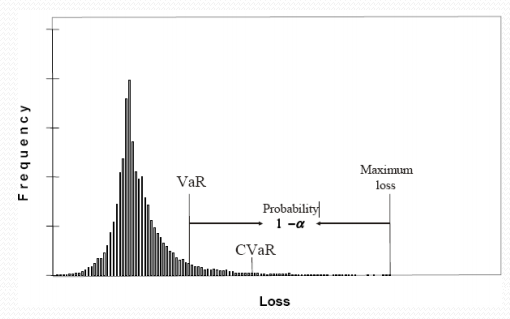
**STATISTIK**

**Expected Tail Loss**

Expected Tail Loss (ETL) merupakan ukuran resiko yang sifatnya diturunkan untuk distribusi kerugian. ETL atau juga dikenal dengan Conditional VaR (CVaR) secara umum didefenisikan sebagai ekspektasi ukuran resiko yang nilainya di atas Value at Risk (VaR), yaitu:

ETL = E (L | L > VaR)

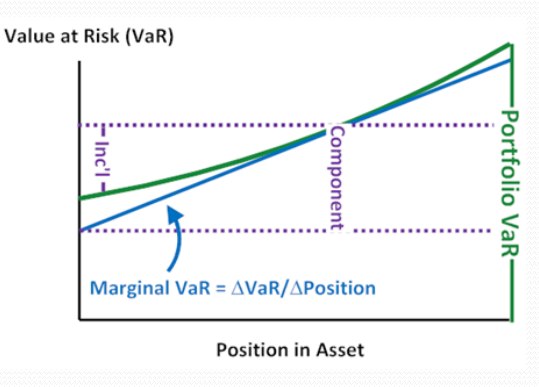


Pada grafik terlihat bahwa CVaR lebih besar dari VaR, pada distribusi kerugian, nilai CvaR terletak di sebelah kanan nilai VaR.

Pendekatan untuk menghitung nilai ETL dapat dilakukan dengan menghitung rata-rata nilai VaR untuk sejumlah nilai interval konfidensi 1 – αk , dengan

Untuk suatu bilangan bulat n yang cukup besar yang biasanya disarankan n>50

Value at Risk (VaR) adalah kerugian yang dapat ditoleransi dengan tingkat kepercayaan (keamanan) tertentu. VaR diartikan sebagai estimasi kerugian maksimum yang akan dialami pada periode waktu dan tingkat kepercayaan tertentu. Oleh karena itu, terdapat kemungkinan kerugian yang didapat oleh investor selama periode kepemilikan akan lebih rendah dibanding limit yang dibentuk dengan VaR. Namun, bisa saja terdapat kemungkinan kerugian lebih buruk, karena keterbatasan dari VaR adalah tidak dapat menyatakan seberapa besar kerugian yang benar-benar terjadi dan secara definitif tidak menegaskan kemungkinan kerugian paling buruk. VaR hanya menyatakan kerugian yang mungkin didapat oleh investor, tetapi investor dapat menggunakan VaR sebagai salah satu tolok ukur menetapkan seberapa besar target risiko



**ANOVA**

Anova merupakan singkatan dari Analysis of variance. Anova merupakan analisis statistik yang menguji perbedaan rerata antar kelompok data.

Anova dalam perhitungan regresi linier membandingkan nilai mean square dan hasilnya adalah menilai apakah model prediksi linear tidak berbeda nyata dengan nilai koefisien estimasi dan standar error.

**Contoh perhitungan sesuai dengan koding Anova satu arah**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rope A** | **Rope B** | **Rope C** | **Rope D** | **Rope E** |
| **414** | **377** | **326** | **273** | **449** |
| **434** | **493** | **337** | **299** | **472** |
| **500** | **521** | **348** | **323** | **485** |
| **412** | **373** | **225** | **248** | **371** |

Langkah pertama dalam Anova satu arah sesuai dengan data di atas adalah :

1. Membuat grafik sesuai data dari rope A ke E dimana mempunyai perbedaan nilai dan disesuaikan dalam grafik vertikal.
2. Saat menjalankan analysis varians disini adalah mencoba untuk mengkonfigurasikan bahwa data point yang ada tiap rope dari populasi normal dimana nilai rata-rata mu (A), mu (B). Untuk rope C, D, dan E nilai rata-rata sama ataupun tidak sama menghasilkan hipotesis nilai yang berbeda dalam grafik.
3. Pertama dibuat sebuah matriks dimana setiap rope berisi data dari kelima ropes. Lalu dibuat label tiap rope A sampai ke E. Fungsi yang dipakai untuk proses satu arah dalam matlab adalah anoval1. Lalu dipanggil group dari ropes A sampai E dalam variabel kedua ataupun dapat digunakan fungsi anoval1 saja.
4. Bila dijalankan pada script dimana data nilai P sebagai variabel p dan dimasukkan huruf p pada command line akan menghasilkan nilai p dibawah 0.05.
5. Fungsi anova1 akan menampilkan analysis varians dalam tabel. Untuk tiap 5 ropes didapatkan nilai merah yaitu merepresentasikan nilai median setiap 4 ukuran. Pada tiap ropes terdapat ujung atas dan ujung bawah merepresentasikan nilai max dan minimum. Pada ropes ke 3 terdapat higher flat ini merepresentasikan 0.75% dari data dan lower flat merepresentasikan 0.25% dari data.
6. Untuk menghasilkan command limit y, diset lower limit dari 0 dan upper limit sebesar 10 % lebih besar dari nilai max dari data set.

**CLUSTERING**

Algoritma Clustering adalah salah satu algoritma yang digunakan untuk klasifikasi atau pengelompokan data. Pada algoritma K-Means Clustering, nilai tengah dihitung dengan rata-rata (mean) dan perhitungan jarak dihitung dari data pada masing-masing mean, sedangkan pada algoritma ini, data akan digunakan sebagai nilai tengah / disebut dengan medoid, dan perhitungan jarak dihitung dari jarak antar masing-masing data.

Proses partisi data didasarkan pada jarak terdekat antara data dengan centroid masing-masing kluster. Pengelompokkan dilakukan berdasarkan centroid atau titik pusat massa nya. Titik pusat massa ini berupa rerata (mean) dari sekumpulan data yang telah di kelompokkan kebeberapa kluster. Sehingga nilai pusat titik massa ini akan berubah-ubah hingga mencapai titik dimana pusat titik massa tidak lagi mengalami perubahan

Langkah-langkah dari clustering :

1.Tentukan jumlah kluster yang diinginkan (k)

2. Tentukan nilai centroids awal.

3. Hitung jarak tiap data terhadap masing-masing centroid

4. Kelompokkan data-data tersebut ke kluster berdasarkan jarak paling dekat (minimum) terhadap sebuah kluster, dalam hal ini saya menggunakan Euclidean Distance.

5. Hitung ulang nilai Centroids dengan menghitung nilai rerata (mean) data dari masing-masing kluster.

6. Lakukan langkah 3-5 hingga nilai centroids tidak lagi mengalami perubahan

**STATISTICAL INFERENCE**

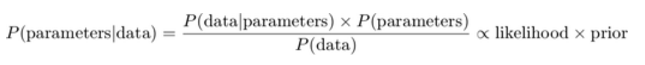
Inferensi statistik adalah untuk mengambil kesimpulan dari data sampel yang dimana diketahui populasi yang tidak ada datanya. Terdapat dua pendekatan statistik inferens yaitu pendekatan frequentist atau terkadang disebut sebagai pendekatan klasik. Dalam pendekatan ini, prosedur dikembangkan hanya melihat performa seluruh kemungkinan sampel acak. Informasi sampel acak yang diperoleh sebelumnya pun diabaikan.

