Pengaplikasian approximate dynamic programming dalam permasalahan perekomendasiaan dinamis

Calvin sadewa and 13512066
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512066@std.stei.itb.ac.id

***Abstract*—rekomendasi adalah proses menseleksi kandidat dari himpunan kandidat berdasarkan parameter-parameter yang telah didefinisikan. rekomendasi dinamis adalah proses rekomendasi yang parameter - parameternya terdefinisi, hanya tidak bisa dimasukan ke program karena suatu hal , sehingga sistem harus mengubah perilakunya sendiri untuk bisa memberikan rekomendasi yang bagus. paper ini akan membahas bagaimana menerapkan rekomendasi dinamis pada komputer dengan menggunakan approximate dynamic programming**

***Index Terms*—approximate dynamic programming, rekomendasi dinamis, kategori,**

I. PENDAHULUAN

rekomendasi adalah proses yang sering kita lakukan atau rasakan dalam kehidupan sehari - hari, seperti rekomendasi baju, laptop, pekerjaan , dll.

rekomendasi membuat kehidupan kita menjadi lebih mudah karena kita tidak perlu melihat semua produk yang ada, tetapi kita bisa mendapatkan hasil

yang optimal (atau sub-optimal). dalam rekomendasi , biasanya ada parameter - parameter yang terdefinisi (misalnya : suka kemeja, ingin harga murah)

dalam kehidupan sehari - hari, rekomendasi yang tepat akan membantu anda dalam membuat keputusan karena anda tidak perlu menghabiskan waktu untuk mencari barang yang tepat dengan keinginan anda.

rekomendasi dinamis dalam hal ini adalah proses pencarian rekomendasi yang bagus meskipun parameter - parameternya tidak bisa didefinisikan, entah karena satu atau lain hal. dalam dunia nyata, rekomendasi dinamis ini ibarat mencari baju, pertama kali anda membeli baju, mungkin anda tidak tahu baju mana yang cocok dengan anda, namun seiring berjalannya waktu, anda biasanya mengetahui baju mana yang cocok untuk anda, biasanya dalam kategori - kategori tertentu (kemeja, batik, dll).

tujuan dari paper ini adalah bagaimana menerapkan rekomendasi dinamis dengan menggunakan teknik yang dinamakan approximate dynamic programming, atau lebih dikenal sebagai reinforcement learning.

II. DASAR TEORI

Dynamic Programming

Dynamic programming adalah salah satu strategi pemecahan permasalahan, strategi ini memecah permasalahan awal menjadi subpersoalan, dan ada sub persoalan yang saling bertumpang-tindih (overlap)

S -> {k1,k2,k3,...}

ki intersect kj != 0

disini , S adalah permasalahan awal , yang dipecah menjadi beberapa k subpersoalan sedemikian rupa sehinggan ada subpersoalan (ki) yang memiliki subpersoalan yang sama dengan subpersoalan (kj), padahal ki dan kj sama-sama subpersoalan dari S, kalau properti ini tidak dipenuhi, maka Dynamic Programmming tidak ada bedanya dengan metode pemecahan yang lain.

nantinya subpermasalah yang overlap akan di simpan untuk bisa dipakai ketika dibutuhkan, proses penyimpanan ini disebut memoisasi.

Salah satu aplikasi dynamic programming adalah menghitung fibonacci

```
function fib(n)
    if n = 0
       return 0
    else
       var previousFib := 0, currentFib := 1
       repeat n − 1 times // loop is skipped if n = 1
          var newFib := previousFib + currentFib
          previousFib := currentFib
          currentFib  := newFib
    return currentFib
```

fungsi fibonacci diatas menyimpan nilai dari fibonacci sebelumnya, sehiingga program tidak perlu melakukan komputasi berulang – ulang ketika mengevaluasi fibonacci

Approximate dynamic programming / reinforcement learning

reinforcement learning adalah cabang dari machine learning yang terinspirasi dari "perilaku" di dunia nyata yang bisa berubah berdasarkan kondisi.

