nilai elemen “-1” sudah menghasilkan int\_encode sebanyak  $r$ , yaitu [2. 9.] dari proses prediksinya sebelumnya.

Misal hasil ketika ***i\_k = 5***, Misal didapat int\_encode = 10., maka array PF di-update jadi [9. 2. 10.] dan dilakukan hstack dengan array [10. 2.], dari proses prediksinya sebelumnya.

Misal hasil ketika ***i\_k = 6***, Misal didapat int\_encode = 6., maka array PF di-update jadi [2. 10. 6.] dan dilakukan hstack dengan array [6. 10.], dari proses prediksinya sebelumnya.

**Stop**, karena *i\_k = 7*

- Jika data test terdiri dari 2 fitur, misal dari array [-1. 15. 4.] yang juga merupakan fitur pola (PF) dilakukan hstack dengan array [-1., -1.], sepanjang dari nilai *r*-nya. Misal hasil ketika ***i\_k = 0***, dari variabel pada sebagian kode program python yaitu “lokasi\_maks\_as\_int\_encode (int\_encode)”,

Misal didapat int\_encode = 8., maka array PF di-update jadi [15. 4. 8.] dan dilakukan hstack dengan array [-1., -1.].

Misal hasil ketika ***i\_k = 1***, Misal didapat int\_encode = 0., maka array PF di-update jadi [4. 8. 0.] dan dilakukan hstack dengan array [-1., -1.].

Misal hasil ketika ***i\_k = 2***, Misal didapat int\_encode = 9., maka array PF di-update jadi [8. 0. 9.] dan dilakukan hstack dengan array [9. 0.], tidak lagi menggunakan array [-1., -1.], karena di dalam PF yang sudah tidak mengandung nilai elemen “-1” sudah menghasilkan int\_encode sebanyak *r*, yaitu [9. 0.] dari proses prediksinya sebelumnya.

Misal hasil ketika ***i\_k = 3***, Misal didapat int\_encode = 2., maka array PF di-update jadi [0. 9. 2.] dan dilakukan hstack dengan array [2. 9.], dari proses prediksinya sebelumnya.

Misal hasil ketika ***i\_k = 4***, Misal didapat int\_encode = 10., maka array PF di-update jadi [9. 2. 10.] dan dilakukan hstack dengan array [10. 2.], dari proses prediksinya sebelumnya.

Misal hasil ketika ***i\_k = 5***, Misal didapat int\_encode = 6., maka array PF di-update jadi [2. 10. 6.] dan dilakukan hstack dengan array [6. 10.], dari proses prediksinya sebelumnya.

Misal hasil ketika ***i\_k = 6***, Misal didapat int\_encode = 0., maka array PF di-update jadi [10. 6. 0.] dan dilakukan hstack dengan array [0. 6.], dari proses prediksinya sebelumnya.

**Stop**, karena ***i\_k = 7***

- o Jika data test terdiri dari  $\geq n$  fitur, misal diketahui dari array [15. 4. 8. 0. 9.], maka diambil  $n$  fitur terakhir yaitu menjadi array [8. 0. 9.] yang juga merupakan fitur pola (PF) dilakukan hstack dengan array [-1., -1.], sepanjang dari nilai  $r$ -nya.

Misal hasil ketika ***i\_k = 0***, Misal didapat int\_encode = 2., maka array PF di-update jadi [0. 9. 2.] dan dilakukan hstack dengan array [-1., -1.].

Misal hasil ketika ***i\_k = 1***, Misal didapat int\_encode = 10., maka array PF di-update jadi [9. 2. 10.] dan dilakukan hstack dengan array [10. 2.], tidak lagi menggunakan array [-1., -1.], karena di dalam PF yang sudah tidak mengandung nilai elemen “-1” sudah menghasilkan int\_encode sebanyak  $r$ , yaitu [10. 2.] dari proses prediksinya sebelumnya.

Misal hasil ketika ***i\_k = 2***, Misal didapat int\_encode = 6., maka array PF di-update jadi [2. 10. 6.] dan dilakukan hstack dengan array [6. 10.], dari proses prediksinya sebelumnya.

Misal hasil ketika ***i\_k = 3***, Misal didapat int\_encode = 0., maka array PF di-update jadi [10. 6. 0.] dan dilakukan hstack dengan array [0. 6.], dari proses prediksinya sebelumnya.

Misal hasil ketika ***i\_k = 4***, Misal didapat int\_encode = 11., maka array PF di-update jadi [6. 0. 11.] dan dilakukan hstack dengan array [11. 0.], dari proses prediksinya sebelumnya.

Misal hasil ketika ***i\_k = 5***, Misal didapat int\_encode = 14., maka array PF di-update jadi [0. 11. 14.] dan dilakukan hstack dengan array [14. 11.], dari proses prediksinya sebelumnya.

Misal hasil ketika ***i\_k = 6***, Misal didapat int\_encode = 16., maka array PF di-update jadi [11. 14. 16.] dan dilakukan hstack dengan array [16. 14.], dari proses prediksinya sebelumnya.

**Stop**, karena ***i\_k = 7***

## Source Code 18.2 Koding Algoritme Modified Long Short-Term Memory (MLSTM)

```
#Koding MLSTM (part. 1 of 8)

=====
By: Imam Cholissodin | imamcs@ub.ac.id
Dosen Fakultas Ilmu Komputer (Filkom)
Universitas Brawijaya (UB)
Tgl 26 October 2021
Semoga Bermanfaat. Aamin :D
=====
#
11th ICGT 2021 dan lainnya
#
from google.colab import drive
drive.mount('/content/drive')

Buat Folder
import os
os.chdir("/content/drive/My Drive")
path = "Publikasi GreenTech 2021/Koding LSTM utk Manualisasi"
path = "#Penelitian&Pengmas 2021/Publikasi GreenTech 2021/Ko-
ding LSTM utk Manualisasi"
path hasil = "#Penelitian&Pengmas 2021/Publikasi GreenTech 2021/Ko-
ding LSTM utk Manualisasi/HasilTrainNTest"
if not os.path.exists(path):
 os.makedirs(path)
if not os.path.exists(path_hasil):
 os.makedirs(path_hasil)
os.chdir("Publikasi GreenTech 2021/Koding LSTM utk Manualisasi")
os.chdir("#Penelitian&Pengmas 2021/Publikasi GreenTech 2021/Ko-
ding LSTM utk Manualisasi")

!pwd
!ls -l -a --block-size=K

base buku ajar AI, Machine learning & Deep Learning
Link https://www.researchgate.net/publica-
tion/348003841_Buku_Ajar_AI_Machine_Learning_Deep_Learning
=====
By: Imam Cholissodin | imamcs@ub.ac.id
Dosen Fakultas Ilmu Komputer (Filkom)
Universitas Brawijaya (UB)
Tgl 26 October 2021
Semoga Bermanfaat. Aamin :D
=====

2_RNN berbasis Simplified/Modified LSTM (MLSTM) menggunakan ELM pada pred-
iksi data Teks untuk Pembuatan Karya Sastra (by Extend Long-Short Feature pada
Fitur Pola dan Fitur Recurrent pada Mono Cell Gate)

download file data teks sebagai contoh
!gdown --id 1eJX6-5fwI--dU1hYQOUYeWgWYKCjNOX1
import os
path_hasil = "HasilTrainNTest"
if not os.path.exists(path_hasil):
 os.makedirs(path_hasil)
```

### Output:

```
Downloading...
From: https://drive.google.com/uc?id=1eJX6-5fwI--dU1hYQOUYeWgWYKCjNOX1
To: /content/drive/My Drive/#Penelitian&Pengmas 2021/Publikasi GreenTech
2021/Koding LSTM utk Manualisasi/KataBijak_new.txt
100% 133/133 [00:00<00:00, 61.5kB/s]
```

### #Koding MLSTM (part. 2 of 8)

```
=====
By: Imam Cholissodin | imamcs@ub.ac.id
Dosen Fakultas Ilmu Komputer (Filkom)
Universitas Brawijaya (UB)
Tgl 26 October 2021
Semoga Bermanfaat. Aamiin :D
=====

RNN berbasis Simplified/Modified LSTM (MLSTM) menggunakan ELM pada prediksi d
ata Teks
untuk Pembuatan Karya Sastra (by Extend Long-Short Feature pada Fitur Pola dan Fitur Recurrent pada Mono Cell Gate)
#
Load data
import pandas as pd
import numpy as np
import math
import time
from datetime import datetime, timezone
from time import gmtime, strftime
import pytz

file_name = 'KataBijak_new.txt'
df_kataBijak = pd.read_csv(file_name,header=None)
display(df_kataBijak)

batasan:
tdk dilakukan stemming
tidak ada tanda baca karya sastra yg digunakan

case folding
df_kataBijak = df_kataBijak.apply(lambda x: x.astype(str).str.lower())

tokenisasi

print("\n", df_kataBijak.values)
print("\n", token_df_kataBijak.values)

def alt_loadtxt(fname,aktifkan_EOF=None):
 '''Load file txt'''
 f = open(fname,'r')

 data = []
 for line in f.readlines():
 # split tiap line
 # data.append(line.replace('\n','').split(' '))

 # split tiap kata untuk semua line jadi satu line
 # serta menggunakan term "new line" sebagai fitur sekuens juga
 data.extend(line.replace('\n', ' new_line').split(' '))
 if(aktifkan_EOF):
 data.extend(['end_of_file'].split(' ')) # penambahan eof

 f.close()

 return data
def fn_onehot_encoding_1d(integer_encoding_dataInput, n_unik_term=None):
 init_encode = np.zeros((integer_encoding_dataInput.size, np.max(integer_en-
coding_dataInput) + 1 if n_unik_term is None else n_unik_term))
 init_encode[np.arange(integer_encoding_dataInput.size),integer_encoding_da-
taInput] = 1

 get_minus_satu = np.where(integer_encoding_dataInput== -1)[0]
 for i in range(get_minus_satu.size):
 init_encode[get_minus_satu[i],:] = 0

 return init_encode

def fn_onehot_encoding_multi_d(integer_encoding_dataInput, n_unik_term=None):
 return (np.arange(integer_encoding_dataIn-
put.max() + 1 if n_unik_term is None else n_unik_term) == integer_encoding_da-
taInput[... ,None]).astype(float)
```

```
def create_matrix_delay(isTimeSeriesData, cara, r, Ntrain, Ntest, Ytrain, Ytest, Target, byk_data):
 # s_tr_train = np.zeros([Ntrain,r])
 # s_tr_test = np.zeros([Ntest,r])
 s_tr_train = -1*np.ones([Ntrain,r])
 s_tr_test = -1*np.ones([Ntest,r])
 # diset -1, agar nanti ketika di fn_onehot_encoding multi_d nilai fiturnya menjadi nol semua,
 # byk data = Ntrain + Ntest
 if isTimeSeriesData == False:
 if(cara==1):
 # isTimeSeriesData = False | cara ke-1

 for i in range(r):
 s_tr_train [r-((r-1)-i)::,i,None] = Ytrain[:Ntrain-(r-((r-1)-i)),:]
 # print("s_tr_train: \n",s_tr_train)

 # print("\n")

 for i in range(r):
 s_tr_test[r-((r-1)-i)::,i,None] = Ytest[:Ntest-(r-((r-1)-i)),:]
 # print("s_tr_test: \n",s_tr_test)

 return s_tr_train, s_tr_test
 if(cara==2):
 # isTimeSeriesData = False | cara ke-2

 s_tr_train_cara2 = np.zeros([Ntrain,r])
 for tt in range(Ntrain):
 for rr in range(r):
 if (tt > rr):
 # (tt - (n + rr) + n) - 1 = (tt-rr) - 1, jika indeksnya dimulai dari 0
 s_tr_train_cara2[tt,rr] = Ytrain.flatten()[((tt-rr) - 1)]
 else:
 # s_tr_train_cara2[tt,rr] = 0.
 s_tr_train_cara2[tt,rr] = -1.
 # diset -1, agar nanti ketika di fn_onehot_encoding multi_d nilai fiturnya menjadi nol semua,
 # yaitu tidak termasuk semua integer token apa-paun, kecuali jika nanti mengembangkan penggunaan type data spiral pada dataset
 # print("\n s_tr_train_cara2: \n",s_tr_train_cara2)
 # print("\n")

 s_tr_test_cara2 = np.zeros([Ntest,r])
 for tt in range(Ntest):
 for rr in range(r):
 if (tt > rr):
 # (tt - (n + rr) + n) - 1 = (tt-rr) - 1, jika indeksnya dimulai dari 0
 s_tr_test_cara2[tt,rr] = Ytest.flatten()[((tt-rr) - 1)]
 else:
 # s_tr_test_cara2[tt,rr] = 0.
 s_tr_test_cara2[tt,rr] = -1.
 # diset -1, agar nanti ketika di fn_onehot_encoding multi_d nilai fiturnya menjadi nol semua,
 # yaitu tidak termasuk semua integer token apa-paun, kecuali jika nanti mengembangkan penggunaan type data spiral pada dataset
 # print("\n s_tr_test_cara2: \n",s_tr_test_cara2)
 return s_tr_train_cara2, s_tr_test_cara2

 if isTimeSeriesData == True:
 if(cara==1):
 # isTimeSeriesData = True | cara ke-1
 # s_tr_train = np.zeros([Ntrain,r])
 for i in range(r):
 s_tr_train [r-((r-1)-i)::,i,None] = Target[:Ntrain-(r-((r-1)-i)),:]
 # print("s_tr_train: \n",s_tr_train)

 # print("\n")

 # s_tr_test = np.zeros([Ntest,r])
 for i in range(r):
 a_a = r
 b_b = -1
 # mencari suku ke-n dgn kaidah deret aritmatika
```

```
un_un = (a_a-1) + ((i+1)-1)*(b_b)
s_tr_test [:,:,None] = Target[Ntrain-r+un_un:Ntrain-r+Ntest+un_un,:]

print("s_tr_test: \n",s_tr_test)

return s_tr_train, s_tr_test
if(cara==2):
 # isTimeSeriesData = True | cara ke-2
 s_tr_train_n_test_cara2 = np.zeros([byk_data,r])
 for tt in range(byk_data):
 for rr in range(r):
 if (tt > rr):
 # (tt - (n + rr) + n) - 1 = (tt-rr) - 1, jika indeksnya dimulai dari 0
 s_tr_train_n_test_cara2[tt,rr] = Target.flatten()[((tt-rr) - 1)]
 else:
 # s_tr_train_n_test_cara2[tt,rr] = 0.
 s_tr_train_n_test_cara2[tt,rr] = -1.
 # diset -1, agar nanti ketika di fn_onehot_encoding_multi_d nilai fiturnya menjadi nol semua,
 # yaitu tidak termasuk semua integer token apa-paun, kecuali jika nanti mengembangkan penggunaan type data spiral pada dataset
 # print("\n s_tr_train_n_test_cara2: \n",s_tr_train_n_test_cara2)

 # s_tr_train = np.zeros([Ntrain,r])
 # s_tr_test = np.zeros([Ntest,r])
 s_tr_train = s_tr_train_n_test_cara2[:Ntrain,:,:]
 s_tr_test = s_tr_train_n_test_cara2[Ntrain:,:,:]
 return s_tr_train, s_tr_test

def create_matrix_delay_roll(isTimeSeriesData, cara, r, Ntrain, Ntest, Ytrain, Ytest, Target):
def create_matrix_delay_roll(r, Ntrain, Ntest, Ytrain, Ytest, Target):
 # jika pada create_matrix_delay_roll
 # maka pasti isTimeSeriesData == False

 # s_tr_train = -1*np.ones([Ntrain,r])
 # s_tr_test = -1*np.ones([Ntest,r])
 s_tr_train = np.zeros([Ntrain,r])
 s_tr_test = np.zeros([Ntest,r])
 # diset -1, agar nanti ketika di fn_onehot_encoding_multi_d nilai fiturnya menjadi nol semua,

 for i in range(r):
 # s_tr_train [r-((r-1)-i)::,i,None] = Ytrain[:Ntrain-(r-((r-1)-i)),:]
 s_tr_train[:,i,None] = np.roll(Ytrain,(i+1))
 # print("s_tr_train: \n",s_tr_train)

 # print("\n")

 for i in range(r):
 # s_tr_test[r-((r-1)-i)::,i,None] = Ytest[:Ntest-(r-((r-1)-i)),:]
 s_tr_test[:,i,None] = np.roll(Ytest,(i+1))
 # print("s_tr_test: \n",s_tr_test)

 return s_tr_train, s_tr_test

def create_matrix_delay_roll_mono_gate(r, Ntrain, Ntest, Ytrain, Ytest, Target):
 # jika pada create matrix delay roll
 # maka pasti isTimeSeriesData == False

 # s_tr_train = -1*np.ones([Ntrain,r])
 # s_tr_test = -1*np.ones([Ntest,r])
 s_tr_train = np.zeros([Ntrain,r])
 s_tr_test = np.zeros([Ntest,r])
 # diset -1, agar nanti ketika di fn_onehot_encoding_multi_d nilai fiturnya menjadi nol semua,

 for i in range(r):
 # s_tr_train [r-((r-1)-i)::,i,None] = Ytrain[:Ntrain-(r-((r-1)-i)),:]
 s_tr_train[:,i,None] = np.roll(Ytrain,(i+1))
 # print("s_tr_train: \n",s_tr_train)

 # print("\n")
 s_tr_train_final = np.vstack((-1*np.ones([Ntrain,r]),s_tr_train))

 for i in range(r):
 # s_tr_test[r-((r-1)-i)::,i,None] = Ytest[:Ntest-(r-((r-1)-i)),:]

```

```
s_tr_test[:,i,None] = np.roll(Ytest,(i+1))
print("s_tr_test: \n",s_tr_test)

s_tr_test_final = np.vstack((-1*np.ones([Ntest,r]),s_tr_test))

return s_tr_train_final, s_tr_test_final

def myrandfloat(mbaris,nkolom,lower,upper):
 #BatasRANDplusOne=10000
 BatasRANDplusOne=max(10000,2*np.math.ceil(upper))
 #mbaris=1
 #nkolom=1
 Rand_Sample=np.random.randint(BatasRANDplusOne,size=(mbaris,nkolom))
 min_Rand_Sample = 0
 max_Rand_Sample = BatasRANDplusOne - 1
 upper_boundary= upper
 lower_boundary= lower
 normalize_Rand_Sample_minMax=((Rand_Sample-min_Rand_Sample)/(max_Rand_Sample-min_Rand_Sample))*((upper_boundary-lower_boundary))+lower_boundary
 #print(normalize_Rand_Sample_minMax)
 #x = normalize_Rand_Sample_minMax
 return normalize_Rand_Sample_minMax

def fn_sigmoid(x):
 return 1. / (1. + np.e**(-x))

def fn_tanh(x):
 return (np.e**x-np.e**(-x)) / (np.e**x+np.e**(-x))

def y_predict_to_onehot_binary(y_predict):
 # axis = 1 artinya mengambil nilai indeks, yang nilainya pal-ing maks tiap barisnya
 argmax_index_arr = np.argmax(y_predict, axis=1)
 onehot_binary_arr = np.zeros((y_predict.shape[0],y_predict.shape[1]))
 onehot_binary_arr[np.arange(y_predict.shape[0]),argmax_index_arr] = 1
 return onehot_binary_arr,argmax_index_arr

def onehot_binary_to_arr_token(onehot_binary_arr,get_unik_term_terurut_alpha-
bet):
 # axis = 1 artinya mengambil nilai indeks, yang nilainya pal-ing maks tiap barisnya
 argmax_index_arr = np.argmax(onehot_binary_arr, axis=1)
 return get_unik_term_terurut_alphabet[argmax_index_arr]

def arr_token_to_txt(fname, arr_token):
 with open('./HasilTrainNTest/'+fname, "w") as fhandle:
 for line in arr_token:
 if line == "new_line":
 line = '.. (ini penanda new_line)'
 fhandle.write(f'{line}\n')

 with open('./HasilTrainNTest/'+fname, 'r') as file:
 data_txt = file.read()

 # print(data_txt)
 return data_txt

def arr_token_to_txt_sastr(a,fname, arr_token):
 with open('./HasilTrainNTest/'+fname, "w") as fhandle:
 for line in arr_token:
 if line == "new_line":
 line = '\n'
 fhandle.write(f'{line}')
 else:
 fhandle.write(f'{line} ')

 with open('./HasilTrainNTest/'+fname, 'r') as file:
 data_txt = file.read()

 # print(data_txt)
 return data_txt

def get_MAPE_0_100(aktual,predict):
 banyak_fitur_target = aktual.shape[1]
 # Ru-
 # mus MAPE yang digunakan, jika data aktualnya ada nol atau tidak ada yg nol
 # dan untuk memastikan MAPE pada interval = [0%;100%] --> dengan kondisi ini
 # Ref: (Berretti,Thampi,danSrivastava,2015) dalam
 # Hapsari,KD.,Cholissodin,I.,Santoso,E.,2016
 konstanta_smoothing = 0.00000001
 c = konstanta_smoothing
 mape_init = np.abs(((aktual+c) - (predict+c)) / (aktual+c))*100
```

```
 mape_norm = np.sum(np.where(mape_init>100, 100, mape_init))/(len(predict)*banyak_fitur_target)
 return mape_norm
def fn_n_gram_generator(kalimat_in,n= 2,n_gram= False):
 """
 N-Gram generator dengan parameters kalimat yang dimasukkan
 n menyatakan banyaknya n_grams
 parameter n_gram = False artinya menghilangkan proses perulangan pada n_grams
 """
 kalimat_in = kalimat_in.lower() # convert menjadi lower case
 sent_arr = np.array(kalimat_in.split()) # split menjadi string arrays
 length = len(sent_arr)

 word_list = []
 for i in range(length+1):
 if i < n:
 continue
 word_range = list(range(i-n,i))
 s_list = sent_arr[word_range]
 string = ' '.join(s_list) # converting list menjadi strings
 word_list.append(string) # append menjadi word_list
 if n_gram:
 word_list = list(set(word_list))
 return word_list

def fn_bleu_score(original_refernce_atau_data_aktual, candidate_atau_hasil_predict_atau_hasil_sistem):
 """
 Bleu score untuk kalimat original_refernce_atau_data_aktual dan candidate_atau_hasil_predict_atau_hasil_sistem
 candidate_atau_hasil_predict_atau_hasil_sistem bisa berupa hasil kalimat generate secara global, misal membuat kalimat dengan algoritma RNN_Native/RNN_LSTM atau hasil mesin translate dll.
 """
 cand_length = len(candidate_atau_hasil_predict_atau_hasil_sistem.split())
 ori_length = len(original_refernce_atau_data_aktual.split())

 # Brevity Penalty
 if cand_length > ori_length:
 BP=1
 else:
 penalty=1-(cand_length/ori_length)
 BP = np.exp(penalty)

 # Clipped precision
 clipped_precision_score = []
 for ngram_level in range(1, 5): # 1-gram sampai 4-gram

 original_ngram_list = fn_n_gram_generator(original_refernce_atau_data_aktual, ngram_level)
 # original_n_gram = Counter(original_ngram_list) # jika dat besar, gunakan ini
 original_n_gram = dict(zip(*np.unique(original_ngram_list, return_counts=True)))

 machine_ngram_list = fn_n_gram_generator(candidate_atau_hasil_predict_atau_hasil_sistem, ngram_level)
 # machine_n_gram = Counter(machine_ngram_list) # jika dat besar, gunakan ini
 machine_n_gram = dict(zip(*np.unique(machine_ngram_list, return_counts=True)))

 num_ngrams_in_translation = sum(machine_n_gram.values()) # jumlah ngrams yang digunakan

 # iterate the unique ngrams in translation (candidate)
 for j in machine_n_gram:

 if j in original_n_gram: # if found in reference

 if machine_n_gram[j] > original_n_gram[j]: # CLIPPING - if found in translation more than in source, clip
 machine_n_gram[j] = original_n_gram[j]

 else:
 machine_n_gram[j] = 0
```

```
#print (sum(machine_n_gram.values()), c)
clipped_precision_score.append(float(sum(machine_n_gram.values()) / num_ngrams_in_translation))

#print (clipped_precision_score)

secara default, BLEU score menghitung cumulative 4-
gram atau disebut dengan BLEU-4,
maka weights untuk BLEU-4 adalah 1/4 (atau 25%) atau 0.25 untuk tiap 1-
gram, 2-gram, 3-gram dan 4-gram scores.
sehingga weights = [0.25]*4
weights = [0.25]*4

s = (w_i * np.log(p_i) for w_i, p_i in zip(weights, clipped_preci-
sion_score))
s = (w_i * np.log(p_i+0.001) if p_i==0 else np.log(p_i) for w_i, p_i in zip(w-
eights, clipped_precision_score)) # untuk menghandle di-
vide by zero di log dgn menambah nilai konstanta kecil, misal 0.001
s = BP * np.exp(math.fsum(s)) #fsum sama dengan sum dengan type float
return s
```

### #Koding MLSTM (part. 3 of 8)

```
load data KataBijak_new.txt
file_name = 'KataBijak_new.txt'
aktifkan_EOF = True # dapat diisikan True/False
data = alt_loadtxt(file_name, aktifkan_EOF)
case folding
data = [name.lower() for name in data]
print(data)
print(len(data))

convert list menjadi array semua token dalam numpy
arr_token = np.array(data)
len_token = arr_token.shape[0]
print('Array semua Token: \n', arr_token)
print('\nPanjang Token:', len_token)
```

### Output:

```
Array semua Token:
['tujuan' 'hidup' 'kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat'
 'new_line' 'kesabaran' 'adalah' 'obat' 'terbaik' 'untuk' 'segala'
 'kesulitan' 'new_line' 'siapapun' 'bisa' 'jadi' 'apapun' 'end_of_file']

Panjang Token: 22
```

### #Koding MLSTM (part. 4 of 8)

```
one-hot encoded vector sebagai salah satu teknik untuk merepresentasikan se-
tiap term dalam ruang d dimensi
dimana d = len_unik_term
#
get unik term
get_unik_term_terurut_alphabet = np.sort(np.array(list(set(arr_token))))
len_unik_term = len(get_unik_term_terurut_alphabet)
print('get_unik_term_terurut_alphabet: \n', get_unik_term_terurut_alphabet)
print('\nlen_unik_term: ', len_unik_term, '\n')
```

```
nilai integrer untuk tiap kata unik tersebut
integer_encoding = np.arange(0,len_unik_term)
print('integer_encoding: \n', integer_encoding, '\n')

proses membuat one hot encoding berupa vektor biner berdasarkan nilai inte-
ger encoding diatas
misal terdapat sekueuns data berikut:
raw_data = '''hidup dan
hidup bahagia adalah tujuan terbaik''' # jika ada kata yg tidak ada da-
lam unik term, maka dianggap "new_line"
misal kata "dan", dengan coding tambahan berupa
np.where(get_unik_term_terurut_alphabet==token_in)[0][0] \
if np.where(get_unik_term_terurut_alphabet==token_in)[0].size \
else np.where(get_unik_term_terurut_alphabet=='new_line')[0][0]

unraw_data = []
unraw_data.extend(raw_data.replace('\n','new_line ').split(' '))
if (aktifkan_EOF):
 rawData.extend(['\n']) # menambahkan eof (optional)
unraw_data = [name.lower() for name in rawData]
arr_token_unraw_data = np.array(unraw_data)
dataInput = arr_token_unraw_data
print('dataInput: \n', dataInput, '\n')

get nilai kode integer
integer_encoding_dataInput,len_sekuens_dataInput = np.ar-
ray([np.where(get_unik_term_terurut_alphabet==token_in)[0][0] \
 if np.where(get_un-
ik_term_terurut_alphabet==token_in)[0].size \
 else np.where(get_-
unik_term_terurut_alphabet=='new_line')[0][0] \
 for to-
ken_in in dataInput]),dataInput.shape[0]
print('integer_encoding_dataInput: \n', integer_encoding_dataInput, '\n')
print('len_sekuens_dataInput: \n', len_sekuens_dataInput, '\n')

mengubah nilai kode integer menjadi one hot encod-
ing berupa vektor biner, ke depannya dapat dikembangkan menggunakan probabilis-
tic encoding
Ref:
https://stackoverflow.com/questions/36960320/convert-a-2d-matrix-to-a-3d-one-
hot-matrix-numpy
https://stackoverflow.com/questions/29831489/convert-array-of-indices-to-1-
hot-encoded-numpy-array
#
onehot_integer_encoding_dataInput = fn_onehot_encoding_1d(integer_encoding_da-
taInput, len_unik_term)
print('onehot_integer_encoding_dataInput: \n', onehot_integer_encoding_dataIn-
put, '\n')

persiapan buat data train
semua_token_dataInput = arr_token
print('semua_token_dataInput: \n', semua_token_dataInput, '\n')

get nilai kode integer
integer_encoding_semua_token_dataInput,len_sekuens_semua_token_dataIn-
put = np.array([np.where(get_unik_term_terurut_alphabet==token_in)[0][0] \
 if np.where(get_un-
ik_term_terurut_alphabet==token_in)[0].size \
 else np.where(get_-
unik_term_terurut_alphabet=='new_line')[0][0] \
 for to-
ken_in in semua_token_dataInput]),semua_token_dataInput.shape[0]
print('integer_encoding_semua_token_dataInput: \n', integer_encoding_semua_to-
ken_dataInput, '\n')
print('len_sekuens_semua_token_dataInput: \n', len_sekuens_semua_token_dataIn-
put, '\n')

onehot_integer_encoding_semua_token_dataInput = fn_onehot_encoding_1d(inte-
ger_encoding_semua_token_dataInput, len_unik_term)
print('onehot_integer_encoding_semua_token_dataInput: \n', onehot_integer_en-
coding_semua_token_dataInput, '\n')
```

## Output:

### #Koding MLSTM (part. 5 of 8)

```
set nilai parameter RNN berbasis simplified/Modified LSTM (MLSTM) menggunakan ELM
inisialisasi jumlah fitur (n) yang digunakan sebanyak 3,
sedangkan jumlah hidden neuron (m) sebanyak 2,
dan jumlah context neuron (r) sebanyak 2.
Sedangkan datasets dibagi menjadi 2 bagian,
untuk data training sebanyak misal diset 75% dan data testing sebanyak 25% data.
n = 3
r = 2
m = int(np.ceil((2/3)*((n+r)*len_unik_term)))
m = int(np.ceil((2/3)*((n+r)*len_unik_term)))

buat dataset cara ke-1, di mana hanya membuat sekvens sebanyak n fitur saja
sehingga tidak ada sekvens sebanyak 1 fitur, 2 fitur, ..., n fitur
byk_data_cara1 = len_sekuens_semua_token_dataInput-n
dataset_cara1 = np.zeros((byk_data_cara1,n+1))

byk data = dataset.shape[0]
Ntrain = int((75/100)*dataset.shape[0]) # misal diset 75%
Ntrain_cara1= int((100/100)*dataset_cara1.shape[0]) # misal diset 100%
Ntest = byk data - Ntrain # untuk yang tidak "100%" sebagai data train, maka data testing disamakan dgn data training"
Ntest_cara1 = Ntrain_cara1 # misal dibuat semua data latih juga sebagai data uji

apakahTrain100persen = True
if(apakahTrain100persen):

print('byk data cara1:', byk_data_cara1,'')
print('Ntrain_cara1:', Ntrain_cara1,'')
print('Ntest_cara1:', Ntest_cara1,'\n')

for i in range(n+1):
 if i == n:
 dataset_cara1[:,i,None] = integer_encoding_semua_token_dataInput[...,None][i,:,:]
 else:
 dataset_cara1[:,i,None] = integer_encoding_semua_token_dataInput[...,None][1:-n+i,:]

print('dataset_cara1: \n', dataset_cara1,'\n')
print('dataset_cara1.shape: ', dataset_cara1.shape,' \n')
print('integer encoding semua token_dataInput: \n', integer_encoding_semua_token_dataInput,' \n')

buat dataset cara ke-2, di mana selain membuat sekvens sebanyak n fitur,
juga ada sekvens sebanyak 1 fitur, 2 fitur, ..., n-1 fitur, n fitur
byk_data_cara2 = 0
for i in range(n):
 byk_data_cara2 += len_sekuens_semua_token_dataInput-(i+1)

dikembangkan menggunakan dataset dengan type circle atau roll (aktifkan_type_dataset_circle_or_roll = True)
True artinya setiap data pada integer_encoding dipastikan pernah menjadi nilai target, dan sebaliknya
aktifkan_type_dataset_circle_or_roll = True
if(aktifkan_type_dataset_circle_or_roll):
 # atau dengan cara agak panjang
 penambah_utk_byk_data_cara2 = 0
 for i in range(n):
 penambah_utk_byk_data_cara2 += (i+1)
 byk_data_cara2 += penambah_utk_byk_data_cara2

 # atau dengan cara singkat
 # byk_data_cara2 = n*len_sekuens_semua_token_dataInput

dataset_cara2 = np.zeros((byk_data_cara2,n+1))

Ntrain_cara2 = int((75/100)*byk_data_cara2) # misal diset 75%
Ntrain_cara2 = int((100/100)*byk_data_cara2) # misal diset 100%
```

```
Ntest cara2 = byk_data cara2 - Ntrain cara2 # untuk yang tidak "100% sebagai data train, maka data testing disamakan dgn data training"
Ntest_cara2 = Ntrain_cara2 # misal dibuat semua data latih juga sebagai data uji
print('byk data cara2:', byk_data_cara2,'')
print('Ntrain_cara2:', Ntrain_cara2,'')
print('Ntest_cara2:', Ntest_cara2,'\\n')

time.sleep(3600)

if(aktifkan_type_dataset_circle_or_roll==False):
 for i_fitur in range(n):
 iter_fitur = i_fitur + 1
 byk_data_Temp = len(sekuens_semu_token_dataInput-iter_fitur)
 dataset_Temp = np.zeros((byk_data_Temp,n+1))
 for i in range(n+1):
 if i == n:
 dataset_Temp[:,i,None] = integer_encoding_semu_token_dataInput[...,None][iter_fitur,:,:]
 elif(i<(n-iter_fitur)):
 # dataset_Temp[:,i,None] = 0 # dengan recurrent = nol
 dataset_Temp[:,i,None] = -1 # dengan recurrent = -1
 else:
 dataset_Temp[:,i,None] = integer_encoding_semu_token_dataInput[...,(i-(n-iter_fitur))-iter_fitur+i-(n-iter_fitur),:]
 # print(dataset_Temp)
 # print(dataset_Temp.shape)
 if(i_fitur==0):
 dataset_cara2 = dataset_Temp
 else:
 dataset_cara2 = np.vstack((dataset_cara2,dataset_Temp))

 elif(aktifkan_type_dataset_circle_or_roll):
 for i_fitur in range(n):
 iter_fitur = i_fitur + 1
 # byk data Temp = len sekuens semua token dataInput-iter_fitur
 byk_data_Temp = len(sekuens_semu_token_dataInput)
 dataset_Temp = np.zeros((byk_data_Temp,n+1))
 for i in range(n+1):
 if i == n:
 # dataset.Temp[:,i,None] = integer_encoding_semu_token_dataInput[...,None][iter_fitur,:,:]
 dataset_Temp[:,i,None] = np.roll(integer_encoding_semu_token_dataInput[...,None], -iter_fitur)
 elif(i<(n-iter_fitur)):
 # dataset.Temp[:,i,None] = 0 # dengan recurrent = nol
 dataset_Temp[:,i,None] = -1 # dengan recurrent = -1
 else:
 # dataset.Temp[:,i,None] = integer_encoding_semu_token_dataInput[...,(i-(n-iter_fitur))-iter_fitur+i-(n-iter_fitur),:]
 dataset_Temp[:,i,None] = np.roll(integer_encoding_semu_token_dataInput[...,(i-(n-iter_fitur))], -1)
 # print(dataset_Temp)
 # print(dataset_Temp.shape)
 if(i_fitur==0):
 dataset_cara2 = dataset_Temp
 else:
 dataset_cara2 = np.vstack((dataset_cara2,dataset_Temp))

 print('dataset_cara2: \\n', dataset_cara2,'\\n')
 print('dataset_cara2.shape: \\n', dataset_cara2.shape,'\\n')

 print('integer_encoding_semu_token_dataInput: \\n', integer_encoding_semu_token_dataInput,'\\n')

 # misal menggunakan dataset_cara1 atau dataset_cara2 sebagai dataset, untuk upaya peningkatan hasil kinerja pembelajaran sistem
 set_cara_membuat_dataset = 2
 if(set_cara_membuat_dataset==1):
 dataset = dataset_cara1
 byk_data = dataset.shape[0]
 Ntrain = Ntrain_cara1
 Ntest = Ntest_cara1
 elif(set_cara_membuat_dataset==2):
 dataset = dataset_cara2
 byk_data = dataset.shape[0]
 Ntrain = Ntrain_cara2
 Ntest = Ntest_cara2
 Ntrain_ori = Ntrain_cara2
 Ntest_ori = Ntest_cara2
```

```
onehot_dataset = fn_onehot_encoding_multi_d(dataset, len_unik_term)
print('onehot_dataset: \n', onehot_dataset, '\n')
print('onehot_dataset.shape: \n', onehot_dataset.shape, '\n')

Xtrain_feature,Xtrain = dataset[:Ntrain,:-1],dataset[:Ntrain,:]
print('\n Xtrain_feature: \n',Xtrain_feature)
print('\n Xtrain: \n',Xtrain)

Xtest_feature,Xtest = dataset[Ntrain:,:-1],dataset[Ntrain:,:] # untuk yang tidak "100% dataset sebagai data train, maka data testing disamakan dgn data training"
Xtest_feature,Xtest = dataset[Ntrain,:-1],dataset[:Ntrain,:] # untuk yang "100% dataset sebagai data train, maka data testing disamakan dgn data training"

if(aktifkan_type_dataset_circle_or_roll):
 Xtrain_feature,Xtrain = np.vstack((Xtrain_feature,Xtrain_feature)),np.vstack((Xtrain,Xtrain))
 Xtest_feature,Xtest = np.vstack((Xtest_feature,Xtest_feature)),np.vstack((Xtest,Xtest))

Ytrain, Ytest = dataset[:Ntrain,-1], dataset[Ntrain:,-1] # untuk yang tidak "100% dataset sebagai data train, maka data testing disamakan dgn data training"
Ytrain, Ytest = dataset[:Ntrain,-1], dataset[:Ntrain,-1] # untuk yang "100% dataset sebagai data train, maka data testing disamakan dgn data training"
Ytrain,Ytest = Ytrain[...,None],Ytest[...,None]
print('\n Ytrain: \n',Ytrain)
print('\n Ytest: \n',Ytest)

Ytrain_non_norm, Ytest_non_norm = arr_df_idr_us[:Ntrain,-1], arr_df_idr_us[Ntrain:,-1]
Ytrain_non_norm,Ytest_non_norm = Ytrain_non_norm[...,None],Ytest_non_norm[...,None]

Ytrain_onehot_encode, Ytest_onehot_encode = fn_onehot_encoding_multi_d(dataset[:Ntrain,-1], len_unik_term), \
fn_onehot_encoding_multi_d(dataset[Ntrain:,-1], len_unik_term) # untuk yang tidak "100% dataset sebagai data train, maka data testing disamakan dgn data training"

Ytrain_onehot_encode, Ytest_onehot_encode = fn_onehot_encoding_multi_d(dataset[:Ntrain,-1], len_unik_term), \
fn_onehot_encoding_multi_d(dataset[:Ntrain,-1], len_unik_term) # untuk yang "100% dataset sebagai data train, maka data testing disamakan dgn data training"

if(aktifkan_type_dataset_circle_or_roll):
 Ytrain_onehot_encode, Ytest_onehot_encode = np.vstack((Ytrain_onehot_encode,Ytrain_onehot_encode)),np.vstack((Ytest_onehot_encode,Ytest_onehot_encode))

print('\n Ytrain_onehot_encode: \n',Ytrain_onehot_encode)
print('\n Ytest_onehot_encode: \n',Ytest_onehot_encode)
```

## Output:

```
byk_data_caral: 19
Ntrain_caral: 19
Ntest_caral: 19

dataset_caral:
[[18. 6. 10. 0.]
 [6. 10. 0. 11.]
 [10. 0. 11. 2.]
 [0. 11. 2. 4.]
 [11. 2. 4. 15.]
 [2. 4. 15. 12.]
 [4. 15. 12. 8.]
 [15. 12. 8. 0.]
```

```
[12. 8. 0. 13.]
[8. 0. 13. 17.]
[0. 13. 17. 19.]
[13. 17. 19. 14.]
[17. 19. 14. 9.]
[19. 14. 9. 12.]
[14. 9. 12. 16.]
[9. 12. 16. 3.]
[12. 16. 3. 7.]
[16. 3. 7. 1.]
[3. 7. 1. 5.]]]

dataset_cara1.shape:
(19, 4)

byk_data_cara2: 66
Ntrain_cara2: 66
Ntest_cara2: 66

dataset_cara2:
[[-1. -1. 18. 6.]
 [-1. -1. 6. 10.]
 [-1. -1. 10. 0.]
 [-1. -1. 0. 11.]
 [-1. -1. 11. 2.]
 [-1. -1. 2. 4.]
 [-1. -1. 4. 15.]
 [-1. -1. 15. 12.]
 [-1. -1. 12. 8.]
 [-1. -1. 8. 0.]
 [-1. -1. 0. 13.]
 [-1. -1. 13. 17.]
 [-1. -1. 17. 19.]
 [-1. -1. 19. 14.]
 [-1. -1. 14. 9.]
 [-1. -1. 9. 12.]
 [-1. -1. 12. 16.]
 [-1. -1. 16. 3.]
 [-1. -1. 3. 7.]
 [-1. -1. 7. 1.]
 [-1. -1. 1. 5.]
 [-1. -1. 5. 18.]
 [-1. 18. 6. 10.]
 [-1. 6. 10. 0.]
 [-1. 10. 0. 11.]
 [-1. 0. 11. 2.]
 [-1. 11. 2. 4.]
 [-1. 2. 4. 15.]
 [-1. 4. 15. 12.]
 [-1. 15. 12. 8.]
 [-1. 12. 8. 0.]
 [-1. 8. 0. 13.]
 [-1. 0. 13. 17.]
 [-1. 13. 17. 19.]
 [-1. 17. 19. 14.]
 [-1. 19. 14. 9.]
 [-1. 14. 9. 12.]
 [-1. 9. 12. 16.]
 [-1. 12. 16. 3.]
 [-1. 16. 3. 7.]
 [-1. 3. 7. 1.]
 [-1. 7. 1. 5.]
 [-1. 1. 5. 18.]
 [-1. 5. 18. 6.]
 [18. 6. 10. 0.]
 [6. 10. 0. 11.]
 [10. 0. 11. 2.]
 [0. 11. 2. 4.]
 [11. 2. 4. 15.]
 [2. 4. 15. 12.]
 [4. 15. 12. 8.]
 [15. 12. 8. 0.]
 [12. 8. 0. 13.]
 [8. 0. 13. 17.]
 [0. 13. 17. 19.]
 [13. 17. 19. 14.]
 [17. 19. 14. 9.]
 [19. 14. 9. 12.]
 [14. 9. 12. 16.]
 [9. 12. 16. 3.]
 [12. 16. 3. 7.]
```

```
[16. 3. 7. 1.]
[3. 7. 1. 5.]
[7. 1. 5. 18.]
[1. 5. 18. 6.]
[5. 18. 6. 10.]]]

dataset_cara2.shape:
(66, 4)

integer_encoding_semua_token_dataInput:
[18 6 10 0 11 2 4 15 12 8 0 13 17 19 14 9 12 16 3 7 1 5]

Xtrain_feature:
[[-1. -1. 18.]
 [-1. -1. 6.]
 [-1. -1. 10.]
 [-1. -1. 0.]
 [-1. -1. 11.]
 [-1. -1. 2.]
 [-1. -1. 4.]
 [-1. -1. 15.]
 [-1. -1. 12.]
 [-1. -1. 8.]
 [-1. -1. 0.]
 [-1. -1. 13.]
 [-1. -1. 17.]
 [-1. -1. 19.]
 [-1. -1. 14.]
 [-1. -1. 9.]
 [-1. -1. 12.]
 [-1. -1. 16.]
 [-1. -1. 3.]
 [-1. -1. 7.]
 [-1. -1. 1.]
 [-1. -1. 5.]
 [-1. 18. 6.]
 [-1. 6. 10.]
 [-1. 10. 0.]
 [-1. 0. 11.]
 [-1. 11. 2.]
 [-1. 2. 4.]
 [-1. 4. 15.]
 [-1. 15. 12.]
 [-1. 12. 8.]
 [-1. 8. 0.]
 [-1. 0. 13.]
 [-1. 13. 17.]
 [-1. 17. 19.]
 [-1. 19. 14.]
 [-1. 14. 9.]
 [-1. 9. 12.]
 [-1. 12. 16.]
 [-1. 16. 3.]
 [-1. 3. 7.]
 [-1. 7. 1.]
 [-1. 1. 5.]
 [-1. 5. 18.]
 [18. 6. 10.]
 [6. 10. 0.]
 [10. 0. 11.]
 [0. 11. 2.]
 [11. 2. 4.]
 [2. 4. 15.]
 [4. 15. 12.]
 [15. 12. 8.]
 [12. 8. 0.]
 [8. 0. 13.]
 [0. 13. 17.]
 [13. 17. 19.]
 [17. 19. 14.]
 [19. 14. 9.]
 [14. 9. 12.]
 [9. 12. 16.]
 [12. 16. 3.]
 [16. 3. 7.]
 [3. 7. 1.]
 [7. 1. 5.]
 [1. 5. 18.]
 [5. 18. 6.]]]
```

```
Xtrain:
[[-1. -1. 18. 6.]
[-1. -1. 6. 10.]
[-1. -1. 10. 0.]
[-1. -1. 0. 11.]
[-1. -1. 11. 2.]
[-1. -1. 2. 4.]
[-1. -1. 4. 15.]
[-1. -1. 15. 12.]
[-1. -1. 12. 8.]
[-1. -1. 8. 0.]
[-1. -1. 0. 13.]
[-1. -1. 13. 17.]
[-1. -1. 17. 19.]
[-1. -1. 19. 14.]
[-1. -1. 14. 9.]
[-1. -1. 9. 12.]
[-1. -1. 12. 16.]
[-1. -1. 16. 3.]
[-1. -1. 3. 7.]
[-1. -1. 7. 1.]
[-1. -1. 1. 5.]
[-1. -1. 5. 18.]
[-1. 18. 6. 10.]
[-1. 6. 10. 0.]
[-1. 10. 0. 11.]
[-1. 0. 11. 2.]
[-1. 11. 2. 4.]
[-1. 2. 4. 15.]
[-1. 4. 15. 12.]
[-1. 15. 12. 8.]
[-1. 12. 8. 0.]
[-1. 8. 0. 13.]
[-1. 0. 13. 17.]
[-1. 13. 17. 19.]
[-1. 17. 19. 14.]
[-1. 19. 14. 9.]
[-1. 14. 9. 12.]
[-1. 9. 12. 16.]
[-1. 12. 16. 3.]
[-1. 16. 3. 7.]
[-1. 3. 7. 1.]
[-1. 7. 1. 5.]
[-1. 1. 5. 18.]
[-1. 5. 18. 6.]
[18. 6. 10. 0.]
[6. 10. 0. 11.]
[10. 0. 11. 2.]
[0. 11. 2. 4.]
[11. 2. 4. 15.]
[2. 4. 15. 12.]
[4. 15. 12. 8.]
[15. 12. 8. 0.]
[12. 8. 0. 13.]
[8. 0. 13. 17.]
[0. 13. 17. 19.]
[13. 17. 19. 14.]
[17. 19. 14. 9.]
[19. 14. 9. 12.]
[14. 9. 12. 16.]
[9. 12. 16. 3.]
[12. 16. 3. 7.]
[16. 3. 7. 1.]
[3. 7. 1. 5.]
[7. 1. 5. 18.]
[1. 5. 18. 6.]
[5. 18. 6. 10.]]]

Ytrain:
[[6.]
[10.]
[0.]
[11.]
[2.]
[4.]
[15.]
[12.]
[8.]
[0.]
[13.]
[17.]
```

```
[19.]
[14.]
[9.]
[12.]
[16.]
[3.]
[7.]
[1.]
[5.]
[18.]
[10.]
[0.]
[11.]
[2.]
[4.]
[15.]
[12.]
[8.]
[0.]
[13.]
[17.]
[19.]
[14.]
[9.]
[12.]
[16.]
[3.]
[7.]
[1.]
[5.]
[18.]
[6.]
[0.]
[11.]
[2.]
[4.]
[15.]
[12.]
[8.]
[0.]
[13.]
[17.]
[19.]
[14.]
[9.]
[12.]
[16.]
[3.]
[7.]
[1.]
[5.]
[18.]
[6.]
[10.]]
```