Pendekatan frequentist berlandaskan pada ide-ide dibawah ini:

1. Parameter, yaitu karakteristik dari populasi, adalah konstan namun tidak diketahui.
2. Probabilita selalu diinterpretasikan sebagai frekuensi relatif jangka panjang, tak peduli datanya.
3. Prosedur statistik dinilai dengan seberapa baik prosedur itu dalam jangka panjang dengan mengulang-ulang percobaan sampai tak hingga.

Pendekatan kedua adalah pendekatan berdasarkan teori Bayes.

Inferensi Bayesian adalah proses menganalisis model statistik dengan penggabungan pengetahuan sebelumnya tentang model atau parameter model. Akar inferensi tersebut adalah teorema Bayes:



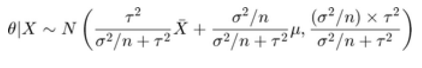
Sebagai contoh, misalkan kita memiliki pengamatan normal



dimana sigma diketahui dan distribusi sebelumnya untuk theta adalah



Dalam rumus ini mu dan tau, kadang-kadang dikenal sebagai hiperparameter, juga dikenal. Jika kita mengamati n sampel X, kita dapat memperoleh distribusi posterior untuk theta sebagai



Bayesian berlandaskan pada ide-ide berikut:

1. Sejak kita tak pernah yakin akan nilai sebenarnya dari parameter, maka parameter dianggap sebagai suatu random variabel.
2. Aturan probabilita digunakan secara langsung untuk melakukan inferens tentang parameter.
3. Pernyataan probabilita tentang parameter diinterpretasikan sebagai “derajat kepercayaan”. Distribusi prior adalah subyektif. Setiap orang bisa memilih priornya sendiri, yang mengandung bobot relatif yang diberikannya pada parameter tersebut, yang mengukur bagaimana sejauh mana bisa diterima/dipercaya setiap parameter tersebut sebelum percobaan.
4. Setelah itu kita menyesuaikan kepercayaan/penerimaan kita pada parameter tersebut setelah memperoleh data dengan menggunakan teorema Bayes, sehingga akan menghasilkan distribusi posterior, yang memberikan bobot relatif tiap parameter setelah data dianalisis. Distribusi posterior diperoleh dari dua sumber, yaitu: **distribusi prior**dan **data pengamatan**.

Inferensi statistik biasanya didasarkan pada estimasi kemungkinan maksimum (MLE). MLE memilih parameter yang memaksimalkan kemungkinan data, dan secara intuitif menarik. Dalam MLE, parameter diasumsikan tidak diketahui tetapi tetap, dan diperkirakan dengan keyakinan. Dalam statistik Bayesian, ketidakpastian tentang parameter yang tidak diketahui dikuantifikasi menggunakan probabilitas sehingga parameter yang tidak diketahui dianggap sebagai variabel acak.

Grafik dalam koding menunjukkan prior, likelihood, dan posterior untuk theta.

**MULTIVARIATE ANALYSIS / MULTIVARIATE REGRESSION**

Metode analisis multivariat adalah suatu metode statistika yang tujuan digunakannya adalah untuk menganalisis data yang terdiri dari banyak [variabel](https://www.statistikian.com/2012/10/variabel-penelitian.html) serta diduga antar [variabel](https://www.statistikian.com/2012/10/variabel-penelitian.html) tersebut saling berhubungan satu sama lain. Analisis multivariat adalah salah satu dari teknik statistik yang diterapkan untuk memahami struktur data dalam dimensi tinggi. Dimana variabel-variabel yang dimaksud tersebut saling terkait satu sama lain.

Analisis regresi majemuk juga dapat mengakomodasi sifat data yang tidak linear (curvilinear). Data tersebut dapat ditransformasi dengan cara di logaritma atau di akar kuadrat. Akan tetapi, metode ini hanya dapat berlaku untuk mengubah sedikit plot data. Selain itu, metode ini tidak memiliki landasan statistik yang memadai mengenai ketepatan model linear yang akan dibentuk. Oleh karena itu, dapat digunakan alternatif lain berupa polinomial. Polinomial dibentuk dengan mengikutsertakan suatu komponen nonlinear dari variabel independen ke dalam model regresi.

Penjelasan Koding :

X adalah matriks Data Anda. 500 titik data dengan 5 dimensi. Cara lain untuk melihat ini adalah 500 sampel dari 5 variabel independen. Y adalah vektor pengamatan 0 dengan 1. Untuk menemukan pasangan polinom yang baik dari kolom X ke Y. Katakanlah Anda memutuskan cocok dengan polinomial derajat 2 untuk semua 5 variabel independen. Untuk itu untuk menyesuaikan basis polinomial standar tanpa mencampurkan lebih lanjut dengan ketentuan.

**Generalized Linear Model**

Banyak metode regresi yang spesifikasi ke dalam Generalized Linear Models (GLM). Menggunakan GLM, model linier dapat digeneralisasikan untuk pola lain menggunakan link function dimana lebih untuk hubungan antara x dan y yang memungkinkan regresi dapat diterapkan di setiap tipe data.

Contoh ini adalah mengkalkulasikan beberapa ukuran dalam bidang statistik. Digunakan variabel X dan Y dalam struct dimana harus terdapat satu variabel per elemen.

Contohnya :

Y.Jarak

X.Kecepatan

X.Waktu

Setiap variabel yang ada harus mempunyai N x 1 vektor kolom dimana setiap barisnya akan diamati. Ini juga memungkinkan untuk menghasilkan bobot (weight) dalam struct dan akan diperhitungakan dalam prosedur estimasinya. Saat menggunakan distribusi binomial, Y seharusnya mempunyai sebuah variabel dengan 2 kolom dimana kolom 1sebagai perespon dan kolom 2 sebagai sample ukuran tiap perespon. Jika tidak mempunyai contoh ukurannya, dapat membuat 2 kolom vektor sekali saja.

Untuk mengestimasi model GLM terdiri dari 3 tahap :

1. Buat objek model GLM mdl =GLM;
2. Tentukan tautan yang dibutuhkan dan distribusinya mdl.Distribution = .. mdl.Link = ..
3. Estimasikan parameter dengan data mdl.Estimate(Y,X);

Setelah mengestimasikan, memungkinkan untuk menyesuaikan nilai baru ke model yang terestimasi yaitu fit = mdl.Fit(Xnew);

Hasilnya akan sering sama dengan yang ada di R.

**Supervised ML**

**Generalized Linear Models**

Generalized Linear Models (GLM) mengestimasi model regresi untuk hasil setelah distribusi eksponensial. Selain distribusi Gaussian (yaitu normal), termasuk distribusi Poisson, binomial, dan gamma. Masing-masing melayani tujuan yang berbeda, dan tergantung pada distribusi dan pilihan fungsi tautan, dapat digunakan baik untuk prediksi atau klasifikasi.

Suite GLM meliputi:

* Regresi gaussian
* Regresi Poisson
* Regresi binomial (klasifikasi)
* Klasifikasi multinomial
* Regresi gamma
* Regresi ordinal

GLM dapat menghasilkan dua kategori model: klasifikasi dan regresi. Regresi logistik adalah GLM yang melakukan klasifikasi biner. Jenis data dari kolom respons menentukan kategori model. Jika respons adalah variabel kategori (juga disebut faktor atau enum), maka model klasifikasi dibuat. Jika tipe data kolom respons adalah numerik (integer), maka model regresi bisa dibuat.

Ketika GLM melakukan regresi (dengan kolom faktor), satu kategori dapat ditinggalkan untuk menghindari multikolinieritas. Jika regularisasi dinonaktifkan (lambda = 0), maka satu kategori akan ditinggalkan. Namun, ketika menggunakan parameter lambda standar, semua kategori disertakan.