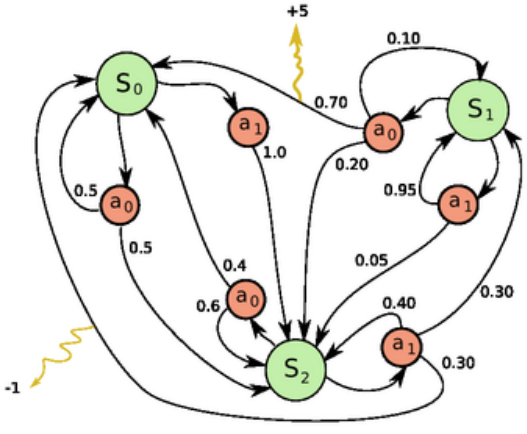
Secara formal , reinforcement learning[1] adalah sekumpulan alat untuk menyelesaikan permasalahan kontrol dalam Markov decision process (MDP),

dalam reinforcement learning , fungsi P atau R tidak diketahui.

alasan nama lain dari teknik ini adalah approximate dynamic programming karena P atau R tidak diketahui, maka program harus menerka P atau R dari pengalaman lampau, mirip dengan dynamic programming (dengan problem = sekarang dan subproblem = pengalaman lampau)

reinforcement learning telah banyak menghasilkan hal[2], salah satunya adalah pengendalian helikopter secara terbalik (rotor di bawah) dan program pemain catur.

reinforcement learning sering di bahas di lingkaran ilmuwan Evolution programming, AI.



Markov Decision Process

Markov decision processes (MDPs), adalah sebuah mathemahical framework yang di develop oleh andrey markov untuk memodelkan sistem decision-making dimana hasil dari decision adalah sebagian random dan sebagian di tangan pembuat decision. MDP sngat berguna dalam mempelajari banyak persoalan optimasi yang berhubungan dengan dynamic programming dan reinforcement learning . MDPs diketahui mulai dari tahu 1950-an (cf. Bellman 1957). MDP digunakan di banyak area seperti robotika, kontrol otomatis, ekonomi, and manufactoring.

Markov decision process dapat didefinisikan sebagi tuple (S,A,P.(.,.),R.(.,.))

dimana :
S adalah kumpulan state yang ada
A adalah kumpulan aksi yang bisa di lakukan
Pa(s,s') adalah probabilitas suatu state s ke state s' melalui aksi a.
Ra(s,s') adalah reward suatu state s ke state s' melalui aksi a.

Q-learning

Q-learning adalah sebuah tehnik reinforcement learning. Q-learning bisa memberikan solusi optimal dalam Markov decision process. Q-learning berkerja dengan mempelajari nilai action-state maka solusi optimal hanyalah tinggal memilih nilai action-state maksimal untuk setiap state Salah satu kelebihan Q-learning adalah ia bisa mengkomparasikan aksi tanpa perlu mengetahui detil model ia sekarang.

Q-learning menggunakan prinsip temporal difference untuk menentukan fungsi kemungkinan $Q^*(s,a)$. *dalam Q-learning, program menyimpan tabel* $Q(s,a)$, Dimana s adalah sebuah state dan a adalah sebuah aksi. $Q[s,a]$ merepresentasikan Q(s,a) dengan $Q^*(s,a)$, *yaitu tabel yang paling baru*.

Untuk setiap pengalaman, Q-learning akan mengupdate Q[s,a] sesuai rumus :

$$Q[s,a] \leftarrow Q[s,a] + \alpha(r + \gamma max_{a'} Q[s',a'] - Q[s,a])$$

or, equivalently,

$Q[s,a] \leftarrow (1-\alpha)\, Q[s,a] + \alpha(r + \gamma \max_{a'} Q[s',a'])$.

legenda : s : state dalam himpunan State
a : aksi yang dilakukan sistem
r : reward yang di dapat
y : discount factor (yaitu seberapa penting reward kumulatif masa depan)

dalam rumus ini, tahap program dinamis ada di $\max_{a'} Q[s',a']$ *yang*

controller *Q-learning*(*S*,*A*,*γ*,*α*)
2: **Inputs**
3: *S* is a set of states
4: *A* is a set of actions
5: *γ* the discount
6: *α* is the step size
7: **Local**
8: real array *Q[S,A]*
9: previous state *s*
10: previous action *a*
11: initialize *Q[S,A]* arbitrarily
12: observe current state *s*
13: **repeat**
14: select and carry out an action *a*
15: observe reward *r* and state *s'*
16: $Q[s,a] \leftarrow Q[s,a] + \alpha(r + \gamma \max_{a'} Q[s',a'] - Q[s,a])$
17: $s \leftarrow s'$ **until** termination