Ytest:

```
[[6.]
[10.]
[0.]
[11.]
[2.]
[4.]
[15.]
[12.]
[8.]
[0.]
[13.]
[17.]
[19.]
[14.]
[9.]
[12.]
[16.]
[3.]
[7.]
[1.]
[5.]
[18.]
[10.]
[0.]
[11.]]
```

```
[2.]
[4.]
[15.]
[12.]
[8.]
[0.]
[13.]
[17.]
[19.]
[14.]
[9.]
[12.]
[16.]
[3.]
[7.]
[1.]
[5.]
[18.]
[6.]
[0.]
[11.]
[2.]
[4.]
[15.]
[12.]
[8.]
[0.]
[13.]
[17.]
[19.]
[14.]
[9.]
[12.]
[16.]
[3.]
[7.]
[1.]
[5.]
[18.]
[6.]
[10.]]
```

```
Ytrain_onehot_encode:
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 1. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

```
Ytest_onehot_encode:
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 1. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

### #Koding LSTM (part. 6 of 8)

```
proses training

#
1. inisialisasi matriks delay (s)
Target = dataset[:, -1]
Target = Target[..., None]
print(Target)
print(Target.shape)

isTimeSeriesData=False # dapat diisikan True/False
```

```
cara=2 # dapat disikan 1 atau 2, 1 maknanya menggunakan cara lain dgn memodifikasi cara akses indeks yg tidak persis dgn dipaper referensi, tetapi hasil sama dengan cara 2 yg base paper
if(aktifkan_type_dataset_circle_or_roll==False):
 s_tr_train,s_tr_test = create_matrix_delay(isTimeSeriesData, cara, r, Ntrain, Ntest, Ytrain, Ytest, Target, byk_data)
elif(aktifkan_type_dataset_circle_or_roll):
 s_tr_roll mono_gate(r, Ntrain_ori, Ntest_ori, Ytrain, Ytest, Target)
 byk_data = 2*Ntrain # atau 2*s_tr_train.shape[0]
 Ntrain = s_tr_train.shape[0]
 Ntest = s_tr_test.shape[0]

print(" s_tr_train: \n", s_tr_train)
print()
print("\n s_tr_test : \n", s_tr_test)

2.1 Random nilai matrik W (bobot input)
W = myrandfloat(m,n+r,lower=-1,upper=1)
print("\n W : \n", W)

2.2 Random nilai matrik bias
bias = myrandfloat(1,m,lower=0,upper=1)
print("\n bias : \n", bias)

3. Menghitung nilai Htrain, Hplus
#
3.1 Menggabungkan data latih dengan matriks delay
if(aktifkan_type_dataset_circle_or_roll==False):
Xtrain_merge_s = np.hstack((Xtrain_feature,s_tr_train))
elif(aktifkan_type_dataset_circle_or_roll):
Xtrain_merge_s = np.hstack((np.vstack((Xtrain_feature,Xtrain_feature)),s_tr_train))
print("\n Xtrain_merge_s : \n", Xtrain_merge_s,"\n")

onehot_Xtrain_merge_s = fn_onehot_encoding_multi_d(Xtrain_merge_s, len_unik_term)
print('onehot_Xtrain_merge_s: \n', onehot_Xtrain_merge_s,' \n')
print('onehot_Xtrain_merge_s.shape: \n', onehot_Xtrain_merge_s.shape,' \n')

full connected, yaitu join semua fitur di dalam onehot_Xtrain_merge_s menjadi array dgn ukuran [Ntrain x (n+r)*len_unik_term]
dimana setiap datanya yang awalnya berukuran [(n+r) x len_unik_term] menjadi berukuran [1 x (n+r)*len_unik_term]
full_conn_onehot_Xtrain_merge_s = np.zeros((Ntrain,(n+r)*len_unik_term))
full_conn_onehot_Xtrain_merge_s = onehot_Xtrain_merge_s.reshape(Ntrain,(n+r)*len_unik_term)
print('full conn onehot Xtrain merge s: \n', full_conn_onehot_Xtrain_merge_s,' \n')
print('full_conn_onehot_Xtrain_merge_s.shape: \n', full_conn_onehot_Xtrain_merge_s.shape,' \n')

is_singular_matrix = True
while(is_singular_matrix):
 # 2.1 Random nilai matrik W (bobot input dgn ordo banyak_hidden_neuron(m), banyak_fitur(n)+recurrent(r))
 # bobot = np.random.rand
 # W = myrandfloat(m,n+r,lower=-1,upper=1)
 W = myrandfloat(m, (n+r)*len_unik_term,lower=-1,upper=1)
 bobot_input = W
 # print("\n W : \n", W)

 # 2.2 Random nilai matrik bias
 bias = myrandfloat(1,m,lower=0,upper=1).flatten()
 # print("\n bias : \n", bias)

 # hitung matrik h_train dengan fungsi aktivasi jenis sigmoid
 # h_train = fn sigmoid(np.dot(Xtrain_merge_s, np.transpose(bobot_input)) + bias*np.ones((Ntrain,1)))

 h_train = fn.sigmoid(np.dot(full_conn_onehot_Xtrain_merge_s, np.transpose(bobot_input)) + bias*np.ones((Ntrain,1)))
 # h_train = fn_tanh(np.dot(full_conn_onehot_Xtrain_merge_s, np.transpose(bobot_input)) + bias*np.ones((Ntrain,1)))

 print('h_train: \n',h_train, '\n')
```

```
time.sleep(100)

cek matrik singular
cek_matrik = np.dot(np.transpose(h_train), h_train)
det_cek_matrik = np.linalg.det(cek_matrik)
if det_cek_matrik != 0:
 #proceed

#if np.linalg.cond(cek_matrik) < 1/sys.float_info.epsilon:
i = np.linalg.inv(cek_matrik)
is_singular_matrix = False
else:
 is_singular_matrix = True

h_plus = np.dot(np.linalg.inv(cek_matrik), np.transpose(h_train))

print("h_plus: \n", h_plus, "\n")

4. Beta_topi
output weight disebut juga sebagai Beta_topi atau bobot_output
Beta_topi = np.dot(h_plus, Ytrain)
Beta_topi = np.dot(h_plus, Ytrain_onehot_encode)

bobot_output = Beta_topi
print("Beta_topi: \n", Beta_topi, "\n")

Coba testing dengan data Xtrain
h_test = fn_sigmoid(np.dot(full_conn_onehot_Xtrain_merge_s, np.transpose(bob-
bot_input)) + bias*np.ones((Ntrain,1)))
h_test = fn_tanh(np.dot(full_conn_onehot_Xtrain_merge_s, np.transpose(bob-
bot_input)) + bias*np.ones((Ntrain,1)))

Ytrain_predict = np.dot(h_test, bobot_output)
onehot_binary_arr_Ytrain_predict,lokasi_maks_as_int_encode = y_pre-
dict_to_onehot_binary(Ytrain_predict)
arr_token_Ytrain_predict = onehot_binary_to_arr_token(onehot_bi-
nary_arr_Ytrain_predict, get_unik_term_terurut_alphabet)
print("arr_token_Ytrain_predict: \n", arr_token_Ytrain_predict, "\n")

membuat name_unik2save utk simpan hasil
name_unik2save = str(datetime.datetime.now().astimezone(pytz.timezone('Asia/Jakar-
ta'))).strftime('%d-%m-%Y-%H-%M-%S')
hasil_txt_Ytrain_predict = arr_token_to_txt('filename_Hasil-
Train_'+name_unik2save+'.txt',arr_token_Ytrain_predict)

simpan bobot_input, bias dan bobot_output
set info param
af mewakili parameter activation function
hd mewakili parameter jumlah_hidden
fi mewakili jumlah fitur input yg diset
fif mewakili jumlah fitur input full connected
ft mewakili jumlah fitur target
elm_tr_predict (etrp) memakili label kegunaan ELM-nya

af= 'sigmoid'
af= 'tanh'

info_param = '-Ntrain-'+str(Ntrain)+'-Ntest-'+str(Ntest)+'-af-'+af+'-hd-
'+str(m)+'-r-'+str(r)+'-fi-'+str(n)+'-fif-' \
+str((n+r)*len_unik_term)+'-ft-'+str(len_unik_term)

nama_path_hasil = './HasilTrainNTTest/'+name_unik2save+'/'
nama_path_hasil = './HasilTrainNTTest/'+name_unik2save
nama_file_csv_bobot_input = nama_path_hasil+info_param+'_bobot_input.csv'
nama_file_csv_bias = nama_path_hasil+info_param+'_bias.csv'
nama_file_csv_bobot_output = nama_path_hasil+info_param+'_bobot_output.csv'

pd.DataFrame(bobot_input).to_csv(nama_file_csv_bobot_input, header=None, in-
dex=None)
pd.DataFrame(bias).to_csv(nama_file_csv_bias, header=None, index=None)
pd.DataFrame(bobot_output).to_csv(nama_file_csv_bobot_output, header=None, in-
dex=None)
```

```
Cara baca file csv diatas
baca_bobot_input = pd.read_csv(nama_file_csv_bobot_input, header=None).values
print('baca_bobot_input: \n', baca_bobot_input, '\n')

baca_bias = pd.read_csv(nama_file_csv_bias, header=None).values.flat[:]
print('baca bias: \n', baca_bias, '\n')

baca_bobot_output = pd.read_csv(nama_file_csv_bobot_output, header=None).values
print('baca_bobot_output: \n', baca_bobot_output, '\n')

print("hasil_txt_Ytrain_predict:")
print(hasil_txt_Ytrain_predict, "\n")

Hitung MAPE dari semua data train sebagai data testing
aktual = Ytrain_onehot_encode
predict = onehot_binary_arr_Ytrain_predict # Ytest_predict dlm bentuk biner (onehot encode)
mape = get_MAPE_0_100(aktual,predict)
print('MAPE dgn Ytest_predict dari Ytrain_predict dlm bentuk biner (onehot encode) = ', mape.round(2),'%')
predict = Ytrain_predict # Ytest_predict dlm bentuk desimal (non onehot encode)
mape = get_MAPE_0_100(aktual,predict) # => nilai ini nantinya bisa menjadi nilai fitness saat RNN base ELM tersebut dioptimasi dengan PSO/GA
print('MAPE dgn Ytest_predict dari Ytrain_predict dlm bentuk desimal (non onehot encode) = ', mape.round(2),'%') # lebih rekom untuk digunakan karena Ytest_predict dlm bentuk desimal

proses testing
#
1. load Xtest, bobot_input, bias dan bobot_output
#
2. get matrik delay data test (s_tr test)
print("\n s_tr_test: \n", s_tr_test)

3. hitung matrik h_test dengan fungsi aktivasi jenis sigmoid
Xtest_merge_s = np.hstack((Xtest_feature,s_tr_test))
onehot_Xtest_merge_s = fn_onehot_encoding_multi_d(Xtest_merge_s, len_unik_term)
print('onehot_Xtest_merge_s: \n', onehot_Xtest_merge_s, '\n')
print('onehot_Xtest_merge_s.shape: \n', onehot_Xtest_merge_s.shape, '\n')

full connected, yaitu join semua fitur di dalam onehot_Xtest_merge_s menjadi array dgn ukuran [Ntest x (n+r)*len_unik_term]
dimana setiap datanya yang awalnya berukuran [(n+r) x len_unik_term] menjadi berukuran [1 x (n+r)*len_unik_term]
full_conn_onehot_Xtest_merge_s = np.zeros((Ntest,(n+r)*len_unik_term))
full_conn_onehot_Xtest_merge_s = onehot_Xtest_merge_s.reshape(Ntest,(n+r)*len_unik_term)
print('full_conn_onehot_Xtest_merge_s: \n', full_conn_onehot_Xtest_merge_s, '\n')
print('full_conn_onehot_Xtest_merge_s.shape: \n', full_conn_onehot_Xtest_merge_s.shape, '\n')

h_test = fn_sigmoid(np.dot(full_conn_onehot_Xtest_merge_s, np.transpose(bobot_input)) + bias*np.ones((Ntest,1)))
print('h_test: \n', h_test, '\n')

4. Coba testing dengan data Xtest
Ytest_predict = np.dot(h_test, bobot_output)
print("\n Ytest_predict: \n", Ytest_predict)

onehot_binary_arr_Ytest_predict,lokasi_maks_as_int_encode = y_predict_to_onehot_binary(Ytest_predict)
arr_token_Ytest_predict = onehot_binary_to_arr_token(onehot_binary_arr_Ytest_predict,get_unik_term_terurut_alphabet)
print("arr_token_Ytest_predict: \n", arr_token_Ytest_predict, "\n")

membuat nama_unik2save utk simpan hasil
name_unik2save = str(datetime.datetime.now().astimezone(pytz.timezone('Asia/Jakarta'))).strftime('%d-%m-%Y-%H-%M-%S'))
hasil_txt_Ytest_predict = arr_token_to_txt('filename_HasilTest '+name_unik2save+'.txt',arr_token_Ytest_predict)
print("hasil_txt_Ytest_predict:")
print(hasil_txt_Ytest_predict, "\n")

Hitung MAPE dari semua data testing (meskipun bisa jadi sebelumnya diskenariokan, semua data testing merupakan data training atau tidak)
aktual = Ytest_onehot_encode
```

```
predict = onehot binary arr_Ytest_predict # Ytest_predict dlm bentuk biner (onehot encode)
mape = get_MAPE_0_100(aktual,predict)
print('MAPE dgn Ytest_predict dlm bentuk biner (onehot encode) = ', mape.round(2), '%')
predict = Ytest_predict # Ytest_predict dlm bentuk desimal (non onehot encode)
mape = get_MAPE_0_100(aktual,predict) # => nilai ini nantinya bisa menjadi nilai i fitness saat RNN base ELM tersebut dioptimasi dengan PSO/GA
print('MAPE dgn Ytest_predict dlm bentuk desimal (non onehot encode) = ', mape.round(2), '%') # lebih rekom untuk digunakan karena Ytest_predict dlm bentuk desimal
```

## Output:

```
full_conn_onehot_Xtrain_merge_s:
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]]
```

```
h_train:
[[0.66268164 0.83015958 0.74615468 ... 0.66838065 0.74541521 0.77537665]
 [0.54909622 0.85694922 0.70240693 ... 0.42032676 0.45769713 0.61473189]
 [0.71109129 0.52103967 0.74744062 ... 0.3926364 0.58967845 0.39416383]
 ...
 [0.68862082 0.7548988 0.70719264 ... 0.76313001 0.1549662 0.20831149]
 [0.55300541 0.95608152 0.83521703 ... 0.37466593 0.88050275 0.83731218]
 [0.68771944 0.84587013 0.69482662 ... 0.47551714 0.34822225 0.71236348]]
```

```
h_plus:
[[0.14153418 -0.33325338 0.07132244 ... -0.0510968 0.04283721
 -0.01438937]
 [0.07837094 0.16858436 0.12787906 ... -0.02267887 0.04233356
 0.15443911]
 [-0.05030069 0.40926252 -0.13427298 ... -0.00432646 0.15961595
 0.13125652]
 ...
 [0.09053341 0.05826373 -0.1755814 ... -0.03448717 0.05324
 0.10434712]
 [0.31808055 0.18289536 -0.0266534 ... -0.07587025 -0.05727513
 -0.10173059]
 [-0.044664 -0.25724175 -0.08847894 ... -0.11545714 -0.13484063
 -0.05189649]]
```

```
Beta_topi:
[[0.63226176 0.1624548 -0.0849652 ... 0.05321409 0.02727835
 -0.06778946]
 [-0.17101841 0.1672491 -0.04077286 ... -0.13575277 0.01968845
 0.38868342]
 [-0.07122763 -0.05145998 -0.32647438 ... -0.14515881 0.15858811
 -0.15468656]
 ...
 [0.21193842 -0.00940845 -0.22307239 ... 0.10629986 0.29412993
 -0.02062864]
 [-0.29241284 0.15196209 -0.24251531 ... -0.07898043 -0.25063983
 0.07193468]
 [-0.01689059 -0.1236735 0.09779437 ... -0.13856821 -0.12525153
 -0.49262197]]
```

```
hasil_txt_Ytrain_predict:
MAPE dgn Ytest_predict dari Ytrain_predict dlm bentuk biner (onehot encode) = 0.15 %
MAPE dgn Ytest_predict dari Ytrain_predict dlm bentuk desimal (non onehot encode) = 95.54 %
full_conn_onehot_Xtest_merge_s:
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]]
```

```
h_test:
[[0.66268164 0.83015958 0.74615468 ... 0.66838065 0.74541521 0.77537665]
[0.54909622 0.85694922 0.70240693 ... 0.42032676 0.45769713 0.61473189]
[0.71109129 0.52103967 0.74744062 ... 0.3926364 0.58967845 0.39416383]
...
[0.68862082 0.7548988 0.70719264 ... 0.76313001 0.1549662 0.20831149]
[0.55300541 0.95608152 0.83521703 ... 0.37466593 0.88050275 0.83731218]
[0.68771944 0.84587013 0.69482662 ... 0.47551714 0.34822225 0.71236348]]

hasil_txt_Ytest_predict:
hidup
kita
adalah
menjadi
bahagia
dan
selamat
.. (ini penanda new_line)
siapapun
adalah
menjadi
terbaik
untuk
segala
kesulitan
.. (ini penanda new_line)
siapapun
bisa
jadi
apapun
end_of_file
tujuan
kita
adalah
menjadi
bahagia
dan
selamat
.. (ini penanda new_line)
kesabaran
adalah
obat
terbaik
untuk
segala
kesulitan
.. (ini penanda new_line)
siapapun
bisa
jadi
apapun
end_of_file
tujuan
hidup
adalah
menjadi
bahagia
dan
selamat
.. (ini penanda new_line)
kesabaran
adalah
obat
terbaik
untuk
segala
kesulitan
.. (ini penanda new_line)
siapapun
bisa
jadi
apapun
end_of_file
tujuan
hidup
kita
hidup
kita
adalah
menjadi
bahagia
```

```
dan
selamat
.. (ini penanda new_line)
kesabaran
adalah
obat
terbaik
untuk
segala
kesulitan
.. (ini penanda new_line)
siapapun
bisa
jadi
apapun
end_of_file
tujuan
kita
adalah
menjadi
bahagia
dan
selamat
.. (ini penanda new_line)
kesabaran
adalah
obat
terbaik
untuk
segala
kesulitan
.. (ini penanda new_line)
siapapun
bisa
jadi
apapun
end_of_file
tujuan
hidup
adalah
menjadi
bahagia
dan
selamat
.. (ini penanda new_line)
kesabaran
adalah
obat
terbaik
untuk
segala
kesulitan
.. (ini penanda new_line)
siapapun
bisa
jadi
apapun
end_of_file
tujuan
hidup
kita
```

```
MAPE dgn Ytest_predict dlm bentuk biner (onehot encode) = 0.15 %
MAPE dgn Ytest_predict dlm bentuk desimal (non onehot encode) = 95.54 %
```

### #Koding MLSTM (part. 7 of 8)

```
Tanpa load file bobot_input, bias dan bobot_output, tetapi menggunakan yg masih ada di memory
mencoba melakukan test melalui kata,
dimana kata yang dimasukkan minimal 1 kata, maksimal sebanyak n,
n merupakan banyak fitur yang digunakan saat proses training RNNELM
#
jika kata yang dimasukkan lebih dari n, maka diambil n kata terakhir

raw_data_Xtest_by_token = 'kita'
raw_data_Xtest_by_token = 'bahagia'
raw_data_Xtest_by_token = 'hidup'
raw_data_Xtest_by_token = 'hidup bahagia'
raw_data_Xtest_by_token = 'hidup bahagia'
raw_data_Xtest_by_token = 'tujuan hidup adalah bahagia'
raw_data_Xtest_by_token = 'tujuan hidup bahagia'

#
raw_data_Xtest_by_token = '''hidup
bahagia''' # jika ada kata yg tidak ada dalam unik term, maka dianggap "new line"
misal kata "dan", dengan koding tambahan berupa
np.where(get_unik_term_terurut_alphabet==token_in)[0][0] \
if np.where(get_unik_term_terurut_alphabet==token_in)[0].size \
else np.where(get_unik_term_terurut_alphabet=='new_line')[0][0]

ubah raw_data_Xtest_by_token menjadi onehot_encode
unraw_raw_data_Xtest_by_token = []
unraw_raw_data_Xtest_by_token.extend(raw_data_Xtest_by_token.strip().replace('\n', 'new line').split(' '))
unraw_raw_data_Xtest_by_token = [name.lower() for name in unraw_raw_data_Xtest_by_token]
arr_token_unraw_data_Xtest_by_token = np.array(unraw_raw_data_Xtest_by_token)
if(arr_token_unraw_data_Xtest_by_token.size==n):
 dataInput_Xtest_by_token = arr_token_unraw_data_Xtest_by_token[-n:] # diam-bil n kata terakhir
 print("dataInput_Xtest_by_token awal: \n", arr_token_unraw_data_Xtest_by_token, '\n')
 print("dataInput_Xtest_by_token diambil", str(n), 'kata terakhir: \n', dataInput_Xtest_by_token, '\n')
else:
 dataInput_Xtest_by_token = arr_token_unraw_data_Xtest_by_token
 print("dataInput_Xtest_by_token: \n", dataInput_Xtest_by_token, '\n')

get nilai kode integer
integer_encoding_dataInput_Xtest_by_token, len_sekuens_dataInput_Xtest_by_token = np.array([np.where(get_unik_term_terurut_alphabet==token_in)[0][0] \
 if np.where(get_unik_term_terurut_alphabet==token_in)[0].size \
 else np.where(get_unik_term_terurut_alphabet=='new_line')[0][0] \
 for token_in in dataInput_Xtest_by_token]), dataInput_Xtest_by_token.shape[0]
print('integer encoding dataInput_Xtest_by_token: \n', integer_encoding_dataInput_Xtest_by_token, '\n')
print('len_sekuens_dataInput_Xtest_by_token: \n', len_sekuens_dataInput_Xtest_by_token, '\n')

mengubah nilai kode integer menjadi one hot encoding berupa vektor biner, ke depannya dapat dikembangkan menggunakan probabilistic encoding
onehot_integer_encoding_dataInput_Xtest_by_token = fn_onehot_encoding_1d(integer_encoding_dataInput_Xtest_by_token, len_unik_term)

#
untuk extend onehot encode jika masuk kondisi berikut, dgn hasil np.vstack(([integer_encode], ..., [full_zeros]))
if(len_sekuens_dataInput_Xtest_by_token<n):
len_extend_onehot_encode = np.abs(len_sekuens_dataInput_Xtest_by_token-n)
for i in range(len_extend_onehot_encode):
onehot_integer_encoding_dataInput_Xtest_by_token = np.vstack((onehot_integer_encoding_dataInput_Xtest_by_token,np.zeros((1,len_unik_term))))
#
untuk extend onehot encode jika masuk kondisi berikut, dgn hasil np.vstack(([full_zeros], ..., [integer_encode]))
```

```
if(len_sekuens_dataInput_Xtest_by_token<n):
 len_extend_onehot_encode = np.abs(len_sekuens_dataInput_Xtest_by_token-n)
 for i in range(len_extend_onehot_encode):
 onehot_integer_encoding_dataInput_Xtest_by_token = np.vstack((np.zeros((1,len_unik_term)),onehot_integer_encoding_dataInput_Xtest_by_token))

print('onehot integer encoding dataInput_Xtest_by_token: \n', onehot_integer_encoding_dataInput_Xtest_by_token,'\\n')
Ntest_Xtest_by_token = 1

Jika ingin memprediksi sampai i_k sekuens
i_k = 27

raw_data_Xtest_by_token_interator = raw_data_Xtest_by_token
recurrent_delay = None
for i in range(i_k):
 print('i = ', i)
 unraw_raw_data_Xtest_by_token = []
 unraw_raw_data_Xtest_by_token.extend(raw_data_Xtest_by_token_interator.strip().replace('\\n','\\n line ').split(' '))
 unraw_raw_data_Xtest_by_token = [name.lower() for name in unraw_raw_data_Xtest_by_token]
 arr_token_unraw_data_Xtest_by_token = np.array(unraw_raw_data_Xtest_by_token)
 if(arr_token_unraw_data_Xtest_by_token.size==n):
 dataInput_Xtest_by_token = arr_token_unraw_data_Xtest_by_token[-n:] # diambil n kata terakhir
 print('dataInput_Xtest_by_token awal: \\n', arr_token_unraw_data_Xtest_by_token,'\\n')
 print('dataInput_Xtest_by_token diambil', str(n),'kata terakhir: \\n', dataInput_Xtest_by_token,'\\n')
 else:
 dataInput_Xtest_by_token = arr_token_unraw_data_Xtest_by_token
 print('dataInput_Xtest_by_token: \\n', dataInput_Xtest_by_token,'\\n')

 # get nilai kode integer
 integer_encoding_dataInput_Xtest_by_token,len_sekuens_dataInput_Xtest_by_token = np.array([np.where(get_unik_term_terurut_alphabet==token_in)[0][0] \
 if np.where(get_unik_term_terurut_alphabet==token_in)[0].size \
 else np.where(get_unik_term_terurut_alphabet=='new_line')[0][0] \
 for token_in in dataInput_Xtest_by_token]),dataInput_Xtest_by_token.shape[0]
 print('integer encoding dataInput_Xtest_by_token: \\n', integer_encoding_dataInput_Xtest_by_token,'\\n')
 print('len_sekuens_dataInput_Xtest_by_token: \\n', len_sekuens_dataInput_Xtest_by_token,'\\n')

 # if(i==0):
 # Proses penentuan Ytest_predict
 # hitung matrik h_test dengan fungsi aktivasi jenis sigmoid
 #
 # pada i=0, karena raw data Xtest_by_token ini tidak dalam dataset, maka matrik delay atau s_tr_test, semua nilainya diset nol
 # dan pada saat i > 0,
 #
 # if(i==0):
 # s_tr_test_Xtest_by_token = -1*np.ones((1,r)) # dikali -1, agar nanti ketika di fn_onehot_encoding_multi_d nilai fiturnya menjadi nol semua, yaitu tidak termasuk semua integer token apa-paun, kecuali jika nanti mengembangkan penggunaan type data spiral pada dataset
 # recurrent_delay = s_tr_test_Xtest_by_token[0,:-1]
 # else:
 # # rl =
 # s_tr_test_Xtest_by_token = np.zeros((1,r))
 # s_tr_test_Xtest_by_token[0,0] = lokasi_maks_as_int_encode
 # s_tr_test_Xtest_by_token[0,1:] = recurrent_delay
 # recurrent_delay = s_tr_test_Xtest_by_token[0,:-1]
 # print('s_tr_test_Xtest_by_token: \\n',s_tr_test_Xtest_by_token,'\\n')
 # print('recurrent_delay: \\n',recurrent_delay,'\\n')

 # untuk MLSTM
 if(arr_token_unraw_data_Xtest_by_token.size < n+r):
 s_tr_test_Xtest_by_token = -1*np.ones((1,r)) # dikali -1, agar nanti ketika di fn_onehot_encoding_multi_d nilai fiturnya menjadi nol semua, yaitu tidak termasuk semua integer token apa-paun, kecuali jika nanti mengembangkan penggunaan type data spiral pada dataset
```

```
if(arr_token_unraw_data_Xtest_by_token.size > n):
 s_tr_test_Xtest_by_token_temp = np.zeros((1,r))
 s_tr_test_Xtest_by_token_temp[0,0] = lokasi_maks_as_int_encode
 s_tr_test_Xtest_by_token_temp[0,1:] = recurrent_delay
 recurrent_delay = s_tr_test_Xtest_by_token_temp[0,:-1]
else:
 s_tr_test_Xtest_by_token = np.zeros((1,r))
 s_tr_test_Xtest_by_token[0,0] = lokasi_maks_as_int_encode
 s_tr_test_Xtest_by_token[0,1:] = recurrent_delay
 recurrent_delay = s_tr_test_Xtest_by_token[0,:-1]
print('s_tr_test_Xtest_by_token: \n',s_tr_test_Xtest_by_token, '\n')
print('recurrent_delay: \n',recurrent_delay, '\n')

#
if(arr_token_unraw_data_Xtest_by_token.size>=n):
 Xtest_feature_Xtest_by_token = integer_encoding_dataInput_Xtest_by_token[-n:]
 Xtest_feature_Xtest_by_token = Xtest_feature_Xtest_by_token[None,...]
else:
 # Xtest feature Xtest by token = np.hstack((integer encoding dataIn-
put_Xtest_by_token[None,...],np.zeros((1,np.abs(len_sekvens_dataIn-
put_Xtest_by_token-n)))))

 Xtest_feature_Xtest_by_token = np.hstack((-1*np.ones((1,np.abs(len_sek-
vens_dataInput_Xtest_by_token-n))),integer_encoding_dataInput_Xtest_by_to-
ken[None,...])))

 Xtest_merge_s_Xtest_by_token = np.hstack((Xtest_feature_Xtest_by_to-
ken,s_tr_test_Xtest_by_token))

 onehot_Xtest_merge_s_Xtest_by_token = fn_onehot_encod-
ing_multi_d(Xtest_merge_s_Xtest_by_token, len_unik_term)
 # print('onehot_Xtest_merge_s: \n', onehot_Xtest_merge_s, '\n')
 # print('onehot_Xtest_merge_s.shape: \n', onehot_Xtest_merge_s.shape, '\n')

 # full connected, yaitu join semua fitur di dalam onehot_Xtest_merge_s men-
jadi array dgn ukuran [Ntest x (n+r)*len_unik_term]
 # dimana setiap datanya yang awalnya berukuran [(n+r) x len_unik_term] men-
jadi berukuran [1 x (n+r)*len_unik_term]
 full_conn_onehot_Xtest_merge_s_Xtest_by_to-
ken = np.zeros((Ntest_Xtest_by_to-
ken,(n+r)*len_unik_term))
 full_conn_onehot_Xtest_merge_s_Xtest_by_to-
ken = onehot_Xtest_merge_s_Xtest_by_token.reshape(Ntest_Xtest_by_to-
ken,(n+r)*len_unik_term)
 print('full_conn_onehot_Xtest_merge_s_Xtest_by_to-
ken: \n', full_conn_onehot_Xtest_merge_s_Xtest_by_token, '\n')
 # print('full_conn_onehot_Xtest_merge_s_Xtest_by_to-
ken.shape: \n', full_conn_onehot_Xtest_merge_s_Xtest_by_to-
ken.shape, '\n')

 h_test_Xtest_by_token = fn_sig-
moid(np.dot(full_conn_onehot_Xtest_merge_s_Xtest_by_token, np.transpose(bob-
bot_input)) + bias*np.ones((Ntest_Xtest_by_token,1)))
 print('h_test_Xtest_by_token: \n',h_test_Xtest_by_token, '\n')

4. Coba testing dengan data Xtest
Ytest_predict_Xtest_by_token = np.dot(h_test_Xtest_by_token, bobot_output)
print("\n Ytest_predict: \n", Ytest_predict)

onehot_binary_arr_Ytest_predict, lokasi_maks_as_int_encode = y_pre-
dict_to_onehot_binary(Ytest_predict_Xtest_by_token)
arr_token_Ytest_predict = onehot_binary_to_arr_token(onehot_bi-
nary_arr_Ytest_predict,get_unik_term_terurut_alphabet)
print("arr_token Ytest predict: \n", arr_token_Ytest_predict, "\n")

raw_data_Xtest_by_token_interator += ' '+ arr_token_Ytest_predict[0]
print("raw_data_Xtest_by_token_interator:")
print(raw_data_Xtest_by_token_interator, "\n")

if(i==0):
 arr_token_Ytest_predict_iterator = np.vstack((arr_token_un-
raw_data_Xtest_by_token[... ,None],arr_token_Ytest_predict))
else:
 arr_token_Ytest_predict_iterator = np.vstack((arr_token_Ytest_predict_iterator,
arr_token_Ytest_predict))

else:

if(hasil_txt_Ytest_predict==".. (ini penanda new_line)"):
print()
else:
print(hasil_txt_Ytest_predict)
```

```
membuat name_unik2save utk simpan hasil
name_unik2save = str(datetime.today().astimezone(pytz.timezone('Asia/Jakarta')).strftime('%d-%m-%Y-%H-%M-%S'))
hasil_txt_Ytest_predict_sastraa = arr_token_to_txt_sastraa('file-name_HasilTest_'+name_unik2save+'.txt',arr_token_Ytest_predict_iterator.flaten())
print("hasil_txt_Ytest_predict_sastraa:")
print(hasil_txt_Ytest_predict_sastraa, "\n")

menghitung nilai BLUE Score untuk hasil testing
untuk proses training bobot_input,bias, dan bobot_out-
put utk digunakan pada penghitungan Ytest_predict dalam bentuk onehot en-
code, dapat menggunakan MAPE atau BLUE Score,
tetapi lebih direkomendasikan menggunakan MAPE, karena bentuk dari Ytest_predict dalam bentuk onehot enco
de
misal diketahui berikut
original_reference_atau_data_aktual = "tujuan hidup baha-
gia new_line dan selamat dunia akhirat"
original_reference_atau_data_aktual = "bahagia hidup kita adalah menjadi ba-
hagia"
original_reference_atau_data_aktual = "bahagia hidup kita kita menjadi baha-
gia"
original_reference_atau_data_aktual = '''hidup adalah kita menjadi baha-
gia kita obat terbaik untuk segala kesulitan
siapapun bisa jadi apapun obat hidup menjadi kita menjadi hidup'''

original_reference_atau_data_aktual = '''kita adalah menjadi baha-
gia dan selamat
kesabaran adalah obat terbaik untuk segala kesulitan
siapapun bisa jadi apapun end of file tujuan hidup kita adalah menjadi ba-
hagia'''

candidate_atau_hasil_predict_atau_hasil_sis-
tem = ' '.join(str(term) for term in arr_token_Ytest_predict_iterator.flat-
en())
print('original_reference_atau_data_aktual:')
print(original_reference_atau_data_aktual,'n')

Bilingual Evaluation Understudy Score (BLUE) Score,
Skor BLUE adalah metrik untuk mengukur hasil kinerja atau mengevaluasi kalima-
t yang dihasilkan oleh sistem terhadap kalimat aktual yang dibuat menurut per-
sepsi manusia atau pakar.
original_reference_atau_data_aktual = ' '.join(str(term) for term in origi-
nal_reference_atau_data_aktual.replace("\n"," new line ").split(" "))
print('Hasil Skor BLUE dari Kalimat Lengkap (raw_data_Xtest_by_token + Kalimat
hasil prediksi): ','{0:.6f}'.format(fn_bleu_score(original_refer-
ence_atau_data_aktual, candidate_atau_hasil_predict_atau_hasil_sistem)),"n")

original_reference_atau_data_aktual = ' '.join(str(term) for term in origi-
nal_reference_atau_data_aktual.split(" ")[-i_k:])
original_reference_atau_data_aktual = ' '.join(str(term) for term in origi-
nal_reference_atau_data_aktual.replace("\n"," new line ").split(" ")[-i_k:])
print('original_reference_atau_data_aktual sepanjang Kalimat hasil prediksi: ')
print(original_reference_atau_data_aktual.replace(" new_line ","\n"),"\n")

candidate_atau_hasil_predict_atau_hasil_sis-
tem = ' '.join(str(term) for term in arr_token_Ytest_predict_iterator[-
i_k:].flatlen())
print('candidate_atau_hasil_predict_atau_hasil_sistem: ')
print(candidate_atau_hasil_predict_atau_hasil_sistem.re-
place(" new line ","\n"),"\n")
print('Hasil Skor BLUE dari non Kalimat Lengkap (Kalimat hasil prediksi): ','{0
:.6f}'.format(fn_bleu_score(original_reference_atau_data_aktual, candi-
date_atau_hasil_predict_atau_hasil_sistem)))
```