Solusi terbaik dari implementasi GLM bergantung pada properti data dan informasi sebelumnya mengenai variabel (jika tersedia). Secara umum, data dianggap jarang jika rasio nol untuk tidak nol dalam matriks input yang lebih dari 10. Solusinya jarang ketika hanya subset dari set asli variabel dimaksudkan untuk disimpan dalam model. Semua prediktor memiliki koefisien non-nol dalam model akhir.

Di GLM, dapat ditentukan salah satu solusi yaitu sebagi berikut:

* IRLSM: Metode Kuadrat Minimal Secara Impressative (default)
* L\_BFGS: Memori-terbatas algoritma Broyden-Fletcher-Goldfarb-Shanno
* OTOMATIS: Mengatur solusi berdasarkan data dan parameter yang diberikan.
* KOORDINATE\_DESCENT: Koordinat Layak (eksperimental)
* COORDINATE\_DESCENT\_NAIVE: Coordinate Decent Naive (eksperimental)
* GRADIENT\_DESCENT\_LH: Gradient Descent Likelihood (tersedia untuk keluarga Ordinal saja; standar untuk keluarga Ordinal)
* GRADIENT\_DESCENT\_SQERR: Gradient Descent Squared Error (tersedia hanya untuk keluarga Ordinal)

Contoh ini adalah mengkalkulasikan beberapa ukuran dalam bidang statistik. Digunakan variabel X dan Y dalam struct dimana harus terdapat satu variabel per elemen.

Contohnya :

Y.Jarak

X.Kecepatan

X.Waktu

Setiap variabel yang ada harus mempunyai N x 1 vektor kolom dimana setiap barisnya akan diamati. Ini juga memungkinkan untuk menghasilkan bobot (weight) dalam struct dan akan diperhitungkan dalam prosedur estimasinya. Saat menggunakan distribusi binomial, Y seharusnya mempunyai sebuah variabel dengan 2 kolom dimana kolom 1 sebagai perespon dan kolom 2 sebagai sample ukuran tiap perespon. Jika tidak mempunyai contoh ukurannya, dapat membuat 2 kolom vektor sekali saja.

Untuk mengestimasi model GLM terdiri dari 3 tahap :

1. Buat objek model GLM mdl =GLM;
2. Tentukan tautan yang dibutuhkan dan distribusinya mdl.Distribution = .. mdl.Link = ..
3. Estimasikan parameter dengan data mdl.Estimate(Y,X);

Setelah mengestimasikan, memungkinkan untuk menyesuaikan nilai baru ke model yang terestimasi yaitu fit = mdl.Fit(Xnew);

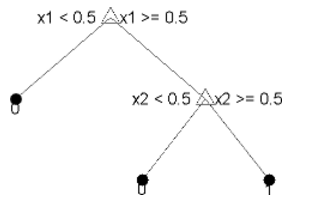
Hasilnya akan sering sama dengan yang ada di R.

**Decision Trees and Random Forests**

Decision tree atau pohon acak merupakan metode klasifikasi yang menggunakan struktur pohon atau struktur berhirarki. Konsep dari decision tree adalah mengubah data menjadi pohon keputusan dan aturan-aturan keputusan. Manfaat utama dari penggunaan decision tree adalah kemampuannya untuk menghasilkan proses pengambilan keputusan yang kompleks menjadi lebih simpel sehingga pengambil keputusan akan lebih menginterpretasikan solusi dari permasalahan.

Decision tree atau pohon klasifikasi dan pohon regresi, memprediksi respons terhadap data. Untuk memprediksi respons, ikuti keputusan di pohon dari akar (awal) node ke simpul daun. Simpul daun mengandung respon. Pohon klasifikasi memberi respon yang terukur, seperti 'benar' atau 'salah'. Pohon regresi juga memberikan tanggapan numerik.

Seperti contoh Decision tree ini ;



Pohon ini memprediksi klasifikasi berdasarkan dua prediktor, x1 dan x2. Untuk memprediksi, mulailah di simpul atas, diwakili oleh segitiga (Δ). Keputusan pertama adalah apakah x1 lebih kecil dari 0,5. Jika demikian, ikuti cabang kiri, dan lihat bahwa pohon mengklasifikasikan data sebagai tipe 0.

Namun, jika x1 melebihi 0,5, maka ikuti cabang kanan ke node segitiga kanan bawah. Di sini pohon bertanya apakah x2 lebih kecil dari 0,5. Jika demikian, maka ikuti cabang kiri untuk melihat bahwa pohon mengklasifikasikan data sebagai tipe 0. Jika tidak, maka ikuti cabang kanan untuk melihat bahwa pohon mengklasifikasikan data sebagai tipe 1.

Decision tree individu cenderung berlebihan. TreeBagger dalam matlab menghasilkan Decision tree di ensemble menggunakan sampel bootstrap dari data. Selain itu, TreeBagger memilih subkumpulan acak prediktor untuk digunakan pada setiap pembagian keputusan seperti dalam algoritma random forests.

Secara default, pohon TreeBagger mengklasifikasikan beberapa trees ke bawah. Untuk menyimpan pohon regresi, tentukan 'Metode', 'regresi'.

TreeBagger bergantung pada fungsi ClassificationTree and RegressionTree untuk menumbuhkan pohon individu. Secara khusus, ClassificationTree dan RegressionTree menerima jumlah fitur yang dipilih secara acak untuk setiap pembagian keputusan sebagai argumen input opsional.

Klasifikasi random forest dilakukan melalui penggabungan pohon (tree) dengan melakukan training pada sampel data yang dimiliki. Penggunaan pohon (tree) yang semakin banyak akan mempengaruhi akurasi yang akan didapatkan menjadi lebih baik. Penentuan klasifikasi dengan random forest diambil berdasarkan hasil voting dari tree yang terbentuk. Pemenang dari tree yang terbentuk ditentukan dengan vote terbanyak. Pembangunan pohon (tree) pada random forest sampai dengan mencapai ukuran maksimum dari pohon data.

Random forest menggunakan Decision Tree untuk melakukan proses seleksi. Pohon yang dibangun dibagi secara rekursif dari data pada kelas yang sama. Pemecahan (split) digunakan untuk membagi data berdasarkan jenis atribut yang digunakan. Pembuatan decision tree pada saat penentuan klasifikasi, pohon yang buruk akan membuat prediksi acak yang saling bertentangan. Sehingga ada decision tree akan menghasilkan jawaban yang baik. Proses Klasifikasi akan berjalan jika semua tree telah terbentuk. Pada saat proses klasifikasi selesai dilakukan, inisialisasi dilakukan dengan sebanyak data berdasarkan nilai akurasinya.

Langkah-langkah yang diambil untuk mengimplementasikan random forest:

1. Anggaplah ada pengamatan N dan fitur M dalam kumpulan data pelatihan. Pertama, sampel dari set data pelatihan diambil secara acak dengan penggantian.

2. Sebuah subset fitur M dipilih secara acak dan fitur mana saja yang memberikan perpecahan terbaik digunakan untuk membagi node secara iteratif.

3. Pohon itu tumbuh ke yang terbesar.

4. Langkah di atas diulang dan prediksi diberikan berdasarkan agregasi prediksi dari n jumlah pohon.

**Ensembles - bagging and boosting**

Metode Ensemble, yang menggabungkan beberapa pohon keputusan untuk menghasilkan kinerja prediksi yang lebih baik daripada menggunakan pohon keputusan tunggal. Prinsip utama di balik model ensemble adalah bahwa sekelompok data yang kecil berkumpul bersama untuk membentuk data yang kuat.