III. IMPLEMENTASI

kita akan menggunakan representasi Q-learning dalam implementasi ini

pertama-tama kita definisi kan dahulu Aksi - aksi apa saja yang bisa sistem kita lakukan, yaitu rekomendasi produk, anggap produk dibagi atas kategori, maka A adalah himpunan rekomendasi kategori.

kemudian kita definisikan S, karena sistem bergantung terhadap personalitas pelanggan, maka S adalah himpunan dari seluruh pelanggan, dan s0 adalah pelanggan pertama yang memakai produk kita. tentu saja, S bisa di update secara dinamik.

setelah itu, inisiliasi semua fungsi Q(s,a) dengan 0.

kemudian buat fungsi learning factor, untuk sekarang set semua menjadi 0.1.

kemudian definisikan reward yang didapat jika benar atau salah, untuk sekarang reward jika benar adalah 1 dan jika salah adalah -1 (benar atau salah disini adalah disukai pelanggan atau tidak)

dan terakhir set discount factor, dalam untuk sekarang set menjadi 0, karena dalam sistem kita berasumsi bahwa semua long-term reward adalah short-term reward, didasari dengan asumsi bahwa jika kita berhasil memberika rekomendasi yang bagus, orang tersebut pasti akan datang lagi.

untuk mendapatkan rekomendasi, hanya tinggal mencari a sehingga Q(s,a) maksimum untuk seorang Pelanggan s

Implementasi dalam java :

```
import java.util.*;

public class makalah{

        public static final double learningFactor = 0.1;
        public static final int banyakKategori = 5;
        public static final int kategoriBaju = 0;
        public static final int kategoriCelana = 1;
        public static final int kategoriSepatu = 2;
        public static final int kategoriKaos = 3;
        public static final int kategoriHandphone = 4;
        public static final String[] namaKategori = {"Baju","Celana","Sepatu","Kaos","Handphone"};

        //representasi Q
        public static LinkedList<Double[]> Q = new LinkedList<>();

        //pelanggan yang lagi di servis
        public static int currentPelanggan;

        //main
        public static void main(String[] args){
                boolean end = false;
                Scanner in = new Scanner(System.in);
                while (!end){

        System.out.println("Masukan input :\n"
                                + "0 untuk memberikan data input\n"
                                + "1 untuk mendapatkan rekomendasi\n"
```

```
                    + "2 untuk menambah pelanggan\n"
                    + "3 untuk mengganti pelanggan\n"
                    + "4 untuk melihat banyak pelanggan\n"
                    + "5 untuk keluar dari program");
                int input = in.nextInt();
                if (input == 0){
                    dataInput();
                }
                if (input == 1){
                    rekomendasi();
                }
                if (input == 2){
                    tambahPelanggan();
                }
                if (input == 3){

        System.out.println("masukan indeks pelanggan");
                    currentPelanggan = in.nextInt();
                }
                if (input == 4){

        System.out.println("Banyak pelanggan : "+Q.size());
                }
                if (input == 5){end = true;}
            }
        }

        // Menambahkan pelanggan
        public static void tambahPelanggan (){
            Double[] A = new Double[banyakKategori];

            //inisialisasi Q(s,a) dengan 0
            for (int i = 0; i < banyakKategori; i++){
                A[i] = (double)0;
            }

            Q.add(A);
        }

        // Memberikan data input ke program yang nantinya akan digunakan untuk rekomendasi
        public static void dataInput(){
            int random = (int )(Math.random() * 5);
            System.out.println("Anda diberikan "+namaKategori[random]
        + "\n Apa reaksi anda? masukan 1 jika setuju dan 0 kalau tidak setuju");
            Scanner in = new Scanner(System.in);
            int reward;
            if (in.nextInt() == 1){
                reward = 1;
            }
            else reward = -1;

            //hitung fungsi Q selanjutnya
            Q.get(currentPelanggan)[random] = (1-learningFactor)*Q.get(currentPelanggan)[random] + learningFactor*reward;
        }

        //rekomendasi
        public static void rekomendasi(){

            int rekomendasi = 0;

            //Cari maksimum untuk di rekomendasikan
            for (int i = 1; i < banyakKategori; i++){
                if (Q.get(currentPelanggan)[rekomendasi] < Q.get(currentPelanggan)[i]) rekomendasi = i;
            }
            System.out.println("Anda diberikan "+namaKategori[rekomendasi]
        + "\n Apa reaksi anda? masukan 1 jika setuju dan 0 kalau tidak setuju");
            Scanner in = new Scanner(System.in);
            int reward;
            if (in.nextInt() == 1){
                reward = 1;
            }
            else reward = -1;

            //hitung fungsi Q selanjutnya
            Q.get(currentPelanggan)[rekomendasi] = (1-learningFactor)*Q.get(currentPelanggan)[rekomendasi] + learningFactor*reward;
        }
    }
```

Fungsi Q(s,a) di update setiap kali user memberikan data input atau meminta rekomendasi.