## Output:





```
[15.]
full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[0. 0. 1. 0.
1. 0.
0. 0. 0. 0. 0. 0. 0. 1. 0.
0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0.]]

h_test_Xtest_by_token:
[[0.54815511 0.36985165 0.75987881 0.30763886 0.77911665 0.57608706
0.62000017 0.14032645 0.4785111 0.44533885 0.71133778 0.91283362
0.78052473 0.69291471 0.42662622 0.69510223 0.49664972 0.68949935
0.69586474 0.56722229 0.66012838 0.63458909 0.41847595 0.39868649
0.72827706 0.65682225 0.69867244 0.19217429 0.79944645 0.77455692
0.78813688 0.52787386 0.92011014 0.85550881 0.15747087 0.81326206
0.58356701 0.15055266 0.80050266 0.69140172 0.98384933 0.67048314
0.51781923 0.6758519 0.82986326 0.66749338 0.42723793 0.42506126
0.938824 0.53076417 0.96430612 0.79180394 0.5285717 0.82760632
0.71314156 0.64879834 0.02785872 0.98079501 0.90188279 0.38635824
0.93669951 0.44393118 0.83238987 0.68703172 0.53141172 0.84545243
0.88773457]]

raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line

i = 6
dataInput_Xtest_by_token awal:
['kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat' 'new_line']

dataInput_Xtest_by_token diambil 3 kata terakhir:
['dan' 'selamat' 'new_line']

integer_encoding_dataInput_Xtest_by_token:
[4 15 12]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[12. 15.]]

recurrent_delay:
[12.]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[0. 0. 0. 1. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 1. 0.
1. 0. 1.
0. 0. 0. 0. 0.]]

h_test_Xtest_by_token:
[[0.75744343 0.93068501 0.82495056 0.61233684 0.2141768 0.68438066
0.49427468 0.09186866 0.09445392 0.15273989 0.41653021 0.65506176
0.81537909 0.98450338 0.97713853 0.65594254 0.06792166 0.74337912
0.8013478 0.29413451 0.55201634 0.10584405 0.59146792 0.5968457
0.40288962 0.36992158 0.76144464 0.41284049 0.44129116 0.43375999
0.61837303 0.34369642 0.35231909 0.82707773 0.49329973 0.192485
0.85544699 0.45409999 0.53143663 0.62790795 0.72738555 0.30870498
0.78637814 0.80972816 0.80565837 0.57782018 0.82878741 0.20082628
0.48370414 0.19279609 0.7131211 0.97279196 0.59581049 0.24458345
0.66669381 0.29675754 0.07364146 0.2780218 0.29226921 0.6306144
0.60397183 0.62486532 0.60294275 0.82510937 0.98658704 0.52128925
0.89609929]]

raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line kesabaran

i = 7
dataInput_Xtest_by_token awal:
['kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat' 'new_line'
'kesabaran']

dataInput_Xtest_by_token diambil 3 kata terakhir:
['selamat' 'new_line' 'kesabaran']

integer_encoding_dataInput_Xtest_by_token:
[15 12 8]

len sekuens dataInput Xtest by token:
```



















```
integer_encoding_dataInput_Xtest_by_token:
[6 10 0]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[0. 10.]]

recurrent_delay:
[10.]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[0. 0. 0. 0. 0. 0. 1. 0.
0. 0. 0. 0. 0. 0. 1. 0.
0.
0.
0. 0. 0. 0. 0.]]

h_test_Xtest_by_token:
[[0.73643147 0.38963544 0.40248068 0.45995693 0.35786139 0.95839396
0.63444994 0.72760364 0.17955697 0.56437223 0.72744504 0.30115915
0.55824029 0.75201928 0.69497505 0.77777126 0.79257775 0.81172319
0.68167414 0.65374991 0.21739228 0.78081584 0.62388019 0.94116476
0.45199352 0.34584271 0.83593152 0.80642451 0.87495454 0.91416126
0.58943645 0.81112638 0.31646245 0.21241472 0.61823141 0.6911029
0.63308037 0.40180743 0.37741141 0.49110005 0.53725953 0.84606561
0.87994392 0.70623911 0.68238986 0.82746358 0.25872513 0.31197994
0.46953473 0.6408506 0.85079806 0.44746435 0.87130998 0.69041928
0.73068481 0.88593802 0.60306246 0.63502961 0.91679222 0.18985616
0.46744287 0.47362187 0.28094196 0.54909622 0.78649572 0.73080288
0.5337022]]

raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line kesabaran adalah obat terbaik
untuk segala kesulitan new_line siapapun bisa jadi apapun end_of_file tujuan
hidup kita adalah menjadi

i = 24
dataInput_Xtest_by_token awal:
['kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat' 'new_line'
'kesabaran' 'adalah' 'obat' 'terbaik' 'untuk' 'segala' 'kesulitan'
'new_line' 'siapapun' 'bisa' 'jadi' 'apapun' 'end_of_file' 'tujuan'
'hidup' 'kita' 'adalah' 'menjadi']

dataInput_Xtest_by_token diambil 3 kata terakhir:
['kita' 'adalah' 'menjadi']

integer_encoding_dataInput_Xtest_by_token:
[10 0 11]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[11. 0.]]

recurrent_delay:
[11.]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
0. 0. 0. 0. 1. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0.]]

h_test_Xtest_by_token:
[[0.09421468 0.90734402 0.94059734 0.87565315 0.84745414 0.69103885
0.16218438 0.36071615 0.92619463 0.33381753 0.77429486 0.36884995
0.63201108 0.52540564 0.85333187 0.70196773 0.12481586 0.74395106
0.33192976 0.25259714 0.50275025 0.15767001 0.87122024 0.70022822
0.32654083 0.8394667 0.29554843 0.32905302 0.51207386 0.30204392
0.78420288 0.78233535 0.66593778 0.73164806 0.35620838 0.81314052
0.93125061 0.59511185 0.75283899 0.39679391 0.9287971 0.72191797
0.50337529 0.34956267 0.92928499 0.6563487 0.55016073 0.87004878
0.21535741 0.44832993 0.51572139 0.49317474 0.34238918 0.82797699
0.21589869 0.87398846 0.53723467 0.59412346 0.95076191 0.35827517
0.44121719 0.70663318 0.75797588 0.65127821 0.71088579 0.68414298
0.7945438]]
```

```
raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line kesabaran adalah obat terbaik
untuk segala kesulitan new_line siapapun bisa jadi apapun end_of_file tujuan
hidup kita adalah menjadi bahagia

i = 25
dataInput_Xtest_by_token awal:
['kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat' 'new_line'
'kesabaran' 'adalah' 'obat' 'terbaik' 'untuk' 'segala' 'kesulitan'
'new_line' 'siapapun' 'bisa' 'jadi' 'apapun' 'end_of_file' 'tujuan'
'hidup' 'kita' 'adalah' 'menjadi' 'bahagia']

dataInput_Xtest_by_token diambil 3 kata terakhir:
['adalah' 'menjadi' 'bahagia']

integer_encoding_dataInput_Xtest_by_token:
[0 11 2]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[2. 11.]]

recurrent_delay:
[2.]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[1. 0.
0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
0.
0. 0. 0. 0.]])

h_test_Xtest_by_token:
[[0.4075658 0.97489263 0.86505033 0.65184584 0.64991415 0.7808843
0.66702708 0.44743962 0.4479342 0.43749737 0.76776244 0.19290506
0.10498582 0.51172402 0.72673062 0.53972011 0.65716032 0.42474356
0.71039222 0.26831267 0.39488052 0.97118586 0.84135796 0.90788909
0.808074 0.11305526 0.81288205 0.88741522 0.85376942 0.77896171
0.17122678 0.5736184 0.16380803 0.73003487 0.30477061 0.045795
0.54582563 0.76473519 0.39600425 0.15556959 0.4455859 0.56712409
0.42274105 0.46408337 0.47110434 0.61221813 0.67422847 0.63354487
0.88399346 0.79398816 0.68817031 0.27989258 0.52642799 0.33486365
0.74896677 0.13863394 0.14168314 0.74140911 0.95790471 0.49724975
0.83343374 0.40669677 0.85003473 0.23770259 0.4309864 0.66379852
0.9765504]])

raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line kesabaran adalah obat terbaik
untuk segala kesulitan new_line siapapun bisa jadi apapun end_of_file tujuan
hidup kita adalah menjadi bahagia dan

i = 26
dataInput_Xtest_by_token awal:
['kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat' 'new_line'
'kesabaran' 'adalah' 'obat' 'terbaik' 'untuk' 'segala' 'kesulitan'
'new_line' 'siapapun' 'bisa' 'jadi' 'apapun' 'end_of_file' 'tujuan'
'hidup' 'kita' 'adalah' 'menjadi' 'bahagia' 'dan']

dataInput_Xtest_by_token diambil 3 kata terakhir:
['menjadi' 'bahagia' 'dan']

integer_encoding_dataInput_Xtest_by_token:
[11 2 4]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[4. 2.]]

recurrent_delay:
[4.]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[0.
0.
0.
0.
0. 0. 0. 0.]])
```

```
h_test_Xtest_by_token:
[[0.88713514 0.30316251 0.96852079 0.58115894 0.78277782 0.75830596
0.7779268 0.51994141 0.82369054 0.17053248 0.68750462 0.43184503
0.25744211 0.67703391 0.34392205 0.43388282 0.68084891 0.78668039
0.60850779 0.48442848 0.77949506 0.49652471 0.86520203 0.95825016
0.9489084 0.04073749 0.59134709 0.89548312 0.67795168 0.57652662
0.53325417 0.11439591 0.5824486 0.58125631 0.52892058 0.36659445
0.87953126 0.21446309 0.46577517 0.88196812 0.73586813 0.51871812
0.61877417 0.25316399 0.66551492 0.53325417 0.14907503 0.39277951
0.72485968 0.5852673 0.63073087 0.71828967 0.45184489 0.13328885
0.44902263 0.73803949 0.81468555 0.70750318 0.6030864 0.78746813
0.78014759 0.89010452 0.82969375 0.51054949 0.07500377 0.84735068
0.4491216]]

raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line kesabaran adalah obat terbaik
untuk segala kesulitan new_line siapapun bisa jadi apapun end_of_file tujuan
hidup kita adalah menjadi bahagia dan selamat

hasil_txt_Ytest_predict_sastraw:
kita adalah menjadi bahagia dan selamat
kesabaran adalah obat terbaik untuk segala kesulitan
siapapun bisa jadi apapun end_of_file tujuan hidup kita adalah menjadi bahagia
dan selamat

original_reference_atau_data_aktual:
kita adalah menjadi bahagia dan selamat
kesabaran adalah obat terbaik untuk segala kesulitan
siapapun bisa jadi apapun end_of_file tujuan hidup kita adalah menjadi bahagia

Hasil Skor BLUE dari Kalimat Lengkap (raw_data_Xtest_by_token + Kalimat hasil
prediksi): 0.730159

original_reference_atau_data_aktual sepanjang Kalimat hasil prediksi:
kita adalah menjadi bahagia dan selamat
kesabaran adalah obat terbaik untuk segala kesulitan
siapapun bisa jadi apapun end_of_file tujuan hidup kita adalah menjadi bahagia

candidate_atau_hasil_predict_atau_hasil_sistem:
adalah menjadi bahagia dan selamat
kesabaran adalah obat terbaik untuk segala kesulitan
siapapun bisa jadi apapun end_of_file tujuan hidup kita adalah menjadi bahagia
dan selamat

Hasil Skor BLUE dari non Kalimat Lengkap (Kalimat hasil prediksi): 0.720798
```

## #Koding MLSTM (part. 8 of 8)

```
Dengan load file bobot_input, bias dan bobot_output, yaitu dari yang te-
lah disimpan sebelumnya
set info param
af mewakili parameter activation function
hd mewakili parameter jumlah hidden atau (m)
dan jumlah context neuron (r)
fi mewakili jumlah fitur input yg diset
fif mewakili jumlah fitur input full connected
ft mewakili jumlah fitur target

misal memilih hasil pelatihan yang berikut
info_param = '30-12-2021-13-09-39-Ntrain-132-Ntest-132-af-sigmoid-hd-67-r-2-fi-
3-fif-100-ft-20'

nama_path_hasil = './HasilTrainNTest/'
nama_file_csv_bobot_input = nama_path_hasil+info_param+'_bobot_input.csv'
nama_file_csv_bias = nama_path_hasil+info_param+'_bias.csv'
nama_file_bobot_output = nama_path_hasil+info_param+'_bobot_output.csv'

pd.DataFrame(bobot_input).to_csv(nama_file_csv_bobot_input, header=None, index=None)
pd.DataFrame(bias).to_csv(nama_file_csv_bias, header=None, index=None)
```

```
pd.DataFrame(bobot_output).to_csv(nama_file_csv_bobot_output, header=None, index=None)

Cara baca file csv diatas
baca_bobot_input = pd.read_csv(nama_file_csv_bobot_input, header=None).values
print('baca_bobot_input: \n', baca_bobot_input, '\n')

baca_bias = pd.read_csv(nama_file_csv_bias, header=None).values.flat[:]
print('baca bias: \n', baca_bias, '\n')

baca_bobot_output = pd.read_csv(nama_file_csv_bobot_output, header=None).values
print('baca_bobot_output: \n', baca_bobot_output, '\n')

mencoba melakukan test melalui kata,
dimana kata yang dimasukkan minimal 1 kata, maksimal sebanyak n,
n merupakan banyak fitur yang digunakan saat proses training RNNELM
#
jika kata yang dimasukkan lebih dari n, maka diambil n kata terakhir

raw_data_Xtest_by_token = 'kita'
raw data Xtest_by_token = 'bahagia'
raw_data_Xtest_by_token = 'hidup'
raw_data_Xtest_by_token = 'hidup bahagia'
raw_data_Xtest_by_token = 'hidup bahagia'
raw_data_Xtest_by_token = 'tujuan hidup adalah bahagia'
raw_data_Xtest_by_token = 'tujuan hidup bahagia'

raw data Xtest_by_token = '''hidup
bahagia''' # jika ada kata yg tidak ada dalam unik term, maka dianggap "new line"
misal kata "dan", dengan koding tambahan berupa
np.where(get_unik_term_terurut_alphabet==token_in)[0][0] \
if np.where(get_unik_term_terurut_alphabet==token_in)[0].size \
else np.where(get_unik_term_terurut_alphabet=='new_line')[0][0]

ubah raw data Xtest_by_token menjadi onehot_encode
unraw_raw_data_Xtest_by_token = []
unraw_raw_data_Xtest_by_token.extend(raw_data_Xtest_by_token.strip().replace('\n', 'new line').split(' '))
unraw_raw_data_Xtest_by_token = [name.lower() for name in unraw_raw_data_Xtest_by_token]
arr_token_unraw_data_Xtest_by_token = np.array(unraw_raw_data_Xtest_by_token)
if(arr_token_unraw_data_Xtest_by_token.size>n):
 dataInput_Xtest_by_token = arr_token_unraw_data_Xtest_by_token[-n:] # diam-bil n kata terakhir
 print("dataInput_Xtest_by_token awal: \n", arr_token_unraw_data_Xtest_by_token, '\n')
 print("dataInput_Xtest_by_token diambil", str(n), "kata terakhir: \n", dataInput_Xtest_by_token, '\n')
else:
 dataInput_Xtest_by_token = arr_token_unraw_data_Xtest_by_token
 print("dataInput_Xtest_by_token: \n", dataInput_Xtest_by_token, '\n')

get nilai kode integer
integer_encoding_dataInput_Xtest_by_token, len_sekuens_dataInput_Xtest_by_token = np.array([np.where(get_unik_term_terurut_alphabet==token_in)[0][0] \
 if np.where(get_unik_term_terurut_alphabet==token_in)[0].size \
 else np.where(get_unik_term_terurut_alphabet=='new_line')[0][0] \
 for token_in in dataInput_Xtest_by_token]), dataInput_Xtest_by_token.shape[0]
print('integer_encoding_dataInput_Xtest_by_token: \n', integer_encoding_dataInput_Xtest_by_token, '\n')
print('len_sekuens_dataInput_Xtest_by_token: \n', len_sekuens_dataInput_Xtest_by_token, '\n')

mengubah nilai kode integer menjadi one hot encoding berupa vektor biner, ke depannya dapat dikembangkan menggunakan probabilistic encoding
onehot_integer_encoding_dataInput_Xtest_by_token = fn_onehot_encoding_1d(integer_encoding_dataInput_Xtest_by_token, len_unik_term)

untuk extend onehot encode jika masuk kondisi berikut, dgn hasil np.vstack(([integer_encode], ..., [full_zeros])):
if(len_sekuens_dataInput_Xtest_by_token<n):
len_extend_onehot_encode = np.abs(len_sekuens_dataInput_Xtest_by_token-n)
```

```
for i in range(len_extend_onehot_encode):
onehot_integer_encoding_dataInput_Xtest_by_token = np.vstack((onehot_integer_encoding_dataInput_Xtest_by_token,np.zeros((1,len_unik_term))))

untuk extend onehot encode jika masuk kondisi berikut, dgn hasil np.vstack(([full_zeros], ..., [integer_encode]))
if(len_sekuens_dataInput_Xtest_by_token<n):
 len_extend_onehot_encode = np.abs(len_sekuens_dataInput_Xtest_by_token-n)
 for i in range(len_extend_onehot_encode):
 onehot_integer_encoding_dataInput_Xtest_by_token = np.vstack((np.zeros((1,len_unik_term)),onehot_integer_encoding_dataInput_Xtest_by_token))

print('onehot_integer_encoding_dataInput_Xtest_by_token: \n', onehot_integer_encoding_dataInput_Xtest_by_token,'\'\n')

Ntest_Xtest_by_token = 1

Jika ingin memprediksi sampai i_k sekuens
i_k = 27

raw_data_Xtest_by_token_interator = raw_data_Xtest_by_token
recurrent_delay = None
for i in range(i_k):
 print('i = ', i)
 unraw_raw_data_Xtest_by_token = []
 unraw_raw_data_Xtest_by_token.extend(raw_data_Xtest_by_token_interator.strip().replace('\n','new_line').split(' '))
 unraw_raw_data_Xtest_by_token = [name.lower() for name in unraw_raw_data_Xtest_by_token]
 arr_token_unraw_data_Xtest_by_token = np.array(unraw_raw_data_Xtest_by_token)
 if(arr_token_unraw_data_Xtest_by_token.size>=n):
 dataInput_Xtest_by_token = arr_token_unraw_data_Xtest_by_token[-n:] # diambil n kata terakhir
 print('dataInput_Xtest_by_token awal: \n', arr_token_unraw_data_Xtest_by_token,'\'\n')
 print('dataInput_Xtest_by_token diambil', str(n),'kata terakhir: \n', dataInput_Xtest_by_token,'\'\n')
 else:
 dataInput_Xtest_by_token = arr_token.unraw_data_Xtest_by_token
 print('dataInput_Xtest_by_token: \n', dataInput_Xtest_by_token,'\'\n')

 # get nilai kode integer
 integer_encoding_dataInput_Xtest_by_token,len_sekuens_dataInput_Xtest_by_token = np.array([np.where(get_unik_term_terurut_alphabet==token_in)[0][0] \
 if np.where(get_unik_term_terurut_alphabet==token_in)[0].size \
 else np.where(get_unik_term_terurut_alphabet=='new_line')[0][0] \
 for token_in in dataInput_Xtest_by_token]),dataInput_Xtest_by_token.shape[0]
 print('integer_encoding_dataInput_Xtest_by_token: \n', integer_encoding_dataInput_Xtest_by_token,'\'\n')
 print('len_sekuens_dataInput_Xtest_by_token: \n', len_sekuens_dataInput_Xtest_by_token,'\'\n')

 # if(i==0):
 # Proses penentuan Ytest_predict
 # hitung matrik h_test dengan fungsi aktivasi jenis sigmoid
 #
 # pada i=0, karena raw_data_Xtest_by_token ini tidak dalam dataset, maka matrik delay atau s_tr_test, semua nilainya diset nol
 # dan pada saat i > 0,
 #
 # if(i==0):
 # s_tr_test_Xtest_by_token = -1*np.ones((1,r)) # dikali -1, agar nanti ketika di fn_onehot_encoding_multi_d nilai fiturnya menjadi nol semua, yaitu tidak termasuk semua integer token apa-paun, kecuali jika nanti mengembangkan penggunaan type data spiral pada dataset
 # recurrent_delay = s_tr_test_Xtest_by_token[0,:-1]
 # else:
 # # r1 =
 # s_tr_test_Xtest_by_token = np.zeros((1,r))
 # s_tr_test_Xtest_by_token[0,0] = lokasi_maks_as_int_encode
 # s_tr_test_Xtest_by_token[0,1] = recurrent_delay
 # recurrent_delay = s_tr_test_Xtest_by_token[0,:-1]
 # print('s_tr_test_Xtest_by_token: \n',s_tr_test_Xtest_by_token,'\'\n')
 # print('recurrent_delay: \n',recurrent_delay,'\'\n')
```

```
untuk MLSTM
if(arr_token.unraw_data_Xtest_by_token.size < n+r):
 s_tr_test_Xtest_by_token = -1*np.ones((1,r)) # dikali -1 agar nanti ketika di fn_onehot_encoding_multi_d nilai fiturnya menjadi nol semua, yaitu tidak termasuk semua integer token apa-paun, kecuali jika nanti mengembangkan penggunaan type data spiral pada dataset

 if(arr_token.unraw_data_Xtest_by_token.size > n):
 s_tr_test_Xtest_by_token_temp = np.zeros((1,r))
 s_tr_test_Xtest_by_token_temp[0,0] = lokasi_maks_as_int_encode
 s_tr_test_Xtest_by_token_temp[0,1:] = recurrent_delay
 recurrent_delay = s_tr_test_Xtest_by_token_temp[0,:-1]

 else:
 s_tr_test_Xtest_by_token = np.zeros((1,r))
 s_tr_test_Xtest_by_token[0,0] = lokasi_maks_as_int_encode
 s_tr_test_Xtest_by_token[0,1:] = recurrent_delay
 recurrent_delay = s_tr_test_Xtest_by_token[0,:-1]
 print('s_tr_test_Xtest_by_token: \n',s_tr_test_Xtest_by_token, '\n')
 print('recurrent_delay: \n',recurrent_delay, '\n')

 #

 if(arr_token.unraw_data_Xtest_by_token.size>=n):
 Xtest_feature_Xtest_by_token = integer_encoding_dataInput_Xtest_by_token[-n:]
 Xtest_feature_Xtest_by_token = Xtest_feature_Xtest_by_token[None,...]
 else:
 # Xtest feature Xtest_by_token = np.hstack((integer_encoding_dataInput_Xtest_by_token[None,...],np.zeros((1,np.abs(len_sekuens_dataInput_Xtest_by_token-n)))))

 Xtest_feature_Xtest_by_token = np.hstack((-1*np.ones((1,np.abs(len_sekuens_dataInput_Xtest_by_token-n))),integer_encoding_dataInput_Xtest_by_token[None,...]))
 Xtest_merge_s_Xtest_by_token = np.hstack((Xtest_feature_Xtest_by_token,s_tr_test_Xtest_by_token))

 onehot_Xtest_merge_s_Xtest_by_token = fn_onehot_encoding_multi_d(Xtest_merge_s_Xtest_by_token, len_unik_term)
 # print('onehot_Xtest_merge_s: \n', onehot_Xtest_merge_s,'\'\n')
 # print('onehot_Xtest_merge_s.shape: \n', onehot_Xtest_merge_s.shape,'\'\n')

 # full connected, yaitu join semua fitur di dalam onehot_Xtest_merge_s menjadi array dgn ukuran [Ntest x (n+r)*len_unik_term]
 # dimana setiap datanya yang awalnya berukuran [(n+r) x len_unik_term] menjadi berukuran [1 x (n+r)*len_unik_term]
 full_conn_onehot_Xtest_merge_s_Xtest_by_token = np.zeros((Ntest_Xtest_by_token,(n+r)*len_unik_term))
 full_conn_onehot_Xtest_merge_s_Xtest_by_token = onehot_Xtest_merge_s_Xtest_by_token.reshape(Ntest_Xtest_by_token,(n+r)*len_unik_term)
 print('full_conn_onehot_Xtest_merge_s_Xtest_by_token: \n', full_conn_onehot_Xtest_merge_s_Xtest_by_token,'\'\n')
 # print('full_conn_onehot_Xtest_merge_s_Xtest_by_token.shape: \n', full_conn_onehot_Xtest_merge_s_Xtest_by_token.shape,'\'\n')

 h_test_Xtest_by_token = fn sigmoid(np.dot(full_conn_onehot_Xtest_merge_s_Xtest_by_token, np.transpose(baca_bobot_input)) + baca_bias*np.ones((Ntest_Xtest_by_token,1)))
 print('h_test_Xtest_by_token: \n',h_test_Xtest_by_token,'\'\n')

 # 4. Coba testing dengan data Xtest
 Ytest_predict_Xtest_by_token = np.dot(h_test_Xtest_by_token, baca_bobot_output)
 # print("\n Ytest_predict: \n", Ytest_predict)

 onehot_binary_arr_Ytest_predict, lokasi_maks_as_int_encode = y_predict_to_onehot_binary(Ytest_predict_Xtest_by_token)
 arr_token_Ytest_predict = onehot_binary_to_arr_token(onehot_binary_arr_Ytest_predict,get_unik_term_terurut_alphabet)
 # print("arr_token_Ytest_predict: \n", arr_token_Ytest_predict,"\'\n')

 raw_data_Xtest_by_token_iterator += ' '+ arr_token_Ytest_predict[0]
 print("raw_data_Xtest_by_token_iterator: ")
 print(raw_data_Xtest_by_token_iterator,"\'\n')

 if(i==0):
 arr_token_Ytest_predict_iterator = np.vstack((arr_token.unraw_data_Xtest_by_token[...],arr_token_Ytest_predict))
 else:
 arr_token_Ytest_predict_iterator = np.vstack((arr_token_Ytest_predict_iterator,arr_token_Ytest_predict))


```

```
else:
if(hasil_txt_Ytest_predict==".. (ini penanda new_line)"):
print()
else:
print(hasil_txt_Ytest_predict)

membuat name unik2save utk simpan hasil
name_unik2save = str(datetime.today().astimezone(pytz.timezone('Asia/Jakarta')).strftime('%d-%m-%Y-%H-%M-%S'))
hasil_txt_Ytest_predict_sastraa = arr_token_to_txt_sastraa('file-name_HasilTest_'+name_unik2save+'.txt',arr_token_Ytest_predict_iterator.flatten())
print("hasil txt Ytest predict sastra:")
print(hasil_txt_Ytest_predict_sastraa, "\n")

menghitung nilai BLUE Score untuk hasil testing
untuk proses training bobot_input,bias, dan bobot_out-
put utk digunakan pada penghitungan Ytest_predict dlm bentuk onehot en-
code, dapat menggunakan MAPE atau BLUE Score,
tetapi lebih direkomendasikan menggunakan MAPE, karena bentuk dari Ytest_predict dalam bentuk onehot enco
de
misal diketahui berikut
original_reference_atau_data_aktual = "tujuan hidup baha-
gia new_line dan selamat dunia akhirat"
original_reference_atau_data_aktual = "bahagia hidup kita adalah menjadi ba-
hagia"
original_reference_atau_data_aktual = "bahagia hidup kita kita menjadi baha-
gia"
original_reference_atau_data_aktual = "'hidup adalah kita menjadi baha-
gia kita obat terbaik untuk segala kesulitan
siapapun bisa jadi apapun obat hidup menjadi kita menjadi hidup'''

original_reference_atau_data_aktual = '''kita adalah menjadi baha-
gia dan selamat
kesabaran adalah obat terbaik untuk segala kesulitan
siapapun bisa jadi apapun end_of_file tujuan hidup kita adalah menjadi ba-
hagia'''

candidate_atau_hasil_predict_atau_hasil_sis-
tem = ' '.join(str(term) for term in arr_token_Ytest_predict_iterator.flat-
ten())
print('original_reference_atau_data_aktual!')
print(original_reference_atau_data_aktual,'n')

Bilingual Evaluation Understudy Score (BLUE) Score,
Skor BLUE adalah metrik untuk mengukur hasil kinerja atau mengevaluasi kalima-
t yang dihasilkan oleh sistem terhadap kalimat aktual yang dibuat menurut per-
sepsi manusia atau pakar.
original_reference_atau_data_aktual = ' '.join(str(term) for term in origi-
nal_reference_atau_data_aktual.replace("\n"," new line ").split(" "))
print('Hasil Skor BLUE dari Kalimat Lengkap raw data Xtest by token + Kalimat
hasil prediksi: ','{0:.6f}'.format(fn_bleu_score(original_refer-
ence_atau_data_aktual, candidate_atau_hasil_predict_atau_hasil_sistem)),"\n")

original_reference_atau_data_aktual = ' '.join(str(term) for term in origi-
nal_reference_atau_data_aktual.split(" ")[-i_k:])
original_reference_atau_data_aktual = ' '.join(str(term) for term in origi-
nal_reference_atau_data_aktual.replace("\n"," new line ").split(" ")[-i_k:])
print('original_reference_atau_data_aktual sepanjang Kalimat hasil prediksi: ')
print(original_reference_atau_data_aktual.replace(" new line ","\n"),"\n")

candidate_atau_hasil_predict_atau_hasil_sis-
tem = ' '.join(str(term) for term in arr_token_Ytest_predict_iterator[-
i_k:]).flatten()
print('candidate atau hasil predict atau hasil sistem: ')
print(candidate_atau_hasil_predict_atau_hasil_sistem.re-
place(" new line ","\n"),"\n")
print('Hasil Skor BLUE dari non Kalimat Lengkap (Kalimat hasil prediksi): ','{0:
.6f}'.format(fn_bleu_score(original_reference_atau_data_aktual, candi-
date_atau_hasil_predict_atau_hasil_sistem)))
```

## Output:





```
[15.]
full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[0. 0. 1. 0.
1. 0.
0. 0. 0. 0. 0. 0. 0. 1. 0.
0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0.]]

h_test_Xtest_by_token:
[[0.54815511 0.36985165 0.75987881 0.30763886 0.77911665 0.57608706
0.62000017 0.14032645 0.4785111 0.44533885 0.71133778 0.91283362
0.78052473 0.69291471 0.42662622 0.69510223 0.49664972 0.68949935
0.69586474 0.56722229 0.66012838 0.63458909 0.41847595 0.39868649
0.72827706 0.65682225 0.69867244 0.19217429 0.79944645 0.77455692
0.78813688 0.52787386 0.92011014 0.85550881 0.15747087 0.81326206
0.58356701 0.1505266 0.80050266 0.69140172 0.98384933 0.67048314
0.51781923 0.6758519 0.82986326 0.66749338 0.42723793 0.42506126
0.938824 0.53076417 0.96430612 0.79180394 0.5285717 0.82760632
0.71314156 0.64879834 0.02785872 0.98079501 0.90188279 0.38635824
0.93669951 0.44393118 0.83238987 0.68703172 0.53141172 0.84545243
0.88773457]]

raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line

i = 6
dataInput_Xtest_by_token awal:
['kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat' 'new_line']

dataInput_Xtest_by_token diambil 3 kata terakhir:
['dan' 'selamat' 'new_line']

integer_encoding_dataInput_Xtest_by_token:
[4 15 12]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[12. 15.]]

recurrent_delay:
[12.]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[0. 0. 0. 1. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 1. 0.
1. 0. 1.
0. 0. 0. 0. 0.]]

h_test_Xtest_by_token:
[[0.75744343 0.93068501 0.82495056 0.61233684 0.2141768 0.68438066
0.49427468 0.09186866 0.09445392 0.15273989 0.41653021 0.65506176
0.81537909 0.98450338 0.97713853 0.65594254 0.06792166 0.74337912
0.8013478 0.29413451 0.55201634 0.10584405 0.59146792 0.5968457
0.40288962 0.36992158 0.76144464 0.41284049 0.44129116 0.43375999
0.61837303 0.34369642 0.35231909 0.82707773 0.49329973 0.192485
0.85544699 0.45409999 0.53143663 0.62790795 0.72738555 0.30870498
0.78637814 0.80972816 0.80565837 0.57782018 0.82878741 0.20082628
0.48370414 0.19279609 0.7131211 0.97279196 0.59581049 0.24458345
0.66669381 0.29675754 0.07364146 0.2780218 0.29226921 0.6306144
0.60397183 0.62486532 0.60294275 0.82510937 0.98658704 0.52128925
0.89609929]]

raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line kesabaran

i = 7
dataInput_Xtest_by_token awal:
['kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat' 'new_line'
'kesabaran']

dataInput_Xtest_by_token diambil 3 kata terakhir:
['selamat' 'new_line' 'kesabaran']

integer_encoding_dataInput_Xtest_by_token:
[15 12 8]

len sekuens dataInput Xtest by token:
```



```
dataInput_Xtest_by_token diambil 3 kata terakhir:
['kesabaran' 'adalah' 'obat']

integer_encoding_dataInput_Xtest_by_token:
[8 0 13]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[13. 0.]]

recurrent_delay:
[13.]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[0. 0. 0. 0. 0. 0. 0. 1. 0.
0.
0. 0. 0. 0. 0. 1. 0.
0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0.]]

h_test_Xtest_by_token:
[[0.46157208 0.7614628 0.66055458 0.54758532 0.82041356 0.79265995
0.73788478 0.67552317 0.79795107 0.76692328 0.6320576 0.35288983
0.36348822 0.64989139 0.2565255 0.95709484 0.32561778 0.5684494
0.12571445 0.1901486 0.41925496 0.44543766 0.82780599 0.64303441
0.82389378 0.78228426 0.71316201 0.69037653 0.82239419 0.2512025
0.85240319 0.22458852 0.70779284 0.45414957 0.6272067 0.82888673
0.96207651 0.45628254 0.80443407 0.77537665 0.56648563 0.79024994
0.81205922 0.22453627 0.6695102 0.64259812 0.65718285 0.53872616
0.71289598 0.57991689 0.91261054 0.21567868 0.49769979 0.92732149
0.27494091 0.87662299 0.10674461 0.63303391 0.82630398 0.22763352
0.79082955 0.44230237 0.27868474 0.28112383 0.5790883 0.58861331
0.41492692]]]

raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line kesabaran adalah obat terbaik

i = 10
dataInput_Xtest_by_token awal:
['kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat' 'new_line'
'kesabaran' 'adalah' 'obat' 'terbaik']

dataInput_Xtest_by_token diambil 3 kata terakhir:
['adalah' 'obat' 'terbaik']

integer_encoding_dataInput_Xtest_by_token:
[0 13 17]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[17. 13.]]

recurrent_delay:
[17.]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[1. 0.
0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 1. 0.
0. 0. 0. 0. 0.]]

h_test_Xtest_by_token:
[[0.08720618 0.91301647 0.56628913 0.69474179 0.53131211 0.76670869
0.28975183 0.47237539 0.32119732 0.86347821 0.47063091 0.91908229
0.39192114 0.65655169 0.831537 0.8196165 0.37696503 0.93869176
0.28812858 0.59337563 0.88376764 0.61053104 0.96897797 0.59554554
0.15789594 0.47858596 0.90481647 0.88753507 0.55678466 0.80715721
0.15112911 0.69569539 0.64635614 0.83976299 0.45194398 0.26543631
0.52563008 0.62207146 0.5964606 0.42110673 0.35352956 0.70295018
0.93946985 0.31856463 0.62178927 0.84031397 0.39108726 0.47302352
0.84255566 0.72764328 0.54113593 0.45330677 0.79297207 0.5702885
0.77042651 0.29579836 0.71457155 0.59489495 0.57153797 0.72390125
0.71444915 0.20923663 0.72635309 0.73354846 0.36975842 0.83321148
0.25242724]]]

raw_data_Xtest_by_token_interator:
```









```
[['kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat' 'new_line'
 'kesabaran' 'adalah' 'obat' 'terbaik' 'untuk' 'segala' 'kesulitan'
 'new_line' 'siapapun' 'bisa' 'jadi' 'apapun']

dataInput_Xtest_by_token diambil 3 kata terakhir:
['bisa' 'jadi' 'apapun']

integer_encoding_dataInput_Xtest_by_token:
[3 7 1]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[1. 7.1]

recurrent_delay:
[1.1]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[0. 0. 0. 1. 0.
 0. 0. 0. 1. 0.
 0.
 0.
 0. 0. 0. 0. 0.]]

h_test_Xtest_by_token:
[[0.4816564 0.78920367 0.70357629 0.68039237 0.20169444 0.85364451
 0.36661768 0.91024513 0.69573773 0.37562712 0.68806299 0.69512343
 0.32825863 0.29326317 0.31430307 0.69856715 0.66281576 0.60136147
 0.86886827 0.37188177 0.8893785 0.83366963 0.79885251 0.78882075
 0.60138544 0.61171941 0.63995239 0.40082226 0.34878993 0.42674854
 0.75035566 0.40867708 0.87432951 0.17930664 0.36016283 0.18005844
 0.39454601 0.5629456 0.87398846 0.89882334 0.89476018 0.76415891
 0.21986957 0.11000314 0.89901417 0.15316748 0.78009612 0.38309131
 0.43826048 0.68990602 0.20579926 0.88966352 0.80466996 0.53822907
 0.91543185 0.30065423 0.19925788 0.51334816 0.41616567 0.2474775
 0.3766832 0.83301682 0.44341279 0.52024096 0.60416318 0.88354143
 0.72485968]]

raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line kesabaran adalah obat terbaik
untuk segala kesulitan new_line siapapun bisa jadi apapun end_of_file

i = 19
dataInput_Xtest_by_token awal:
['kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat' 'new_line'
 'kesabaran' 'adalah' 'obat' 'terbaik' 'untuk' 'segala' 'kesulitan'
 'new_line' 'siapapun' 'bisa' 'jadi' 'apapun' 'end_of_file']

dataInput_Xtest_by_token diambil 3 kata terakhir:
['jadi' 'apapun' 'end_of_file']

integer_encoding_dataInput_Xtest_by_token:
[7 1 5]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[5. 1.1]

recurrent_delay:
[5.1]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0.
 0.
 0. 0. 0. 0. 0. 0.]]

h_test_Xtest_by_token:
[[0.68862082 0.7548988 0.70719264 0.27876517 0.26723417 0.55471056
 0.75082372 0.68461824 0.29096765 0.50462533 0.83915648 0.23584109
 0.70505505 0.83935885 0.72625369 0.96182761 0.98608021 0.78333847
 0.60831718 0.72022835 0.37936385 0.84195773 0.41915757 0.43602122
 0.565036 0.47294873 0.33071113 0.10144983 0.950668694 0.279087
 0.98669775 0.67107945 0.582254 0.74672254 0.55285707 0.56346226
 0.21149597 0.05560963 0.21473279 0.49994999 0.9571482 0.76089922
 0.94340534 0.1976194 0.42225301 0.73198174 0.16221156 0.67723069]
```





```
integer_encoding_dataInput_Xtest_by_token:
[6 10 0]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[0. 10.]]

recurrent_delay:
[10.]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[0. 0. 0. 0. 0. 0. 1. 0.
0. 0. 0. 0. 0. 0. 1. 0.
0.
0.
0. 0. 0. 0. 0.]]

h_test_Xtest_by_token:
[[0.73643147 0.38963544 0.40248068 0.45995693 0.35786139 0.95839396
0.63444994 0.72760364 0.17955697 0.56437223 0.72744504 0.30115915
0.55824029 0.75201928 0.69497505 0.77777126 0.79257775 0.81172319
0.68167414 0.65374991 0.21739228 0.78081584 0.62388019 0.94116476
0.45199352 0.34584271 0.83593152 0.80642451 0.87495454 0.91416126
0.58943645 0.81112638 0.31646245 0.21241472 0.61823141 0.6911029
0.63308037 0.40180743 0.37741141 0.49110005 0.53725953 0.84606561
0.87994392 0.70623911 0.68238986 0.82746358 0.25872513 0.31197994
0.46953473 0.6408506 0.85079806 0.44746435 0.87130998 0.69041928
0.73068481 0.88593802 0.60306246 0.63502961 0.91679222 0.18985616
0.46744287 0.47362187 0.28094196 0.54909622 0.78649572 0.73080288
0.5337022]]

raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line kesabaran adalah obat terbaik
untuk segala kesulitan new_line siapapun bisa jadi apapun end_of_file tujuan
hidup kita adalah menjadi

i = 24
dataInput_Xtest_by_token awal:
['kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat' 'new_line'
'kesabaran' 'adalah' 'obat' 'terbaik' 'untuk' 'segala' 'kesulitan'
'new_line' 'siapapun' 'bisa' 'jadi' 'apapun' 'end_of_file' 'tujuan'
'hidup' 'kita' 'adalah' 'menjadi']

dataInput_Xtest_by_token diambil 3 kata terakhir:
['kita' 'adalah' 'menjadi']

integer_encoding_dataInput_Xtest_by_token:
[10 0 11]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[11. 0.]]

recurrent_delay:
[11.]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
0. 0. 0. 1. 0.
0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0.]]

h_test_Xtest_by_token:
[[0.09421468 0.90734402 0.94059734 0.87565315 0.84745414 0.69103885
0.16218438 0.36071615 0.92619463 0.33381753 0.77429486 0.36884995
0.63201108 0.52540564 0.85333187 0.70196773 0.12481586 0.74395106
0.33192976 0.25259714 0.50275025 0.15767001 0.87122024 0.70022822
0.32654083 0.8394667 0.29554843 0.32905302 0.51207386 0.30204392
0.78420288 0.78233535 0.66593778 0.73164806 0.35620838 0.81314052
0.93125061 0.59511185 0.75283899 0.39679391 0.9287971 0.72191797
0.50337529 0.34956267 0.92928499 0.6563487 0.55016073 0.87004878
0.21535741 0.44832993 0.51572139 0.49317474 0.34238918 0.82797699
0.21589869 0.87398846 0.53723467 0.59412346 0.95076191 0.35827517
0.44121719 0.70663318 0.75797588 0.65127821 0.71088579 0.68414298
0.7945438]]
```

```
raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line kesabaran adalah obat terbaik
untuk segala kesulitan new_line siapapun bisa jadi apapun end_of_file tujuan
hidup kita adalah menjadi bahagia

i = 25
dataInput_Xtest_by_token awal:
['kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat' 'new_line'
'kesabaran' 'adalah' 'obat' 'terbaik' 'untuk' 'segala' 'kesulitan'
'new_line' 'siapapun' 'bisa' 'jadi' 'apapun' 'end_of_file' 'tujuan'
'hidup' 'kita' 'adalah' 'menjadi' 'bahagia']

dataInput_Xtest_by_token diambil 3 kata terakhir:
['adalah' 'menjadi' 'bahagia']

integer_encoding_dataInput_Xtest_by_token:
[0 11 2]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[2. 11.]]

recurrent_delay:
[2.]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[1. 0.
0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
0.
0. 0. 0. 0.]])

h_test_Xtest_by_token:
[[0.4075658 0.97489263 0.86505033 0.65184584 0.64991415 0.7808843
0.66702708 0.44743962 0.4479342 0.43749737 0.76776244 0.19290506
0.10498582 0.51172402 0.72673062 0.53972011 0.65716032 0.42474356
0.71039222 0.26831267 0.39488052 0.97118586 0.84135794 0.90788909
0.808074 0.11305526 0.81288205 0.88741522 0.85376942 0.77896171
0.17122678 0.5736184 0.16380803 0.73003487 0.30477061 0.045795
0.54582563 0.76473519 0.39600425 0.15556959 0.4455859 0.56712409
0.42274105 0.46408337 0.47110434 0.61221813 0.67422847 0.63354487
0.88399346 0.79398816 0.68817031 0.27989258 0.52642799 0.33486365
0.74896677 0.13863394 0.14168314 0.74140911 0.95790471 0.49724975
0.83343374 0.40669677 0.85003473 0.23770259 0.4309864 0.66379852
0.9765504]])

raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line kesabaran adalah obat terbaik
untuk segala kesulitan new_line siapapun bisa jadi apapun end_of_file tujuan
hidup kita adalah menjadi bahagia dan

i = 26
dataInput_Xtest_by_token awal:
['kita' 'adalah' 'menjadi' 'bahagia' 'dan' 'selamat' 'new_line'
'kesabaran' 'adalah' 'obat' 'terbaik' 'untuk' 'segala' 'kesulitan'
'new_line' 'siapapun' 'bisa' 'jadi' 'apapun' 'end_of_file' 'tujuan'
'hidup' 'kita' 'adalah' 'menjadi' 'bahagia' 'dan']

dataInput_Xtest_by_token diambil 3 kata terakhir:
['menjadi' 'bahagia' 'dan']

integer_encoding_dataInput_Xtest_by_token:
[11 2 4]

len_sekuens_dataInput_Xtest_by_token:
3

s_tr_test_Xtest_by_token:
[[4. 2.]]

recurrent_delay:
[4.]

full_conn_onehot_Xtest_merge_s_Xtest_by_token:
[[0.
0.
0.
0.
0. 0. 0. 0.]])
```

```
h_test_Xtest_by_token:
[[0.88713514 0.30316251 0.96852079 0.58115894 0.78277782 0.75830596
0.7779268 0.51994141 0.82369054 0.17053248 0.68750462 0.43184503
0.25744211 0.67703391 0.34392205 0.43388282 0.68084891 0.78668039
0.60850779 0.48442848 0.77949506 0.49652471 0.86520203 0.95825016
0.9489084 0.04073749 0.59134709 0.89548312 0.67795168 0.57652662
0.53325417 0.11439591 0.5824486 0.58125631 0.52892058 0.36659445
0.87953126 0.21446309 0.46577517 0.88196812 0.73586813 0.51871812
0.61877417 0.25316399 0.66551492 0.53325417 0.14907503 0.39277951
0.72485968 0.5852673 0.63073087 0.71828967 0.45184489 0.13328885
0.44902263 0.73803949 0.81468555 0.70750318 0.6030864 0.78746813
0.78014759 0.89010452 0.82969375 0.51054949 0.07500377 0.84735068
0.4491216]]

raw_data_Xtest_by_token_interator:
kita adalah menjadi bahagia dan selamat new_line kesabaran adalah obat terbaik
untuk segala kesulitan new_line siapapun bisa jadi apapun end_of_file tujuan
hidup kita adalah menjadi bahagia dan selamat

hasil_txt_Ytest_predict_sastraw:
kita adalah menjadi bahagia dan selamat
kesabaran adalah obat terbaik untuk segala kesulitan
siapapun bisa jadi apapun end_of_file tujuan hidup kita adalah menjadi bahagia
dan selamat

original_reference_atau_data_aktual:
kita adalah menjadi bahagia dan selamat
kesabaran adalah obat terbaik untuk segala kesulitan
siapapun bisa jadi apapun end_of_file tujuan hidup kita adalah menjadi bahagia

Hasil Skor BLUE dari Kalimat Lengkap (raw_data_Xtest_by_token + Kalimat hasil
prediksi): 0.730159

original_reference_atau_data_aktual sepanjang Kalimat hasil prediksi:
kita adalah menjadi bahagia dan selamat
kesabaran adalah obat terbaik untuk segala kesulitan
siapapun bisa jadi apapun end_of_file tujuan hidup kita adalah menjadi bahagia

candidate_atau_hasil_predict_atau_hasil_sistem:
adalah menjadi bahagia dan selamat
kesabaran adalah obat terbaik untuk segala kesulitan
siapapun bisa jadi apapun end_of_file tujuan hidup kita adalah menjadi bahagia
dan selamat

Hasil Skor BLUE dari non Kalimat Lengkap (Kalimat hasil prediksi): 0.720798
```

## Link kode program lengkapnya:

[https://colab.research.google.com/drive/1Lwy\\_EahfZISvfBKT3RhDg9\\_iPnA76Ve?usp=sharing](https://colab.research.google.com/drive/1Lwy_EahfZISvfBKT3RhDg9_iPnA76Ve?usp=sharing)



## 18.6 Tugas Kelompok

1. Jelaskan pengertian dari Algoritme Recurrent Neural Network (RNN)!
2. Apa keunggulan dari RNN jika dikombinasikan dengan algoritma Extreme Learning Machine (ELM) sehingga menjadi RELMNN!
3. Pada dataset “Data nilai tukar Rupiah terhadap Dolar Amerika Serikat” dari contoh di atas, buatlah manualisasi mulai dari proses *training* sampai *testing*, dengan perbandingan data *training*: data *testing* = 3: 1. Di mana, inisialisasi jumlah fitur (*n*) yang digunakan sebanyak 4, sedangkan jumlah *hidden neuron* (*m*) sebanyak 2, dan jumlah *context neuron* (*r*) sebanyak 3.
4. Buatlah kode program untuk mengimplementasikan RELMNN sesuai dengan kasus yang ada di contoh prediksi nilai tukar. Lalu bandingkan, pastikan hasil koding dengan manualisasi dicontoh menghasilkan nilai-nilai perhitungan yang sama!
5. Jelaskan konsep “**Forget gate** by Extend Long-Short Feature” pada MKSTM berdasarkan bagian kode program berikut!

```
s_tr_train,s_tr_test = create_matrix_delay_roll_mono_gate(r,
Ntrain_ori, Ntest_ori, Ytrain, Ytest, Target)
```

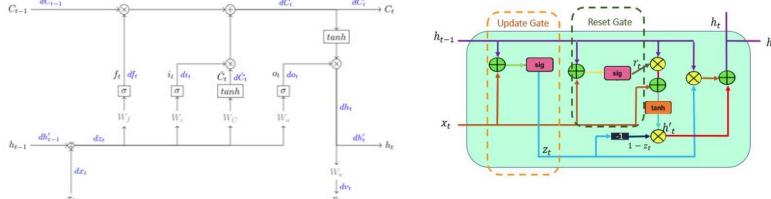
Untuk menyusun **Recurrent gate** lengkap, sebelum dilakukan hstack dengan Xtrain\_feature untuk data train untuk menyusun **Mono Cell gate**, di mana bagian koding tersebut, yaitu

```
Xtrain_feature = np.vstack((Xtrain_feature,Xtrain_feature))
Ytrain_onehot_encode = np.vstack((Ytrain_onehot_en-
code,Ytrain_onehot_encode))
Xtrain_merge_s = np.hstack((Xtrain_feature,s_tr_train))
```

6. Buatlah kode program untuk mengimplementasikan RELMNN dan MLSTM sesuai dengan kasus yang ada di contoh data teks. Lalu bandingkan nilai BLUE Score-nya, manakah hasil yang lebih baik, dan disertai dengan manualisasi masing-masing dari kedua algoritma dengan *n* = 2, *r* = 2 dan parameter lainnya bebas tetapi tetap relevan!
7. Buatlah kode program untuk mengimplementasikan LSTM Native (modifikasi dari link kode google colab yang disediakan, yaitu pada bagian **### 4\_RNN dengan Jenis LSTM Native from Scratch -**

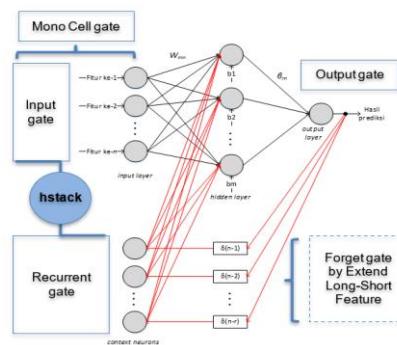
Lama Proses Trainingnya karena tidak Menggunakan ELM") dan MLSTM sesuai dengan kasus yang ada di contoh data teks. Lalu bandingkan nilai BLUE Score-nya, manakah hasil yang lebih baik, dan disertai dengan manualisasi LSTM Native dan perbandingan waktu komputasi dari kedua algoritma dengan parameter bebas tetapi tetap relevan!