Teknik untuk metode Ensemble adalah :

* Bagging *(Bootstrap Aggregation)* digunakan ketika untuk mengurangi varians dari pohon keputusan. Ide di sini adalah membuat beberapa subset data dari sampel pelatihan yang dipilih secara acak dengan penggantian. Setiap kumpulan data subset digunakan untuk melatih pohon keputusan mereka. Sebagai hasilnya dengan ensemble model yang berbeda. Rata-rata semua prediksi dari pohon yang berbeda digunakan yang lebih kuat dari pohon keputusan tunggal.
* Boosting adalah teknik ensembel lain untuk menciptakan koleksi prediktor. Dalam teknik ini, data yang ada dipelajari secara berurutan dengan data awal yang memasang model sederhana ke data dan kemudian menganalisis data untuk kesalahan. Dengan kata lain, menyesuaikan pohon berturut-turut (sampel acak) dan di setiap langkah, tujuannya adalah untuk memecahkan kesalahan alami dari pohon sebelumnya.

**Multilayer Perceptron**

Multi-Layer Perceptron adalah jaringan syaraf tiruan feed-forward yang terdiri dari sejumlah neuron yang dihubungkan oleh bobot-bobot penghubung. Neuron-neuron tersebut disusun dalam lapisan-lapisan yang terdiri dari satu lapisan input (input layer), satu atau lebih lapisan tersembunyi (hidden layer) dan satu lapisan output (output layer). Lapisan input menerima sinyal dari luar, kemudian meneruskannya ke lapisan tersembunyi pertama sehingga mencapai lapisan output.

Berikut ini adalah tahap-tahapan dalam penyelesaian masalah menggunakan metode Jaringan Syarat Tiruan menggunakan Multilayer Percepteron.

1. Identifikasi masalah

Tahap ini merupakan identifikasi masalah yang hendak diselesaikan dengan jaringan syaraf tiruan, meliputi identifikasi jenis dan jumlah masukan serta keluaran pada jaringan.

1. Menyiapkan training data set

Training data set merupakan kumpulan pasangan data masukan-keluaran berdasarkan pengetahuan yang telah dikumpulkan sebelumnya. Banyaknya data set harus mencukupi dan dapat p p mewakili setiap kondisi yang hendak diselesaikan. Terbatasnya data set akan menyebabkan akurasi jaringan menjadi rendah.

1. Inisialisasi dan pembentukan jaringan

Tahap inisialisasi meliputi penentuan topologi, pemilihan fungsi aktivasi, dan pemilihan fungsi pelatihan jaringan. Penentuan topologi adalah penentuan banyaknya hidden layer dan penentuan jumlah neuron pada input layer, hidden layer dan output layer.

1. Simulasi jaringan

Simulasi jaringan dilakukan untuk melihat keluaran jaringan berdasarkan masukan, bobot neuron dan fungsi aktivasinya.

1. Pelatihan / training jaringan

Sebelum melakukan pelatihan, dilakukan penentuan parameter training terlebih dahulu, seperti penentuan jumlah iterasi, learning rate, error yang diijinkan. Setelah itu dilakukan pelatihan yang merupakan proses iteratif untuk menentukan bobot koneksi antar neuron.

6. Menggunakan jaringan untuk pengenalan pola

Setelah pelatihan dilakukan, jaringan siap untuk digunakan untuk pengenalan pola. Kemampuan jaringan dalam mengenal pola sangat bergantung dari bagaimana jaringan tersebut dilatih.

**Bayesian Learning**

Regresi linier berganda merupakan salah satu model linier yang melibatkan variabel respon dan variabel prediktor didalamnya. Pada dasarnya terdapat dua paradigma utama dalam pendugaan parameter yaitu paradigma frekuentist dengan pendekatan klasik dan paradigma Bayesian dengan pendekatan Bayesian. Metode klasik mendasarkan inferensinya hanya pada informasi yang dikandung dalam sampel. Pendekatan Bayes menggabungkan informasi yang dikandung dalam sampel dengan informasi lain yang telah tersedia sebelumnya. Metode klasik parameter yang dihasilkan bersifat konstan sedangkan pada metode Bayesian parameter diperlakukan sebagai variabel acak yang memiliki distribusi.

Dalam statistik Bayesian, nilai parameter dinyatakan menggunakan teori probabilitas. Fungsi kepadatan peluang untuk parameter β dan  mencerminkan kredibilitas nilai yang mungkin untuk parameter ini. Tujuan dari pendekatan Bayesian adalah dengan menggunakan data untuk memperbarui distribusi parameter, dan kemudian menarik kesimpulan menggunakan distribusi yang telah diketahui. Statistik *ɵ* merupakan simbol untuk merepresentasikan general vektor dari parameter. Symbol *f,g,h,k,p,q,r* dan *t* adalah fungsi densitas.

Regresi berganda dengan bayes hampir sama dengan regresi klasik kecuali pada penentuan distribusi prior parameter. Penentun prior sangat penting dalam Bayesian modeling. Pada metode Bayesian, semua parameter diperlakukan sebagai variabel sehingga parameter memiliki sebaran distribusi. Dari alasan tersebut parameter ɵ akan mempunyai nilai dalam domain populasi ϴ, dengan densitas *f(ɵ)*. Dan densitas ini yang akan dinamakan sebagai prior dari ɵ, dengan adanya informasi prior yang dipadukan dengan data,Y, akan membentuk posterior, dengan menghitung densitas bersyarat dari ϴ diberikan Y*=y*. Distribusi prior dikelompokkan menjadi dua macam yaitu *Conjugate prior* dan *non-conjugate prior*. Prior *conjugate* merupakan prior yang sesuai atau terkait dengan pola model *likelihood* datanya. Penggunaan prior *conjugate* mempermudah dalam mengkalkulasikan posterior karena pada prior *conjugate*, distribusi posteriornya telah diketahui.



Persamaan diatas merupakan persamaan Teorema Bayes.

Pemilihan distribusi prior pada pendekatan bayes dapat dilakukan melalui data masa lalu yang telah ada dan distribusi prior bisa disebut dengan distribusi prior "data based" jika data masa lalu tidak tersedia. Distribusi prior dipilh berdasarkan kepercayaan peneliti dan distribusi prior jenis ini disebut non data based. Selain distribusi prior terdapat juga pendekatan distribusi posterior. Distribusi ini berkaitan dengan penentuan masing-masing paramter pada pola distribusi prior tersebut. Distribusi prior informatif mengacu pada pemberian parameter dari distribusi prior yang telah dipilih. Baik prior yang dipilih, pemberian nilai parameter pada distribusi prior ini akan sangat mempengaruhi bentuk distribusi posterior yang akan didapatkan pada informasi data yang diperoleh. Untuk mendapatkan distirbusi posterior dari β , distribusi bersama dari p dan sampel harus dihitung terlebih dahulu.

Misalkan *x* adalah sampel data yang label kelasnya tidak diketahui. Misalkan *H* adalah hipotesa: sedemikian sehingga sampel data *x* termasuk dalam kelas khusus *c*. Penulis ingin menentukan *P*(*H/x*), probabilitas bahwa hipotesa *H* berlaku dengan diberikannya sampel data hasil pengamatan *x*. *P*(*H/x*) adalah probabilitas *posterior* yang menggambarkan keyakinan kita pada hipotesa setelah *x* diberikan. Sebaliknya, *P*(*H*) adalah probabilitas *H* sebelumnya untuk sesuatu sampel, terlepas dari bagaimana bentuk data dalam sampel. Probabilitas *posterior P*(*H/x*) didasarkan pada lebih banyak informasi daripada probabilitas *priori P*(*H*). Teorema Bayes memberikan cara menghitung probabilitas posterior *P*(*H/x*) dengan menggunakan probabilitas *P*(*H*)*, P*(*x*) dan *P*(*x/H*).