Analisis

Cara baca testing :

Tambah Px adalah menambah pelanggan dengan indeks x

Px : nama_kategori setuju/tidak
Memberikan informasi tentang pelanggan ke x kepada sistem mengenai prefensi

Px rek : nama_kategori setuju/tidak
Sistem memberikan prefensi ke pelanggan dan pelanggan mengevaluasi nya

Testing pertama:

Tambah P0
P0 : Handphone setuju
P0 : Kaos tidak setuju
P0 rek : Handphone setuju
Tambah P1
P0 : Handphone tidak
P0 : Celana setuju
P0 : Handphone tidak
P0 : Kaos setuju
P0 : Handphone tidak
P0 : Kaos setuju
P0 rek : Kaos setuju
P1 : kaos tidak
P1 : Handphone setuju
P1 rek : Handphone setuju
P0 rek : Kaos setuju
Tambah P2
P2 : Celana setuju
P2 : kaos Setuju
P2 rek : Celana setuju

Testing ke 2 :

Tambah P0
Tambah P1
Tambah P2
Tambah P3
P0 : Celana setuju
P0 rek : Celana setuju
P0: Celana setuju
P0 : Kaos setuju
P0 : Handphone tidak setuju
P0 : Kaos setuju
P1 : Sepatu setuju
P1 : Celana setuju
P1 : Sepatu setuju
P1 : Sepatu setuju
P1 : Kaos tidak setuju
P1 rek : Celana setuju
P0 rek : Celana setuju

Dari hasil testing, algoritma memberikan hasil yang cukup memuaskan, algoritma dapat bereaksi terhadap konsumen yang berbeda - beda, selain itu, program mampu mengingat prefensi seseorang untuk menggunakannya dalam memprediksi rekomendasi seseorang. Selain itu, program juga bisa mendeteksi perubahan dari prefensi pelanggan.

Dalam implementasi kali iini, program disimplifikasi sedemikian rupa sehingga banyak aspek dari reinforcement learning tidak di tangkap didalam program, salah satu contohnya adalah tidak digunakannya discount factor (ingat asumsi bahwa local optimum pastilah membawa global optimum, atau dengan kata lain pelanngan adalah makhluk independen) , mungkin saja pelanggan saling berkomunikasi satu sama lain dan mempengaruhi rekomendasi produk.

Salah satu improvisasi yang bisa dilakukan adalah dengan mengubah sistim kategori produk menjadi sistem tag, karena dalam dunia nyata pelanggan mungkin memilih produk berdasarkan tag dibandingkan dengan kategori, kategori pada program bisa diganti dengan kombinatorial tag – tag.

Selain itu, pencentralan database bisa membuat program ini menjadi online

IV. KESIMPULAN

Permodelan dengan approximate dynamic programming bisa digunakan untuk memodelkan sistem rekomendasi dinamis yang cukup sederhana.

Selain itu, permodelan ini juga dapat di aplikasikan ke sistem – sistem yang cukup kompleks tanpa mengalami perubahan yang signifikan

REFRENSI

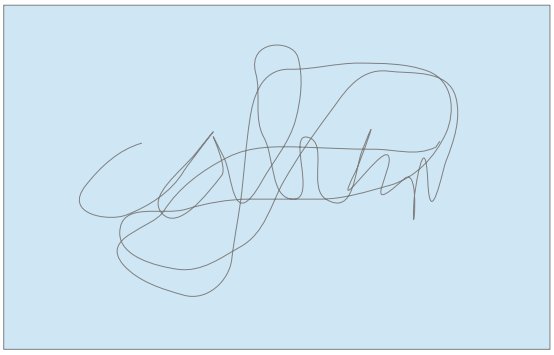
[1] Richard S. Suttor and Andre G. Barto. Reinforcement Learning : An introduction. MIT Press, 1998
[2] http://umichrl.pbworks.com/w/page/7597597/Successes%20of%20Reinforcement%20Learning

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Mei 2014

ttd



Calvin sadewa / 13512066