8. Buatlah kode program untuk mengimplementasikan GRU, LSTM Native (Gunakan dari full lib. dengan Tensorflow atau Keras atau lainnya) dan MLSTM sesuai dengan kasus yang ada di contoh data teks. Lalu bandingkan nilai BLUE Score-nya, manakah hasil yang lebih baik, dan disertai tampilan perbandingan grafik waktu komputasi masing-masing dari ketiga algoritma dengan parameter bebas tetapi tetap relevan!
9. Berdasarkan arsitektur (a) LSTM vs (b) GRU vs (c) MLSTM berikut:



(a)

(b)



(c)

Buatlah representasi individu atau partikel atau kromosom untuk mengoptimasikan nilai evaluasi dari masing-masing algoritma di atas berdasarkan parameter fungsi aktivasi, jenis gate, interval nilai random atau lainnya, menggunakan Algoritma Particle Swarm Optimization (PSO) atau Algoritma Genetika!

## BAB 19 Deep Reinforcement Learning (DRL)

### 19.1 Pengantar

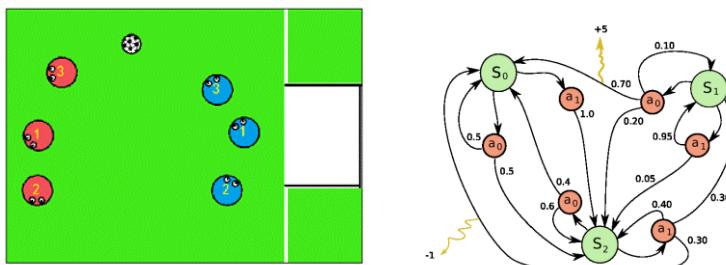
Multi-agent atau social-agent dapat dikatakan sebagai sebuah robot yang mampu bekerja sama dalam proses pembelajaran maupun untuk memberikan solusi pada suatu kasus. Pada teknologi yang telah modern ini, multi-agent tersebut dapat dipelajari dan dikembangkan menggunakan teknik Deep Reinforcement Learning (DRL). Pengembangan multi-agent pada Inverse RL pada soccer game sebagai abstraksi percobaan memiliki hasil yang lebih baik jika dibandingkan dengan single-agent Inverse RL. Hal ini disebabkan karena kurangnya penyeimbangan informasi yang didapat dari lingkungan yang sifatnya memungkinkan dinamis ataupun pasif.

Ketika RL dikonfigurasikan pada multi-agent, maka RL tersebut diimplementasikan pada single-agent. Karena secara komputasi, setiap agent dihitung nilai Q-valuenya untuk memperoleh *policies* yang paling efektif. Dapat dihitung secara sequensial maupun secara paralel. Tantangan dalam menghitung Q-value yaitu banyaknya alternatif/pilihan algoritme yang dapat digunakan. Selain itu juga sedikit yang menggunakan penggabungan 2 algoritme atau lebih.

Penelitian DRL pernah dilakukan pada multi-agent menggunakan CNN (*convolution neural network*) untuk memperoleh nilai estimasi dari Q-value. Penelitian tersebut melakukan pertimbangan pada kasus multi-agent mulai dari bagaimana cara dua agen tidak terkesan belajar dengan hanya mementingkan diri sendiri hingga permasalahan skalabilitas ketika lingkungan berukuran kecil atau besar. DRL terbukti efektif dalam menyelesaikan permasalahan tersebut jika dibandingkan dengan pendekatan tradisional untuk menyelesaikan Markov Decision Process (MDP). Bentuk interaksi agen satu dengan yang lainnya dalam konsep multi-agent dapat dipetakan menjadi *fully cooperative*, *fully competitive*, dan gabungannya, yang membuat agen satu dengan yang lain dapat bekerja sama atau tidak atau ada aturan lain (Bu,Soni, 2006).

## 19.2 Multi-Agent RL (MARL)

Multi-agent memiliki tantangan besar dalam mengoptimalkan kontrol *policy* masing-masing agen. Karena pada waktu yang sama harus memperhitungkan keadaan agen lainnya. Salah satu pendekatan yang dapat digunakan untuk pelatihan, yaitu semua agen sepenuhnya harus terdesentralisasi, yang mana setiap agen memiliki *policy* Deep Q-Network secara mandiri (Punma, 2017).



Gambar 19.1 Contoh Multi-Agent dan MDP

(Jiang, 2019) dan (Karpathy, 2016)

## 19.3 Self-Organizing Map (SOM)

Self-Organizing Map (SOM) pertama kali dipublikasi oleh Teuvo Kohonen dari University of Helsinki tahun 1981. Jaringan SOM Kohonen termasuk dalam pembelajaran *unsupervised* yaitu metode yang secara otomatis mengatur dan menentukan estimasi nilai keluarannya sesuai dengan aturan yang dimiliki. Proses pelatihan SOM (a-e) di bawah ini akan terus dilakukan sampai kondisi berhenti. (Kartika, 2016).

a. Menetapkan:

- Input data yang dinormalisasi
- Jumlah kelas

b. Inisialisasi:

- Bobot input ( $w_{ij}$ ) dengan nilai *random*.
- Set parameter *learning rate* ( $\alpha$ ).
- Set maksimum iterasi (MaxIterasi).

c. Set Iterasi = 0.

d. Kerjakan jika Iterasi < MaxIterasi.

- Pilih data secara acak, misal terpilih data ke- $j$ .
- Cari jarak antara data ke- $j$  dengan tiap bobot input ke- $i$  ( $D_j$ ):

$$D(j) = \sum_i^m (w_{ij} - X_i)^2 \quad (1)$$

- Cari bobot yang terkecil (pemenang).
- Update bobot yang baru:

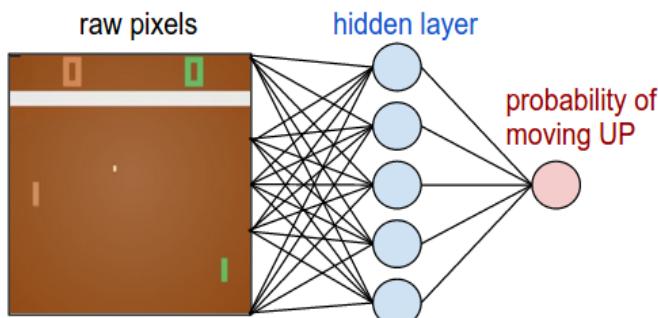
$$w_{ij}(\text{baru}) = w_{ij}(\text{lama}) + \alpha[x_i - w_{ij}(\text{lama})] \quad (2)$$

- Perbarui *learning rate* ( $\alpha$ ).
- Iterasi += 1

e. Selesai

Pada proses pengujian SOM cukup sederhana, yaitu hanya dengan menghitung nilai jarak antara bobot dengan data tanpa harus melakukan *update* bobot. Adapun bobot yang digunakan adalah bobot terbaik yang diperoleh dari hasil proses pelatihan. Penelitian RL dengan SOM yang telah ada menggunakan konsep pemberian *reward* secara tidak langsung (*delay reward*) yang memungkinkan sistem lebih mempertimbangkan *reward* jangka panjang karena kompleksnya *state* yang dilalui agen, sehingga sistem tidak sewenang-wenang dalam memberikan reward yang justru terkadang membuat agen menjadi kurang optimal. Sedangkan SOM yang diterapkan juga hanya fokus pada penentuan nilai Q-value tunggal (Smith, 2001).

## 19.4 Q-learning dan JST



Gambar 19.2 Contoh Multi-Agent dengan JST  
(Karpathy, 2016)

Penyelesaian MDP seperti pada Gambar 19.2 dengan menggunakan pendekatan JST, misal dengan *backpropagation* dan *feedforward* untuk mengestimasi nilai Q-Value membutuhkan cukup banyak variabel yang harus dihitung dan juga hal ini terkait dengan banyaknya iterasi yang dibutuhkan. Kenapa demikian, karena JST yang diterapkan dalam MDP akan menghasilkan nilai prediksi yang kemudian akan dicek dengan hasil Q-Value dari proses feedforward sebagai target sementaranya (Ardiansyah, 2017). Berikut ilustrasi langkah-langkah untuk menyelesaikan Game Pong pada Gambar 2, yang terdiri dari 2 agen, *paddle* kiri sebagai agen ke-1 (berwarna coklat muda) dan *paddle* kanan sebagai agen ke-2 (berwana hijau):

a. Menetapkan

- 1) *State*
- 2) *Action*
- 3) *Reward*

Dan juga ditetapkan untuk nilai parameter *learning rate* ( $\alpha_Q$ ) dan *discount rate* ( $\gamma_Q$ ) dengan nilai sembarang, misal dalam interval [0;1].

b. Proses pada JST

- 1) Inisialisasi bobot
- 2) Ambil *state* dari lingkungan
- 3) Pilih *action* dari *state* (misal dgn  $\epsilon$ -greedy)
- 4) Hitung *Reward* dan *Next state*
- 5) Hitung nilai *update Q-value*
- 6) Latih JST pada *action* yang sudah dipilih dengan *backpropagation*
- 7) *Update Next state* menjadi *state* saat ini
- 8) Cek kondisi penyimpanan bobot dan kondisi berhenti

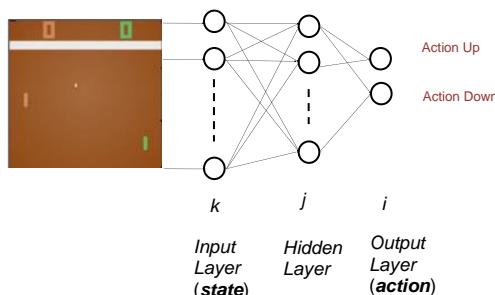
c. Selesai

Sebagai tambahan, proses pada JST, pada nomor tiga, untuk memilih *action* dari *state* dapat digunakan metode *feed-forward*. Dan pada nomor lima, yaitu untuk menghitung nilai *update Q-value* dapat menggunakan Persamaan 3.

$$Q(S,A) = Q(S,A) + \alpha_Q[R + \gamma_Q \max_a Q(S',a) - Q(S,A)] \quad (3)$$

## 19.5 Studi Kasus: DRL untuk Multi-Agent pada Permainan Pim Pong

Bentuk hasil arsitektur SOM pada Gambar 19.3 berdasarkan Gambar 19.2, setiap agen dihitung secara sekuensial, jika bola mendekati agen ke-1, maka yang dihitung nilai Q-value terlebih dahulu adalah untuk agen ke-1. Dan sebaliknya jika bola mendekati agen ke-2, maka yang dihitung nilai Q-value adalah untuk agen ke-2. Misal berikut untuk perhitungan Q-value dari agen ke-1.



Gambar 19.3 Hasil arsitektur SOM

c. Menetapkan

- 1) *State*, misal yang menjadi fitur ke-1 (sudut *paddle* bola dengan agen), fitur ke-2 (jarak *paddle* dengan agen).
- 2) *Action*, ke atas (*Up*) dan ke bawah (*Down*). Jumlah kelas = 2 (*Up* dan *Down*). *Up* sbg kelas ke-1, *Down* sbg kelas ke-2.
- 3) *Reward*, 1 (ketika berhasil mengenai bola), -1 (ketika bola terlewati), 0,1 (ketika scorenya bertambah atau ketika bola masih di udara) dan misal  $\alpha_Q = 0,95$  dan  $\gamma_Q = 0,99$

d. Proses pada SOM (agen ke-1)

- 1) Baca *state*
  - Input data sebaiknya yang telah ternormalisasi [0;1], dalam RL berarti ambil *state* dari lingkungan. Misal didapat fitur ke-1 = 0,9 dan fitur ke-2 = 0,7 atau dapat ditulis  $S = [0,9 \ 0,7]^T$ .
- 2) Inisialisasi:
  - Bobot input ( $w_{ij}$ ) dengan nilai *random*.

| Inisialisasi Bobot [0;1] |         |
|--------------------------|---------|
| Kelas 1                  | Kelas 2 |
| 0,3                      | 0,2     |
| 0,4                      | 0,8     |

- 3) Pilih *action* dari state, misal dengan proses pengujian SOM

- Ambil data dari *state* saat ini, yaitu

| Data yang ada pada <i>state</i> |                                   |
|---------------------------------|-----------------------------------|
|                                 | <i>State</i> saat ini (Data ke-1) |
| Fitur ke-1                      | 0,9                               |
| Fitur ke-2                      | 0,7                               |

- Cari jarak antara data ke-*i* dengan tiap bobot input ke-*j* ( $D_j$ ) dengan Persamaan 1:

| Data ke- <i>i</i> | Jarak pada kelas ke- <i>j</i> |     |
|-------------------|-------------------------------|-----|
|                   | 1                             | 2   |
| 1                 | 0,45                          | 0,5 |

Di mana nilai jarak-jarak tersebut akan dikonversi menjadi nilai *action-value function* dengan Persamaan 4,

$$\text{action-value function} = 1 - D_j \quad (4)$$

sehingga didapatkan tabel *action-value function* berikut yang akan selalu diambil nilai yang terbesar.

| Data ke- <i>i</i> | action-value function pada kelas ke- <i>j</i> |     |
|-------------------|-----------------------------------------------|-----|
|                   | 1                                             | 2   |
| 1                 | 0,55                                          | 0,5 |

- Cari bobot yang terkecil jaraknya (sebagai pemenang), yaitu kelas *k*, yang didapat dari  $k=\text{ArgMin}_j=\{0,45;0,5\}$ , di mana  $k=1$ , maka

| Kelas 1<br>(di-update) | Kelas 2<br>(tdk di-update) |
|------------------------|----------------------------|
| 0,3                    | 0,2                        |
| 0,4                    | 0,8                        |

Karena kelas 1 pemenangnya, maka agen memilih untuk melakukan *action Up*.

- 4) Hitung *Reward* dan *Next state*

Pada saat *state* dgn nilai  $[0,9 \ 0,7]^T$ , agen memilih untuk melakukan *action Up*, maka agen mendapat *reward* +0,1 (dgn asumsi bahwa bola masih diudara atau tidak masuk pada kondisi

Reward 1 dan 2). Dan misal didapat *Next state* dgn nilai  $[0,7 \ 0,5]^T$ , atau dapat ditulis menjadi  $S' = [0,7 \ 0,5]^T$ .

5) Hitung nilai *update Q-value* atau *action-value function*

**Alternatif cara pertama:**

Pertama ambil nilai *action-value function* dari state sebelumnya yaitu  $[0,9 \ 0,7]^T$  yang terbesar, shg  $Q(S,A) = 0,55$ .

Selanjutnya menghitung nilai *action-value function* terbesar untuk *Next state* yaitu  $[0,7 \ 0,5]^T$  pada variabel  $\max_a Q(S',a)$  dengan melakukan proses pengujian SOM untuk mendapat nilai *action-value function* yang kemudian akan dibandingkan untuk mendapat nilai terbesar.

- Cari jarak antara data ke- $i$  =  $[0,7 \ 0,5]^T$  dengan tiap bobot input ke- $j$  ( $D_j$ ):

| Data ke- $i$ | Jarak pada kelas ke- $j$ |      |
|--------------|--------------------------|------|
|              | 1                        | 2    |
| 1            | 0,17                     | 0,34 |

sehingga didapatkan tabel *action-value function* berikut yang akan selalu diambil nilai yang terbesar.

| Data ke- $i$ | action-value function pada kelas ke- $j$ |      |
|--------------|------------------------------------------|------|
|              | 1                                        | 2    |
| 1            | 0,83                                     | 0,66 |

karena *action 1 (Up) > action 2 (Down)*, maka  $\max_a Q(S',a) = 0,83$ .

Hitung nilai *update Q-value* dengan Persamaan 3.

$$\begin{aligned} Q(S,A) &= Q(S,A) + \alpha_Q [R + \gamma_Q \max_a Q(S',a) - Q(S,A)] \\ &= 0,55 + 0,95[0,1 + 0,99 * 0,83 - 0,55] \\ &= 0,903115 \end{aligned}$$

**Alternatif cara kedua:**

Pertama ambil nilai *action-value function* dari state sebelumnya yaitu  $[0,9 \ 0,7]^T$  yang terbesar, shg  $Q(S,A) = 0,55$ .

Selanjutnya menghitung nilai *action-value function* terbesar untuk *Next state* yaitu  $[0,7 \ 0,5]^T$  pada variabel  $\max_a Q(S',a)$  dengan melakukan proses pengujian SOM untuk mendapat nilai *action-value function* yang kemudian akan dibandingkan untuk mendapat nilai terbesar. Tetapi hanya dihitung terhadap salah satu

bobot yang memiliki nilai *action-value function* yang terbesar dari state sebelumnya.

- Cari jarak antara data ke-*i* = [0,7 0,5]<sup>T</sup> dengan pada satu bobot input ke-*j* ( $D_j$ ) yang memiliki nilai *action-value function* terbesar dari state sebelumnya, yaitu menunjuk ke *action* kelas 1 (Up), maka dihitung terhadap kelas ke-1 saja.

| Data ke- <i>i</i> | Jarak pada kelas ke- <i>j</i> |   |
|-------------------|-------------------------------|---|
|                   | 1                             | - |
| 1                 | 0,17                          | - |

sehingga didapatkan tabel *action-value function* berikut.

| Data ke- <i>i</i> | action-value<br>function pada kelas ke- <i>j</i> |   |
|-------------------|--------------------------------------------------|---|
|                   | 1                                                | - |
| 1                 | 0,83                                             | - |

karena *action* 1 (Up) saat > *action* 1 (Up) sebelumnya, yaitu 0,83 > 0,55, maka  $\max_a Q(S', a) = 0,83$ . Lalu Hitung *update Q-value*.

$$\begin{aligned}Q(S, A) &= Q(S, A) + \alpha_Q [R + \gamma_Q \max_a Q(S', a) - Q(S, A)] \\&= 0,55 + 0,95[0,1 + 0,99 * 0,83 - 0,55] \\&= 0,903115\end{aligned}$$

Pilih salah satu dari alternatif kedua cara di atas, yang nantinya dapat dilakukan pengujian, untuk mengetahui manakah yang terbaik jika desain *framework* ini diimplementasikan. Misal dalam penelitian ini digunakan alternatif cara pertama.

6) Latih JST pada *action* yang sudah dipilih dengan pelatihan SOM Karena kelas 1 atau *action* 1 (Up) pada state sebelumnya, yaitu pada  $S = [0,9 0,7]^T$  terpilih sebagai pemenang, maka pada proses pelatihan SOM saat ini, kelas 1 tersebut akan digunakan sebagai *output* yang diharapkan. Dalam hal ini, SOM seolah-olah diarahkan ke *supervised learning* seperti pada Persamaan 5.

$$\min_{y_i} (\varepsilon) = \min_{y_i} (t - \sum_{i=1}^n (y_i - x_i)^2)^2 \quad (5)$$

Di mana  $t$  menyatakan nilai invers dari nilai Q-value yang sudah di-update, sehingga  $t = 1 - 0,903115 = 0,0969$ . Nilai  $t$  ini setara dengan nilai jarak ideal yang diharapkan atau seperti *output* yang

diharapkan. Dan  $y_i$  merupakan bobot yang akan dicari untuk menggantikan bobot awal  $W=[0,3 \ 0,4]^T$  dari kelas 1. Jika ingin mendapatkan  $y_i$ , maka cara yang paling ideal dengan melakukan penghitungan hasil turunan Persamaan 5 terhadap  $y_i$  sehingga sama dengan 0, seperti pada Persamaan 6. Dan  $n$  menyatakan banyak fitur. Kemudian isi dari  $x_i = [0,9 \ 0,7]^T$ , sama dengan nilai dari state sebelumnya.

$$\frac{\partial}{\partial y_i} (\mathcal{E}) = \frac{\partial}{\partial y_i} (t - \sum_{i=1}^n (y_i - x_i)^2)^2 = 0 \quad (6)$$

Jika hanya 2 fitur ( $n=2$ ), maka  $y_i$  dihitung dengan terlebih dahulu dilakukan random nilai  $y_1=[x_1-vt; x_1+vt]$ , di mana hasil random tersebut harus tetap dalam batasan  $y_1 \in [0;1]$ , lalu menghitung  $y_2$  dengan Persamaan 7, di mana nilai hasil bobot update  $W = [y_1 \ y_2]^T$ .

$$y_2 = x_2 \pm \sqrt{t - (y_1 - x_1)^2} \quad (7)$$

Jika random  $y_1=0,6$ , maka  $y_2 = 0,78$  (pembulatan) atau  $y_2 = 0,67$  (pembulatan). Pilih salah satu, maka didapat nilai hasil bobot update  $W = [0,6 \ 0,78]^T$ . Untuk tetap mempertimbangkan dari nilai bobot sebelumnya  $W=[0,3 \ 0,4]^T$ , maka nilai hasil bobot update yang final dapat dihitung dengan hasil rata-rata tiap dimensinya, sehingga didapat  $W=[0,45 \ 0,59]^T$ .

7) *Update Next state menjadi state saat ini*

$$S = S' = [0,7 \ 0,5]^T$$

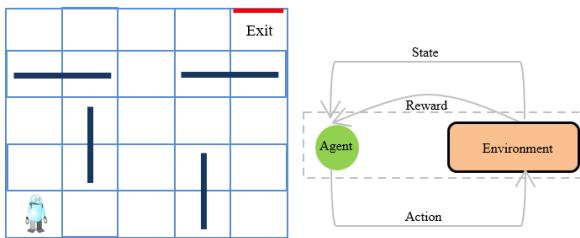
8) Cek kondisi penyimpanan bobot dan kondisi berhenti.

Pada langkah ini, pastikan bobot terakhir telah tersimpan untuk agen ke-1 yang memungkinkan nantinya dapat digunakan atau *load* kembali untuk percobaan berikutnya, atau untuk dilakukan proses *transfer learning* ke agen pada lingkungan yang lain tapi setara dengan lingkungan saat ini (*scalable*), misal pada arena yang lebih besar atau lainnya. Dan proses RL tersebut akan dihentikan dengan misalkan memberikan batasan nilai *score*-nya apabila telah mencapai 100. Jika belum mencapai *score* tersebut, maka lakukan kembali RL, yaitu kembali lagi ke langkah 3. Dan perlu diperhatikan, karena menggunakan penerapan multi-agent, maka setelah agen ke-1 telah selesai dilakukan pembelajaran untuk agen ke-2 dan seterusnya, sampai kedua telah mencapai kondisi berhenti yang dimulai dari langkah ke-1.

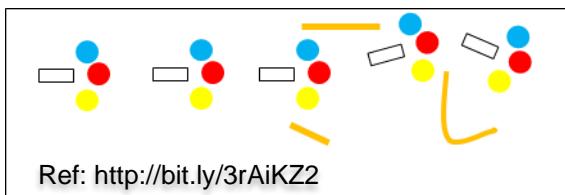
c. Selesai

## 19.6 Tugas Kelompok

1. Dari contoh manualisasi Deep Reinforcement Learning (DRL) di atas, ubahlah algoritma SOM dengan memilih satu dari beberapa algoritma berikut:
  - Extreme Learning Machines (ELM),
  - Learning Vector Quantization (LVQ),
  - Backpropagation,
  - CNN,
  - atau algoritma Deep Learning lainnya
2. Selesaikan kasus Maze berikut dengan menggunakan DRL berbasis ELM.



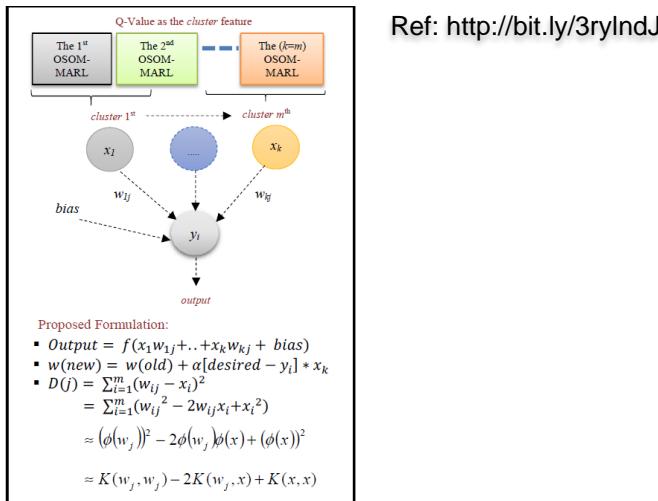
3. Perhatikan skema ilustrasi sensor agen “Smart Car” yang akan dilatih secara mandiri terhadap lingkungannya.



Pada arsitektur algoritma JST, detail dari bagian lapisan input menyatakan state dari lingkungan yang diterima dari sensor agen sebanyak 5 fitur, yaitu sensor merah, sensor kuning, sensor biru, orientasi, -orientasi. Pada 3 fitur pertama semuanya dalam bentuk sensor yang digunakan untuk mendeteksi adanya pasir pada jalan masing-masing dibagian depan (merah), kanan (kuning) dan kiri (biru) dari agen mobil virtual. Sedangkan Pada lapisan output menyatakan action yang memuat tiga keluaran aksi dari agen yaitu Putar 20 derajat searah jarum jam, Putar 20 derajat berlawanan arah jarum jam, dan Tidak berbalik kemanapun (tetap lurus ke depan), yang mana action yang dipilih diambil dari nilai Q-

value hasil prediksi yang paling maksimum. Kemudian setiap aksi tersebut disertai nilai reward, jika mobil menabrak atau menembus pasir (nilainya -5), jika mobil menjauh dari tujuan atau menjauh dari yang menjadi objektifnya (nilainya -0,1), dan jika mendekati objektifnya (nilainya 0,1). Dari penjelasan di atas, buatlah visualisasi dari Arsitektur JST-nya.

#### 4. Perhatikan skema design MARL berikut



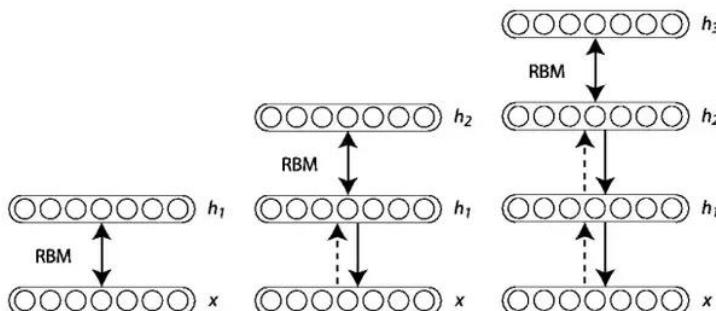
Buatlah contoh manualisasi sederhana untuk perhitungan dari pernyataan bahwa jarak antar data ke- $i$  terhadap setiap bobot input ke- $j$  ( $D_j$ ) dapat menggunakan pendekatan rumus Kernel atau disebut sebagai *kernel trick*.

#### 5. Buatlah kode program untuk mengimplementasikan Deep Reinforcement Learning (DRL) sesuai dengan kasus yang ada di contoh. Lalu bandingkan, pastikan hasil koding dengan manualisasi dicontoh menghasilkan nilai-nilai perhitungan yang sama!

## BAB 20 Algoritma Deep Belief Net (DBN)

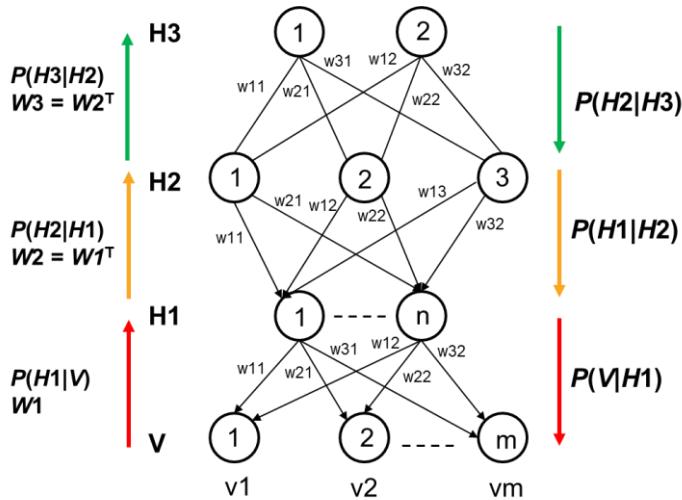
Deep Belief Network adalah salah satu metode non-convolutional pertama yang berhasil menerima dan melakukan proses pelatihan sesuai standar arsitektur dari deep learning. Restricted Boltzmann Machine (RBM) adalah blok bentuk visual dari model probabilistik yang tidak berarah yang mengandung lapisan variabel yang dapat diobservasi dan minimal terdapat satu layer sebagai variabel laten (hidden layer/ darknet layer). RBM dapat ditumpuk (satu di atas yang lain) untuk membentuk model yang lebih dalam. Langkah-langkah dalam DBN:

- Contract Fitur
- Re-Contract Input



Gambar 20.1 Tentang skema DBN

Deep Belief Network adalah generative models (model yang dapat dikembangkan) dengan satu atau beberapa lapisan variabel laten (hidden layer/ darknet layer). Dalam DBN, variabel laten biasanya biner, sedangkan layer/ unit yang terlihat (visible (v) ) mungkin berupa biner atau non-biner (layer input dan layer output, tetapi ada yang menyebutkan bahwa layer output merupakan hidden layer). Tidak ada koneksi intralayer (koneksi dari node ke node dalam 1 layer). Kemudian pada DBN juga biasanya, setiap unit di setiap lapisan terhubung ke setiap unit di setiap lapisan tetangga, meskipun dimungkinkan untuk membangun DBN yang terhubung lebih jarang (misal dibuat dinamis). Koneksi antara dua lapisan teratas tidak diberikan arah. Sedangkan koneksi antara semua lapisan lainnya diberikan arah, dengan panah mengarah ke lapisan yang paling dekat dengan data (layer input).



Gambar 20.2 Ilustrasi Contoh Arsitektur Umum dari DBN

## 20.1 Langkah dan Pembuktian DBN

Langkah-langkah Training DBN base RBM menggunakan Contrastive divergence dengan Gibbs Sampling:

- Ambil data *training* pada *dataset*, satu demi satu, set dalam variabel *visible* (*v*) di input *layer*.
- Positive Phase (bottom-up):** *Update* semua *hidden* unit (Construct) secara paralel menggunakan persamaan berikut (sebagai nilai edge Positive( $E_{ij}$ )).

$$P(H_j = 1 | V) = \sigma\left(B_j + \sum_{i=1}^m w_{ij} v_i\right) \quad (6)$$

- Negative Phase (top-down):** *Update* kembali semua *hidden* (Re-Construct) dengan cara yang sama seperti langkah (2), menggunakan persamaan berikut (sebagai nilai edge Negative( $E_{ij}$ )).

$$P(V_i = 1 | H) = \sigma\left(A_i + \sum_{j=1}^n w_{ij} H_j\right) \quad (6)$$

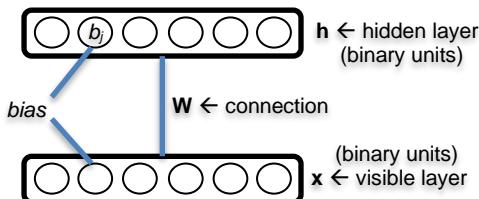
- Update bobot *edge* dengan persamaan berikut (*L* merupakan *learning rate*).

$$Updt(W_{ij}) = W_{ij} + L * (Pos(E_{ij}) - Neg(E_{ij})) \quad (6)$$

- Ulangi langkah (1) sampai (4) untuk semua data *training*, sampai memenuhi kondisi berhenti, misal dengan kondisi

dari  $(\text{Max}(\text{All layer dari nilai Updt}(W_{ij}))) \leq \epsilon \text{psilon}$  atau dengan telah mencapai *iterasiMax*.

Bentuk sederhana dari RBM, *visible layer*, *hidden layer* dan *energy function*.

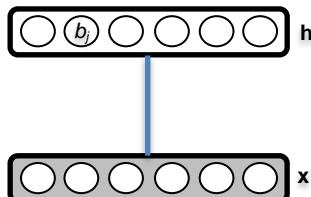


Di mana, *energy* dan distribusi *function* dapat dituliskan sebagai berikut.

$$\begin{aligned} E(x, h) &= -h^T W x - c^T x - b^T h \\ &= -\sum_j \sum_k W_{j,k} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j \end{aligned}$$

$$p(x, h) = \exp(-E(x, h)) / Z$$

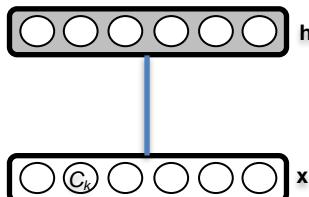
Perhatikan persamaan baku dari probabilitas conditional (*conditional distribution*) dari ilustrasi *hidden* dan *visible* layer. Pada skema yang pertama arahnya dari  $x$  menuju kepada  $h$ , yang pada halaman berikutnya akan diberikan pembuktianya, yaitu kenapa  $p(h|x)$  nilainya sama dengan fungsi sigmoid.



$$p(h|x) = \prod_j p(h_j | x)$$

$$\begin{aligned} p(h_j = 1 | x) &= \frac{1}{1 + \exp(-b_j - W_j \cdot x)} \\ &= \text{sigm}(b_j + W_j \cdot x) \end{aligned}$$

Kemudian pada skema yang kedua arahnya dari  $h$  menuju kepada  $x$ . Pembuktian  $p(x|h)$  juga sama caranya dengan  $p(h|x)$ , sehingga didapatkan fungsi sigmoid sebagai persamaan penyederhanaan akhir.



$$p(x|h) = \prod_k p(x_k | h)$$

$$\begin{aligned} p(x_k = 1 | h) &= \frac{1}{1 + \exp(-c_k + h^T W_k)} \\ &= \text{sigm}(c_k + h^T W) \end{aligned}$$

Pembuktian *conditional probability* pada proses inferensi RBM, misal pada  $p(h|x)$  dapat dibuktikan sebagai berikut.

$$\begin{aligned}
 p(h|x) &= p(x,h)/\sum_{h'} p(x,h') \\
 &= \frac{\exp(h^T Wx + c^T x + b^T h)/Z}{\sum_{h' \in \{0,1\}^H} \exp(h'^T Wx + c^T x + b^T h')/Z} \\
 &= \frac{\exp(\sum_j h_j W_j \cdot x + b_j h_j)}{\sum_{h'_1 \in \{0,1\}} \dots \sum_{h'_H \in \{0,1\}} \exp(\sum_j h'_j W_j \cdot x + b_j h'_j)} \\
 &= \frac{\prod_j \exp(h_j W_j \cdot x + b_j h_j)}{\sum_{h'_1 \in \{0,1\}} \dots \sum_{h'_H \in \{0,1\}} \prod_j \exp(h'_j W_j \cdot x + b_j h'_j)} \\
 &= \frac{\prod_j \exp(h_j W_j \cdot x + b_j h_j)}{\prod_j \left( \sum_{h'_1 \in \{0,1\}} \exp(h'_1 W_1 \cdot x + b_1 h'_1) \right) \dots \left( \sum_{h'_H \in \{0,1\}} \exp(h'_H W_H \cdot x + b_H h'_H) \right)} \\
 &= \frac{\prod_j \exp(h_j W_j \cdot x + b_j h_j)}{\prod_j \left( \sum_{h'_j \in \{0,1\}} \exp(h'_j W_j \cdot x + b_j h'_j) \right)} \\
 &= \frac{\prod_j \exp(h_j W_j \cdot x + b_j h_j)}{\prod_j (1 + \exp(b_j + W_j \cdot x))} \\
 &= \prod_j \left[ \frac{\exp(h_j W_j \cdot x + b_j h_j)}{1 + \exp(b_j + W_j \cdot x)} \right] \\
 &= \prod_j p(h_j | x)
 \end{aligned}$$

Sehingga, ketika  $h_j = 1$ , maka

$$\begin{aligned}
 p(h_j | x) &= \frac{\exp(h_j W_j \cdot x + b_j h_j)}{1 + \exp(b_j + W_j \cdot x)} \\
 p(h_j = 1 | x) &= \frac{\exp(W_j \cdot x + b_j)}{1 + \exp(b_j + W_j \cdot x)} = \frac{\exp(b_j + W_j \cdot x)}{1 + \exp(b_j + W_j \cdot x)} \\
 &= \frac{1}{1 + \exp(-b_j - W_j \cdot x)} = \text{sigm}(b_j + W_j \cdot x)
 \end{aligned}$$

## 20.2 Studi Kasus: DBN untuk Klasifikasi Data Sederhana

### 20.2.1 Proses Training

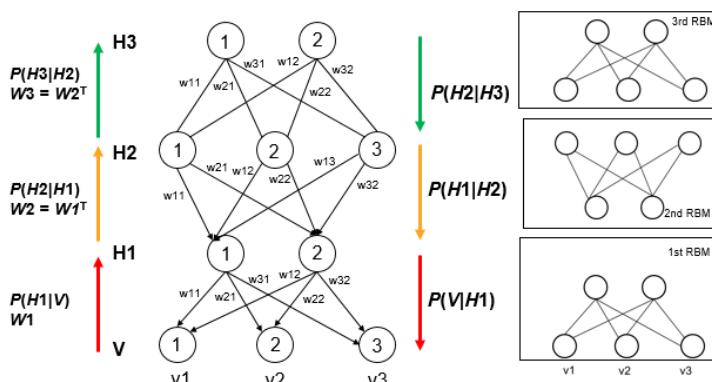
- Perhatikan dataset sederhana berikut:

| Data ke- <i>i</i> | Suka Sepak Bola (0/1) | Suka Volley (0/1) | Suka Catur Bola (0/1) | Kelas (Out/In = 1/0) |
|-------------------|-----------------------|-------------------|-----------------------|----------------------|
| 1                 | 1                     | 1                 | 0                     | Out                  |
| 2                 | 0                     | 0                 | 1                     | In                   |
| 3                 | 0                     | 1                 | 0                     | Out                  |

#### Keterangan:

Pada Fitur “0” = tidak suka, “1” = suka. Sedangkan kelas “Out” menyatakan “Permainan Outdoor”, sedangkan “In” menyatakan “Permainan Indoor”.