Hubungan dasar adalah

*P*(*H/x*) = [*P*(*x/H*)(*P*(*H*)]*/P* (*x*) (3.1)

Andaikan sekarang bahwa terdapat suatu himpunan dari*m* sampel *S* = {*S*1*, S*2*, . . . ,S*m} ( himpunan data ) di mana setiap sampel *S*i digambarkan sebagai vektor dimensi-*n* {*x*1*, x*2*, . . . ,x*n}. Nilai *x*i bersesuaian dengan sifat-sifat *A*1*,A*2*, . . . ,A*n. Juga, terdapat *k* kelas *c*1*, c*2*, . . . , c*k, dan setiap sampel termasuk ke dalam salah satu kelas ini. Diberikan sampel data tambahan *x* ( kelasnya tidak diketahui ), dimungkinkan memprediksi kelas untuk *x* dengan menggunakan probabilitas bersyarat tertinggi *P*(*c*t*/x*), di mana *i* = 1*, . . . ,k*. Yaitu ide dasar dari klasifikator Bayes Naif. Probabilitas ini dihitung dengan menggunakan Teorema Bayes:

*P*(*c*t*/x*) = [*P*(*x/c*i)(*P*(*c*i)]*/P* (*x*)

Karena *P*(*x*) adalah konstan untuk semua kelas, maka hanya perkalian *P*(*x/c*i)(*P*(*c*i) yang perlu dimaksimalkan. Probabilitas priori dihitung dari kelas sebagai *P*(*c*i) = jumlah sampel dari kelas *c*t*/m*, (*m* adalah jumlah total sampel). Karena penghitungan *P*(*x/c*i) sangat kompleks, terutama untuk himpunan data besar, diajukanlah asumsi naif atas sifat saling lepas bersyarat. Dengan menggunakan asumsi ini, dapat dikatakan *P*(*x/c*i) sebagai perkalian, yaitu:

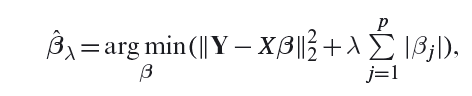
*P*(*x/c*i) = [*P*(*x/c*i)(*c*t*/m*]*/P* (*x*)

di mana *x*t adalah nilai-nilai untuk sifat-sifat dalam sampel *x*. Probabilitas *P*(*x*t*/c*i) dapat ditaksir dari himpunan data.

**Linear/Logistic Regression**

Teknik pemilihan fitur yang kuat yang sangat berguna untuk masalah regresi adalah Lasso *(Least Absolute Shrinkage and Selection Operator).* Lasso pada dasarnya adalah metode regularisasi. Dalam situasi dengan klasifikasi dengan banyak variabel independen, lasso menawarkan cara yang rapi untuk memodelkan variabel dependen sementara secara otomatis memilih variabel yang signifikan dengan mengecilkan koefisien prediktor yang tidak penting ke nol.

Dalam regresi linier biasa memiliki respons yaitu Y∈Rn kontinyu, sebuah matriks X × p matriks dan parameter vektor β∈Rp. Penaksir lasso kemudian didefinisikan sebagai :



Regresi linear adalah teknik statistik yang paling sederhana dan paling banyak digunakan untuk pemodelan prediktif. Pada dasarnya ini memberi persamaan dimana memiliki fitur sebagai variabel independen. Persamaan nya sebagai berikut :

https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2017/06/05185714/snip2.png

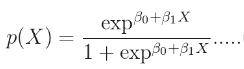
Di sini terdapat Y sebagai variabel dependen , X adalah variabel independen dan semua theta adalah koefisien. Koefisien pada dasarnya adalah bobot yang ditentukan untuk fitur-fitur, berdasarkan kepentingannya. Jadi, pahami regresi linier hanya dengan satu fitur, yaitu, hanya satu variabel independen. Karena itu persamaan menjadi,



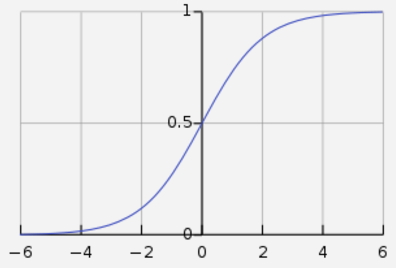
Persamaan ini disebut persamaan regresi linier sederhana, yang mewakili garis lurus, di mana ‘θ0’ adalah intersep, ‘θ1’ adalah kemiringan garis.

Untuk logistik regresi diasumsikan bahwa hasil (variabel prediksi) dan prediktor dilambangkan oleh Y dan X masing-masing dan dua kelas minat dilambangkan dengan + dan - masing-masing. Probabilitas bersyarat dimodelkan bahwa hasil Y adalah +, mengingat bahwa variabel input (prediktor) adalah X. Probabilitas kondisional dilambangkan dengan p (Y = + | X) yang disingkat sebagai p (X) dimana mengacu pada hasil positif Y = +.

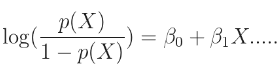
Seperti yang disebutkan sebelumnya, kemungkinan keanggotaan kelas sehingga harus memastikan bahwa fungsi hipotesis (kata yang bagus untuk model) selalu terletak antara 0 dan 1. Fungsi yang diasumsikan dalam regresi logistik adalah:



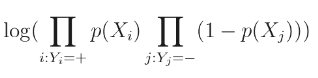
p (X) memang berada di antara 0 dan 1 karena X bervariasi. Biasanya, bagaimanapun, nilai-nilai X dibatasi yang ditunjukkan pada Gambar 1. Gambar tersebut juga menggambarkan karakteristik kurva berbentuk S khas regresi logistik.



Ekivalen persamaan dari gambar di atas adalah :

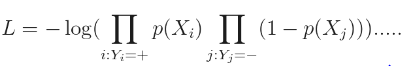


Kuantitas di sebelah kiri adalah log. Jadi model tersebut adalah regresi linear dari log-peluang dan kadang dinamakan logistik. Untuk menemukan nilai-nilai θ0 dan θ1 yang menghasilkan p (X) yang paling akurat mengklasifikasikan semua titik data yang diamati yaitu, yang termasuk kelas positif memiliki probabilitas sedekat mungkin ke 1 dan yang termasuk kelas negatif memiliki probabilitas sedekat mungkin ke 0. Salah satu cara mengatasi masalah ini adalah dengan mengatakan bahwa kita ingin memaksimalkan probabilitas ini, sering disebut sebagai kemungkinan:



Di mana Πmewakili produk di atas i dan j, yang berjalan di atas i dan j poin yang digolongkan masing-masing. Pendekatan ini disebut estimasi kemungkinan maksimum, cukup umum di banyak pembelajaran mesin, terutama yang melibatkan probabilitas.

Satu cara untuk meminimalkan kemungkinan log negatif yang, tentu saja, adalah sama dengan memaksimalkan kemungkinan log. Kuantitas yang diminimalkan adalah sebagai berikut:



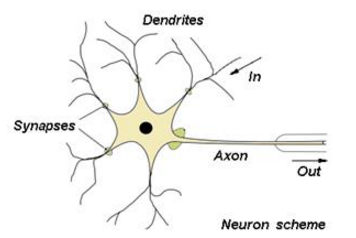
Namun memiliki pengaruh sedikit saja pada regresi logistik.

**Neural Network (MLP's, CNN's, RNN's)**

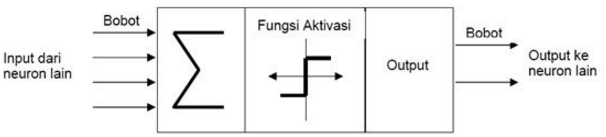
Neural Network merupakan kategori ilmu Soft Computing yang mengadopsi dari kemampuan otak manusia yang mampu memberikan stimulasi/rangsangan, melakukan proses, dan memberikan output. Output diperoleh dari variasi stimulasi dan proses yang terjadi di dalam otak manusia. Kemampuan manusia dalam memproses informasi merupakan hasil kompleksitas proses di dalam otak. Kekuatan komputasi yang luar biasa dari otak manusia ini merupakan sebuah keunggulan di dalam kajian ilmu pengetahuan.