- Perhatikan arsitektur DBN berikut:



Di mana goal DBN ini untuk mengoptimalkan nilai-nilai bobot (w) di antara layer

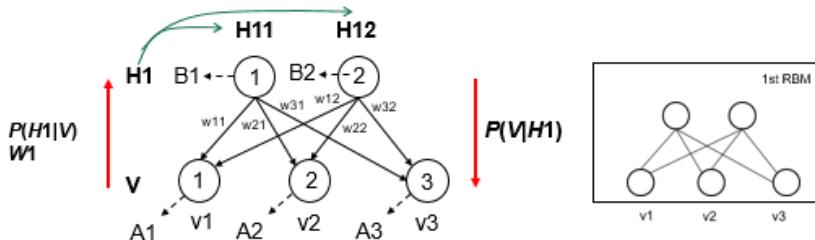
- Misal diketahui nilai awal parameter sebagai berikut:

$v1 = 1, v2 = 1$  dan  $v3 = 0 \rightarrow$  Data train ke-1

$w11 = w21 = w31 = 0,5, w12 = 0, w22 = 1, w32 = 0 \rightarrow$  misal dari  $\text{rand}[-1;1]$

$$\left. \begin{array}{l} A1 = A2 = A3 = 0,2 \\ B1 = B2 = 0,2 \end{array} \right\} \text{nilai bias}$$

### Positive Phase



### Update H11:

$$P(H_{1,j=1} = 1 | V) = \sigma \left( B_{j=1} + \sum_{i=1}^{m=3} w_{i,j=1} v_i \right) = \sigma(B_1 + w_{11}v_1 + w_{21}v_2 + w_{31}v_3)$$

$$= \sigma(0,2 + 0,5*1 + 0,5*1 + 0,5*0) = \sigma(1,2) = \frac{1}{1+e^{(-1,2)}} = 0,7685$$

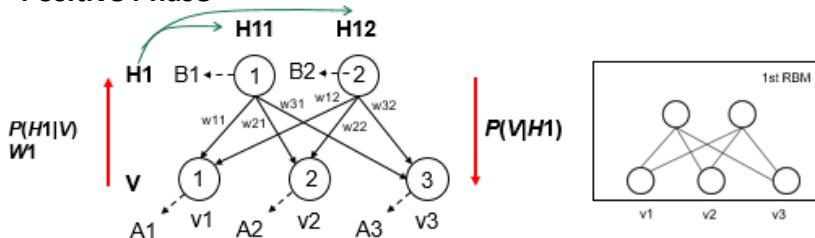
▪ Misal diketahui nilai awal parameter sebagai berikut:

$v1 = 1, v2 = 1$  dan  $v3 = 0 \rightarrow$  Data train ke-1

$w11 = w12 = w21 = w22 = w31 = w32 = 0,5 \rightarrow$  misal dari  $\text{rand}[-1;1]$

$$\left. \begin{array}{l} A1 = A2 = A3 = 0,2 \\ B1 = B2 = 0,2 \end{array} \right\} \text{nilai bias}$$

### Positive Phase



### Update H12:

$$P(H_{1,j=2} = 1 | V) = \sigma\left(B_{j=2} + \sum_{i=1}^{m=3} w_{i,j=2} v_i\right) = \sigma(B_2 + w_{12}v_1 + w_{22}v_2 + w_{32}v_3)$$

$$= \sigma(0,2 + 0,5*1 + 0,5*1 + 0,5*0) = \sigma(1,2) = \frac{1}{1+e^{(-1,2)}} = 0,7685$$

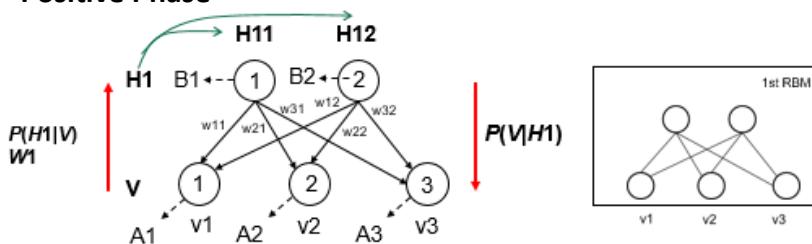
- Misal diketahui nilai awal parameter sebagai berikut:

$v1 = 1, v2 = 1$  dan  $v3 = 0 \rightarrow$  Data train ke-1

$w11 = w21 = w31 = 0,5, w12 = 0, w22 = 1, w32 = 0 \rightarrow$  misal dari  $\text{rand}[-1;1]$

$$\begin{aligned} A1 &= A2 = A3 = 0,2 \\ B1 &= B2 = 0,2 \end{aligned} \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{nilai bias}$$

### Positive Phase



### Update H12:

$$P(H_{1,j=2} = 1 | V) = \sigma\left(B_{j=2} + \sum_{i=1}^{m=3} w_{i,j=2} v_i\right) = \sigma(B_2 + w_{12}v_1 + w_{22}v_2 + w_{32}v_3)$$

$$= \sigma(0,2 + 0*1 + 0,5*1 + 0*0) = \sigma(0,7) = \frac{1}{1+e^{(-0,7)}} = 0,6681$$

- Misal diketahui nilai awal parameter sebagai berikut:

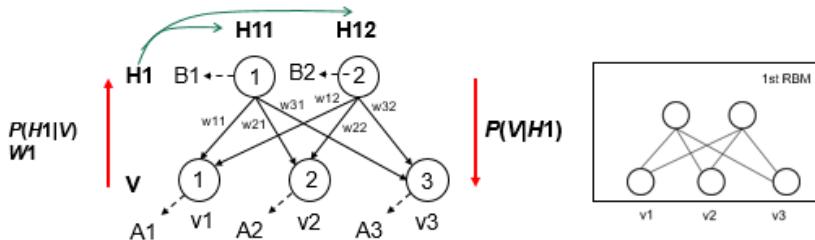
$v1 = 1, v2 = 1$  dan  $v3 = 0 \rightarrow$  Data train ke-1

$w11 = w21 = w31 = 0,5, w12 = 0, w22 = 1, w32 = 0 \rightarrow$  misal dari  $\text{rand}[-1;1]$

$$\begin{aligned} A1 &= A2 = A3 = 0,2 \end{aligned} \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{nilai bias}$$

$$B1 = B2 = 0,2$$

### Negative Phase



### Update V1:

$$P(V_{i=1} = 1 | H1) = \sigma \left( A_{i=1} + \sum_{j=1}^{n=2} w_{i=1,j} H_{1j} \right) = \sigma(A_1 + w_{11}H_{11} + w_{12}H_{12}) \\ = \sigma(0,2 + 0,5 * 0,7685 + 0 * 0,6681) = 0,6420$$

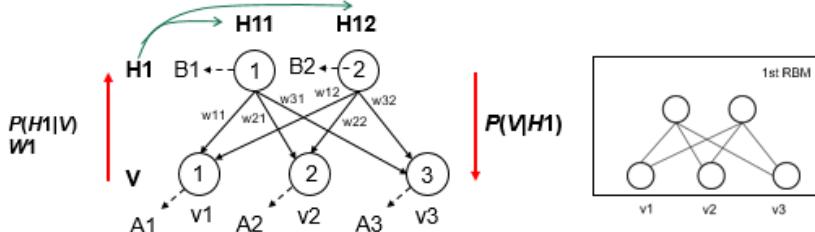
- Misal diketahui nilai awal parameter sebagai berikut:

$v1 = 1, v2 = 1$  dan  $v3 = 0 \rightarrow$  Data train ke-1

$w11 = w21 = w31 = 0,5, w12 = 0, w22 = 1, w32 = 0 \rightarrow$  misal dari  $\text{rand}[-1;1]$

$$\begin{aligned} A1 &= A2 = A3 = 0,2 \\ B1 &= B2 = 0,2 \end{aligned} \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{nilai bias}$$

### Negative Phase



### Update V2:

$$P(V_{i=2} = 1 | H1) = \sigma \left( A_{i=2} + \sum_{j=1}^{n=2} w_{i=2,j} H_{1j} \right) = \sigma(A_2 + w_{21}H_{11} + w_{22}H_{12})$$

$$= \sigma(0,2 + 0,5 * 0,7685 + 1 * 0,6681) = 0,7777$$

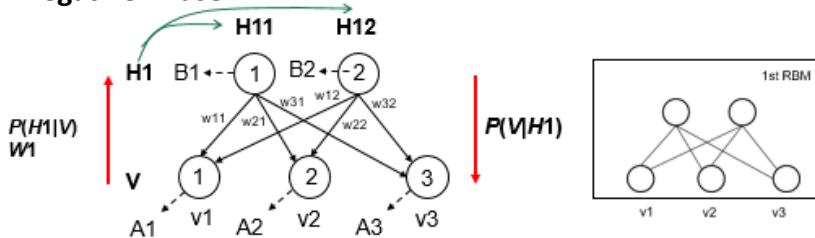
- Misal diketahui nilai awal parameter sebagai berikut:

$v_1 = 1, v_2 = 1$  dan  $v_3 = 0 \rightarrow$  Data train ke-1

$w_{11} = w_{21} = w_{31} = 0,5, w_{12} = 0, w_{22} = 1, w_{32} = 0 \rightarrow$  misal dari  $\text{rand}[-1;1]$

$$\left. \begin{array}{l} A_1 = A_2 = A_3 = 0,2 \\ B_1 = B_2 = 0,2 \end{array} \right\} \text{nilai bias}$$

### Negative Phase



### Update V3:

$$P(V_{i=3} = 1 | H1) = \sigma \left( A_{i=3} + \sum_{j=1}^{n=2} w_{i=3,j} H_{1j} \right) = \sigma(A_3 + w_{31}H_{11} + w_{32}H_{12})$$

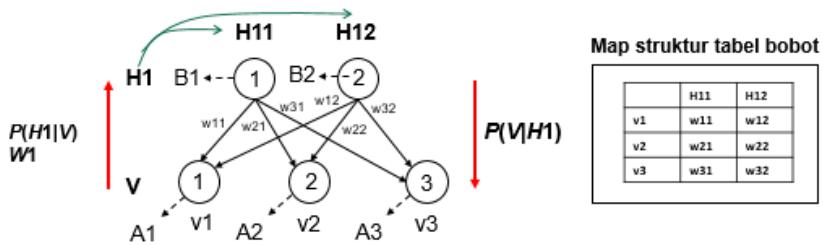
$$= \sigma(0,2 + 0,5 * 0,7685 + 0 * 0,6681) = 0,6420$$

- Misal diketahui nilai awal parameter sebagai berikut:

$v_1 = 1, v_2 = 1$  dan  $v_3 = 0 \rightarrow$  Data train ke-1

$w_{11} = w_{21} = w_{31} = 0,5, w_{12} = 0, w_{22} = 1, w_{32} = 0 \rightarrow$  misal dari  $\text{rand}[-1;1]$

$$\left. \begin{array}{l} A_1 = A_2 = A_3 = 0,2 \\ B_1 = B_2 = 0,2 \end{array} \right\} \text{nilai bias, dan } L = 0,5$$



### Update Bobot:

$$Updt(W_{ij}) = W_{ij} + L * (Positive(E_{ij}) - Negative(E_{ij}))$$

$$Updt(W_{11}) = W_{11} + L * (P(H_{1,j=1} = 1 | V) - P(V_{i=1} = 1 | H1))$$

$$= 0,5 + 0,5 * (0,7685 - 0,6420) = 0,5633$$

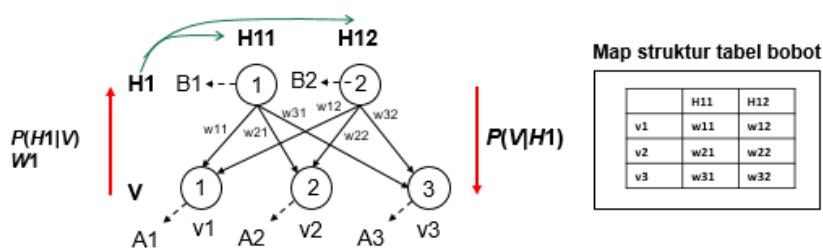
- Misal diketahui nilai awal parameter sebagai berikut:

$v1 = 1, v2 = 1$  dan  $v3 = 0 \rightarrow$  Data train ke-1

$w11 = w21 = w31 = 0,5, w12 = 0, w22 = 1, w32 = 0 \rightarrow$  misal dari rand[-1;1]

$A1 = A2 = A3 = 0,2$   
 $B1 = B2 = 0,2$

} nilai bias, dan  $L = 0,5$



### Update Bobot W12:

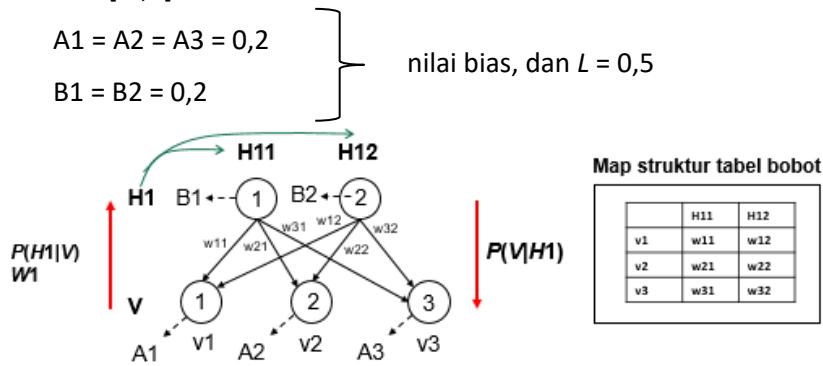
$$Updt(W_{i=1,j=2}) = W_{i=1,j=2} + L * (P(H_{1,j=2} = 1 | V) - P(V_{i=1} = 1 | H1))$$

$$= 0 + 0,5 * (0,6681 - 0,6420) = 0,0131$$

- Misal diketahui nilai awal parameter sebagai berikut:

$v1 = 1, v2 = 1$  dan  $v3 = 0 \rightarrow$  Data train ke-1

$w_{11} = w_{21} = w_{31} = 0,5$ ,  $w_{12} = 0$ ,  $w_{22} = 1$ ,  $w_{32} = 0 \rightarrow$  misal dari  $\text{rand}[-1;1]$



### Update Bobot W21:

$$\begin{aligned} Updt(W_{i=2,j=1}) &= W_{i=2,j=1} + L * (P(H_{1,j=1} = 1 | V) - P(V_{i=2} = 1 | H1)) \\ &= 0,5 + 0,5 * (0,7685 - 0,7777) = 0,4954 \end{aligned}$$

### Update Bobot W22:

$$\begin{aligned} Updt(W_{i=2,j=2}) &= W_{i=2,j=2} + L * (P(H_{1,j=2} = 1 | V) - P(V_{i=2} = 1 | H1)) \\ &= 1 + 0,5 * (0,6681 - 0,7777) = 0,9452 \end{aligned}$$

### Update Bobot W31:

$$\begin{aligned} Updt(W_{i=3,j=1}) &= W_{i=3,j=1} + L * (P(H_{1,j=1} = 1 | V) - P(V_{i=3} = 1 | H1)) \\ &= 0,5 + 0,5 * (0,7685 - 0,6420) = 0,5633 \end{aligned}$$

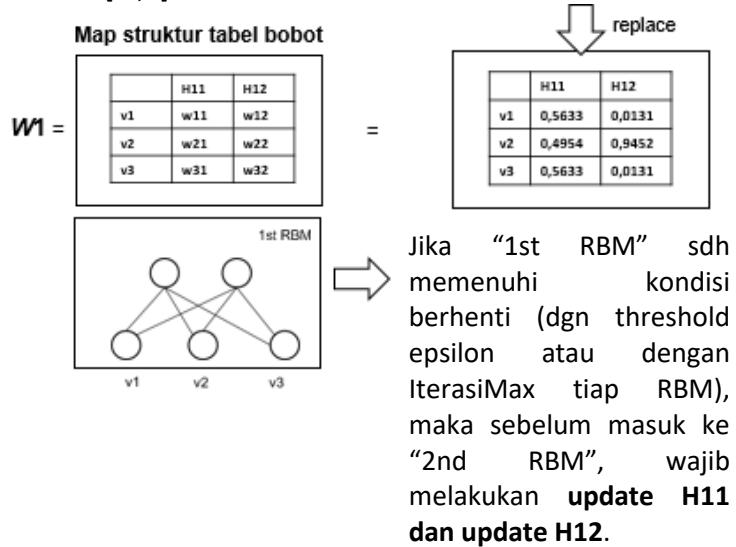
### Update Bobot W32:

$$\begin{aligned} Updt(W_{i=3,j=2}) &= W_{i=3,j=2} + L * (P(H_{1,j=2} = 1 | V) - P(V_{i=3} = 1 | H1)) \\ &= 0 + 0,5 * (0,6681 - 0,6420) = 0,0131 \end{aligned}$$

- Update bobot:

$v1 = 1, v2 = 1$  dan  $v3 = 0 \rightarrow$  Data train ke-1, untuk data train selanjutnya akan diload jika telah menyelesaikan “3rd RBM”, dst sampai semua data train telah dilakukan proses pelatihan.

$w11 = w21 = w31 = 0,5, w12 = 0, w22 = 1, w32 = 0 \rightarrow$  misal dari  $\text{rand}[-1;1]$



- Jika telah masuk ke “2nd RBM”, maka set  $W2$  dengan persamaan berikut

$$W2 = W1^T$$

**Map struktur tabel bobot**

$W1 =$ 

|    | H11 | H12 |
|----|-----|-----|
| v1 | w11 | w12 |
| v2 | w21 | w22 |
| v3 | w31 | w32 |

$=$

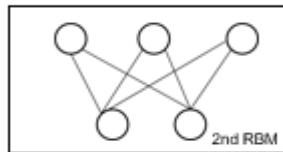
|    | H11    | H12    |
|----|--------|--------|
| v1 | 0,5633 | 0,0131 |
| v2 | 0,4954 | 0,9452 |
| v3 | 0,5633 | 0,0131 |

$W2 =$ 

|     | H21 | H22 | H23 |
|-----|-----|-----|-----|
| H11 | w11 | w12 | w13 |
| H12 | w21 | w22 | w23 |

$=$

|     | H21    | H22    | H23    |
|-----|--------|--------|--------|
| H11 | 0,5633 | 0,4954 | 0,5633 |
| H12 | 0,0131 | 0,9452 | 0,0131 |



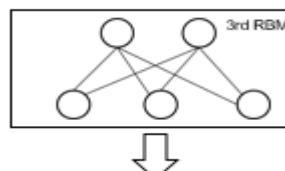
Lakukan perhitungan “2nd RBM” seperti pada “1st RBM”, jika sdh memenuhi kondisi berhenti (dgn threshold epsilon atau dengan IterasiMax tiap RBM), maka sebelum masuk ke “3rd RBM”, wajib melakukan **update H21, update H22 dan update H23**.

- Jika telah masuk ke “3rd RBM”, maka set  $W3$  dengan persamaan berikut

$$W3 = W2^T$$

Map struktur tabel bobot

$$W2 = \begin{array}{|c|c|c|c|} \hline & H21 & H22 & H23 \\ \hline H11 & w11 & w12 & w13 \\ \hline H12 & w21 & w22 & w23 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline & H21 & H22 & H23 \\ \hline H11 & .. & .. & .. \\ \hline H12 & .. & .. & .. \\ \hline \end{array}$$
  
$$W3 = \begin{array}{|c|c|c|} \hline & H31 & H32 \\ \hline H21 & w11 & w12 \\ \hline H22 & w21 & w22 \\ \hline H23 & w31 & w32 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline & H31 & H32 \\ \hline H21 & .. & .. \\ \hline H22 & .. & .. \\ \hline H23 & .. & .. \\ \hline \end{array}$$

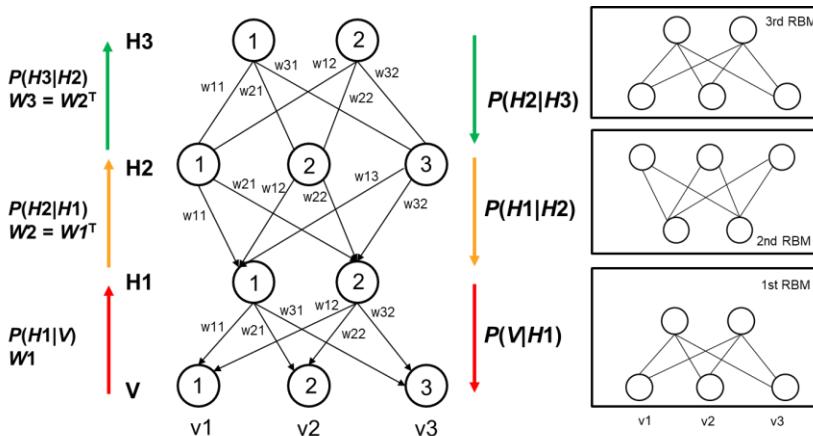


Lakukan perhitungan “3rd RBM” seperti pada “1st RBM”, jika sdh memenuhi kondisi berhenti (dgn threshold epsilon atau dengan IterasiMax tiap RBM), maka sebelum load data train ke-2, wajib melakukan **update H31, dan update H32**.

- Jika pada “3rd RBM” telah selesai memproses data train terakhir, Lakukan perulangan sampai IterasiMaxGlobal (iterasi ini beda dengan IterasiMax di tiap RBM).

## 20.2.2 Proses Testing

Untuk proses testing, misal data uji-nya adalah  $v1 = 0$ ,  $v2 = 0$  dan  $v3 = 0$ , maka gunakan bobot  $W1$ ,  $W2$ , dan  $W3$  yang telah didapatkan dari proses training kemudian cukup lakukan “Positive Phase” saja, hingga ketika telah sampai pada “3rd RBM” dan telah dihitung update  $H31$  dan  $H32$ , ambil yang nilainya paling Max dengan  $k = \text{ArgMax}\{H31, H32\}$ , jika  $k = 0$ , maka masuk kelas “In”, dan sebaliknya jika  $k = 1$ , maka masuk kelas “Out”.



## 20.3 Tugas Kelompok

- Perhatikan dataset sederhana berikut:

| Data ke- <i>i</i> | Suka Sepak Bola | Suka Volley (0/1) | Suka Catur (0/1) | Kelas (Out/In = 1/0) |
|-------------------|-----------------|-------------------|------------------|----------------------|
| 1                 | 0               | 1                 | 0                | Out                  |
| 2                 | 1               | 0                 | 1                | In                   |
| 3                 | 0               | 1                 | 1                | Out                  |

Keterangan:

Pada Fitur “0” = tidak suka, “1” = suka. Sedangkan kelas “Out” menyatakan “Permainan Outdoor”, sedangkan “In” menyatakan “Permainan Indoor”.

Selesaikan Proses Traning menggunakan Deep Belief Net!

2. Dari hasil no.1, lakukan proses Testing untuk menentukan Kelas dari data uji = [1 1 1]!
3. Apa kepanjangan dari RBM? dan Jelaskan peranan RBM dalam Deep Belief Network (DBN)!
4. Pada Langkah-langkah Training DBN base RBM menggunakan Contrastive divergence dengan Gibbs Sampling, terdapat istilah “Positive Phase” dan “Negative Phase”. Jelaskan perbedaan kedua istilah tersebut.
5. Buatlah Pembuktian *conditional probability* pada proses inferensi RBM, misal pada  $p(x|h)$  seperti pada pembuktian  $p(h|x)$ , sehingga didapatkan rumus sigmoid seperti berikut.

$$p(x|h) = \prod_k p(x_k | h)$$

$$\begin{aligned} p(x_k = 1 | h) &= \frac{1}{1 + \exp(-c_k + h^T W_k)} \\ &= \text{sigm}(c_k + h^T W) \end{aligned}$$

6. Buatlah kode program untuk mengimplementasikan DBN sesuai dengan kasus yang ada di contoh. Lalu bandingkan, pastikan hasil koding dengan manualisasi dicontoh menghasilkan nilai-nilai perhitungan yang sama!

## BAB 21 Topik Lanjut Pada Deep Learning

### 21.1 Pengantar

Pada topik lanjut Deep Learning ini, digunakan lebih dari satu algoritma atau dilakukan hibridasi dengan metode atau Teknik lainnya yang semuanya berbasis AI. Hibridasi ini dilakukan untuk mendapatkan hasil yang lebih baik dari metode standar yaitu dengan harapan didapatkan peningkatan hasil nilai evaluasi yang lebih signifikan. Makna lainnya dari hibridasi adalah membuat gabungan antar algoritma untuk dapat saling bekerja sama dalam membantu menyelesaikan kasus dari yang sederhana maupun sampai yang kompleks. Di dalam hibridasi ini diberikan beberapa sub bab antara algoritma yang sama-sama termasuk jaringan syaraf titruan, algoritma metaheuristik maupun lainnya.

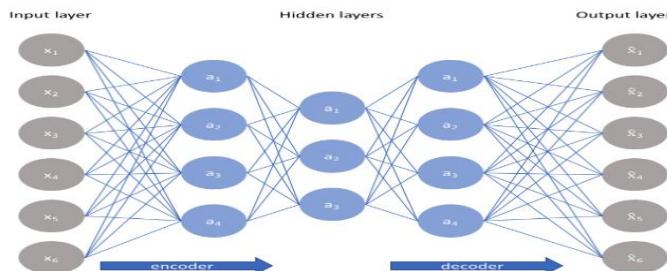


## 21.2 Improve Autoencoder untuk Fast Mapping Fitur dengan Kernel ELM (KELM) dan ELM Murni

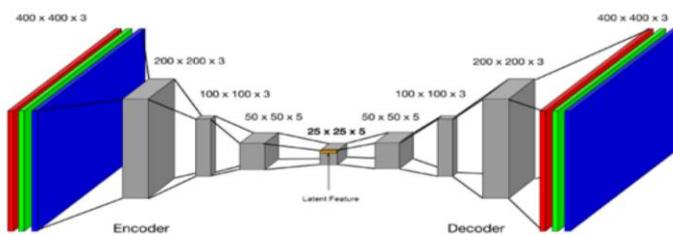
Deep Convolution NN (DCNN) Autoencoder atau Convolution Autoencoder (CAE) berbasis Kernel Extreme Learning Machine (ELM), digunakan untuk mendapatkan *latent feature* secara mudah dan cepat serta mendapatkan hasil fitur mapping, yang secara umum dari dimensi tinggi ke dimensi rendah meskipun juga bisa berapapun dimensinya, dan dapat diterapkan pada berbagai macam data.

### 21.2.1 Autoencoder (AE) vs CAE vs PCA

Selain CNN, terdapat juga CAE yang dilatih secara *supervised learning*, tetapi tidak membutuhkan kelas dari data sebagai target. Berikut ilustrasi Arsitektur AE vs CAE.



Gambar 21.1 Contoh arsitektur Autoencoder (AE)

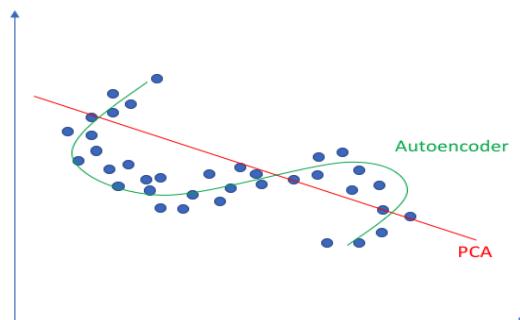


Gambar 21.2 Contoh arsitektur CAE

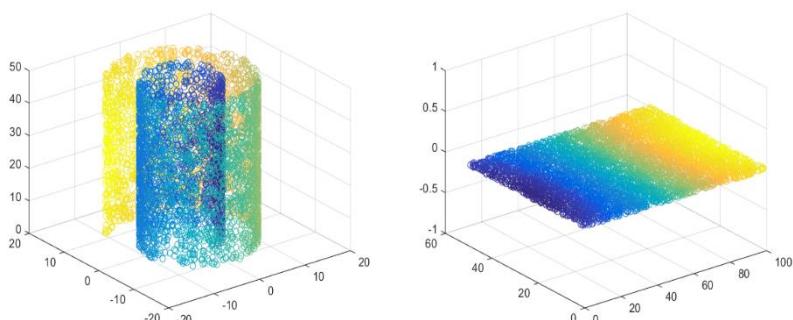
*Autoencoder* atau disebut dengan istilah *auto-associator* adalah model *neural network* yang memiliki masukan dan keluaran yang sama

(set output *layer* dengan input *layer*), memiliki bobot (*weight*) yang sama antara tahap *encoder* dan *decoder*-nya atau disebut *tied weight*, serta berusaha mempelajari data masukan dan berusaha merekonstruksi kembali masukan tersebut dengan informasi yang lebih sedikit (atau disebut *latent feature/bottleneck*).

Kemudian, jika dibandingkan dengan Principal Component Analysis (PCA), di mana PCA ini adalah metode paling umum untuk Dimensionality Reduction, karena sederhana dan mampu mengeliminasi berdasarkan korelasi antar variabel masukan. Namun, **PCA** hanya mereduksi data secara **linear**, dan tidak memperhatikan hubungan antar data secara spasial (misal, jika data gambar, satu data pixel yang nilainya dekat tetapi jauh dari segi letaknya) ataupun data yang **non-linear**. **Autoencoder** sangat *capable* dalam melakukan reduksi data linear maupun non-linear.



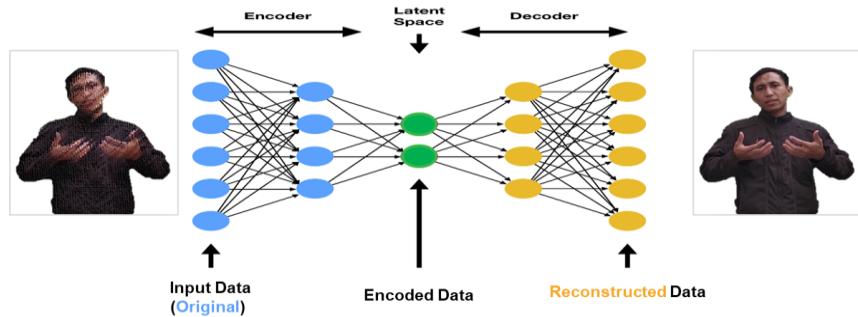
Gambar 21.3 Linear vs nonlinear reduksi dimensi pada 2D



Gambar 21.4 Contoh Sebaran Data pada 3D nonlinear, dan Sebaran Data 2D linear pada 3D

## 21.2.2 Evaluasi Autoencoder

Perhatikan arsitektur AE berikut, untuk memudahkan pemahaman dalam menghitung nilai evaluasi hasil dari Autoencoder.



Gambar 21.5 Ilustrasi arsitektur AE

Hitung performa Autoencoder sebagai nilai Evaluasi, pada kasus misal untuk reduksi dimensi.

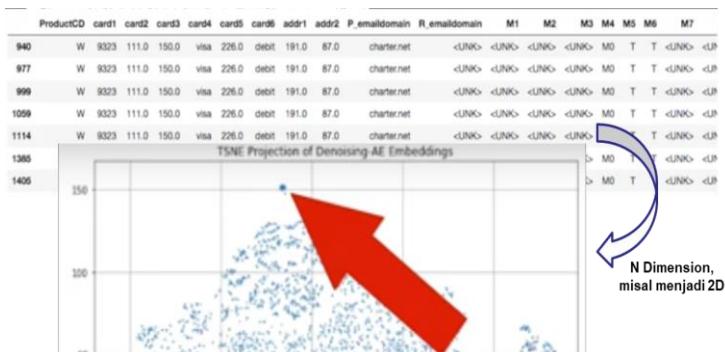
**Error = Reconstructed – Original, atau**

**Error = Decoder(Encoder( $X$ )) –  $X$ , di mana  $X$  = Original**

Pada kasus misal untuk denoising.

**Error = Decoder(Encoder( $X$  + noise )) –  $X$ , di mana  $X$  = Original**

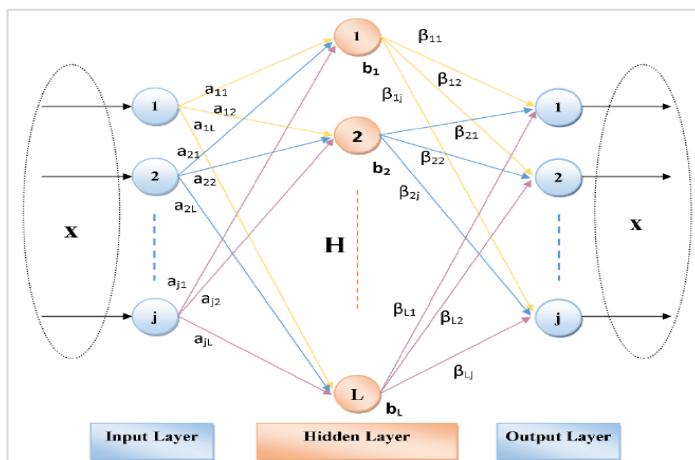
Sedangkan pada contoh kasus misal untuk deteksi *anomaly*, misal berasal dari data berdimensi tinggi, kemudian diubah menjadi dimensi 3D atau 2D dengan model Autoencoder, seperti pada Gambar berikut. Di mana Anomaly Score = Reconstruction Error.



Gambar 21.6 Bagian yang ditunjuk Panah Merah merupakan Outlier

### 21.2.3 Studi Kasus: Deep ELM Kernel atau non-Kernel dengan Autoencoder

Deep ELM sangat mumpuni dalam *reducing time* dan *generalization capabilities* dari segi hasil yang didapat. Meskipun Deep ELM menggunakan CPU, tetapi tetap lebih cepat dari CNN models yang menggunakan GPU. Deep ELM model didalamnya terdapat ELM auto-encoder kernels (maupun digunakan tanpa kernel juga bisa) yang sangat direkomendasikan untuk *real-time application*. Hal tersebut berdasarkan hasil penelitian Deep ELM vs CNN (Altan, G., Kutlu, Y., 2018).



Gambar 21.7 Structure ELM Autoencoder untuk generating presentations “Deep model”

Dan berikut adalah detail hasil perbandingan dari (Altan, G., Kutlu, Y., 2018) terkait CNN vs Deep ELM berdasarkan kecepatan *training* dan nilai performasi akurasinya.

Tabel 21.1 Perbandingan CNN vs Deep ELM

|                             | Metode   | Akurasi | Waktu <i>Training</i> |
|-----------------------------|----------|---------|-----------------------|
| MNIST character recognition | Deep ELM | 99,14%  | 281,37 detik (CPU)    |
|                             | CNN      | 99,05%  | 19 Jam (GPU)          |
| 3D Shape Classification     | Deep ELM | 81,39%  | 306,4 detik (CPU)     |
|                             | CNN      | 77,32%  | > 2 hari (GPU)        |

|                          |           |        |                 |
|--------------------------|-----------|--------|-----------------|
| Traffic sign recognition | Deep ELM  | 99,56% | 209 detik (CPU) |
|                          | CNN       | 99,46% | > 37 Jam (GPU)  |
|                          | CNN + ELM | 99,48% | > 5 Jam (CPU)   |

Untuk memudahkan dalam memahami manualisasi detail dari Autoencoder ELM dan Kernel ELM, Anda harus membaca dan pahami dulu manualisasi ELM pada buku Analisis Big Data dari link "<http://bit.ly/2x8ta9S>" dan konsep Kernel SVM pada buku ini.

#### Source Code 21.1 Autoencoder dengan Kernel ELM (KELM) & ELM

Autoencoder dengan ELM dan Kernel ELM (KELM) ~ Deep KELM dan Deep ELM

```
import pandas as pd
import numpy as np
--
def myrandfloat(mbaris,nkolom,lower,upper):
 #BatasRANDplusOne=10000

 BatasRANDplusOne=max(10000,2*np.math.ceil(upper)
)
 #mbaris=1
 #nkolom=1

 Rand_Sample=np.random.randint(BatasRANDplusOne,s
ize=(mbaris,nkolom))
 min_Rand_Sample = 0
 max_Rand_Sample = BatasRANDplusOne - 1
 upper_boundary= upper
 lower_boundary= lower
 normalize_Rand_Sample_minMax=((Rand_Sample-
min_Rand_Sample)/(max_Rand_Sample-
min_Rand_Sample))*(upper_boundary-
lower_boundary)+lower_boundary
 #print(normalize_Rand_Sample_minMax)
 #x = normalize_Rand_Sample_minMax
 return normalize_Rand_Sample_minMax
--
```

```
----- batas awal proses Encoder &
Decoder dari Autoencoder base Kernel ELM -----
Rev. 1
--

langkah-langkahnya (alternatif ke-1): --> Rev
1
1. buat matrik omega atau matrik kernel antar
data
2. lanjutkan seperti Autoencoder base ELM
murni

langkah-langkahnya (alternatif ke-2): --> Rev
2
1. lakukan seperti Autoencoder base ELM murni
2. buat matrik omega atau matrik kernel antar
data hasil encode
3. lanjutkan seperti Autoencoder base ELM
murni

langkah-langkahnya (alternatif ke-3):
1. lakukan seperti Autoencoder base ELM murni
2. buat matrik omega atau matrik kernel antar
data hasil encode
3. lanjutkan seperti Autoencoder base ELM
murni
--

membuat fungsi-fungsi
fungsi kernel polynomial, dimana c menyatakan
konstanta, d menyatakan derajat/ degree
def kernelPoly(x,y,c,d):
 return np.math.pow((np.dot(x,y)+c),d)
--

Tahap 1: Load data atau generate data (ini
untuk ilustrasi memudahkan pemahaman)
Misal menggunakan "alternatif ke-1" dengan
dataset (ada fitur + target)
misal untuk kasus Regresi
generate dataset, dlm range
[Rnd_Lower;Rnd_Upper]
N_Train_kelm = 9
N_dimFeature_kelm = 3
N_dimTarget_kelm = 1
N_dim_kelm = N_dimFeature_kelm +
N_dimTarget_kelm
print('N_Train_kelm = ',N_Train_kelm)
print('N_dimFeature_kelm = ',N_dimFeature_kelm)
```

```
print('N_dimFeature_kelm = ', N_dimTarget_kelm)
print()
Rnd_Lower = 1
Rnd_Upper = 15

dataset disini, hanya berisi fitur-fitur data
arr_dataset_kelm =
myrandfloat(N_Train_kelm, N_dim_kelm, Rnd_Lower, Rn
d_Upper)
print('Hasil Generate Dataset (dlm bentuk Array
2D):')
print(arr_dataset_kelm.round(2))

Creating pandas dataframe from numpy array
agar nama kolom-kolom jadi terlihat sekaligus
print()
print('Hasil Generate Dataset (dlm bentuk
Dataframe):')
df_dataset_kelm = \
pd.DataFrame({'Fitur ke-1': arr_dataset_kelm[:, 0], \
 'Fitur ke-2': arr_dataset_kelm[:, 1], \
 'Fitur ke-3': arr_dataset_kelm[:, 2], \
 'Target': arr_dataset_kelm[:, 3]})
display(df_dataset_kelm.round(2))
--

Tahap 2: inisialisasi untuk normalisasi data
dan memilih mana data train dan test
& hitung matrik kernel untuk data train dan
test
misal diset langsung dari nilai Rnd_Lower dan
Rnd_Upper
minimum = Rnd_Lower
maksimum = Rnd_Upper

print('Nilai minimum dataset = ', minimum)
print('Nilai maksimum dataset = ', maksimum)
print()

arr_dataset_kelm_norm = (arr_dataset_kelm-
minimum) / (maksimum-minimum)
print('Hasil Normalisasi Data:')
print(arr_dataset_kelm_norm.round(2))

memilih data training dan testing, misal diset
berikut
```

```
data_train_kelm_norm =
arr_dataset_kelm_norm[:6,:]
fitur_data_train_kelm_norm =
data_train_kelm_norm[:,3]
target_data_train_kelm_norm =
data_train_kelm_norm[:,3]
NTrain = len(data_train_kelm_norm)
print()
print('Data Train kelm norm:')
print(data_train_kelm_norm.round(2))
#print(fitur_data_train_kelm_norm)
#print(target_data_train_kelm_norm)

data_test_kelm_norm =
arr_dataset_kelm_norm[6:9,:]
fitur_data_test_kelm_norm =
data_test_kelm_norm[:,3]
target_data_test_kelm_norm =
data_test_kelm_norm[:,3]
NTest = len(data_test_kelm_norm)
print()
print('Data Test kelm norm:')
print(data_test_kelm_norm.round(2))
#print(fitur_data_test_kelm_norm)
#print(target_data_test_kelm_norm)

hitung matrik kernel untuk data training
matrik_kernel_data_train_kelm_norm =
np.zeros((NTrain,NTrain))
for i in range(NTrain):
 for j in range(i,NTrain):
 matrik_kernel_data_train_kelm_norm[i][j] = \
kernelPoly(fitur_data_train_kelm_norm[i],fitur_
data_train_kelm_norm[j],1,2)
 matrik_kernel_data_train_kelm_norm[j][i] =
matrik_kernel_data_train_kelm_norm[i][j]
print()
print('Matrik Kernel Data Train kelm norm:')
print(matrik_kernel_data_train_kelm_norm.round(2))

hitung matrik kernel untuk data testing
matrik_kernel_data_test_kelm_norm =
np.zeros((NTest,NTrain))
for i in range(NTest):
 for j in range(i,NTrain):
```

```
matrik_kernel_data_test_kelm_norm[i][j] = \
kernelPoly(fitur_data_test_kelm_norm[i],fitur_d
ata_train_kelm_norm[j],1,2)
 #matrik_kernel_data_train_kelm_norm[j][i] =
matrik_kernel_data_train_kelm_norm[i][j]
print()
print('Matrik Kernel Data Test kelm norm:')
print(matrik_kernel_data_test_kelm_norm.round(2)
)
--
Tahap 3.1: Proses Encode
lanjutkan seperti Autoencoder base ELM murni
#set bebas, bisa direduksi (dikecilkan dari
dimensi awalnya) atau disparse (dinaikkan
diemsinya)
banyak_hidden_neuron_ae_kelm = 2
banyak_fitur_kelm = NTrain
bobot =
myrandfloat(banyak_hidden_neuron_ae_kelm,
banyak_fitur_kelm,-0.5,0.5)
#bias =
np.random.rand(banyak_hidden_neuron_ae_kelm)
flatten() digunakan utk convert ND to 1D array
bias =
myrandfloat(1,banyak_hidden_neuron_ae_kelm,0.000
0001,0.999999).flatten()
#print(bobot.round(2))
#print()
#print(bias.round(2))

#print()
#Hitung Matrik H sebagai hasil Encoder data
training pada Autoencoder
print('Hitung Matrik H sebagai hasil Encoder
pada Autoencoder:')
h = 1/(1 + np.exp(-
(np.dot(matrik_kernel_data_train_kelm_norm,
np.transpose(bobot)) + bias)))
print(h.round(2))

#
Tahap 3.2: Proses Decode
lanjutkan seperti Autoencoder base ELM murni
Load matrik H, untuk Hitung Matrik Hplus
```

```
h_plus =
np.dot(np.linalg.inv(np.dot(np.transpose(h),h)),
np.transpose(h))
#print(h_plus.round(2))

#print()

Hitung beta topi atau output_weight, dgn
ukuran [j x i]
output_weight = np.dot(h_plus,
matrik_kernel_data_train_kelm_norm)
#print(output_weight.round(2))

#print()

Hitung Matrik Y_topi sebagai hasil Decoder
pada Autoencoder
Di mana, Y_topi sebagai target yang diharapkan
hasilnya sama dengan
X atau arr_dataset input
Y_topi = np.dot(h, output_weight)
predict = Y_topi
print('Hitung Matrik Y_topi sebagai hasil
Decoder pada Autoencoder:')
print(Y_topi.round(2))

Tahap 3.3: Hitung nilai error rate, misal
dengan MAPE (sbg bilai Evaluasi)
aktual = matrik_kernel_data_train_kelm_norm

print()

Rumus MAPE yang digunakan, jika data aktualnya
ada yang nol atau tidak ada yg nol
dan untuk memastikan MAPE pada interval =
[0%;100%] --> dengan kondisi ini
Ref: (Berretti,Thampi,danSrivastava,2015)
dalam
Hapsari,KD.,Cholissodin,I.,Santoso,E.,2016
konstanta_smoothing = 0.00000001
c = konstanta_smoothing
mape_init = np.abs(((aktual+c) - (predict+c))
/ (aktual+c))*100
mape_norm = np.sum(np.where(mape_init>100, 100,
mape_init))/(len(predict)*banyak_fitur_kelm)
print('MAPE Norm [0% ; 100%] = ',
mape_norm.round(2), '%')
```

```
--
Tahap 4.1 Melakukan proses encode untuk data
testing
lanjutkan seperti Autoencoder base ELM murni
print('Matrik Kernel Data Test kelm_norm:')
print(matrik_kernel_data_test_kelm_norm.round(2))

print()
#Hitung Matrik H sebagai hasil Encoder data
training pada Autoencoder
print('Hitung Matrik H Data Test sebagai hasil
Encoder pada Autoencoder:')
h_test = 1/(1 + np.exp(-
(np.dot(matrik_kernel_data_test_kelm_norm,
np.transpose(bobot)) + bias)))
print(h_test.round(2))

Tahap 4.2: Proses Decode
lanjutkan seperti Autoencoder base ELM murni
Load matrik H Data Test, untuk Hitung Matrik
Hplus Test
h_plus_test =
np.dot(np.linalg.inv(np.dot(np.transpose(h_test)
, h_test)), np.transpose(h_test))
#print(h_plus.round(2))

#print()

Hitung beta topi atau output_weight, dgn
ukuran [j x i]
output_weight_test = np.dot(h_plus_test,
matrik_kernel_data_test_kelm_norm)
#print(output_weight.round(2))

print()

Hitung Matrik Y_topi sebagai hasil Decoder
pada Autoencoder
Di mana, Y_topi sebagai target yang diharapkan
hasilnya sama dengan
X atau arr_dataset input
Y_topi_test = np.dot(h_test, output_weight_test)
predict_test = Y_topi_test
print('Hitung Matrik Y_topi_test sebagai hasil
Decoder pada Autoencoder:')
```

```
print(Y_topi_test.round(2))

Tahap 4.3: Hitung nilai error rate, misal
dengan MAPE (sbg bilai Evaluasi)
aktual_test = matrik_kernel_data_test_kelm_norm

print()

Rumus MAPE yang digunakan, jika data aktualnya
ada yang nol atau tidak ada yg nol
dan untuk memastikan MAPE pada interval =
[0%;100%] --> dengan kondisi ini
Ref: (Berretti,Thampi,danSrivastava,2015)
dalam
Hapsari,KD.,Cholissodin,I.,Santoso,E.,2016
konstanta_smooting = 0.00000001
c = konstanta_smooting
mape_init_test = np.abs(((aktual_test+c) -
(predict_test+c)) / (aktual_test+c))*100
mape_norm_test =
np.sum(np.where(mape_init_test>100, 100,
mape_init_test))/(len(predict_test)*banyak_fitur_
_kelm)
print('MAPE Norm Data Test [0% ; 100%] = ',
mape_norm_test.round(2), '%')
--
----- batas akhir proses Encoder &
Decoder dari Autoencoder base Kernel ELM -----
Rev. 1
--
--
----- batas awal Menggunakan hasil
encoder KELM untuk training dan testing ELM
Regresi/lainnya -----
Rev. 1
--
1. Proses training dengan KELM (bukan ELM
murni)
1.1 Dgn mengambil hasil encode data training
plus load targetnya
dataset awal terdiri dari 3 fitur menjadi 2
print('Load Hasil Encoder data training pada
Autoencoder:')
print(h.round(2))
fitur_data_train_kelm_norm_regresi = h
print()
print('Target Data Traning:')
```

```
print(target_data_train_kelm_norm.round(2))

1.2 Hitung matrik kernel untuk data training
untuk regresi
matrik_kernel_data_train_kelm_norm_regresi =
np.zeros((NTrain,NTrain))
for i in range(NTrain):
 for j in range(i,NTrain):

 matrik_kernel_data_train_kelm_norm_regresi[i][j] =
\

 kernelPoly(fitur_data_train_kelm_norm_regresi[i],
],fitur_data_train_kelm_norm_regresi[j],1,2)

 matrik_kernel_data_train_kelm_norm_regresi[j][i] =
 matrik_kernel_data_train_kelm_norm_regresi[i][j]
print()
print('Matrik Kernel Data Train kelm norm untuk
Regresi:')
print(matrik_kernel_data_train_kelm_norm_regresi
.round(2))

1.3 Hitung beta topi atau output_weight KELM
Regresi
koefisien_regulasi = 0.01
I_identity_matrix_kelm = np.eye(NTrain,
dtype=float)
output_weight_kelm_regresi = \
np.dot(np.linalg.inv(
(I_identity_matrix_kelm/koefisien_regulasi) +
matrik_kernel_data_train_kelm_norm_regresi),\
target_data_train_kelm_norm)
print()
print('Hasil Output Weight untuk Regresi:')
print(output_weight_kelm_regresi.round(2))

2. Proses testing dengan KELM (bukan ELM
murni)
2.1 Dgn mengambil hasil encode data testing
plus load targetnya
kode **: data test bisa ambil dari data
training untuk validasi
atau
kode *: data test bisa ambil dari data testing
```

```
dataset awalnya terdiri dari 3 fitur menjadi 2
kode **
print('Load Hasil Encoder data test pada
Autoencoder:')
h_test_regresi = h
print(h_test_regresi.round(2))
fitur_data_test_kelm_norm_regresi =
h_test_regresi
NTest_regresi =
len(fitur_data_test_kelm_norm_regresi)
print()
print('Target Data Testing:')
target_data_test_kelm_norm_regresi =
target_data_train_kelm_norm
print(target_data_test_kelm_norm_regresi.round(2
))

print()

kode *
print('Load Hasil Encoder data test pada
Autoencoder:')
h_test_regresi = h_test
print(h_test_regresi.round(2))
fitur_data_test_kelm_norm_regresi =
h_test_regresi
NTest_regresi =
len(fitur_data_test_kelm_norm_regresi)
print()
print('Target Data Testing:')
target_data_test_kelm_norm_regresi =
target_data_test_kelm_norm
#
print(target_data_test_kelm_norm_regresi.round(2
))

2.2 Hitung matrik kernel untuk data testing
untuk regresi
matrik_kernel_data_test_kelm_norm_regresi =
np.zeros((NTest_regresi,NTrain))
for i in range(NTest_regresi):
 for j in range(NTrain):

 matrik_kernel_data_test_kelm_norm_regresi[i][j]
 = \
```

```
kernelPoly(fitur_data_test_kelm_norm_regresi[i,]
,fitur_data_train_kelm_norm_regresi[j,],1,2)
print()
print('Matrik Kernel Data Test kelm norm untuk
Regresi:')
print(matrik_kernel_data_test_kelm_norm_regresi.
round(2))

2.3 Hitung Y topi KELM Regresi
Di mana, Y_topi sebagai target yang diprediksi
Y_topi_test_regresi =
np.dot(matrik_kernel_data_test_kelm_norm_regresi
, output_weight_kelm_regresi)
predict_test_regresi = Y_topi_test_regresi

proses denormalisasi
predict_test_regresi = predict_test_regresi *
(maksimum - minimum) + minimum
print()
print('Hitung Matrik Y_topi_test sebagai hasil
Prediksi untuk Regresi:')
print(predict_test_regresi.round(2))

Tahap 4.3: Hitung nilai error rate, misal
dengan MAPE (sbg bilai Evaluasi)
aktual_test_regresi =
target_data_test_kelm_norm_regresi

proses denormalisasi
aktual_test_regresi = aktual_test_regresi *
(maksimum - minimum) + minimum
print()
print('Hitung Matrik Y test Aktual untuk
Regresi:')
print(aktual_test_regresi.round(2))

print()

Rumus MAPE yang digunakan, jika data aktualnya
ada yang nol atau tidak ada yg nol
dan untuk memastikan MAPE pada interval =
[0%;100%] --> dengan kondisi ini
Ref: (Berretti,Thampi,danSrivastava,2015)
dalam
Hapsari,KD.,Cholissodin,I.,Santoso,E.,2016
```

```
get banyak_fitur_target_kelm_regresi
if
bool(str(target_data_test_kelm_norm_regresi.shape.replace('(',')').replace(')', '')).split(',')[1].strip()):
 banyak_fitur_target_kelm_regresi =
int(str(target_data_test_kelm_norm_regresi.shape.replace('(',')').replace(')', '')).split(',')[1])
else:
 banyak_fitur_target_kelm_regresi = 1

konstanta_smooting = 0.00000001
c = konstanta_smooting
mape_init_test_regresi = np.abs(((aktual_test_regresi+c) -
(predict_test_regresi+c)) /
(aktual_test_regresi+c))*100)
mape_norm_test_regresi =
np.sum(np.where(mape_init_test_regresi>100, 100,
mape_init_test_regresi))/(len(predict_test_regresi)*banyak_fitur_target_kelm_regresi)
print('MAPE Norm Data Test utk Regresi [0% ; 100%] = ', mape_norm_test_regresi.round(2), '%')
--
----- batas akhir Menggunakan hasil encoder KELM untuk training dan testing ELM Regresi/lainnya -----
Rev. 1
--
#
#
----- batas awal proses Encoder & Decoder dari Autoencoder base ELM -----
Rev. 2
--
#
Tahap 1: Load data atau generate data (ini untuk ilustrasi memudahkan pemahaman)
Misal menggunakan "alternatif ke-2" dengan dataset (ada fitur + target)
misal untuk kasus Regresi
generate dataset, dlm range
[Rnd_Lower;Rnd_Upper]
N_Train_elm = 9
N_dimFeature_elm = 3
N_dimTarget_elm = 1
N_dim_elm = N_dimFeature_elm + N_dimTarget_elm
print('N_Train_elm = ', N_Train_elm)
print('N_dimFeature_elm = ', N_dimFeature_elm)
```

```
print('N_dimFeature_elm = ', N_dimTarget_elm)
print()
Rnd_Lower = 1
Rnd_Upper = 15

dataset disini, hanya berisi fitur-fitur data
arr_dataset_elm =
myrandfloat(N_Train_elm, N_dim_elm, Rnd_Lower, Rnd_
Upper)
print('Hasil Generate Dataset (dlm bentuk Array
2D):')
print(arr_dataset_elm.round(2))

Creating pandas dataframe from numpy array
agar nama kolom-kolom jadi terlihat sekaligus
print()
print('Hasil Generate Dataset (dlm bentuk
Dataframe):')
df_dataset_elm = \
pd.DataFrame({'Fitur ke-1': arr_dataset_elm[:, 0], \
 'Fitur ke-2': arr_dataset_elm[:, 1], \
 'Fitur ke-3': arr_dataset_elm[:, 2], \
 'Target': arr_dataset_elm[:, 3]})
display(df_dataset_elm.round(2))
--

Tahap 2: inisialisasi untuk normalisasi data
dan memilih mana data train dan test
misal diset langsung dari nilai Rnd_Lower dan
Rnd_Upper
minimum = Rnd_Lower
maksimum = Rnd_Upper

print('Nilai minimum dataset = ', minimum)
print('Nilai maksimum dataset = ', maksimum)
print()

arr_dataset_elm_norm = (arr_dataset_elm-
minimum) / (maksimum-minimum)
print('Hasil Normalisasi Data:')
print(arr_dataset_elm_norm.round(2))

memilih data training dan testing, misal diset
berikut
data_train_elm_norm = arr_dataset_elm_norm[:, 6, :]
fitur_data_train_elm_norm =
data_train_elm_norm[:, :3]
```

```
target_data_train_elm_norm =
data_train_elm_norm[:,3]
NTrain_elm = len(data_train_elm_norm)
print()
print('Data Train elm norm:')
print(data_train_elm_norm.round(2))
#print(fitur_data_train_kelm_norm)
#print(target_data_train_kelm_norm)

data_test_elm_norm = arr_dataset_elm_norm[6:9,:]
fitur_data_test_elm_norm =
data_test_elm_norm[:,3]
target_data_test_elm_norm =
data_test_elm_norm[:,3]
NTest_elm = len(data_test_elm_norm)
print()
print('Data Test kelm norm:')
print(data_test_elm_norm.round(2))
#print(fitur_data_test_kelm_norm)
#print(target_data_test_kelm_norm)
--_
Tahap 3.1: Proses Encode data training dengan
ELM murni
#set bebas, bisa direduksi (dikecilkan dari
dimensi awalnya) atau disparse (dinaikkan
dimensinya)
banyak_hidden_neuron_ae_elm = 2
banyak_fitur_elm = N_dimFeature_elm
bobot_elm =
myrandfloat(banyak_hidden_neuron_ae_elm,
banyak_fitur_elm,-0.5,0.5)
#bias =
np.random.rand(banyak_hidden_neuron_ae_elm)
flatten() digunakan utk convert ND to 1D array
bias_elm =
myrandfloat(1,banyak_hidden_neuron_ae_elm,0.0000
001,0.999999).flatten()
#print(bobot.round(2))
#print()
#print(bias.round(2))

#print()
#Hitung Matrik H sebagai hasil Encoder data
training pada Autoencoder
print('Hitung Matrik H sebagai hasil Encoder
pada Autoencoder:')
```

```
h_elm = 1/(1 + np.exp(-
 (np.dot(fitur_data_train_elm_norm,
 np.transpose(bobot_elm)) + bias_elm)))
print(h_elm.round(2))

Tahap 3.2: Proses Decode data training base
ELM murni
Load matrik H, untuk Hitung Matrik Hplus
h_plus_elm =
np.dot(np.linalg.inv(np.dot(np.transpose(h_elm),
h_elm)),np.transpose(h_elm))
print(h_plus.round(2))

print()

Hitung beta topi atau output_weight, dgn
ukuran [j x i]
output_weight_elm = np.dot(h_plus_elm,
fitur_data_train_elm_norm)
print(output_weight.round(2))

print()

Hitung Matrik Y_topi sebagai hasil Decoder
pada Autoencoder
Di mana, Y_topi sebagai target yang diharapkan
hasilnya sama dengan
X atau arr_dataset input
Y_topi_elm = np.dot(h_elm, output_weight_elm)
predict_elm_train = Y_topi_elm
proses denormalisasi
predict_elm_train = predict_elm_train *
(maksimum - minimum) + minimum
print()
print('Hitung Matrik Y_topi Data Training
sebagai hasil Decoder pada Autoencoder:')
print(predict_elm_train.round(2))

Tahap 3.3: Hitung nilai error rate, misal
dengan MAPE (sbg bilai Evaluasi)
print()
print('Matrik Data Training Aktual pada
Autoencoder:')
atau ambil langsung dari dataset dgn
aktual_elm_train = "arr_dataset_elm[:6,:3]"
aktual_elm_train = fitur_data_train_elm_norm
```

```
proses denormalisasi
aktual_elm_train = aktual_elm_train * (maksimum
- minimum) + minimum
print(aktual_elm_train.round(2))

print()

Rumus MAPE yang digunakan, jika data aktualnya
ada yang nol atau tidak ada yg nol
dan untuk memastikan MAPE pada interval =
[0%;100%] --> dengan kondisi ini
Ref: (Berretti,Thampi,danSrivastava,2015)
dalam
Hapsari,KD.,Cholissodin,I.,Santoso,E.,2016
konstanta_smooting = 0.00000001
c = konstanta_smooting
mape_init_elm_train = np.abs(((aktual_elm_train+c) - (predict_elm_train+c)) /
(aktual_elm_train+c))*100
mape_norm_elm_train =
np.sum(np.where(mape_init_elm_train>100, 100,
mape_init_elm_train))/(len(predict_elm_train)*banyak_fitur_elm)
print('MAPE Norm [0% ; 100%] = ',
mape_norm_elm_train.round(2), '%')
--

Tahap 4.1 Melakukan proses encode untuk data
testing base ELM murni
banyak_fitur_elm = N_dimFeature_elm
print('Fitur Data Test elm norm:')
print(fitur_data_test_elm_norm.round(2))

print()
#Hitung Matrik H sebagai hasil Encoder data
training pada Autoencoder
print('Hitung Matrik H Data Test sebagai hasil
Encoder pada Autoencoder:')
h_test_elm = 1/(1 + np.exp(-
(np.dot(fitur_data_test_elm_norm,
np.transpose(bobot_elm)) + bias_elm)))
print(h_test_elm.round(2))

Tahap 4.2: Proses Decode untuk data testing
base ELM murni
Load matrik H Data Test, untuk Hitung Matrik
Hplus Test
```

```
h_plus_test_elm =
np.dot(np.linalg.inv(np.dot(np.transpose(h_test_elm),h_test_elm)),np.transpose(h_test_elm))
#print(h_plus.round(2))

#print()

Hitung beta topi atau output_weight, dgn
ukuran [j x i]
output_weight_test_elm = np.dot(h_plus_test_elm,
fitur_data_test_elm_norm)
#print(output_weight.round(2))

print()

Hitung Matrik Y_topi sebagai hasil Decoder
pada Autoencoder
Di mana, Y_topi sebagai target yang diharapkan
hasilnya sama dengan
X atau arr_dataset input
Y_topi_test_elm = np.dot(h_test_elm,
output_weight_test_elm)
predict_test_elm = Y_topi_test_elm
proses denormalisasi
predict_test_elm = predict_test_elm * (maksimum
- minimum) + minimum
print('Hitung Matrik Y_topi_test sebagai hasil
Decoder pada Autoencoder:')
print(predict_test_elm.round(2))

Tahap 4.3: Hitung nilai error rate, misal
dengan MAPE (sbg bilai Evaluasi)
print()
print('Matrik Data Testing Aktual pada
Autoencoder:')
atau ambil langsung dari dataset dgn
aktual_test_elm = "arr_dataset_elm[6:9,:3]"
aktual_test_elm = fitur_data_test_elm_norm

proses denormalisasi
aktual_test_elm = aktual_test_elm * (maksimum -
minimum) + minimum
print(aktual_test_elm.round(2))

print()
```

```
Rumus MAPE yang digunakan, jika data aktualnya
ada yang nol atau tidak ada yg nol
dan untuk memastikan MAPE pada interval =
[0%;100%] --> dengan kondisi ini
Ref: (Berretti,Thampi,danSrivastava,2015)
dalam
Hapsari,KD.,Cholissodin,I.,Santoso,E.,2016
konstanta_smooting = 0.00000001
c = konstanta_smooting
mape_init_test_elm = np.abs(((aktual_test_elm+c) - (predict_test_elm+c)) /
(aktual_test_elm+c))*100
mape_norm_test_elm =
np.sum(np.where(mape_init_test_elm>100, 100,
mape_init_test_elm))/(len(predict_test_elm)*banyak_fitur_elm)
print('MAPE Norm Data Test [0% ; 100%] = ',
mape_norm_test.round(2), '%')
--
----- batas akhir proses Encoder &
Decoder dari Autoencoder base ELM -----
Rev. 2
--
--
```

Link kode program lengkapnya:

<https://colab.research.google.com/drive/1Xevd0wxlvvSy0vrF0vlGJVMCs12341Lt?usp=sharing>



## 21.3 Improve Deep Learning dengan Particle Swarm Optimization (PSO)

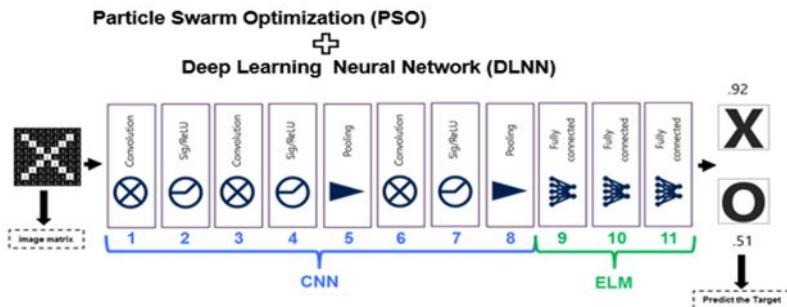
Untuk menjelaskan siklus Deep Learning dengan optimasi PSO maka diberikan contoh sederhana masalah prediksi (mencari nilai prediksi curah hujan) dalam rangka untuk menyiapkan pembuatan kalender tanam bagi petani di Indonesia. Curah Hujan merupakan faktor alam yang sangat penting bagi para petani atau lembaga tertentu untuk memprediksi masa tanam suatu tumbuhan. Permasalahan yang ada, curah hujan sangat sulit untuk diprediksi kejadiannya. Uji coba untuk mendapatkan prediksi curah hujan yang optimal telah banyak dilakukan oleh BMKG melalui penelitian dengan berbagai metode diberbagai bidang, diantaranya adalah bidang meteorologi, klimatologi dan geofisika. Hasil dari penelitian tersebut, memperoleh tingkat keberhasilan yang kurang optimal dalam memprediksi curah hujan. Sekarang ini, banyak metode-metode baru untuk memprediksi suatu kejadian. Metode tersebut diantaranya adalah Deep Learning (DL) dan Particle Swarm Optimization (PSO). Penggunaan metode Deep Learning sangat rentan terhadap pembobotan awal yang kurang optimal, sehingga diperlukan proses optimalisasi dengan menggunakan teknik metaheuristik, yaitu dengan algoritme PSO, karena algoritme ini mempunyai tingkat kompleksitas yang jauh lebih rendah dari algoritme genetika. Pada hasil Penelitian yang disitasi dalam buku ini menggunakan metode tersebut untuk memprediksi curah hujan yaitu dengan cara menentukan model persamaan regresi yang tepat sesuai dengan banyaknya lapisan pada node tersembunyi berdasarkan ukuran kernel dan bobot diantara lapisan tersebut.

### 21.3.1 Tentang Improve Deep Learning dengan PSO (Deep PSO)

Algoritma Deep Learning dengan PSO dapat digunakan untuk melakukan prediksi curah hujan di kabupaten Malang, di mana cara kerjanya menggunakan ekstraksi fitur dan transformasi data menjadi bentuk semacam matrik seperti pada citra. Jumlah dari convolution dan pooling layer bergantung pada kompleksitas dari kasusnya. Convolution layer terdiri dari beberapa kelompok fitur, dan pooling

layer terdiri dari seperti reduce atau melakukan *summary* dari beberapa kelompok atau groups of fitur. Berikut detail Langkah Deep Learning dengan optimasi algoritma PSO.

1. Set Map untuk Network Arsitektur Pre-trained bisa menggunakan ResNet, VGG, Inception atau lainnya. Pada kode yang kami lampirkan, Map tersebut kami buat sederhana dengan tetap memenuhi kaidah CNN pada Deep Learning, dan dikoding *from scratch*.



Gambar 21.8 Map Simplified Deep Learning CNN based ELM with PSO (Deep PSO)

2. Set Parameter value.
3. Deep PSO Process

Dimana representasi 4 cluster dimensi pada tiap partikel PSO pada Hybrid PSO-DLNN (Deep PSO) dapat dilihat pada Table berikut.

| $x_i(t)$ | $k$ | $FC1\_W_{jk}$ | $FC2\_W_{jk}$ | $FC3\_W_{jk}$ |
|----------|-----|---------------|---------------|---------------|
|----------|-----|---------------|---------------|---------------|

### 3.1 Training Process of SDLCNN-ELM

### 3.2 Testing Process of SDLCNN-ELM

dimana,

$k$  terdiri dari 1 dimensi = [Kmin=1;Kmaks=5], ketika perhitungan DL,  $k$  akan dikonversi nilainya menjadi  $2*k + 1$ .

$FC1\_W_{jk}$  terdiri dari  $1 \times (5 \times 7) = [-0.5;0.5]$

$FC2\_W_{jk}$  terdiri dari  $1 \times (7 \times 7) = [-0.5;0.5]$

$FC3\_W_{jk}$  terdiri dari  $1 \times (4 \times 7) = [-0.5;0.5]$

Sehingga panjang dimensi tiap partikelnya adalah 113

### 21.3.1 Evaluasi Deep PSO

Nilai fitness diambil dari nilai  $(1/(1+ \text{MAD}))$ , di mana nilai Mean Absolute Deviation (MAD) merupakan hasil nilai evaluasi dari proses testing Deep Learning dengan optimasi algoritma PSO.

### 21.3.2 Studi Kasus: Prediksi Curah Hujan menggunakan Improve Deep Learning dengan Particle Swarm Optimization (PSO)

Untuk memudahkan dalam memahami manualisasi detail dari Improve Deep Learning dengan PSO, Anda harus membaca dan pahami dulu manualisasi PSO pada buku Swarm Intelligence dari link "<http://bit.ly/2Ww0wYa>" dan CNN pada buku ini.

Source Code 21.2 Improve Deep Learning dengan PSO

```
#Snippet code of improve deep learning with PSO
(deep PSO)

function
[MeanMADeachIteration]=FnMyIPSO_DLCNNneLM_TestConv(typeFeature,IterMaxPSO
)

for t=0:IterMaxPSO
 % calculate value of w, c1, c2, r1, r2
 w=wmin+((wmax-wmin)*((tmax-t)/tmax));
 c1=((c1f-c1i)*(t/tmax))+c1i;
 c2=((c2f-c2i)*(t/tmax))+c2i;
 r1=rand(1,1); % random [0,1] element of uniform distribution
 r2=rand(1,1);

 if(t==0)
 % initialization position of particle
 X=repmat_SLCcLR_lower +
(random('unif',0,1,pop_size,num_dim).*repmat_SLCcLR_delta);
 % initialization velocity of particle = 0
 V;

 % initialization Pbest and Gbest
 Pbest=X;
 [FitnessAllPbest,FitnessGbest,Gbest]=...
 FnGetFitnessNbestIndividuIPSODL(Pbest);
 else
 % update velocity
 %V=w*V+(c1*r1*(Pbest-X))+(c2*r2*(Gbest-X))
 V=(w.*V)+(c1*r1.* (Pbest-X))+(c2*r2.* (ones(pop_size,1)*Gbest)-X));

 V=FnBringtoRangeLowUpIPSODL(V,repmat_V_SLCcLR_lower,repmat_V_SLCcLR_uppe
r);
```

```
% update position
X=X+V;

X=FnBringtoRangeLowUpIPSOGL(X,repmat_SLCCLR_lower,repmat_SLCCLR_upper);

% calculate fitness of each Particles (X)
[FitnessAllX,IndexSortingDesc]=FnGetFitnessIPSOGL(X);

% update Pbest and Gbest

[FitnessAll_Update_Pbest,Fitness_Update_Gbest,Update_Pbest,Update_Gbest]
=...

FnUpdatePbestGbestIPSOGL(FitnessAllPbest,FitnessAllX,FitnessGbest,X,Pbest,Gbest);
FitnessAllPbest=FitnessAll_Update_Pbest;
FitnessGbest=Fitness_Update_Gbest;
Pbest=Update_Pbest;
Gbest=Update_Gbest;

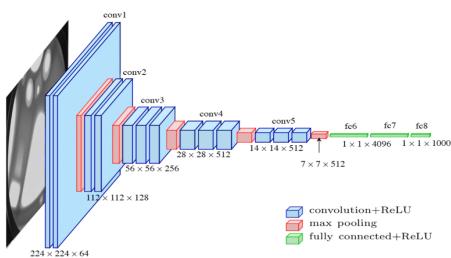
% save mean of fitness each iteration
MeanFitness(t)=mean(FitnessAllPbest);
end
end
```

Link kode program lengkapnya:

<https://github.com/imamcs19/Improve-Deep-Learning-with-PSO>



## 21.4 Tugas Kelompok

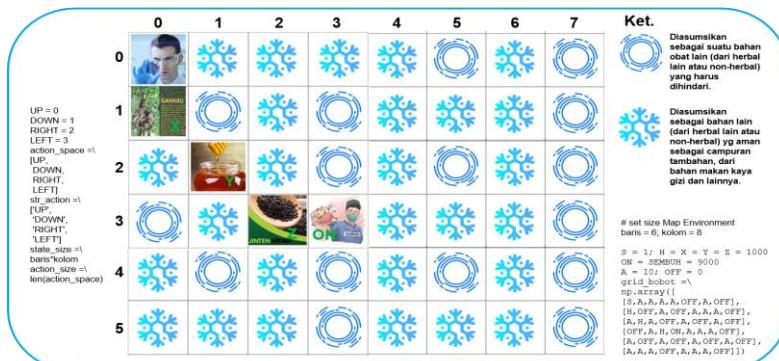
1. Buatlah kode program berdasarkan dari “Improve Autoencoder untuk Fast Mapping Fitur dengan Kernel ELM (KELM) dan ELM Murni” yang menggunakan Kernel berikut:
  - a. Radial Basis Function (RBF)
  - b. Additive Kernel
  - c. Sigmoid Kernel
2. Dari soal no. 1 di atas, ganti metode training dan testing dari hasil Autoencoder KELM dan ELM dengan menggunakan Kernel KNN.
3. Buatlah kode program berdasarkan dari “Improve Deep Learning dengan Partcile Swarm Optimization (PSO)” yang menggunakan Teknik Optimasi berikut:
  - a. Algoritma Genetika (GA)/ Semut (ACO)/ Bee Colony (ABC)
  - b. Local Search dengan SA.
4. Dari soal no. 3 di atas, ubahlah Map untuk Network Arsitektur Pre-trained dengan menggunakan berikut:
  - a. VGG16
  - b. VGG19
  - c. Inception
5. Buatlah kode program yang menggabungkan antara Deep PSO dengan Autoencoder untuk kasus regresi atau klasifikasi dengan dataset misal diambil dari UCI Repository, lalu bandingkan waktu komputasi serta hasil nilai evaluasinya antara Deep PSO dengan dan tanpa Autoencoder!
6. Buatlah Representasi Partikel PSO untuk membuat Optimasi Set Map Arsitektur secara meta-heuristik!

## BAB 22 Project Pilihan

### 22.1 Simulasi Reinforcement Learning untuk Pencarian Kandidat Obat Covid-19

#### 22.1.1 Konsep

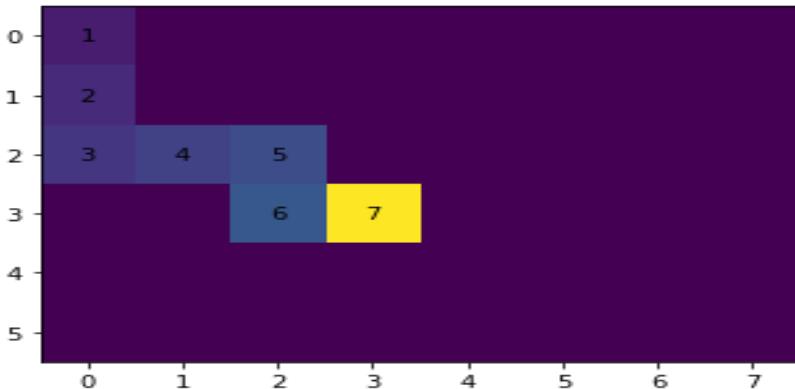
Beberapa ilmuan berbagai bidang berkumpul untuk membuat suatu mesin *docking* untuk uji efek positif obat terhadap corona virus (Covid-19), yaitu dengan mengkombinasikan 5 jenis campuran herbal, yaitu herbal ke-1 ( $X$  adalah dari dahan kayu India), herbal ke-2 ( $Y$  adalah dari Madu Asli), herbal ke-3 ( $Z$  adalah dari Jinten Hitam), serta 2 Herbal/Non-Herbal lainnya (yaitu dinotasikan dengan  $A_1$  &  $A_2$ ), sebagai kandidat obatnya atau suplemennya dalam uji dengan bantuan simulasi computer terhadap kesehatan pasien. Dalam uji simulasi (*in silico*) tersebut merupakan suatu bentuk abstraksi yang segala kondisinya didalam komputer adalah seperti kondisi real di kasus aslinya. Pada ilmuan tersebut kesulitan untuk mempelajari pola interaksi antara campuran obat  $\{X, Y, Z, A_1, A_2\}$  yang diberikan dengan terhadap kondisi pasien Covid-19. Akhirnya salah seorang dari ilmuan tersebut meminta bantuan Anda sebagai ahli dalam bidang AI pada computer untuk mengatasi permasalahan tersebut, yaitu menggunakan Reinforcement Learning dengan Q-Learning sebagai pendekatan untuk membuat dan mendapatkan modelling terbaik dari interaksi campuran herbal terhadap kondisi pasien Covid-19.



Gambar 22.1 Reinforcement Learning untuk Kandidat Obat Covid-19

## 22.1.2 Tampilan Implementasi

Hasil Tracking solusi Pasien Berhasil Sembuh 🤞 dgn animasi matplotlib (6 x 8):



Alur algoritma Q-Learning:

1. inisialisasi Q-values yaitu  $Q(s,a)$ , dengan nilai sembarang, dimana pasangan state-action,  $s \in S$ ,  $a \in A(s)$ , dan  $Q(\text{terminal-state}, \cdot) = 0$
2. repeat (untuk tiap episode):
  - 2.1. inisialisasi  $s$  (dari environment)
  - 2.2. repeat (untuk tiap langkah dari episode):
    - 2.2.1. Pilih action ( $a$ ) pada current state  $s$  (misal, dgn  $\epsilon$ -greedy berdasarkan current Q-value untuk mengestimasi  $Q(s, \cdot)$ )
    - 2.2.2. Lakukan action ( $a$ ) lalu dapatkan reward ( $r$ ) dan keluaran berupa next state ( $s'$ )
    - 2.2.3. Update action-value function pada  $(s, a)$ , dengan menghitung  $Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ , dan meng-update state  $s = s'$
  - 2.3. until (sampai max\_steps)
3. repeat (sampai total\_episodes)

## 22.1.3 Source Code

Kode program terdiri dari beberapa bagian, yaitu Set Environment, Membuat method update state berdasarkan action, dan proses utama Reinforcement Learning dengan Q-Learning, yang semuanya di koding *from scratch* tanpa menggunakan “!pip install gym”.

```
#Set Environment

Reinforcement Learning dengan Q-Learning
import numpy as np
import matplotlib.pyplot as plt
from google.colab import output

set size Map Environment
baris = 6
kolom = 8

Buat Environment
grid_reset = np.zeros(baris*kolom).reshape(baris, kolom)
grid_jelajah = np.zeros(baris*kolom).reshape(baris, kolom)
dimana nilainya [0;255] atau bermakna 0<= nilai <=255
grid_reset[0, 0] = 255
fig, ax = plt.subplots()
reset_env = ax.imshow(grid_reset, interpolation='nearest')

Set keadaan Environment utk bobot reward
grid_bobot = np.zeros(baris*kolom).reshape(baris, kolom)
misal,
init_tempat_start = S = 1
cell_herb_gaharu = cell_herb_madu = cell_herb_jinten = H = 1000
cell_sembuh = ON = 9000
cell_tree_as_bahan_lain_yg_aman = A = 10
cell_lubang_as_bahan_yg_dihindari = OFF = 0
#
S = 1
H = X = Y = Z = 1000
ON = SEMBUH = 9000
A = A1 = A2 = 10
OFF = 0
grid_bobot = np.array([[S,A,A,A,A,OFF,A,OFF],
 [H,OFF,A,OFF,A,A,OFF],
 [A,H,A,OFF,A,OFF,A,OFF],
 [OFF,A,H,ON,A,A,OFF],
 [A,OFF,A,OFF,A,OFF,A,OFF],
 [A,A,A,OFF,A,A,OFF]])

Inisialisasi nilai parameter
state_size = baris*kolom
action_size = 4, knp 4, karena dari banyaknya arah kemung-kinan gerakan yg dapat dilakukan
yaitu ke atas, ke bawah, ke kanan, dan ke kiri, misal dikodekan dgn [0,1,2,3]
misal ke atas = 0, ke bawah = 1, ke kanan = 2, ke kiri = 3.
#
UP = 0
DOWN = 1
RIGHT = 2
LEFT = 3
action_space = [UP,DOWN,RIGHT,LEFT]
str_action =['UP','DOWN','RIGHT','LEFT']

state_size = baris*kolom
action_size = len(action_space)

Buat Q table dengan ukuran baris = state_size rows dan kolom = action_size
qtable = np.zeros((state_size, action_size))
print(qtable)

Inisialisasi nilai parameter utama lainnya (hyperparameters)
total_episodes = 20000 # Total episodes
```

```
total_episodes = 100 # Total episodes
learning_rate = 0.7 # Learning rate
max_steps = 99 # Max steps per episode
max_steps = 19 # Max steps per episode
gamma = 0.95 # Discounting rate

Exploration parameters
epsilon = 1.0 # Exploration rate
max_epsilon = 1.0 # Exploration probability at start
min_epsilon = 0.01 # Minimum exploration probability
decay_rate = 0.005 # Exponential decay rate for exploration prob
```

```
#Membuat method update state berdasar action
```

```
def updateState(i,j,action):
 if(action == UP):
 i_baru = i - 1
 j_baru = j
 if(i_baru < 0):
 i_baru = i
 elif(action == DOWN):
 i_baru = i + 1
 j_baru = j
 if(i_baru > baris - 1):
 i_baru = i
 elif(action == RIGHT):
 i_baru = i
 j_baru = j + 1
 if(j_baru > kolom - 1):
 j_baru = j
 elif(action == LEFT):
 i_baru = i
 j_baru = j - 1
 if(j_baru < 0):
 j_baru = j
 return i_baru,j_baru

membuat method index2XY
def index2XY(index):
 Vs = index
 i_ = int(np.ceil((Vs+1)/kolom)) - 1
 j_ = (Vs+1)%kolom
 if(j_ ==0):
 j_ += kolom
 j_ += -1
 return i_,j_

membuat method XY2index
def XY2index(i,j):
 Vs=(i+1)*kolom + (j+1) - kolom - 1
 return Vs

membuat method env_step(action)
def env_step(action,state):
 # convert state as Index to XY
 (i,j) = index2XY(state)

 # update state dgn action
 (i_baru,j_baru) = updateState(i,j,action)

 # convert XY to Index
 new_state = XY2index(i_baru,j_baru)

 # get bobot grid as reward
 reward = grid_bobot[i_baru,j_baru]

 # cek apakah sudah sampai di cell ("Sembuh" yaitu reward==ON) atau
 # ("Ada bahan obat yg kontradiksi" yaitu reward==OFF)
 if(reward==ON or reward==OFF):
 done = True
 else:
 done = False

 # set info
```

```
 if (reward==OFF):
 info = 'Gagal'
 else:
 info = 'Aman, Lanjukan'

 return new_state, reward, done, info
```

### #Q-Learning bag. 1 dari 4

```
import time
List of rewards
rewards = []

2. repeat (untuk tiap episode)
for episode in range(total_episodes):

 # 2.1. Reset the environment
 # state = env.reset()
 state = 0
 #start jelajah, misal di titik (0,0) atau setara dengan state = 0
 memori_ij =[(0,0)]

 #step = 0
 done = False
 total_rewards = 0

 for step in range(max_steps):
 print()
 print(episode,'-',step)
 # 2.2.1. Choose an action a in the current world state (s)
 ## First we randomize a number base ε-greedy
 exp_exp_tradeoff = np.random.uniform(0, 1)

 ## If this number > greater than epsilon --> exploitation
 ## (taking the biggest Q value for this state)
 if exp_exp_tradeoff > epsilon:
 action = np.argmax(qtable[state,:])
 #print(exp_exp_tradeoff, "action", action)

 # Else doing a random choice --> exploration
 else:
 # action = env.action_space.sample()
 action = np.random.randint(action_size)
 #print("action random", action)

 # 2.2.2. Take the action (a) and observe the outcome state(s') and
 # reward (r)
 #new_state, reward, done, info = env.step(action)
 #new_state, reward, done, info = env_step(action,state)

 get_new_state = True
 while get_new_state:
 new_state, reward, done, info = env_step(action,state)
 print('state =',state,' new state =',new_state)
 print('memori_ij = ',memori_ij)
 #convert new_state ke XY
 (i_baru,j_baru) = index2XY(new_state)

 if ((i_baru,j_baru) not in memori_ij):
 memori_ij.append((i_baru,j_baru))
 print('update memori_ij = ',memori_ij)
 get_new_state = False
 break
 else:
 action = np.random.randint(action_size)
 print("action random", action)

 # 2.2.3. Update
 # Q(s,a):= Q(s,a) + lr [R(s,a) + gamma * max Q(s',a') - Q(s,a)]
 ## qtable[new_state,:]: all the actions we can take from new state
 qtable[state, action] = qtable[state, action] + learning_rate *
 (reward + gamma * np.max(qtable[new_state,:]) - qtable[state, action])
```

```
total_rewards += reward

Our new state is state
state = new_state

If done == True jika ("Sembuh" yaitu reward==ON) atau ("Ada bahan
obat yg kontradiksi" yaitu reward==OFF)), lalu finish episode
if done == True:
 if(grid_bobot[i_baru,j_baru]==SEMBUH):
 #if new_state == 27:
 print("Pasien Berhasil Sembuh 🌟")
 else:
 print("Pasien Masih Belum Sembuh, ada bahan obat yg
 harus dihindari 🚫")
 break

Reduce epsilon (because we need less and less exploration)
epsilon = min_epsilon + (max_epsilon - min_epsilon)*np.exp(-decay_rate*
episode)
rewards.append(total_rewards)

print()
print ("Score over time: " + str(sum(rewards)/total_episodes))
print(qtable)
```

### #Q-Learning bag. 2 dari 4

```
Menggunakan hasil Q-table utk tracking solusi yg didapatkan ! 🌟
print("Hasil Tracking solusi yg didapatkan:")
iter = 0
go_track = True
while go_track:
 i,j = index2XY(iter)
 i_baru,j_baru = updateState(i,j,np.argmax(qtable[iter,:]))
 print(i,' ',j,' = ',str ac-
tion[np.argmax(qtable[iter,:])], '> ',i_baru,',',j_baru)
 iter = XY2Index(i_baru,j_baru)
 if(grid_bobot[i_baru,j_baru]==SEMBUH):
 print("Pasien Berhasil Sembuh 🌟")
 go_track = False
```

### Output:

```
Hasil Tracking solusi yg didapatkan:
0 , 0 = DOWN > 1 , 0
1 , 0 = DOWN > 2 , 0
2 , 0 = RIGHT > 2 , 1
2 , 1 = RIGHT > 2 , 2
2 , 2 = DOWN > 3 , 2
3 , 2 = RIGHT > 3 , 3
Pasien Berhasil Sembuh 🌟
```

### #Q-Learning bag. 3 dari 4

```
from google.colab import output

Menggunakan hasil Q-table utk tracking solusi yg didapatkan ! ↴
grid = np.zeros(baris*kolom).reshape(baris, kolom)
x_awal = 0
y_awal = 0
grid[x_awal, y_awal] = 255

print("\nHasil Tracking solusi Pasien Berhasil Sembuh ↴ dgn
animasi matplotlib (6 x 8):")
iter = 0
memori_ij = [(0,0)]
go_track = True
while go_track:
 i,j = index2XY(iter)
 i_baru,j_baru = updateState(i,j,np.argmax(qtable[iter,:]))
 #print(i,',',j,',', = ',str
 #action[np.argmax(qtable[iter,:])], ' > ',i_baru,',',j_baru)
 iter = XY2index(i_baru,j_baru)

 # Animasi dengan matplotlib
 newGrid = grid.copy()
 print("\nHasil Tracking solusi Pasien Berhasil Sembuh ↴ dgn
animasi matplotlib (6 x 8):")
 fig, ax = plt.subplots()
 img = ax.imshow(grid, interpolation='nearest')
 newGrid[i,j] = 10 + 10*len(memori_ij)
 memori_ij.append((i_baru,j_baru))
 #img.set_data(newGrid)
 newGrid[i_baru, j_baru] = 255

 grid = newGrid.copy()

 plt.pause(2)
 output.clear()

 if(grid_bobot[i_baru,j_baru]==SEMBUH):
 #print("Pasien Berhasil Sembuh ↴")
 go_track = False

 # Hasil urutan jelajah pada grid
print("\nHasil Tracking solusi Pasien Berhasil Sembuh ↴ dgn
animasi matplotlib (6 x 8):")
fig, ax = plt.subplots()
img = ax.imshow(grid, interpolation='nearest')
for ii in range(len(memori_ij)):
 i = memori_ij[ii][0]
 j = memori_ij[ii][1]
 plt.text(j, i, str(ii+1),
 horizontalalignment='center',
 verticalalignment='center',)
```

### #Q-Learning bag. 4 dari 4

```
Menggunakan hasil Q-table utk tracking solusi yg didapatkan maupun
yg blm optimal hasilnya ! ↴
Misal di-set
bykPercobaan di-set dari banyaknya Herbal A/A1/A2 + 1,
di mana + 1 diambil dari
misal state 0 atau di posisi (0,0)
idx_A_A1_A2_1 = np.argwhere(grid_bobot==A)
bykPercobaan = len(np.argwhere(grid_bobot==A))+1

#jumlahJelajah = 19

for episode in range(bykPercobaan):
```

```
mencoba dgn tempat awal start secara sequential dari state yang memungkin
if episode == 0:
 state = 0
else:
 state = XY2index(idx_A_A1_A2_1[episode-1][0],idx_A_A1_A2_1[episode-1][1])

done = False
print("*****")
print("EPISODE ", episode)

for step in range(max_steps):
#for step in range(jumlahJelajah):
 print()
 print(episode,'-',step)

 (i,j) = index2XY(state)

 # Take the action (index) that have
 # the maximum expected future reward given that state
 action = np.argmax(qtable[state,:])
 print('step = ',step,'| action = ',action)

 new_state, reward, done, info = env_step(action,state)
 (i_baru,j_baru) = index2XY(new_state)
 #print(new_state,'-',reward,'-',done,'-',info)
 print(i,'.',j,' = ',
 str_action[np.argmax(qtable[state,:])],' > ',i_baru,',',j_baru)

 # If done == True jika ("Sembuh" yaitu reward==ON) atau ("Ada bahan
 # obat yg kontradiksi" yaitu reward==OFF), lalu finish episode
 if done:
 if(grid_bobot[i_baru,j_baru]==SEMBUH):
 #if new_state == 27:
 print("Pasien Berhasil Sembuh 🌟")
 else:
 print("Pasien Masih Belum Sembuh, ada bahan obat
 yg harus dihindari 🚫")
 break

 state = new_state
```

Link kode lengkapnya:

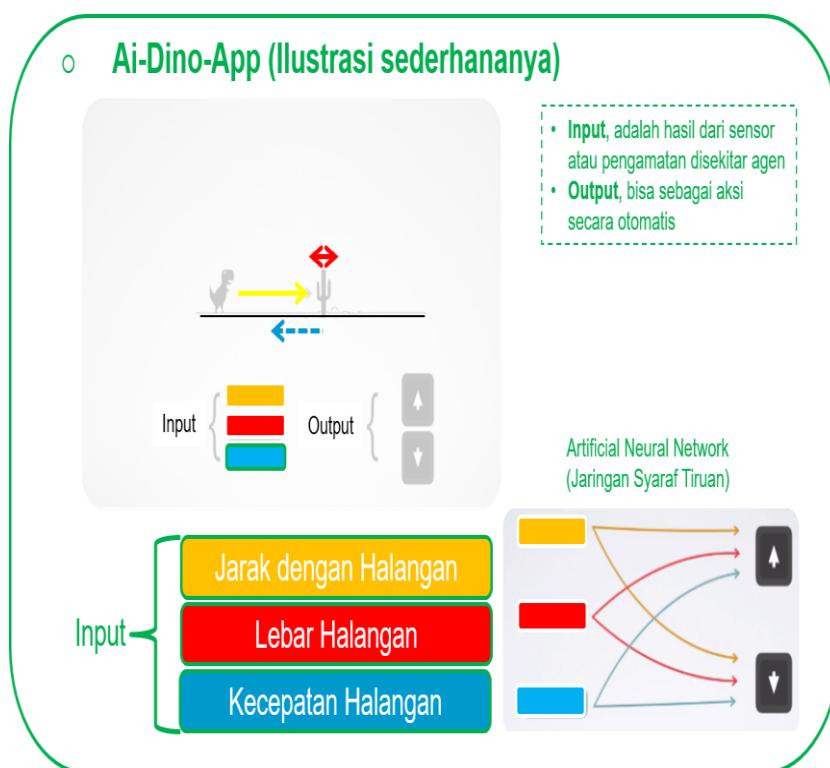
<https://colab.research.google.com/drive/1SYUSU3-wyCptSZWyyULWpaMMZ8PEC0ko?usp=sharing>



## 22.2 Ai Dino, Flappy Bird & Frozen Lake dengan Reinforcement Learning untuk Otomasi Pergerakan

### 22.2.1 Konsep

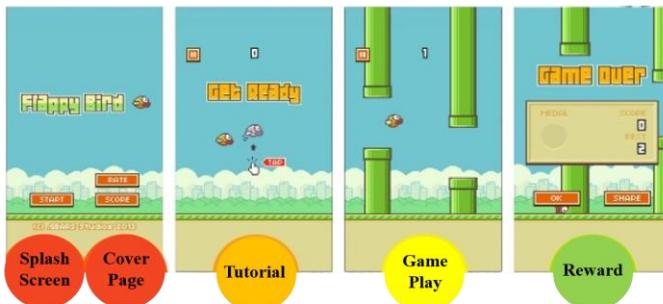
Pada Ai Dino sebelumnya telah dilakukan pembelajaran mandiri berdasarkan beberapa parameter input dan output yang digunakan, di mana parameter-parameter tersebut disesuaikan dengan rules yang ada pada keadaan lingkungan agen Dino sehingga dapat juga parameternya ditambahkan parameter lain atau dikurangi. Inputnya misalkan, Jarak dengan halangan, Lebar halangan, Kecepatan halangan (bisa juga missal ditambahkan ketinggian halangan, dll), sedangkan Outputnya misalkan hanya Up dan Down. Ilustrasinya ditunjukkan seperti pada Gambar berikut.



Gambar 22.2 Reinforcement Learning pada Ai Dino

Kemudian yang kedua Ai Flappy Bird dan yang ketiga Frozen Lake, juga selaras dengan penjelasan dari Ai Dino sebelumnya, dan berikut ilustrasinya.