Konsep Neural Network yaitu :

Ide dasar Neural Network dimulai dari otak manusia, dimana otak memuat sekitar 1011 neuron. Neuron ini berfungsi memproses setiap informasi yang masuk. Satu neuron memiliki 1 akson, dan minimal 1 dendrit. Setiap sel syaraf terhubung dengan syaraf lain, jumlahnya mencapai sekitar 104 sinapsis. Masing-masing sel itu saling berinteraksi satu sama lain yang menghasilkan kemampuan tertentu pada kerja otak manusia.



Dari struktur neuron pada otak manusia, dan proses kerja yang dijelaskan di atas, maka konsep dasar pembangunan neural network buatan (Artificial Neural Network) terbentuk. Ide mendasar dari Artificial Neural Network (ANN) adalah mengadopsi mekanisme berpikir sebuah sistem atau aplikasi yang menyerupai otak manusia, baik untuk pemrosesan berbagai sinyal elemen yang diterima, toleransi terhadap kesalahan/error, dan juga parallel processing.

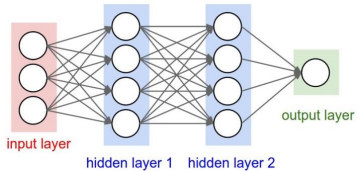


Neural network dibangun dari banyak node/unit yang dihubungkan oleh link secara langsung. Link dari unit yang satu ke unit yang lainnya digunakan untuk melakukan propagasi aktivasi dari unit pertama ke unit selanjutnya. Setiap link memiliki bobot numerik. Bobot ini menentukan kekuatan serta penanda dari sebuah konektivitas.

Proses pada ANN dimulai dari input yang diterima oleh neuron beserta dengan nilai bobot dari tiap-tiap input yang ada. Setelah masuk ke dalam neuron, nilai input yang ada akan dijumlahkan oleh suatu fungsi perambatan (summing function), dengan lambang sigma (∑). Hasil penjumlahan akan diproses oleh fungsi aktivasi setiap neuron, disini akan dibandingkan hasil penjumlahan dengan threshold (nilai ambang) tertentu. Jika nilai melebihi threshold, maka aktivasi neuron akan dibatalkan, sebaliknya, jika masih dibawah nilai threshold, neuron akan diaktifkan. Setelah aktif, neuron akan mengirimkan nilai output melalui bobot-bobot outputnya ke semua neuron yang berhubungan dengannya. Proses ini akan terus berulang pada input-input selanjutnya.

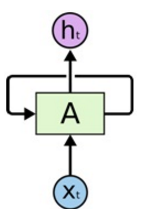
Convolutional Neural Network (CNN) adalah pengembangan dari Multilayer Perceptron (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis Deep Neural Network karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada kasus klasifikasi citra, MLP kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data citra dan menganggap setiap piksel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik.

CNN setiap neuron dipresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap neuron hanya berukuran satu dimensi.



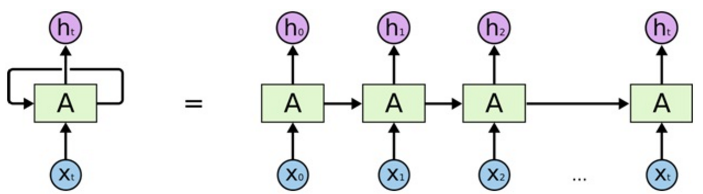
Pada CNN, data yang dipropagasikan pada jaringan adalah data dua dimensi, sehingga operasi linear dan parameter bobot pada CNN berbeda. Pada CNN operasi linear menggunakan operasi konvolusi, sedangkan bobot tidak lagi satu dimensi saja, namun berbentuk empat dimensi yang merupakan kumpulan kernel konvolusi.

Recurrent NN sangatlah sederhana yaitu membantu kita dalam mengolah informasi secara bertahap. Jika mengacu pada Neural Network biasa semua input dan ouput tidak bergantung satu sama lainnya. Hanya jika menggunakan konsep ini maka yang terjadi adalah tugas pada neural network sangatlah banyak dan bertumpuk-tumpuk. Disebut RNN karena RNN melakukan tugas yang sama pada setiap elemen di sebuah urutan, lalu memproses output yang mengacu pada komputasi sebelumnya.



Gambar di atas adalah visualisasi contoh potongan dari sebuah RNN. RNN tersebut mendapat input xt dan menghasilkan output ht . Dan alur loop tersebut memungkinkan informasi untuk dapat dilempar dari satu step menuju step selanjutnya.

Dibawah merupakan gambar sederhana RNN loop. Dapat dilihat bahwa looping dari RNN sebenarnya akan memproses input dari skala waktu 0 sampai t.



Seperti gambar diatas, RNN akan memproses data input satu per satu secara sekuensial dari huruf “h” sampai “l”, hidden layer pun akan melempar data menuju ke hidden layer pada skala waktu selanjutnya. Begitu seterusnya secara sekuensial.

**Deep Learning**

Ide dari deep learning adalah membuat hierarki konsep-konsep dengan arsitektur yang terdiri dari beberapa lapisan, menyusun konsep yang kompleks dari konsep-konsep yang lebih sederhana. Sifat ini memberikan deep learning kemampuan untuk merepresentasikan sebuah permasalahan dengan fleksibilitas yang tinggi.

Deep Learning adalah salah satu jenis algoritma jaringan saraf tiruan yang menggunakan metadata sebagai input dan mengolahnya menggunakan sejumlah lapisan tersembunyi (hidden layer) transformasi non linier dari data masukan untuk menghitung nilai output. Algoritma pada Deep Learning memiliki fitur yang unik yaitu sebuah fitur yang mampu mengekstraksi secara otomatis. Hal ini berarti algoritma yang dimilikinya secara otomatis dapat menangkap fitur yang relevan sebagai keperluan dalam pemecahan suatu masalah. Algoritma semacam ini sangat penting dalam sebuah kecerdasan buatan karena mampu mengurangi beban pemrograman dalam memilih fitur yang eksplisit. Dan algortima ini dapat digunakan untuk memecahkan permasalahan yang perlu pengawasan (supervised), tanpa pengawasan (unsupervised), dan semi terawasi (semi supervised).

Dalam jaringan saraf tiruan tipe Deep Learning setiap lapisan tersembunyi bertanggung jawab untuk melatih serangkaian fitur unik berdasarkan output dari jaringan sebelumnya. Algoritma ini akan menjadi semakin komplek dan bersifat abstrak ketika jumlah lapisan tersembunyi (hidden layer) semakin bertambah banyak. Jaringan saraf yang dimiliki oleh Deep Learning terbentuk dari hirarki sederhana dengan beberapa lapisan hingga tingkat tinggi atau banyak lapisan (multi layer). Berdasarkan hal itulah Deep Learning dapat digunakan untuk memecahkan masalah kompleks yang lebih rumit dan terdiri dari sejumlah besar lapisan transformasi non linier.

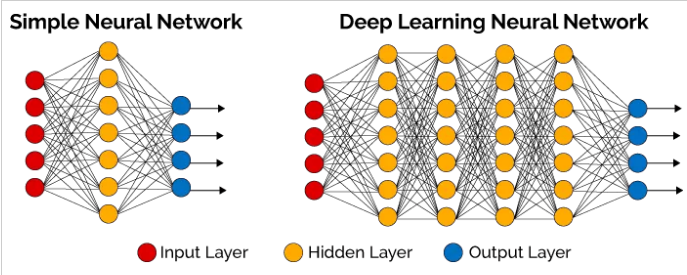
Jenis Deep Learning

* Deep Learning untuk Pembelajaran Tanpa Pengawasan (Unsupervised Learning): Deep Learning tipe ini digunakan pada saat label dari variabel target tidak tersedia dan korelasi nilai yang lebih tinggi harus dihitung dari unit yang diamati untuk menganalisis polanya.
* Hybrid Deep Networks (Deep Learning gabungan): Pendekatan tipe ini bertujuan agar dapat dicapai hasil yang baik dengan menggunakan pembelajaran yang diawasi untuk melakukan analisis pola atau dapat juga dengan menggunakan pembelajaran tanpa pengawasan.