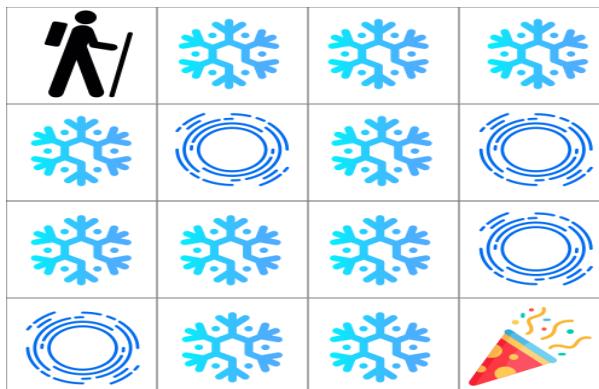
- **Ai-FlappyBird-App (Ilustrasi sederhananya)**



Ai-FlappyBird-App tersebut dengan implementasi algoritma Reinforcement Learning yang dapat berbasis dengan atau tanpa menggunakan algoritma Artificial Neural Network (ANN) atau Jaringan Syaraf Tiruan, yang dapat belajar secara otomasi dari pengalaman.

Gambar 22.3 Reinforcement Learning pada Ai Flappy Bird

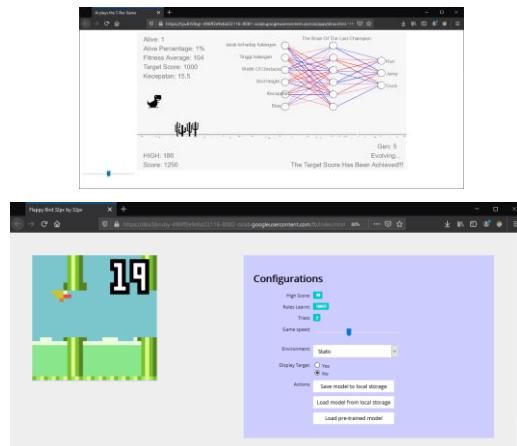
Khusus pada Frozen Lake ini, digunakan gabungan koding sebagian from Scratch algoritma Q-Learning-nya dan untuk sebagian lainnya menggunakan library dari “!pip install gym”, untuk contoh yang *full from scratch* dapat dilihat pada contoh “Simulasi Reinforcement Learning untuk Pencarian Kandidat Obat Covid-19”. Berikut bentuk ilustrasi Map Frozen Lake sederhana, yang juga dapat dicustom untuk set ukuran dan keadaan lingkungannya.



Gambar 22.4 Reinforcement Learning pada Ai Frozen Lake

## 22.2.2 Tampilan Implementasi

Hasil running Ai Dino, Flappy Bird, dan Frozen Lake (hanya cell base) dengan menggunakan Google Colab. Di mana, Ai Dino, Flappy Bird menggunakan Java Script (JS) yang dijalankan pada localhost Google Colab dengan mengenerate link Web App untuk run Script JS tersebut, sedangkan Frozen Lake dengan Python.



Reinforcement Learning-nya berbasis algoritma Q-Learning yang alurnya sudah dijelaskan pada “Simulasi Reinforcement Learning untuk Pencarian Kandidat Obat Covid-19”.

## 22.2.3 Source Code

Kode program terdiri dari beberapa bagian, yaitu Ai Dino, Flappy Bird dan Frozen Lake.

```
#Ai Dino App bag. 1

Ref:
[1] https://arxiv.org/abs/2008.06799
[2] https://github.com/Ethan0104/AI-learns-to-play-the-dino-game
[3] https://medium.com/acing-ai/how-i-build-an-ai-to-play-dino-run-e37f37bdf153

if not os.path.exists('ai'):
 !git clone https://github.com/Ethan0104/AI-learns-to-play-the-dino-game.git

 # lalu rename "AI-learns-to-play-the-dino-
 # gam" > "ai" & "ai/Dino\ game" > "ai/app"
 !mv AI-learns-to-play-the-dino-game/ ai/
 !mv ai/Dino\ game ai/app
else:
 print('Ai-Dino-App sudah di setup sebelumnya')

melihat file pada case study ke-1, dlm KB
!ls./ai/app -l -a --block-size=K
```

```
#Ai Dino App bag. 2
```

```
run ai-app-dino di localhost-nya Google Colab
from google.colab.output import eval_js
print("Klik Link berikut:")
#print(eval_js("google.colab.kernel.proxyPort(8085)")+"drive/My%20Drive/Ai_Project_Sep_2020/ai/app/dino.html")
print(eval_js("google.colab.kernel.proxyPort(8081)")+"ai/app/dino.html")
#!python -m http.server 8081
#!python -m CGIHTTPServer 8360
#!python -m http.server --cgi 8360
#!python3 -m http.server --cgi 8360
get_ipython().system_raw('python3 -m http.server 8081 &')
```

Misal didapatkan Output hasil Link generate-nya berikut:

<https://u84t5...-8081-colab.googleusercontent.com/ai/app/dino.html>

```
#Ai Flappy Bird App bag. 1
```

```
Ref:
[1] http://cs229.stanford.edu/proj2015/362_report.pdf
atau http://cs231n.stanford.edu/reports/2016/pdfs/111_Report.pdf
[2] https://github.com/nileshsah/reinforcement-learning-flappybird
[3] https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-0-q-learning-with-tables-and-neural-networks-d195264329d0
#
if not os.path.exists('fb'):
 !git clone https://github.com/nileshsah/reinforcement-learning-flappybird.git
 # lalu rename "reinforcement-learning-flappybird" > "fb"
 !mv reinforcement-learning-flappybird/ fb/
else:
 print('Ai-FlappyBird-App sudah di setup sebelumnya')
```

```
#Ai Flappy Bird App bag. 2
```

```
run ai-app-flappybird
from google.colab.output import eval_js
print("Klik Link berikut:")
#print(eval_js("google.colab.kernel.proxyPort(8082)")+"fb/index.html")
#!python -m http.server 8081
#!python -m CGIHTTPServer 8360
#!python -m http.server --cgi 8360
#!python3 -m http.server --cgi 8360
get_ipython().system_raw('python3 -m http.server 8082 &')
```

### #Ai Frozen Lake App

```
Ref:
[1] https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf
[2] https://github.com/simoninithomas/Deep_reinforcement_learning_Course/tree/master/Q%20learning/FrozenLake
[3] https://www.freecodecamp.org/news/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe/
[4] https://www.researchgate.net/publication/338171196_Pengembangan_Reinforcement_Learning_RL_Single-Agent_Menggunakan_Improve_Q-Learning

install dependencies
!pip install numpy
!pip install gym
```

Link kode lengkapnya:

<https://colab.research.google.com/drive/11THAT0ZUciWf2tsivgWDS3Gxzd4K9GmV?usp=sharing>



## 22.3 Generative Adversarial Network (GAN) by Example

### 22.3.1 Konsep

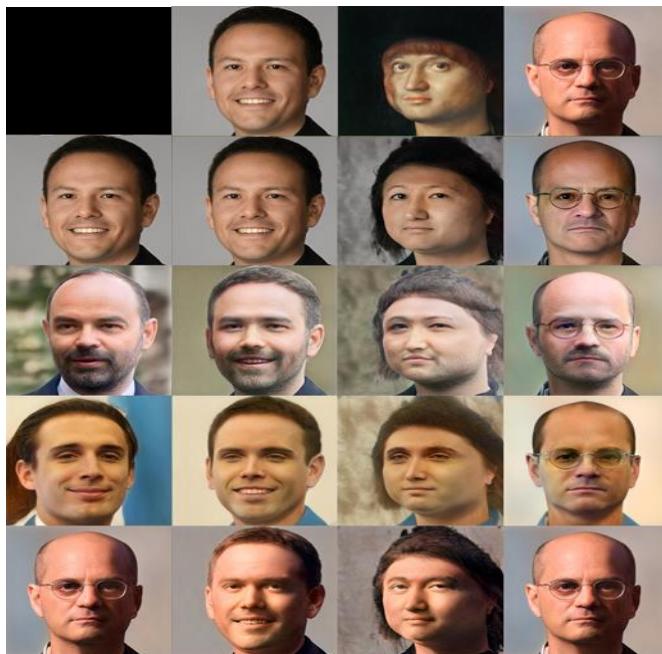
Pada Generative Adversarial Network (GAN) merupakan suatu algoritma yang digunakan untuk membuat gambar sintesis atau dummy dari gabungan dua atau beberapa gambar secara otomatis, serta juga dapat digunakan lebih luas, misal pada multimedia yang mengabungkan Audio, Video, Subtitles dan lainnya. Bentuk sederhana ilustrasi untuk GAN.



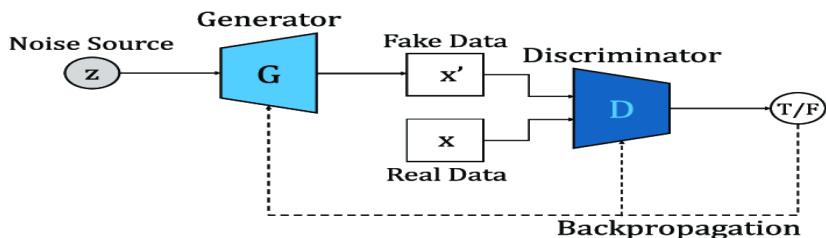
Gambar 22.5 Ilustrasi Generative Adversarial Network (GAN)

### 22.3.2 Tampilan Implementasi

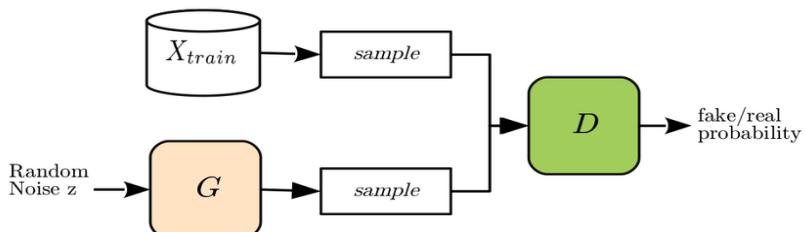
Di dalam GAN, dapat dilakukan misalkan beberapa pilihan sebagai dasar dalam pembuatan sintesis hasil gambarnya. Misalnya dapat berdasarkan “List expressions” seperti age, angle\_horizontal, angle\_pitch, beauty, emotion\_angry, emotion\_disgust, emotion\_easy, emotion\_fear, emotion\_happy, emotion\_sad, emotion\_surprise, eyes\_open, face\_shape, glasses, height, race\_black, race\_white, race\_yellow, smile, width, motion behavior, dan lainnya. Berikut adalah contoh hasil running keluaran tampilan map hasil GAN dengan mengkombinasikan dua foto wajah dalam bentuk matriks baris kolom, dengan ukuran empat baris dan 3 kolom.



Alur algoritma GAN, di mana  $G$  dan  $D$  adalah model yang akan di-update jika hasil masih belum optimal ( $F$  atau Fake), sampai memenuhi kondisi  $T$  atau Real, ini semacam klasifikasi biner (*binary classification*).



Atau



Gambar 22.6 Arsitektur algoritma GAN

### 22.3.3 Source Code

Kode program terdiri dari beberapa bagian, yaitu mulai dari clone kode dari github yang sudah kami modifikasi:D, sampai pada record proses GAN menjadi bentuk video. Berikut akan diberikan sampel Sebagian kode untuk clone-nya.

```
#Ai GAN App

Ref:
[1] https://arxiv.org/pdf/1812.04948.pdf
atau https://arxiv.org/pdf/1904.03189.pdf
atau https://arxiv.org/pdf/1912.04958.pdf
[2] https://github.com/woctezuma/stylegan2-projecting-images
#
sebelum jalankan script ini,
klik "Runtime" > "Change runtime type" > pd Hardware accelerator pilih "GPU"
lalu klik "Save"
from google.colab import drive
drive.mount('/content/drive')

Buat Folder, misal "try_Ai_Project_Sep_2020" di Gdrive
import os
os.chdir("/content/drive/My Drive")
if not os.path.exists('try_Ai_Project_Sep_2020'):
 os.makedirs('try_Ai_Project_Sep_2020')
 #print('Path blm ada')
else:
 print('Path sdh ada')
os.chdir("/content/drive/My Drive/try_Ai_Project_Sep_2020")
!pwd

%tensorflow_version 1.x

%rm -rf ./stylegan2/
!git clone https://github.com/woctezuma/stylegan2.git

#%cd stylegan2/
os.chdir("/content/drive/My Drive/try_Ai_Project_Sep_2020/stylegan2")
!nvcc test_nvcc.cu -o test_nvcc -run
```

Link kode lengkapnya:

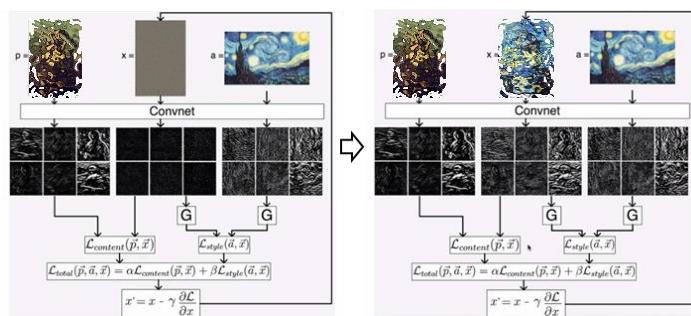
<https://colab.research.google.com/drive/11THATOZUciWf2tsivgWDS3Gxzd4K9GmV?usp=sharing>



## 22.4 Prototype Style Transfer dengan CNN Net. (*Build from Scratch*) like ResNet, VGG, Inception, etc

### 22.4.1 Konsep

Dalam beberapa penelitian sebelumnya, dibuat efek style transfer menggunakan algoritma Convolutional Neural Networks (CNN) dengan memanfaatkan hasil layer-layer pada Convolution dan Pooling sebagai hasil ekstraksi fitur, dan mengabaikan layer pada fully connected dan softmax (karena kedua layer ini digunakan proses klasifikasi, dan proses ini tidak dibutuhkan pada proses style transfer) [10][Gatys, L.A., 2015][Mordvintsev, et al., 2018].



Gambar 22.7 Ilustrasi Cara Kerja Style Transfer

### 22.4.2 Tampilan Implementasi

Hasil Style transfer dari beberapa percobaan dengan Teknik-teknik auto grad untuk optimasi yang berbeda.



Alur algoritma Style Transfer, terdapat original image (p), kemudian terdapat style image (a), dan citra hasil akhir atau generated image (x). Berikut langkah-langkah style transfer:

1. Hitung square-error Content Loss untuk p terhadap x pada layer diantara dua feature representation F dan P pada layer l, di mana  $F^l(i,j)$  merupakan nilai aktivasi dari filter ke-i pada posisi j pada layer l.

$$Loss_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{i,j}^l - P_{i,j}^l)^2$$

$F^l(i,j) \in \mathcal{R}^{N_l} \times M_l$ ,  $N_l$  merupakan banyaknya filter yang tiap feature maps berukuran  $M_l$ , ( $M_l = H \times W$ ).

2. Hitung feature correlations (Gram matrix), di mana  $G^l(i,j)$  merupakan inner product dari feature map i dan j pada layer l.

$$G_{i,j}^l = \sum_k F_{i,k}^l F_{j,k}^l$$

3. Hitung square-error Style Loss untuk a terhadap x pada layer diantara dua feature representation G dan A pada layer l, di mana  $A^l(i,j)$  merupakan nilai style representations dari filter ke-i pada posisi j pada layer l.

$$Loss_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L-1} \left( w_l \left( \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{i,j}^l - A_{i,j}^l)^2 \right) \right)$$

4. Hitung Loss Function total

$$Loss_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha Loss_{content}(\vec{p}, \vec{x}) + \beta Loss_{style}(\vec{a}, \vec{x})$$

$\alpha$  dan  $\beta$  merupakan faktor bobot, untuk content dan style. Rasio yang digunakan  $\alpha/\beta$  berdasarkan Gatys et. al (2015) adalah  $1 \times 10^{-3}$  atau  $1 \times 10^{-4}$ .

### 22.4.3 Source Code

Kode program terdiri dari beberapa bagian, yaitu Deklarasi fungsi-fungsi (terdapat juga kode untuk convolution 2D dan 3D pada matrik citra), inisialisasi *cell* sebagai penampung data bervariasi dan dinamis untuk menampung hasil-hasil convolution dan pooling, load

data yang digunakan serta kode program utama. Pada kode yang kami ditampilkan dibatasi hanya dari deklarasi kode program utama.

```
#Prototype Deep-Close Style Transfer

set nilai parameter
alpha = 18
betha = 1e-1
alpha = 1
betha = 1e-3
alpha = 1
betha = 1e-1
batch_size = 4

gamma = 50.0
gamma_awal = copy.copy(gamma)
Factor_adaptive_gamma = 1.5

belum optimal
gamma = .5

gamma = 2
gamma = 0.001
Iter_Max = 125
Iter_Max = 100

info parameter utk di-insert pada file Total_Loss.csv
info_param = 'a-'+str(alpha)+'-b-'+str(betha)+'-g-' \
+str(gamma_awal)+ '-f-' +str(Factor_adaptive_gamma) +'-eph-' +str(Iter_Max)

Target utk p_go di featureMap_p
#idx_layer_content = 0
#idx_channel_content = 4

x_model_search = np.ar-
ray(scaleTo(x_go.copy()), list_fix_size_tiap_layer[0], list_fix_size_tiap_layer[0])
x_model_search = 255. + x_model_search.astype('float64')

Tampung Total Loss utk grafik
save_Total_Loss = np.zeros(Iter_Max,np.float64)

Tampung Total Loss old untuk momentum agar bisa update gamma dalam iterasi
Total_Loss_old = 0.

Buat Folder, dgn name unik_path2save utk simpan hasil style transfer dan to-
tal loss-nya
name_unik_path2save = str(datetime.datetime.today().astimezone(pytz.timezone('Asia/Ja-
karta'))).strftime('%d-%m-%Y-%H-%M-%S')
import os
os.chdir("/content/drive/My Drive")
if not os.path.exists('./img 2d/hasil style/'+name_unik_path2save):
 os.makedirs('./img 2d/hasil style/'+name_unik_path2save)
 #print('Path blm ada')
else:
 print('Path sdh ada')
os.chdir("/content/drive/My Drive/KeStar Algorithm")
!pwd

misal, utk spesific layer dan channel
ini harus benar-benar dipilih, karena ketika berbeda jenis network akan ber-
beda
ini saya coba seperti Gatys, hasilnya tidak begitu bagus, krn bisa jadi ka-
rena Network-nya beda:
focus_feature_map_content = ['conv4_2']
focusfea-
ture map style = ['conv1_1', 'conv2_1','conv3_1', 'conv4_1', 'conv5_1']
#
Coba ganti dengan melihat secara visual hasil dari 3D dan 2D Conv & Max-Pool-
ing:D

focus_feature_map_content = ['pool1_2']
focusfea-
ture_map_style = ['conv1_1', 'pool1_1', 'conv2_1', 'conv2_2', 'pool2_1']
```

```
focus feature_map_content = ['pooll_1'] # sdah, hsl tdk bagus
focus_feature_map_content = ['pooll_4'] # sdah, hsl lumayan bagus

focus_feature_map_content = ['pool1_2']
focus_feature_map_style = ['conv1_1', 'conv1_2', 'conv1_3', 'conv1_4', 'conv1_5']
focus feature_map_style = ['conv1_1', 'conv1_2', 'conv1_3', 'conv1_4', 'conv1_5',
'pool1_1', 'pool1_2', 'pool1_3']
focus feature_map_style = ['conv1_3', 'conv1_3', 'conv1_3', 'conv1_3', 'conv1_3', 'pool1_3',
'pool1_3', 'pool1_3']
focus feature_map_style = ['conv1_3', 'conv1_3', 'conv1_3', 'conv1_3', 'conv1_3', 'pool1_3',
'pool1_3', 'pool1_3']

optimizer = ["SGD", "SGDAnn", "Momentum", "Adam", "AdamAnn"]
pilih_optimizer = 'AdamAnn'

parameter optimizer = Momentum
betha_gamma = 0.15
avg_grads = 0.*x_model_search.copy()

parameter optimizer = Adam
lr_basis = copy.copy(gamma)
anneal_epoch_freq = 4.
betha_gamma1 = 0.15
betha_gamma2 = 0.15
avg_squared_grads = 0.*x_model_search.copy()

Matrix_lr_basis & Matrix_gamma dgn nilai semuanya sama (same value)
Matrix_lr_basis = lr_basis + avg_squared_grads.copy()
Matrix_gamma = lr_basis + avg_squared_grads.copy()

Misal, mencoba Matrix_lr_basis & Matrix_gamma dgn nilai semuanya random (dynamic value)
Matrix_lr_basis = avg_squared_grads.copy()
Matrix_gamma = avg_squared_grads.copy()
Matrix_lr_basis[:, :, 0] = myrand-
float(list_fix_size_tiap_layer[0], list_fix_size_tiap_layer[0], lr_basis, 2*lr_basis)
Matrix_lr_basis[:, :, 1] = myrand-
float(list_fix_size_tiap_layer[0], list_fix_size_tiap_layer[0], lr_basis, 2*lr_basis)
Matrix_lr_basis[:, :, 2] = myrand-
float(list_fix_size_tiap_layer[0], list_fix_size_tiap_layer[0], lr_basis, 2*lr_basis)
Matrix_gamma[:, :, 0] = myrand-
float(list_fix_size_tiap_layer[0], list_fix_size_tiap_layer[0], lr_basis, 2*lr_basis)
Matrix_gamma[:, :, 1] = myrand-
float(list_fix_size_tiap_layer[0], list_fix_size_tiap_layer[0], lr_basis, 2*lr_basis)
Matrix_gamma[:, :, 2] = myrand-
float(list_fix_size_tiap_layer[0], list_fix_size_tiap_layer[0], lr_basis, 2*lr_basis)

Factor_smoothing utk menghindari pembagian nol
Factor_smoothing = 0.001

Iterasi untuk Optimasi
for i_final in range(Iter_Max):

 # Update featureMap_x
 size_ksize = 3
 size_stride = 1
 for i_byk_layer in range(byk_layer):
 byk_channel = byk_channel_tiap_layer[i_byk_layer]
 fix_size = list_fix_size_tiap_layer[i_byk_layer]
 method = methods[i_byk_layer]
 #stride = size_stride, size_stride

 #x_new_cnn = np.array(scaleTo(x_model_search.copy(), fix_size, fix_size))
 if i_byk_layer == 0:
 x_new_cnn = np.array(
 scaleTo(np.clip(x_model_search, 0, 255).astype('float64').copy(), fix_size, fix_size))

 if i_byk_layer%2==0:
 str_init ='conv'+str(int(((i_byk_layer-0)/2)) + 1)
 #print('utk '+str_init)
```

```
else:
 size_stride += 1
 str_init = 'pool'+str(int(((i_byk_layer-1)/2)) + 1)
 #print('utk '+str_init)

 # reset size_ksize
 #size_ksize = 3
 stride = size_stride,size_stride
 ksize = size_ksize,size_ksize

for i_byk_channel in range(byk_channel):
 pad = True
 #ksize = size_ksize,size_ksize
 if i_byk_layer == 0 & i_byk_channel == 0:
 featureMap_x[i_byk_layer][i_byk_channel] =ReLU(cov_or_pool_3D(x_new_cnn,ksize,stride,method,pad))
 featureMap_x_b[i_byk_layer][i_byk_channel] =ReLU(cov_or_pool_2D(x_new_cnn[:,0],ksize,stride,method,pad))
 featureMap_x_g[i_byk_layer][i_byk_channel] =ReLU(cov_or_pool_2D(x_new_cnn[:,1],ksize,stride,method,pad))
 featureMap_x_r[i_byk_layer][i_byk_channel] =ReLU(cov_or_pool_2D(x_new_cnn[:,2],ksize,stride,method,pad))
 remember_ii = i_byk_layer
 remember_jj = i_byk_channel
 else:
 if i_byk_channel > 0:
 size_stride = 1
 stride = size_stride,size_stride

 #featureMap_x[i_byk_layer][i_byk_channel]=ReLU(cov_or_pool_2D(np.array(scaleTo(featureMap_x[remember_ii][remember_jj], fix_size, fix_size)),ksize,stride,method,pad))
 #featureMap_x_b[i_byk_layer][i_byk_channel]=ReLU(cov_or_pool_2D(np.array(scaleTo(featureMap_x_b[remember_ii][remember_jj], fix_size, fix_size)),ksize,stride,method,pad))
 #featureMap_x_g[i_byk_layer][i_byk_channel]=ReLU(cov_or_pool_2D(np.array(scaleTo(featureMap_x_g[remember_ii][remember_jj], fix_size, fix_size)),ksize,stride,method,pad))
 #featureMap_x_r[i_byk_layer][i_byk_channel]=ReLU(cov_or_pool_2D(np.array(scaleTo(featureMap_x_r[remember_ii][remember_jj], fix_size, fix_size)),ksize,stride,method,pad))

 featureMap_x[i_byk_layer][i_byk_channel]=ReLU(cov_or_pool_2D(featureMap_x[remember_ii][remember_jj],ksize,stride,method,pad))
 featureMap_x_b[i_byk_layer][i_byk_channel]=ReLU(cov_or_pool_2D(featureMap_x_b[remember_ii][remember_jj],ksize,stride,method,pad))
 featureMap_x_g[i_byk_layer][i_byk_channel]=ReLU(cov_or_pool_2D(featureMap_x_g[remember_ii][remember_jj],ksize,stride,method,pad))
 featureMap_x_r[i_byk_layer][i_byk_channel]=ReLU(cov_or_pool_2D(featureMap_x_r[remember_ii][remember_jj],ksize,stride,method,pad))

 ## jika dgn filter lbp
 ##featureMap_x[i_byk_layer][i_byk_channel]=ReLU(calc_lbp(cov_or_pool_2D(np.array(scaleTo(featureMap_x[i_byk_layer][i_byk_channel-1], fix_size, fix_size),ksize,stride,method,pad)))
 ##featureMap_x_b[i_byk_layer][i_byk_channel]=ReLU(calc_lbp(cov_or_pool_2D(np.array(scaleTo(featureMap_x_b[i_byk_layer][i_byk_channel-1], fix_size, fix_size),ksize,stride,method,pad)))
 ##featureMap_x_g[i_byk_layer][i_byk_channel]=ReLU(calc_lbp(cov_or_pool_2D(np.array(scaleTo(featureMap_x_g[i_byk_layer][i_byk_channel-1], fix_size, fix_size),ksize,stride,method,pad)))
 ##featureMap_x_r[i_byk_layer][i_byk_channel]=ReLU(calc_lbp(cov_or_pool_2D(np.array(scaleTo(featureMap_x_r[i_byk_layer][i_byk_channel-1], fix_size, fix_size),ksize,stride,method,pad)))

 remember_ii = i_byk_layer
 remember_jj = i_byk_channel

 # non aktifkan size ksize, misal utk squential-Gate
 #size_ksize = size_ksize + 2
 # non aktifkan jika memang ukuran semua layernya dibuat sama, jika beda aktifkan dgn ketentuan yg berikutnya
 # ukurannya setengahnya dari sebelumnya
 #if byk_channel_tiap_layer[0] == byk_channel_tiap_layer[1]:
 #else:
```

```
#size_stride = size_stride + 1

untuk set target di content image
#idx_layer_content = 0
#idx_channel_content = 4
#print('set target utk p.go featureMap pada idx_layer content = ',idx_layer_content,' - idx_channel_content = ',idx_channel_content)

untuk all layer dan all channel
#Matrix_Total_Loss, Total_Loss = \
#Fn_Matrix_Total_Loss_Lite_v2(idx_layer_content,idx_channel_content,\
#featureMap_p,featureMap_x,featureMap_a,\
#featureMap_p_b, featureMap_p_g, featureMap_p_r, \
#featureMap_x_b, featureMap_x_g, featureMap_x_r, \
#featureMap_a_b, featureMap_a_g, featureMap_a_r, \
#byk_layer,byk_channel tiap layer,\
#list_fix_size_tiap_layer,batch_size,alpha,betha)

untuk spesifik layer dan channel
misal, focus_feature_map_content = ['conv4_2']
focus fea-
ture_map_style = ['conv1_1', 'conv2_1','conv3_1', 'conv4_1', 'conv5_1']
#
Matrix_Total_Loss, Total_Loss = \
Fn_Matrix_Total_Loss_Lite_v2_1(focus_feature_map_content,focus_fea-
ture_map_style,\
 featureMap_p,featureMap_x,featureMap_a,\
 featureMap_p_b, featureMap_p_g, featureMap_p_r, \
 featureMap_x_b, featureMap_x_g, featureMap_x_r, \
 featureMap_a_b, featureMap_a_g, featureMap_a_r, \
 featureVec_featureMap_x_b, featureVec_featureMap_x_g, fea-
tureVec_featureMap_x_r, \
 featureVec_featureMap_a_b, featureVec_featureMap_a_g, fea-
tureVec_featureMap_a_r, \
 gram_matrix_x_b, gram_matrix_x_g, gram_matrix_x_r, \
 gram_matrix_a_b, gram_matrix_a_g, gram_matrix_a_r, \
 byk_layer,byk chan-
nel_tiap_layer,list_fix_size_tiap_layer,batch_size,alpha,betha)

#save_Total_Loss[i_final-1] = Total_Loss
save_Total_Loss[i_final] = Total_Loss

Pilih Optimizer:

if optimizer == "SGDAnn" atau SGD Annealling
if pilih_optimizer == 'SGD':
 # gamma atau learning rate tetap
 gamma = gamma

elif pilih_optimizer == 'SGDAnn':
 if i_final ==0:
 Total_Loss_old = Total_Loss
 else:
 if(Total_Loss >= Total_Loss_old):
 #gamma *= 0.1
 gamma -= Factor_adaptive_gamma

 # menjaga agar gamma tidak negatif
 if(gamma <= Factor_adaptive_gamma):
 gamma = copy.copy(Factor_adaptive_gamma)

 #update Factor_adaptive_gamma
 Factor_adaptive_gamma = Factor_adaptive_gamma/10.

 # menjaga agar gamma tidak negatif
 # gamma = max(gamma,1.5)

 #gamma = gamma
else:
 Total_Loss_old = Total_Loss
elif pilih_optimizer == 'Momentum':
 #avg_grads = avg_grads*(gamma/100) + (1.0-(gamma/100))*Matrix_Total_Loss
 avg_grads = avg_grads*beta_gamma + (1.0-beta_gamma)*Matrix_Total_Loss
 Matrix_Total_Loss = avg_grads.copy()
 #x model_search += - gamma*Matrix_Total_Loss
elif pilih_optimizer == 'Adam_with_Matrix_lr_same_value':
 if i_final == 0:
 Matrix_lr_basis = lr_basis + 0.*avg_squared_grads.copy()
 Matrix_gamma = lr_basis + 0.*avg_squared_grads.copy()
```

```
 avg_grads = avg_grads*beta_gamma1 + (1.0-beta_gamma1)*Matrix_Total_Loss
 avg_squared_grads = avg_squared_grads*beta_gamma2 + (1.0-beta_gamma2)*(Matrix_Total_Loss**2)
 Matrix_gamma = Matrix_lr_basis / np.sqrt(avg_squared_grads+Factor_smoothing)
 Matrix_Total_Loss = avg_grads.copy()
 #x_model_search += - gamma*Matrix_Total_Loss
 elif pilih_optimizer == 'Adam_with_Matrix_lr_random_value':
 if i_final == 0:
 Matrix_lr_basis = 0.*avg_squared_grads.copy()
 Matrix_gamma = 0.*avg_squared_grads.copy()
 Matrix_lr_basis[:, :, 0] = myrand_
 float(list_fix_size_tiap_layer[0],list_fix_size_tiap_layer[0],lr_basis,2*lr_basis)
 Matrix_lr_basis[:, :, 1] = myrand_
 float(list_fix_size_tiap_layer[0],list_fix_size_tiap_layer[0],lr_basis,2*lr_basis)
 Matrix_lr_basis[:, :, 2] = myrand_
 float(list_fix_size_tiap_layer[0],list_fix_size_tiap_layer[0],lr_basis,2*lr_basis)
 Matrix_gamma[:, :, 0] = myrand_
 float(list_fix_size_tiap_layer[0],list_fix_size_tiap_layer[0],lr_basis,2*lr_basis)
 Matrix_gamma[:, :, 1] = myrand_
 float(list_fix_size_tiap_layer[0],list_fix_size_tiap_layer[0],lr_basis,2*lr_basis)
 Matrix_gamma[:, :, 2] = myrand_
 float(list_fix_size_tiap_layer[0],list_fix_size_tiap_layer[0],lr_basis,2*lr_basis)
 avg_grads = avg_grads*beta_gamma1 + (1.0-beta_gamma1)*Matrix_Total_Loss
 avg_squared_grads = avg_squared_grads*beta_gamma2 + (1.0-beta_gamma2)*(Matrix_Total_Loss**2)
 Matrix_gamma = Matrix_lr_basis / np.sqrt(avg_squared_grads+Factor_smoothing)
 Matrix_Total_Loss = avg_grads.copy()
 #x_model_search += - gamma*Matrix_Total_Loss
 elif pilih_optimizer == 'Adam':
 avg_grads = avg_grads*beta_gamma1 + (1.0-beta_gamma1)*Matrix_Total_Loss
 avg_squared_grads = avg_squared_grads*beta_gamma2 + (1.0-beta_gamma2)*(Matrix_Total_Loss**2)
 gamma = lr_basis / np.sqrt(avg_squared_grads+Factor_smoothing)
 Matrix_Total_Loss = avg_grads.copy()
 #x_model_search += - gamma*Matrix_Total_Loss
 elif pilih_optimizer == 'AdamAnn':
 if (i_final+1)%anneal_epoch_freq==0:
 lr_basis = lr_basis/4.0
 avg_grads = avg_grads*beta_gamma1 + (1.0-beta_gamma1)*Matrix_Total_Loss
 avg_squared_grads = avg_squared_grads*beta_gamma2 + (1.0-beta_gamma2)*(Matrix_Total_Loss**2)
 gamma = lr_basis / np.sqrt(avg_squared_grads+Factor_smoothing)
 Matrix_Total_Loss = avg_grads.copy()
 #x_model_search += - gamma*Matrix_Total_Loss
 #
 # -----
 # dLoss / dx_go dengan pendekatan langsung menggunakan hasil turunan pertamanya (SGD):
 # -----
 #misal menjadi x_model_search as result of style transfer dgn Optimizer Standar/ Vanilla SGD atau lainnya
 #x_model_search = x_model_search - (gamma*Matrix_Total_Loss)
 if pilih_optimizer == 'Adam_with_Matrix_lr_same_value' or pilih_optimizer == 'Adam_with_Matrix_lr_random_value':
 x_model_search += - Matrix_gamma*Matrix_Total_Loss
 else:
 #if gamma.shape[0] == 1:
 #print('Cek Learning Rate: ',gamma)
 x_model_search += - gamma*Matrix_Total_Loss
 # menampilkan x_new
 #print('i_final:', i_final, ' x_model_search by Standard/ Vanilla SGD:')
 print('i_final:' + str(i_final) + ' x_model_search by Optimizer ' + pilih_optimizer +':')
 #print('total error bgr = ',total_error_bgr)

 plt.imshow(cv2.cvtColor(np.clip(x_model_search, 0, 255).astype('uint8'), cv2.COLOR_BGR2RGB))
 plt.axis('off')
```

```
plt.show()

menyimpan hasil style transfer
nama_content_img = img_ct_
nama_style_img = img_st_
#nama_path_hasil = './img 2d/hasil style/'
nama_path_hasil = './img 2d/hasil style/'+name_unik_path2save+''
#nama_file_generate = 'hasil_style_transfer'+str(i_final)+'.png'
nama_file_generate = 'ct-'+nama_content_img+'-st-'+nama_style_img+'-iter-
'+str(i_final)+'-v2.jpg'

#name_unik_path2save = str(datetime.today().strftime('%d-%m-%Y-%H-%M-%S'))
cv2.imwrite(nama_path_hasil+nama_file_generate, np.clip(x_model_search, 0, 255).astype('uint8'))

Simpan Total Loss, nantinya utk grafik evaluasi
nama_file_csv = nama_path_hasil+info_param+'-Total_Loss.csv'
np.savetxt(nama_file_csv, save_Total_Loss, fmt="%d", delimiter=",")
np.savetxt('Total_Loss.csv', save_Total_Loss, fmt="%d", delimiter=",")

Tampilkan Grafik Total Loss
print()
style.use('ggplot')

#per_data=genfromtxt('result.csv',delimiter=','
path_dir = nama_path_hasil
items_hasil_style = os.listdir(path_dir)
per_data=genfromtxt(nama_file_csv,delimiter=',')
#per_data=genfromtxt('./img 2d/hasil style/21-08-2020-17-37-05/To-
tal_Loss.csv',delimiter=',')
#per_data=genfromtxt('Total_Loss.csv',delimiter=',')
nama_file_jpg = nama_path_hasil+info_param+'-Total_Loss.jpg'
plt.xlabel ('Iteration')
plt.ylabel ('Total Loss')
plt.title('Result of | '+info_param)
plt.ylim(np.min(per_data)-1,np.max(per_data)+1)
plt.xlim(1,len(per_data))
plt.plot(np.arange(1, len(per_data)+1),per_data)
plt.savefig(nama_file_jpg,dpi=100)
plt.show()
plt.style.use('default')
print('Done..!')
```

Link kode lengkapnya:

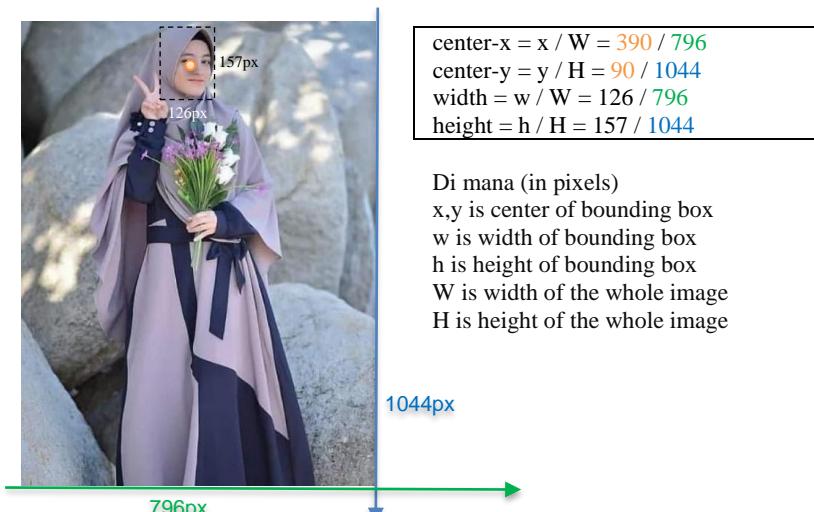
[https://colab.research.google.com/drive/17Ny\\_fUOZetginw2PlvXeSpCc3YKoEmDF?usp=sharing](https://colab.research.google.com/drive/17Ny_fUOZetginw2PlvXeSpCc3YKoEmDF?usp=sharing)



## 22.5 Deteksi Otomatis General Fashion (case Study: Jilbab Fashion) dengan Yolo untuk Level Up Market Bidang Ekonomi & Bisnis

### 22.5.1 Konsep

Pada era millenial saat ini, indentifikasi dan analisis general fashion secara otomatis dari pakaian merupakan case yang menarik dalam menunjang Level Up nilai ekonomi dari penjualan di pasar modern, sehingga butuh terus dikembangkan dari tahun ke tahun terkait dengan banyak aplikasi yang memiliki struktur yang sangat kompleks dan memiliki karakteristik masing-masing. Dari karakteristik-karakteristik unik tersebut, penelitian menyebutkan bahwa terkandung begitu banyak informasi yang dapat diperoleh dari analisis citra pakaian seseorang seperti ekspresi, kesukaan, emosi dan lain sebagainya. Dalam bidang *computer vision* sendiri terus berusaha bagaimana caranya agar komputer dapat meniru cara kerja indra manusia. Hal ini menjadi menarik untuk membuat suatu sistem yang dapat mendeteksi jenis pakaian misal Jilbab untuk wanita yang menggunakan hijab yang syar'i. Berikut ilustrasi pengambilan datanya untuk persiapan proses training algoritma Yolo yang berbasis Deep Learning.



Gambar 22.8 Pengambilan data dengan crop berupa bounding box pada bagian wajah dan Jilbab

## 22.5.2 Tampilan Implementasi

Pada Gambar (a), merupakan hasil deteksi seorang ibu mengenakan hijab pada foto keluarga dengan kondisi kompleks, yang mana sebagian anggota keluarga tersebut ada yang memang tidak berhijab (sang suami) dan anak-anaknya mereka yang memang masih kecil. Dan pada Gambar (b), merupakan hasil deteksi seorang Ibu yang sedang menggendong anaknya, di samping suami.



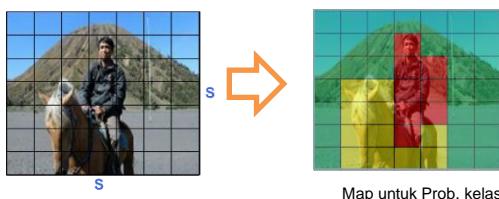
(a)



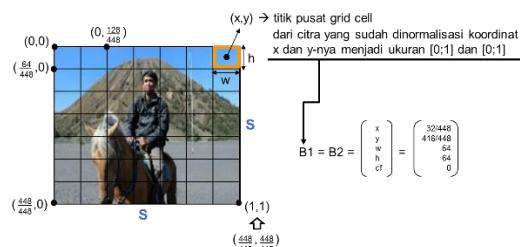
(b)

You Only Look Once (Yolo) adalah sebuah algoritma yang dikembangkan untuk mendeteksi sebuah objek secara real-time. Sistem pendekripsi yang dilakukan adalah dengan menggunakan repurpose classifier atau localizer untuk melakukan deteksi. Sebuah model diterapkan pada sebuah citra di beberapa lokasi dan skala. Daerah dengan citra yang diberi score paling tinggi akan dianggap sebagai sebuah pendekripsi (Unsky, 2017). Yolo menggunakan pendekatan jaringan saraf tiruan (JST) untuk mendekripsi objek pada sebuah citra. Jaringan ini membagi citra menjadi beberapa wilayah dan memprediksi setiap kotak pembatas dan probabilitas untuk setiap wilayah. Kotak-kotak pembatas ini kemudian dibandingkan dengan setiap probabilitas yang diprediksi. Yolo memiliki beberapa kelebihan dibandingkan dengan sistem yang berorientasi pada classifier, terlihat dari seluruh citra pada saat dilakukan test dengan prediksi yang diinformasikan secara global pada citra (Redmon, 2016). Hal tersebut juga membuat prediksi dengan sintesis jaringan saraf ini tidak seperti sistem Region-Convolutional Neural Network (R-CNN) yang membutuhkan ribuan untuk sebuah citra sehingga membuat Yolo lebih cepat hingga beberapa kali daripada R-CNN. Tahapan algoritma Yolo dapat dilihat pada delapan langkah berikut:

- a. Baca data citra dengan ukuran sembarang.
- b. Ubah ukuran citra menjadi  $448 \times 448$ , lalu buat *grid* pada citra dengan ukuran  $S \times S$  grids.



- c. Tiap *grid cell*, misal  $nB=2$ , terdapat  $B$  yang berisi 5 nilai, yaitu lokasi koordinat  $x$  diasumsikan berdasarkan baris ( $x_{br}$ ), koordinat  $y$  berdasarkan kolom ( $y_{kol}$ ), ukuran dan nilai *confidence* ( $x, y, w, h, cf$ ) terhadap  $nB$  bbox yang ada, yaitu B1 dan B2.

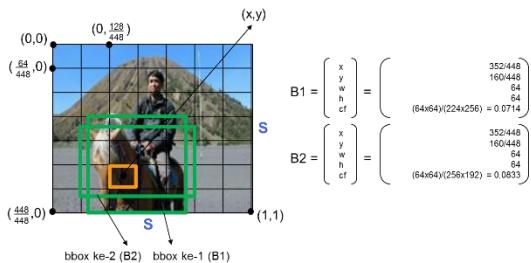


$cf = 0$  (selalu di-set = 0, jika dalam *grid cell* adalah *background*).  $Confidence (cf) = P(\text{object}) * \text{IoU}$ , yang mana,  $nB$  menyatakan banyaknya *bbox*,  $B1$  untuk *bbox1*,  $B2$  untuk *bbox2*, dan *bbox* menyatakan *bounding box*. Persamaan 2 merupakan *Intersection Over Union* ( $\text{IoU}_{\text{pred}}^{\text{truth}}$ ).

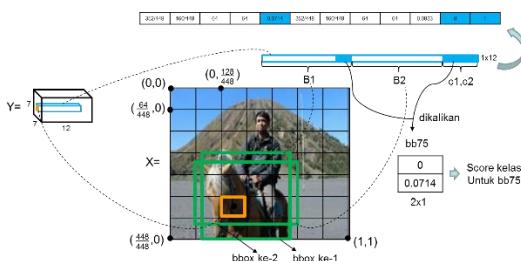
$$\text{IoU} = \frac{\text{Irisan}}{\text{Gabungan}} = \frac{\text{Irisan}}{\text{Pred.} + \text{Gabungan} - \text{Irisan}}$$

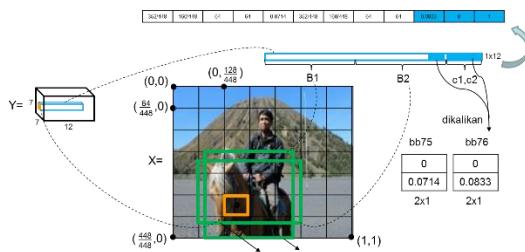
Pada perhitungan  $confidence (cf) = P(\text{object}) * \text{IoU}$ ,  $P(\text{object})$  dapat dilewati terlebih dahulu, sehingga cukup dengan  $cf = \text{IoU}$ . Karena  $P(\text{class}_i | \text{object}) * P(\text{object}) * \text{IoU} = P(\text{class}_i) * \text{IoU}$ .

- d. Jika 2 *bbox* mengacu pada kelas yang sama, maka hasil ukuran tensornya adalah  $(S \times S \times (nB \times 5 + nC)) = (7 \times 7 \times (2 \times 5 + 2)) = (7 \times 7 \times 12)$  tensor.



Di mana, *bbox* ke-1 dan *bbox* ke-2, dibuat bebas dan sebaiknya berbeda. Hitung matriks *bb* dengan ukuran  $(nC \times (S \times S \times nB)) = 2 \times (7 \times 7 \times 2) = (2 \times 98)$ , mulai dari *grid cell* ke-1 sampai ke-49 pada 2 *bbox* yang mengacu pada kelas yang sama, misal untuk bb75 dan bb76.

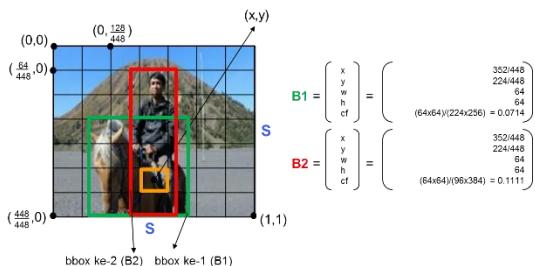




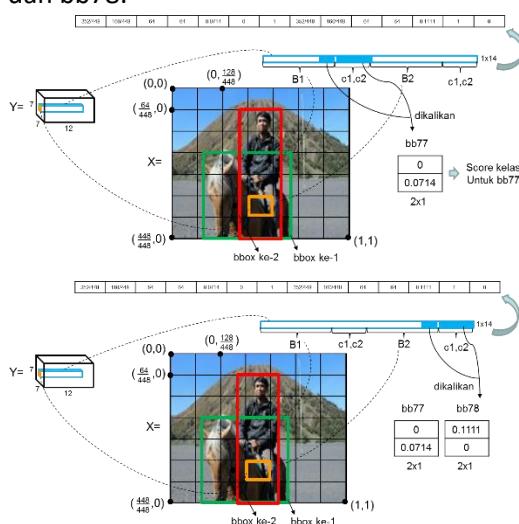
Hasil dari matriks bb.

$$\begin{array}{ll} \text{bb1} & \text{bb2} \\ \begin{pmatrix} 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 \end{pmatrix} \\ 2 \times 1 & 2 \times 1 \end{array} \quad \dots \quad \begin{array}{ll} \text{bb75} & \text{bb76} \\ \begin{pmatrix} 0 & 0.0714 \end{pmatrix} & \begin{pmatrix} 0 & 0.0833 \end{pmatrix} \\ 2 \times 1 & 2 \times 1 \end{array} \quad \dots \quad \begin{array}{ll} \text{bb97} & \text{bb98} \\ \begin{pmatrix} 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 \end{pmatrix} \\ 2 \times 1 & 2 \times 1 \end{array}$$

- e. Jika 2 bbox mengacu pada kelas yang berbeda atau karena terdapat *overlapping object*, maka hasil ukuran tensornya adalah  $(S \times S \times (nB \times 5 + 2 \times nC)) = (7 \times 7 \times (2 \times 5 + 2 \times 2)) = (7 \times 7 \times 14)$  tensor.



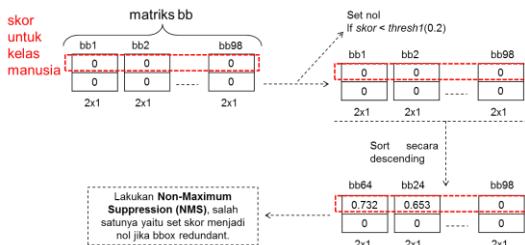
Di mana, bbox ke-1 dan bbox ke-2, mengacu pada objek yang berbeda kelas. Hitung matriks bb dengan ukuran  $(nC \times (S \times S \times nB)) = 2 \times (7 \times 7 \times 2) = (2 \times 98)$ , mulai dari grid cell ke-1 sampai ke-49 pada 2 bbox yang mengacu pada kelas yang berbeda, misal untuk bb77 dan bb78.



### Hasil dari matriks bb.

| Kelas   | bb1 | bb2 | bb77   | bb78   | bb97 | bb98 |
|---------|-----|-----|--------|--------|------|------|
| manusia | 0   | 0   | 0      | 0.1111 | 0    | 0    |
| kuda    | 0   | 0   | 0.0714 | 0      | 0    | 0    |

- f. Tiap kelas pada matriks *bb*, lakukan set skor = 0, jika skor < *thresh1*(0.02), kemudian urutkan secara descending.

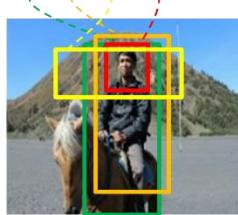


- g. Lakukan *Non-Maximum Suppression* (NMS).

- Daftar semua bbox.

skor untuk setiap bbox pada kelas manusia

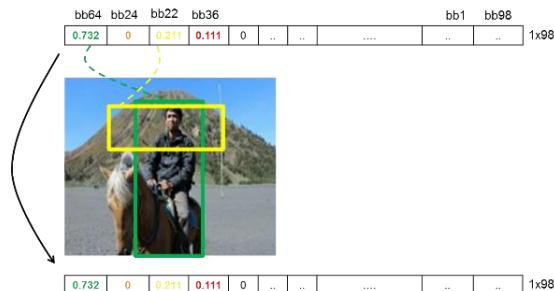
| bb64  | bb24  | bb22  | bb36  | bb1 | bb98 | 1x98 |
|-------|-------|-------|-------|-----|------|------|
| 0.732 | 0.653 | 0.211 | 0.111 | 0   | ...  | ...  |



- Set bbox dengan skor maks, sebagai "bbox\_max". Misal bb64 diketahui sebagai bbox\_max.
- Bandingkan "bbox\_max" dengan bbox lainnya sebagai "bbox\_cur" yang memiliki skor dibawahnya dan tidak nol. Jika  $\text{IoU}(\text{bbox}_{\text{max}}, \text{bbox}_{\text{cur}}) > 0.5$ , maka set skor nol untuk bbox\_cur. Dari hasil perbandingan, jika  $\text{bbox}_{\text{max}} = \text{bb64}$ ,  $\text{bbox}_{\text{cur}} = \text{bb24}$ , maka  $\text{IoU}(\text{bbox}_{\text{max}}, \text{bbox}_{\text{cur}}) > 0.5$  (true), maka set skor  $\text{bb24} = 0$ .



- Lanjutkan ke bb22, dari hasil perbandingan, jika  $\text{bbox\_max} = \text{bb64}$ ,  $\text{bbox\_cur} = \text{bb22}$ , maka  $\text{IoU}(\text{bbox\_max}, \text{bbox\_cur}) > 0.5$  (false), maka skor bb22 tetap.

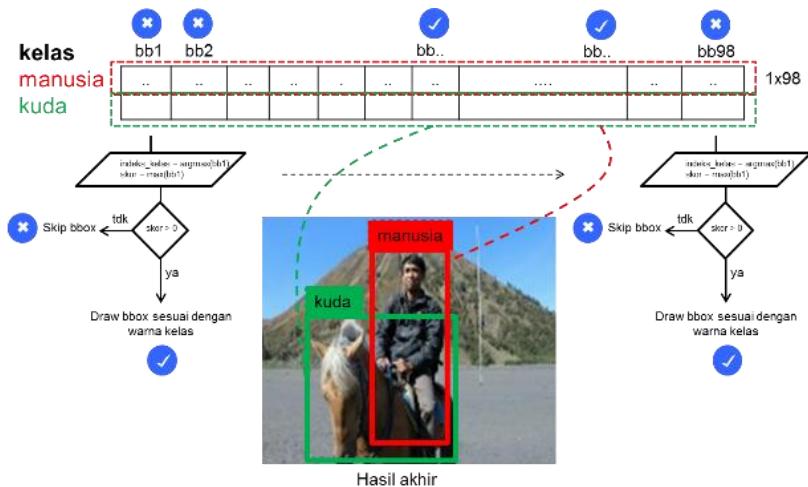


- Lanjutkan ke bb36, dari hasil perbandingan, jika  $\text{bbox\_max} = \text{bb64}$ ,  $\text{bbox\_cur} = \text{bb36}$ , maka  $\text{IoU}(\text{bbox\_max}, \text{bbox\_cur}) > 0.5$  (false), maka skor bb36 tetap.



- Lanjutkan melakukan pembandingan sampai  $\text{bbox\_max} = \text{bb64}$ ,  $\text{bbox\_cur} = \text{bb98}$ . Lalu lanjutkan lagi, set  $\text{bbox\_max} = \text{bb22}$ , dan  $\text{bbox\_cur} = \text{bb36}$ , dst.
- Kemudian lanjutkan ke kelas berikutnya, lakukan hal yang sama seperti yang telah dilakukan pada kelas manusia.

#### h. Plot bounding box berdasar hasil NMS



### 22.5.3 Source Code

Kode program terdiri dari beberapa bagian, yaitu preprosesing (akuisisi data) dan proses utama algoritma Yolo v3.

```
#preprosesing (akuisisi data)

!googleimagesdownload --keywords "hijab syari" --limit 500 --chromedriver \
C:\..\chromedriver.exe
!googleimagesdownload --keywords "foto suami istri hijab" --limit 500 \
--chromedriver C:\..\chromedriver.exe
```

```
#Yolo v3 | Proses Training

from google.colab import drive
drive.mount('/content/drive')
import os
os.chdir("/content/drive/My Drive/darknet/cuDNN")

download cuDNN from thrid-party Nvidia, save on google drive
!wget http://people.cs.uchicago.edu/~kauffm/nvidia/cudnn \
/cudnn-10.0-linux-x64-v7.5.0.56.tgz

Extracts cuDNN files from Drive folder to the VM CUDA folders
!tar -xzvf./cudnn-10.0-linux-x64-v7.5.0.56.tgz -C /usr/local/
!chmod a+r /usr/local/cuda/include/cudnn.h

run only once
os.chdir("/content/gdrive/My Drive/git")

Comment this code on the future runs (cause run only once)
!git clone https://github.com/kriyeng/darknet/
%cd darknet

Run only once
!git checkout feature/google-colab
```

```
#Compile Darknet (run only once)
!make

#Change directory
os.chdir("/content/drive/My Drive/darknet/weights")
!wget https://pjreddie.com/media/files/darknet53.conv.74

os.chdir("/")
!mkdir mydarknet
%cd mydarknet/

Copy the Darknet compiled version to the VM local drive
!cp /content/drive/My\ Drive/darknet/bin/*./darknet

!chmod +x./darknet
!chmod -R 755 /content/drive/My\ Drive/darknet

!./darknet detector train "/content/drive/My Drive/darknet/obj.data" \ "/content/drive/My Drive/darknet/cfg/yolov3.cfg" "/content/drive/My\ Drive/darknet/weights/darknet53.conv.74" -dont_show
```

### #Yolo v3 | Proses Testing

```
os.chdir("/content/drive/My Drive/darknet")
!tar -xzvf.cuDNN/cudnn-10.0-linux-x64-v7.5.0.56.tgz -C /usr/local/
!chmod a+r /usr/local/cuda/include/cudnn.h

os.chdir("/")
!mkdir mydarknet

Copy the Darknet compiled version to the VM local drive
!cp /content/drive/My\ Drive/darknet/bin/*./mydarknet/darknet
!chmod +x./mydarknet/darknet
os.chdir("./mydarknet")
Configure labels
!mkdir -p./data
!cp -R /content/drive/My\ Drive/darknet/labels./data

!./darknet detector test "/content/drive/My Drive/darknet/obj.data" \ "/content/drive/My Drive/darknet/cfg/yolov3.cfg" "/content/drive/My\ Drive/darknet/backup/yolov3_2000.weights" "/content/drive/My\ Drive/darknet/testImgFamily/f1.jpg" -thresh 0.85 -dont-show

 layer filters size input output
 0 conv 32 3 x 3 / 1 416 x 416 x 3 -> 416 x 416 x 32 0.299 BF
 1 conv 64 3 x 3 / 2 416 x 416 x 32 -> 208 x 208 x 64 1.595 BF
...
Done!
/content/drive/My Drive/darknet/testImgFamily/f1.jpg: Predicted in 40.750000
milli-seconds.
hijab: 90%
```

Link kode lengkapnya:

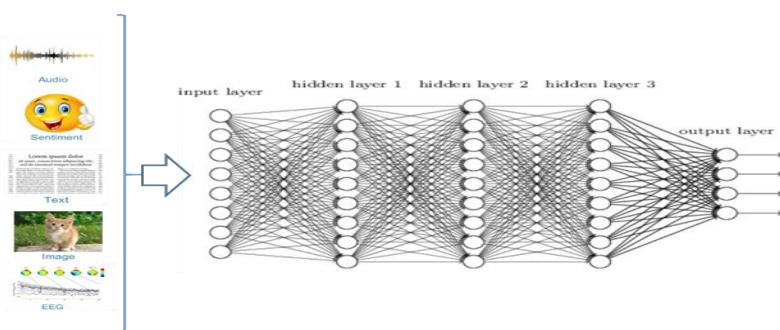
<https://github.com/imamcs19/GreenTech2019-Hijab-Detection>



## 22.6 Prototype Teknik Multimodal Deep Learning untuk Image Captioning

### 22.6.1 Konsep

Berbagai dataset telah banyak dikumpulkan oleh para peneliti, misalnya data dari UCI, Kaggle, github, dan lainnya. Karena data sangat bervariasi, untuk suatu kebutuhan proses komputasi, maka digunakan pendekatan teknik Multimodal Deep Learning yang mampu mengelola proses pembelajaran dalam satu paket training dan testing, tanpa perlu memilah dan membeda-bedakan antara data numerik, citra maupun data dengan tipe lainnya (Mehta, 2018). Ilustrasi Multimodal Deep Learning ditunjukkan pada Gambar berikut.



Gambar 22.9 Multimodal DL dengan data berbeda dan ekstraksi fiturnya

Dalam *case study* ini digunakan teknik multi modal DL dalam Image Captioning, di mana sistem akan secara otomatis mengkonversikan gambar input menjadi keluaran berupa deskripsi keterangan dengan bahasa alami sesuai dengan konten yang diamati dalam gambar tersebut. Image Caption ini menggabungkan keilmuan dari Computer Vision (CV) dan Natural Language Processing (NLP).



Gambar 22.10 Ilustrasi Hasil Auto Image Captioning kata bijak:D

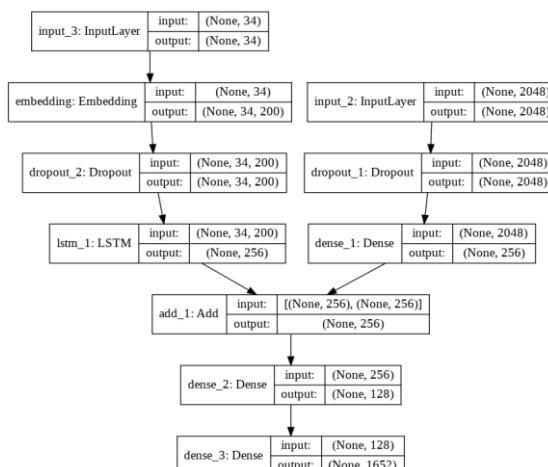
## 22.6.2 Tampilan Implementasi

Contoh hasil Image Captioning pada beberapa Gambar masih belum begitu bagus, dari yang sudah kami coba, sepertinya jika gambarnya sangat jauh berbeda dengan yang ada di data training, maka hasilnya juga masih belum begitu optimal. Berikut adalah contoh hasil image captioning yang dinilai sangat relevan, yang diukur berdasarkan nilai Bilingual Evaluation Understudy (BLEU) Score, di mana nilai evaluasi tersebut digunakan untuk mengukur kalimat alami yang terbentuk dari *generate* sistem dibandingkan dengan kalimat aktualnya.



```
startseq skier is skiing down snowy hill endseq
```

Alur algoritma Multimodal:



Gambar 22.11 Multimodal Neural Network

## 22.6.3 Source Code

Kode program terdiri dari beberapa bagian, yaitu data exploration, glove embeddings, dan caption generator. Pada snippet kode program berikut akan diberikan hasil dari “GloVe Word Embedings” untuk mendapatkan 5 kata terdekat dengan kata kunci masukan.

```
#GloVe Word Embedings

GloVe Word Embedings (Part 2 of 3)

...
menggunakan 100 dimensi kata untuk word embeddings
with open('./glove.6B.200d.txt', 'r') as txt_file:
 glove_embeddings = txt_file.readlines()

membuat dictionary dengan key dari "word" & value adalah embedding-nya
embeddings_dict = {}
for line in glove_embeddings:
 line_content = line.split()
 word, embedding = line_content[0], line_content[1:]
 embeddings_dict[word] = np.asarray(embedding, 'float32')
embeddings_dict['embedding']

Mendeklarasikan Word Similirity
def find_closest_words(embedding):
 return sorted(embeddings_dict.keys(), key=lambda each_word: spatial.distance.euclidean(embeddings_dict[each_word], embedding))

Berikut kode untuk mendapatkan 5 kata terdekat dengan kata kuncil input.
print(find_closest_words(embeddings_dict['indonesia'])[1:6])
print(find_closest_words(embeddings_dict['malang'])[1:6])
```

Output:

```
['indonesian', 'malaysia', 'philippines',
'thailand', 'jakarta']
['blitar', 'semarang', 'jember', 'banyuwangi',
'arema']
```

Link kode lengkapnya:

<https://drive.google.com/file/d/1tmhU3mPWXeZit5EOiQBFzXbKmKPDkzP/view?usp=sharing>



## Daftar Pustaka

- Ardiansyah, A., Rainarli, E. 2017. Implementasi Q-Learning and Backpropagation pada Agen yang Memainkan Permainan Flappy Bird. JNTETI Vol. 6 No. 1, February 2017, ISSN 2301-4156.
- Bellemare, M. G., Naddaf, Y., Veness, J., Dan Bowling, M. 2013. The arcade learning environment: an evaluation platform for general agents. Journal of Artificial Intelligence Research, 47, 253-279.
- Bi, W., Jia, X., & Zheng, M. 2018. A Secure Multiple Elliptic Curves Digital Signature Algorithm for Blockchain. arXiv preprint arXiv:1808.02988.
- Bloembergen, D., Hennes, D., Kaisers, M., Dan Vrancx, P. 2013. Multiagent Reinforcement Learning (MARL), [online] Tersedia di: <<http://www.ecmlpkdd2013.org/wp-content/uploads/2013/09/Multiagent-Reinforcement-Learning.pdf>> [Diakses 4 Juni 2019].
- Brownlee J., 2019. What is Deep Learning?. Diakses pada bulan November tahun 2019 dari <https://machinelearningmastery.com/what-is-deep-learning/>
- Bu, Soniu, L., Babuška, R., and Schutter, B. D. 2006. Multi-agent reinforcement learning: A survey. Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision (ICARCV 2006), Singapore, pp. 527–532, Dec. 2006.
- Chitta, R., Jin, R., Havens, T. C., dan Jain, A. K. 2011. Approximate Kernel K-Means: Solution To Large Scale Kernel Clustering. In KDD, pp. 895-903.
- Cholissodin, I., 2007. Penggunaan Kriptosistem Kurva Elliptik untuk Enkripsi dan Dekripsi Data. Skripsi, Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Airlangga, Surabaya.
- Cholissodin, I., 2014. Classification Of Campus E-Complaint Documents Using Directed Acyclic Graph Multi-Class

SVM Based On Analytic Hierarchy Process. International Conference on Advanced Computer Science and Information Systems (ICACSS), di Jakarta, Indonesia.

Cholissodin I., Setiawan B. D., 2013. Sentiment Analysis Dokumen E-Complaint Kampus Menggunakan Additive Selected Kernel SVM. Seminar Nasional Teknologi Informasi Dan Aplikasinya (SNATIA).

Cholissodin, I., Dewi, C., Dan Surbakti, E. E. 2016. Integrated ANN and Bidirectional Improved PSO for Optimization Of Fertilizer Dose on Palawija Plants. 2nd International Conference on Science in Information Technology (IC-SITech).

Cholissodin, I., Maghfira, T. N. 2017. Pemrograman GPU. Fakultas Ilmu Komputer (FILKOM), Universitas Brawijaya (UB), Malang.

Cholissodin, I., Marji. 2019. Convert Probability Network to Artificial Neural Network based on Position, Time and Speed of Events. 9th Annual Basic Science International Conference 2019 (BaSIC 2019). IOP Publishing.

Cholissodin, I., Palupi, D. E., Putra, M. Y. Y., dan Aprilisia, S. 2020. Detection of Hijab Syar'i as Smart Clothes System for Moslem People Using High Performance of Parallel Computing. IOP Conference Series: Earth and Environmental Science 456 012074.

Cholissodin, I., Riyandani, E. 2016. Analisis Big Data. Fakultas Ilmu Komputer (FILKOM), Universitas Brawijaya (UB), Malang.

Cholissodin, I., Riyandani, E. 2016. Swarm Intelligence. Fakultas Ilmu Komputer (FILKOM), Universitas Brawijaya (UB), Malang.

Cholissodin, I., Setyawan, G. E. 2019. Pengembangan Reinforcement Learning Single-Agent Menggunakan Improve Deep Q-Learning. Seminar Nasional Teknologi & Rekayasa Informasi (SENTRIN) di Senggigi, NTB, Indonesia, 28 – 30 September 2019, ISSN: 2540-9700.

- Cholissodin, I., Sutrisno. 2018. Prediction of Rainfall using Simplified Deep Learning based Extreme Learning Machines. Journal of Information Technology and Computer Science (JITeCS), accredited by number 21/E/KPT/2018 valid from July 9, 2018 to July 9, 2023.
- Cholissodin, I., Sutrisno. 2020. Prediction of Rainfall Using Improved Deep Learning with Particle Swarm Optimization. TELKOMNIKA Telecommunication Computing Electronics and Control, vol. 18, no. 5, pp. 2498~2504.
- Data Science Group at UPB. 2017. Precision, Recall and F1 measure. F.C edited this page on Jan 21, 2017, [Online] Tersedia di: <<https://github.com/dice-group/gerbil/wiki/Precision,-Recall-and-F1-measure>>
- Dean, M. 2018. A Technical Framework for Supply Chain Transformation using Blockchain. School of Computer Science University of Birmingham.
- Dephub. 2019. Angka Kecelakaan Lalin Mudik Tahun Ini Menurun, Menhub Minta Semua Pihak Tetap Waspada, [online] Tersedia di: <<http://www.dephub.go.id/post/read/angka-kecelakaan-lalin-mudik-tahun-ini-menurun,-menhub-minta-semua-pihak-tetap-waspada>> [Diakses 6 Juni 2019].
- Derivations of the LSE for Four Regression Models. <http://facweb.cs.depaul.edu/sjost/csc423/documents/technical-details/lse.pdf>
- Egorov, M. 2016. Multi-Agent Deep Reinforcement Learning. Stanford University.
- Ertugrul, Ö.F., 2016. Forecasting electricity load by a novel recurrent extreme learning machines approach. International Journal of Electrical Power and Energy Systems, 78, hal.429–435.
- Ferguson, M., Ak, R., Lee, Y-T. T., Law, K. H. 2017. Automatic Localization of Casting Defects with Convolutional Neural Networks. DOI: 10.1109/BigData.2017.8258115.
- Gatys, L.A., Ecker, A.S. dan Bethge, M., 2015. A Neural Algorithm of Artistic Style. arXiv:1508.06576v2 [cs.CV] 2 Sep 2015.

- Giro-i-Nieto, X. 2017. Once Perceptron to Rule Them all: Deep Learning for Multimedia. Universitat Politecnica De Catalunya Barcelonatech, Department of Signal Theory and Communications, Image Processing Group.
- Goodfellow I., Bengio Y., Courville A. 2016. Deep Learning. MIT Press. Link <http://www.deeplearningbook.org>
- Hajek, A. and Hartmann, S. 2010. Bayesian Epistemology. in: Dancy, J., Sosa, E., Steup, M. (Eds.) (2001) A Companion to Epistemology, Wiley. ISBN 1-4051-3900- 5 Pre-print.
- Hapsari, K. D., Cholissodin, I., Santoso, E. 2016. Optimasi Radial Basis Function Neural Network Menggunakan Hybrid Particle Swarm Optimization Dan Genetic Algorithm Untuk Peramalan Curah Hujan. DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya, vol. 7, no. 15.
- Hayes, J., Melis, L., Danezis, G., Cristofaro, E. D. 2017. LOGAN: Evaluating Information Leakage of Generative Models Using Generative Adversarial Networks. arXiv:1705.07663v3 [cs.CR] 2 Dec 2017.
- Jauhari, D., Cholissodin I., Dewi, C. 2017. Prediksi Nilai Tukar Rupiah Indonesia Terhadap Dolar Amerika Serikat Menggunakan Metode Recurrent Extreme Learning Machine Neural Network. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK) FILKOM UB, Vol. 1, No. 11, November 2017, hlm. 1188-1197.
- Jayasiri, V. 2017. Vanilla LSTM with numpy. Link [https://blog.varunajayasiri.com/numpy\\_lstm.html](https://blog.varunajayasiri.com/numpy_lstm.html)
- Jiang, S. Multi-Agent-Reinforcement-Learning-Environment. [online] Tersedia di: <<https://github.com/Bigpig4396/Multi-Agent-Reinforcement-Learning-Environment/> [Diakses pada 6 Juni 2019]
- Jozefowicz, R., Zaremba, W., Sutskever, I. 2015. An Empirical Exploration of Recurrent Network Architectures. Proceedings of the 32<sup>nd</sup> International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

- Jupiyandi, S., Saniputra, F., Pratama, Y., Dharmawan, M., & Cholissodin, I. 2019. Pengembangan Deteksi Citra Mobil untuk Mengetahui Jumlah Tempat Parkir Menggunakan CUDA dan Modified YOLO. *Jurnal Teknologi Informasi dan Ilmu Komputer*, 6(4), 413-419. doi:<http://dx.doi.org/10.25126/jtiik.2019641275>
- justHelloWorld. 2017. zero denominator in ROC and Precision-Recall?. [Online] Tersedia di: <<https://stackoverflow.com/questions/44008563/zero-denominator-in-roc-and-precision-recall>>
- Kansal, S. 2020. Develop a blockchain application from scratch in Python. <https://developer.ibm.com/technologies/blockchain/tutorials/develop-a-blockchain-application-from-scratch-in-python/> diakses 17 Juli 2020.
- Karkare, P. 2019. Understanding Recurrent Neural Networks in 6 Minutes. [Online] Tersedia di: <<https://medium.com/x8-the-ai-community/understanding-recurrent-neural-networks-in-6-minutes-967ab51b94fe>> [Dikses 27 Desember 2020].
- Karpathy, A. Deep Reinforcement Learning: Pong from Pixels. [online] Tersedia di: <<http://karpathy.github.io/2016/05/31/rl/>> [Diakses pada 7 Juni 2019]
- Kartika, P., Candra, D., dan Cholissodin, I. 2016. Identification Of Patchouli Leaves Quality Using Self Organizing Maps (SOM) Artificial Neural Network. *Journal of Environmental Engineering & Sustainable Technology*, Vol. 03 No. 01, July 2016, pp 42-50.
- Khatchadourian, R. 2011. What are correct values for precision and recall when the denominators equal 0?. [Online] Tersedia di: <<https://stats.stackexchange.com/questions/8025/what-are-correct-values-for-precision-and-recall-when-the-denominators-equal-0>>
- Kojouharov S. 2017. Cheat Sheets for AI, Neural Networks, Machine Learning, Deep Learning & Big Data. Diakses pada tahun 2018 dari <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

- Lawtomated. 2019. A.I. Technical: Machine vs Deep Learning. [Online] Tersedia di: <<https://lawtomated.com/a-i-technical-machine-vs-deep-learning/>> [Diakses 24 Desember 2020].
- Lendave, V. 2021. LSTM Vs GRU in Recurrent Neural Network: A Comparative Study. [Online] Tersedia di: <<https://analyticsindiamag.com/lstm-vs-gru-in-recurrent-neural-network-a-comparative-study/>>
- Lin, X., Beling, B. A., dan Cogill, R. 2014. Comparison of Multi-agent and Single-agent Inverse Learning on a Simulated Soccer Example. arXiv:1403.6822v1 [cs.LG] 26 March 2014.
- Madzarov, Gjorgji, Gjorgjevikj, Dejan. 2010. Thesis Project: “Evaluation of Distance Measures for Multi-class Classification in Binary SVM Decision Tree”. Macedonia: Department of Computer Science and Engineering.
- Manriquez, C. 2016. Generalized Confusion Matrix for Multiple Classes. DOI: 10.13140/RG.2.2.31150.51523
- McGrayne, Sharon Bertsch. 2011. The Theory That Would Not Die: How Bayes' Rule Cracked The Enigma Code, Hunted Down Russian Submarines, & Emerged Triumphant from Two Centuries of Controversy. New Haven: Yale University Press. 13-ISBN 9780300169690/10-ISBN 0300169698; OCLC 670481486.
- Mehta, P., 2018. Multimodal Deep Learning - Fusion of multiple modalities using Deep Learning. [Online] Tersedia di: <<https://towardsdatascience.com/multimodal-deep-learning-ce7d1d994f4>> [Diakses 30 Maret 2020].
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M. 2013. Playing Atari with Deep Reinforcement Learning. DeepMind Technologies, [Online] Tersedia di: <<https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf>>
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., dan Petersen, S. 2015. Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

- Mordvintsev, et al. 2018. Differentiable Image Parameterizations. Distill.
- Neena, A., dan M. G. 2017. A Review on Deep Convolutional Neural Networks. International Conference on Communication and Signal Processing, April 6-8, 2017, India.
- Newberg, L. 2009. Error statistics of hidden Markov model and hidden Boltzmann model results. BMC Bioinformatics 10: 212. doi:10.1186/1471-2105-10-212.
- Ng, Andrew. 2009. Cs229 The Simplified SMO Algorithm. [Online] Tersedia di: <<http://cs229.stanford.edu/materials smo.pdf>> [Diakses 5 Juni 2014].
- Nguyen, T. T., Nguyen, N. D., dan Nahavandi, S. 2019. Deep Reinforcement Learning for Multi-Agent Systems: A Review of Challenges, Solutions and Applications. arXiv:1812.11794v2 [cs.LG] 6 Feb 2019.
- Phillips, L. D., Edwards, W. 2008. Chapter 6: Conservatism in a simple probability inference task (Journal of Experimental Psychology (1966) 72: 346- 354). In Jie W. Weiss and David J. Weiss. A Science of Decision Making: The Legacy of Ward Edwards. Oxford University Press. pp. 536. ISBN 978-0-19-532298-9.
- Platt, J. C. 1998. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machine. Microsoft Research.
- Prasetyo, E. 2013. Data Mining: Konsep Dan Aplikasi Menggunakan Matlab. Penerbit Andi.
- Punma, C. 2017. Autonomous Vehicle Fleet Coordination With Deep Reinforcement Learning. PwC AI Accelerator.
- Putri, I.R., Cholissodin, I., Setiawan, B. D. 2015. Optimasi Metode Adaptive Fuzzy K-Nearest Neighbor Dengan Particle Swarm Optimization Untuk Klasifikasi Status Sosial Ekonomi Keluarga. DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya, vol. 5, no. 3.
- Qu, J. 2018. I Created A Virtual Self Driving Car With Deep Q-Networks, [online] Tersedia di: <<https://towardsdatascience.com/i-created-a-virtual-self-driving-car-with-deep-q-networks-7f87c0aad7c8>> [Diakses 1 Juni 2019].

- Quora, 2017. What is the difference between a deep belief network (DBN) and recurrent neural network (RNN)?. [Online] Tersedia di: <<https://www.quora.com/What-is-the-difference-between-a-deep-belief-network-DBN-and-recurrent-neural-network-RNN>> [Diakses 27 Desember 2020].
- Raicea, R. 2017. Simulation of a self-driving car game using a Deep Q Learning AI, [online] Tersedia di: <<https://github.com/Radu-Raicea/Self-Driving-Car-Simulation>> [Diakses 2 Juni 2019].
- Rajam, Felci, & Valli, S. 2011. Thesis projects: “Content-Based Image Retrieval Using a Quick SVM-Binary Decision Tree – QSVMBDT”. India: Anna University Chennai dan Joseph’s College of Engineering Jeppiar Nagar.
- Rajkumar, N., Jaganathan., P. 2013. A New RBF Kernel Based Learning Method Applied to Multiclass Dermatology Diseases Classification. Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013).
- Rees, J. 2020. How Blockchain is Revolutionizing the Supply Chain Industry. <https://readwrite.com/2020/02/10/how-blockchain-is-revolutionizing-the-supply-chain-industry/> diakses 23 Juli 2020.
- Rehman, M., Khan, G. M. dan Mahmud, S. A. 2014. Foreign Currency Exchange Rates Prediction Using CGP and Recurrent Neural Network. IERI Procedia, 10, hal.239–244.
- Russell, S., Norvig, P. 2003. Artificial Intelligence: A Modern Approach. International Edition, Edisi 2. Pearson Prentice-Hall Education International. New Jersey.
- Sembiring, Krisantus. 2007. Penerapan Teknik Support Vector Machine untuk Pendekripsi Intrusi pada Jaringan. Sekolah Teknik Elektro dan Informatika, ITB.
- Setyawan, G. E., dan Cholissodin, I. 2019. Development Of Deep Reinforcement Learning Multi-Agent Framework Design Using Self-Organizing Map. 2019 International Conference on Sustainable Information Engineering and

- Technology (SIET), Lombok, Indonesia, 2019, pp. 246-250, doi: 10.1109/SIET48054.2019.8986121.
- Shoham, Y., Leyton-Brown, K. 2010. Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Stanford University dan University of British Columbia.
- Shridhar, K. 2018. Kaggle #1 Winning Approach for Image Classification Challenge. Kaggle competition: Plant Seedlings Classification, 21 Juni 2018, [Online] Tersedia di: <<https://medium.com/neuralspace/kaggle-1-winning-approach-for-image-classification-challenge-9c1188157a86>>
- Simonini, T. 2018. Deep reinforcement learning Course. Implementations from the free course Deep Reinforcement Learning with Tensorflow and PyTorch 2017-2020, [Online] Tersedia di: <[https://github.com/simoninithomas/Deep\\_reinforcement\\_learning\\_Course](https://github.com/simoninithomas/Deep_reinforcement_learning_Course)>
- Simonini, T. 2018. Diving deeper into Reinforcement Learning with Q-Learning. 10 April 2018/#Machine Learning, [Online] Tersedia di: <<https://www.freecodecamp.org/news/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe/>>
- Smith, A. J. 2001. Applications of the Self-Organising Map to Reinforcement Learning. Institute for Adaptive and Neural Computation. The Division of Informatics, University of Edinburgh, 5 Forrest Hill, Edinburgh, UK, EH1 2QL.
- Smith, A. J. 2002. Applications of the Self-Organising Map to Reinforcement Learning. Institute for Adaptive and Neural Computation The Division of Informatics University of Edinburgh 5 Forrest Hill, Edinburgh, UK EH1 2QL.
- Steve Gunn, 1997. Support Vector Machines for Classification and Regression. Image Speech and Intelligent Systems Group.
- Sutton, R. S., dan Barto, A. G. 2017. Reinforcement Learning: An Introduction. MIT Press.

- Swanson, D.A., Tayman, J., Bryan, T.M. 2010. MAPE-R Rescaled Measure of Accuract For Cross-Sectional Forecasts. Department of Sociology, University of California Riverside, USA.
- Taleb, N., 2019. Prospective Applications of Blockchain and Bitcoin Cryptocurrency Technology. TEM Journal. Volume 8, Issue 1, Pages 48-55, ISSN 2217-8309, DOI: 10.18421/TEM81-06
- Vaddadi, S. 2020. Image Captioning-Using Multimodal Neural Networks. [Online] Tersedia di: <<https://github.com/sam-pathv95/image-captioning>> [Diakses 8 Juni 2020].
- Venkatachalam, M. 2019. Recurrent Neural Networks. [Online] Tersedia di: <<https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce>> [Diakses 27 Desember 2020].
- Vijayakumar, S., Wu, S. 1999. Sequential Support Vector Classifiers and Regression. RIKEN Brain Science Institute 1999, pp.610 – 619.
- Wang, H., Zhang, Y., Yu, X. 2020. An Overview of Image Caption Generation Methods. Computational Intelligence and Neuroscience, vol. 2020, Article ID 3062706, 13 pages, 2020. <https://doi.org/10.1155/2020/3062706>
- Wu, Jianxin; Vedaldi, Andrea; Maji, Subhransu; Perronnin, Florent. 2012. Additive Kernel and Explicit Embeddings for Large Scale Computer Vision Problems. Florence.
- Wu, W., Lin, S., Moon, W. 2012. Combining Support Vector Machine with Genetic Algorithm to Classify Ultrasound Breast Tumor Images. Chang Gung University, Taiwan and Seoul National University Hospital, South Korea.
- Zantalis F., Koulouras G., Karabetsos S., Kandris D. 2019. A Review of Machine Learning and IoT in Smart Transportation. Future Internet 2019, 11, 94; doi:10.3390/fi11040094

### Biografi Penulis



**Imam Cholissodin**, lahir di Lamongan pada tanggal 19 Juli 1985, telah menyelesaikan pendidikan S2 di Teknik Informatika FTIF ITS Surabaya pada Tahun 2011. Sejak Tahun 2012 telah aktif sebagai dosen pengajar di jurusan Teknik Informatika Program Teknologi dan Ilmu Komputer (PTIIK), dan Alhamdulillah mulai tahun 2016 telah menjadi Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya (UB) Malang pada beberapa mata kuliah, seperti Information Retrieval, Pengolahan Citra Digital, Probabilitas dan Statistik, Grafika Komputer, Decision Support System, Kecerdasan Buatan, Data Mining, Analisis Big Data, Pemrograman GPU, Algoritma Evolusi, Swarm Intelligence, Pengenalan Pola dan Pemrograman Aplikasi Perangkat Bergerak. Di samping mengajar, peneliti juga aktif dalam Riset Group Sistem Cerdas dan Riset Group Media, Game & Mobile Technology (MGM) di dalam Laboratorium Riset. Selain itu peneliti juga melakukan beberapa publikasi pada seminar maupun jurnal nasional dan internasional (IEEE, Scopus, etc). Riset pada tahun 2015-2018 bersama beberapa tim dosen dan mahasiswa adalah berfokus pada bidang Information Retrieval, teknik optimasi untuk analisis dokumen lembaga pemerintahan secara Real-time, dengan tema “Pengembangan Sistem Audit Dokumen Lembaga Pemerintahan Menggunakan Stream Deep Learning Untuk Mendukung Smart Governance” yang merupakan kombinasi beberapa multi-disiplin keilmuan antara Decision Support System (DSS), Teknik Optimasi, Big Data, Machine Learning, Ilmu Administrasi Pemerintahan serta Information Retrieval (IR). Dan pada tahun 2019 melanjutkan riset bidang Big Data yang dikolaborasikan dengan bidang Ekonomi bersama tim dosen dan mahasiswa Fakultas Ekonomi dan Bisnis (FEB) UB bersama Badan Perencanaan Pembangunan Daerah (BAPPEDA) provinsi JATIM dengan tema “Penyusunan Blue Print Inisiasi Pemanfaatan Open Data Dalam Perencanaan Pembangunan Daerah” untuk mendukung Smart Governance terpadu (integrated dengan semua existing sistem dari berbagai platform) based Artificial Intelligence dalam beberapa tahun ke depan, serta pengembangan Core Engine Deep Learning and Big Data as General Library/ Toolbox & package installer, dan support untuk bahasa pemrograman dan platform OS apapun di bawah payung penelitian Lab. Komputasi Cerdas FILKOM UB pada komputasi Backend dan Frontend pada perangkat desktop, web maupun mobile untuk bidang kesehatan, pemerintahan dan lainnya berbasis local maupun serverless dengan teknologi cloud computing dalam rangka membangun dan menghasilkan produk “Smart App” di era Revolusi Industri 4.0 dan Society 5.0.

Motto: “We Are A Code, We Are The Best Code Of God”.

---



**Sutrisno**, lahir di Tulungagung pada tanggal 25 Maret 1957, telah menyelesaikan Pendidikan sarjana (S1) di Teknik Elektro Institut Teknologi Bandung (ITB) lulus tahun 1982 dan Pendidikan strata dua (S2) di Program Studi Magister Teknik Elektro Universitas Brawijaya (UB) lulus tahun 2008. Sejak tahun 1982 telah menjadi dosen di jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya dan menjadi Ketua Program Studi Teknik Informatika tahun 2009 s.d. tahun 2011 dan pada tahun 2011-2016 menjabat sebagai Ketua Program (Dekan) di Program Teknologi Informasi dan Ilmu Komputer (PTIIK) Universitas Brawijaya sekarang menjadi Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya. Matakuliah yang pernah diajarkan meliputi: Sistem Distribusi, Rangkaian Elektrik/Listrik, Pemrograman Dasar, Pemrograman Lanjut, Desain dan Analisis Algoritma, Analisa Struktur Data dan Rancangan Perangkat Lunak.



**Arief Andy Soebroto**, lahir di Surabaya 25 April 1972. Lulus SMAN 1 Surabaya Tahun 1990, menyelesaikan S1 di Teknik Elektro (Elektronika) Universitas Brawijaya (UB) Tahun 1996 dan S2 Tahun 2007 di Institut Teknologi 10 Nopember Surabaya (ITS) Tahun 2005. Tahun 1996-1999 bekerja di Penyelenggara Jasa Layanan Internet MEGA Net. Tahun 1999-2011 sebagai dosen di JurusanTeknik Elektro (Informatika) Universitas Brawijaya,. Sejak Tahun 2011 mengajar di Program Studi Informatika Program Teknologi Informasi & Ilmu

Komputer Universitas Brawijaya (PTIIK-UB). Beberapa hibah penelitian dari Direktorat Jenderal Pendidikan Tinggi Kementerian Pendidikan dan Kebudayaan Republik Indonesia sejak tahun 2006 sampai dengan 2012 telah diterima, PHB, Strategi Nasional dan Riset Andalan Perguruan Tinggi (RAPID). Sebagai Ketua Kelompok Dosen Keahlian Bidang Informatika teknik Elektro Universitas Brawijaya Tahun 2007-2009.Tahun 2009-2011 menjabat sebagai Pembantu Dekan Bidang Administrasi dan Keuangan Fakultas Teknik Universitas Brawijaya. Bidang ilmu yang ditekuni adalah Sistem Cerdas, yaitu Decision Support and Expert System Research Group (DESSY RG) sebagai koordinator kelompok pada Laboratorium Komputasi Cerdas PTIIK-UB.



**Uswatun Hasanah**, lahir di Probolinggo pada tanggal 01 Juni tahun 1995, telah berhasil menyelesaikan studi S1 Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya Malang dengan tugas akhir yang berjudul “Penentuan Seleksi Atlet Taekwondo Menggunakan Metode Algoritme Support Vector Machine”.



**Yessica Inggir Febiola**, lahir di Tulungagung pada tanggal 27 Februari tahun 1997, telah berhasil menyelesaikan studi S1 Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya Malang dengan tugas akhir yang berjudul “Peramalan Hasil Panen Kelapa Sawit Menggunakan Metode *Multifactors High Order Fuzzy Time Series* yang Dioptimasi dengan *K-Means Clustering*”.