Deep Learning bekerja berdasarkan pada arsitektur jaringan dan prosedural optimal yang digunakan pada arsitektur. Setiap output dari lapisan per lapisan yang tersembunyi dapat dipantau dengan menggunakan grafik khusus yang dirancang untuk setiap output neuron. Kombinasi dan rekombinasi dari setiap neuron yang saling terhubung dari semua unit lapisan tersembunyi dilakukan menggunakan gabungan dari fungsi aktivasi. Prosedur-prosedur tersebut dikenal sebagai Transformasi Non Linier yang digunakan untuk prosedur optimal untuk menghasilkan bobot optimal pada setiap unit lapisan guna mendapatkan nilai target yang dibutuhkan.

Jaringan Saraf Tiruan adalah jaringan saraf yang biasanya menggunakan jaringan seperti umpan maju (feed forward) atau recurrent network yang hanya memiliki 1 atau 2 lapisan tersembunyi. Tetapi, jika lapisan jaringan sarafnya lebih dari 2 layer ke atas atau bahkan mencapai ratusan lapisan itulah yang disebut sebagai Deep Learning. Pada Jaringan Syaraf Tiruan arsitektur jaringan yang dimilikinya kurang kompleks dan membutuhkan lebih banyak informasi tentang data input sehingga dapat menentukan algortima mana yang dapat digunakan. Dalam Jaringan Saraf Tiruan terdiri dari beberapa algoritma yaitu Model Hebb, Perceptron, Adaline, Propagasi Maju, dll. Sedangkan pada algortima jaringan saraf Deep Learning tidak memerlukan informasi apapun terhadap data yang akan dipelajarinya, dan algoritmanya dapat secara mandiri melakuan tuning (penyetelan) dan pemilihan model yang paling optimal.

Ketika dalam proses perancangan, apabila jumlah saraf yang ditambahkan sangat banyak, hal tersebut tidak akan pernah cocok untuk menyelesaikan setiap masalah. Persoalan terpenting dalam Deep Learning adalah jaringan sarafnya dilatih dengan cara penurunan gradien secara sederhana. Pada saat kita menambahkan lapisan jaringan yang semakin banyak, maka sebaliknya penurunan dari gradien semakin berkurang sehingga dapat mempengaruhi nilai outpunya.



**Hidden Markov Models**

Hidden Markov Model (HMM) adalah suatu model probabilitas yang menggambarkan hubungan statistik antara urutan observasi O dan urutan state S yang tidak diobservasi ”hidden” Ciri-ciri HMM adalah sebagai berikut:

a. Observasi diketahui tetapi urutan keadaan (state) tidak diketahui sehingga disebut hidden

b. Observasi adalah fungsi probabilitas keadaan

c. Perpindahan keadaan adalah dalam bentuk probabilitas

HMM mempunyai parameter-parameter distribusi sebagai berikut :

1. Probabilitas Transisi

1. Probabilitas observasi

1. Distribusi keadaan awal

Sedangkan parameter tertentu HMM ada dua yaitu N dan M:

1. N, jumlah keadaan model. Dinotasikan himpunan terbatas untuk keadaan yang mungkin adalah

M, jumlah dari simbol observasi/keadaan, ukuran huruf diskret. Simbol observasi berhubungan dengan keluaran fisik dari sistem yang dimodelkan. Dinotasikan himpunan terbatas untuk observasi yang mungkin adalah

HMM yang harus dipecahkan untuk model yang diterapkan di dunia nyata, yaitu:

1. Menghitung bila diberikan urutan observasi dan λ = (A, B, π).

Solusi:

Cara umum yang biasa digunakan adalah dengan memeriksa setiap kemungkinan urutan *N state* sepanjang *T* (banyaknya observasi). Hal ini tidak mungkin dilakukan karena perhitungannya kurang efisien.

1. Memilih urutan keadaan yang paling optimal yang berhubungan dengan Q = q1,q2 ... qT bila diberikan urutan observasi O = O1,O2, ...,OT dan sebuah model λ = (A, B, π).

Solusi:

Inti dari permasalahan 2 adalah mencari deretan *state* yang tersembunyi (*hidden*) untuk suatu deretan observasi yang dihasilkan dari model λ. Deretan *state* yang dicari harus merupakan deretan yang optimal agar dapat dimodelkan pada deretan obervasi dari model λ. Metode yang biasa digunakan untuk menemukan deretan *state* optimal adalah algoritma *viterbi* (*dynamic programming*). Algoritma *viterbi* dapat memaksimalkan nilai *P*(*O*| λ) sehingga memberikan deretan observasi yang optimal. Berdasarkan aturan Bayes, secara matematis *P*(*Q*|*O*, λ) dapat dinyatakan sebagai berikut:

*P*(*O*|λ)=

1. Mengatur parameter λ agar maksimal

Solusi:

Permasalahan 3 merupakan masalah yang paling sulit jika dibandingkan dengan masalah-masalah sebelumnya. Intinya adalah menentukan suatu metode yang dapat disesuaikan dengan model parameter *A*, *B*, π untuk memenuni kriteria optimasi tertentu. Tidak ada cara untuk menganalisa set model parameter yang memaksimalkan peluang dari deretan observasi secara tertutup.

Akan tetapi, hal ini dapat dilakukan dengan memilih model λ yang mempunyai kemungkinan (likelihood), P(O|λ) yang dimaksimalkan secara local (localy-maximized) dengan prosedur iterasi seperti metode Baum-Welch (*expectation-maximization*, *EM*). Prosedure iterasi merupakan proses *training* yang berlangsung terus-menerus sampai kondisi kritis (*local minimal*) terpenuhi.

Untuk mendeskripsikan prosedur reestimasi (*update* iterasi dan perbaikan) dari parameter HMM, pertama kali didefinisikan ξ*t*(*i*, *j*), yaitu peluang berada pada *state i* waktu *t* dan berpindah menuju *state j* waktu *t*+1 yang dimodelkan sebagai berikut:

ξt *(i,j)*= *p*(*qt =i, qt+1 = j | O, λ*)

**Naive Bayes**

Naive Bayes merupakan pengklasifikasian dengan metode probabilitas yang dikemukakan oleh ilmuan Inggris Thomas Bayes. Pada penerapan teorema Bayes (aturan Bayes) dengan asumsi independensi (ketidaktergantungan) yang kuat (naif). Dengan kata lain, dalam Naive Bayes yang digunakan adalah model fitur independen.

Algoritma *naive bayes classifer* adalah sebagai berikut :

1. Set data latih dengan label kelas yang telah ditentukan. masing-masing data latih diwakili oleh n-vektor atribut, X = (X1, X2, X3, ... , Xn).
2. Misalkan bahwa terdapat kelas sebanyak m, C1, C2, C3, ..., Cm. Terdapat sebuah data X, sistem klasifikasi akan memprediksi bahwa X tergolong dalam kelas yang memiliki nilai probabilitas posterior tertinggi, sesuai dengan kondisi X. Jadi *naive bayes classifier* memprediksi data X tergolong ke kelas Ci jika dan hanya jika

Kelas Ci maksimal dari disebut *maximum posteriori hypothesis.* Berikut rumus menghitung nilai probabilitas posterior berdasarkan teorema *bayesian.*

Keterangan :

= *Posterior* atau probabilitas masing-masing kelas

= Probabilitas *prior*

1. Nilai P(X) konstan atau tetap untuk semua kelas, hanya dan yang lebih diutamakan. Jika probabilitas prior tidak diketahui, maka umumnya diasumsikan bahwa jumlah data dari semua kelas kemungkinan sama. Dengan demikian . Sebagai catatan untuk menghitung probabilitas prior adalah dengan menghitung banyaknya data pada kelas Ci dibagi dengan banyaknya jumlah data latih.
2. Jika data memiliki banyak atribut, maka itu memerlukan komputasi yang besar. Untuk mengurangi komputasi dalam menghitung maka dibuat asumsi *naive* dari kondisi kelas yang bebas. Ini berarti bahwa nilai atribut secara kondisional bebas dari nilai yang lainya. Dengan demikian,

xk mengacu pada nilai atribut Ak dari data X. Untuk tiap-tiap atribut, perlu diperhatikan lagi apakah bersifat kategori atau kontinu. Jadi menghitung terdapat pertimbangan sebagai berikut :

1. Jika Ak bersifat kategori, maka hitung jumlah banyaknya data dari kelas Ci yang memiliki nilai xk untuk atribut ak dibagi dengan |C*i*, D|, banyak nya jumlah data kelas C*i* pada data latih
2. Jika Ak bersifat kontinu, maka biasanya diasumsikan dengan distribusi *gaussian* dengan nilai rata-rata (µ) dan standar deviasi (σ). Berikut rumus dari distribusi *gaussian*

Jadi,

1. Untuk memprediksi label kelas X, dihitung untuk setiap kelas. Label kelas yang diprediksi adalah kelas yang memiliki nilai maksimum.

**Game Theory**

Permainan take-away yang paling terkenal adalah permainan Nim, dimainkan sebagai berikut. Ada tiga tumpukan chip berisi x1, x2, dan x3 chip masing-masing. (Tumpukan ukuran 5, 7, dan 9 membuat permainan yang bagus.) Dua pemain bergiliran bergerak. Setiap langkah terdiri dari memilih salah satu tumpukan dan menghapus chip dari itu. Tidak dapat menghapus chip dari lebih dari satu tumpukan dalam satu giliran, tetapi dari tumpukan yang dipilih, kita dapat menghapus sebanyak mungkin chip yang diinginkan, dari satu chip ke seluruh tumpukan. Pemenangnya adalah pemain yang menghapus chip terakhir.

Analisis awal. Ada tepat satu posisi terminal, yaitu (0, 0, 0), yang karena itu posisi P. Solusi untuk satu-tumpukan Nim : menghapus seluruh tumpukan. Setiap posisi dengan tepat satu tumpukan yang tidak kosong, katakan (0, 0, x) dengan x> 0. Oleh karena itu adalah posisi-N. Pertimbangkan dua tumpukan Nim. Sangat mudah untuk melihat bahwa P-positions adalah mereka yang dua tumpukan memiliki jumlah chip yang sama, (0, 1, 1), (0, 2, 2), dll.

Ini karena jika giliran lawan bergerak dari posisi seperti itu, dia harus berubah ke posisi di mana dua tumpukan memiliki jumlah chip yang tidak sama, dan kemudian bisa segera kembali ke posisi dengan jumlah chip yang sama (mungkin terminal posisi).

Jika ketiga tumpukan itu tidak kosong, situasinya menjadi lebih rumit. Jelas, (1, 1, 1), (1, 1, 2), (1, 1, 3) dan (1, 2, 2) semuanya adalah N-posisi karena mereka dapat dipindahkan ke (1, 1, 0) atau (0, 2, 2). Posisi paling sederhana berikutnya adalah (1, 2, 3) dan harus posisi P karena bisa hanya dipindahkan ke salah satu posisi N yang ditemukan sebelumnya. Kita bisa terus dan menemukan bahwa posisi P yang paling sederhana berikutnya adalah (1, 4, 5), dan (2, 4, 6), tetapi sulit untuk melihat cara menggeneralisasikan ini. Apakah (5, 7, 9) posisi P? Apakah (15, 23, 30) posisi P?

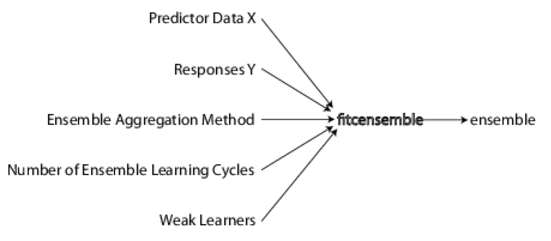
Jika melanjutkan dengan analisis di atas, dapat menemukan sebuah pola. Pola ini berlaku menggunakan konsep dasar P-posisi dan N-posisi.

**Decision Forests**

Dengan menggunakan berbagai metode, dapat menggabungkan hasil dari banyak pembelajaran yang lemah menjadi satu ensembel berkualitas tinggi. Metode ini mengikuti sintaks yang sama, sehingga dapat mencoba metode yang berbeda dengan sedikit perubahan pada perintah di matlab. Anda dapat membuat ensemble untuk klasifikasi dengan menggunakan fitcensemble atau untuk regresi dengan menggunakan fitrensemble. Untuk melatih ensemble untuk klasifikasi menggunakan fitcensemble, gunakan sintaks ini :

ens = fitcensemble(X,Y,Name,Value)

X adalah matriks data. Setiap baris berisi satu pengamatan, dan setiap kolom berisi satu variabel prediktor. Y adalah vektor tanggapan, dengan jumlah pengamatan yang sama seperti baris dalam X. Nama, Nilai menentukan opsi tambahan menggunakan satu atau lebih argumen pasangan nama-nilai. Misalnya, menentukan metode agregasi ensembel dengan argumen 'Metode', jumlah siklus pembelajaran ensemble dengan argumen 'NumLearningCycles', dan jenis pembelajaran yang lemah dengan argumen 'Pembelajar'.



Demikian pula ensemble dapat dilatih untuk regresi dengan menggunakan fitrensemble, yang mengikuti sintaks yang sama seperti fitcensemble. Untuk detail tentang argumen input dan argumen pasangan nama-nilai, lihat halaman fungsi fitrensemble.

**Support Vector Machine**

Support Vector Machines (SVM) adalah sistem pembelajar yang menggunakan sebuah ruang hipotesis fungsi linier dalam ruang fitur berdimensi tinggi, dilatih dengan menggunakan sebuah algoritma pembelajar dari teori optimasi yang mengimplementasikan sebuah bias pembelajar yang diturunkan dari teori pembelajar statistika.

Karena konsep awal *support vector machines*hanya untuk mengatasi masalah klasifikasi dua klas maka SVM tidak dapat diterapkan untuk masalah *multiclass* sehingga dikembangkan metode-metode pengambilan keputusan untuk mengatasi masalah ini. Metode-metode tersebut antara lain metode *one-against-all*, metode *one- against-one* (*pairwise*), metode *decision directed acyclic graph*, dan sebagainya. Perkembangan SVM pun dinilai sangat cepat, hal ini ditandai dengan banyaknya variasi-variasi SVM yang telah berhasil dikembangkan oleh para peneliti. Sedikitnya terdapat lima jenis SVM antara lain :

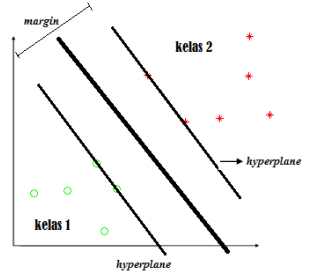
1. Least- Squares Support Vector Machines (LSVM)
2. Linear Programming Support Vector Machines (LPSVM)
3. Sparse Support Vector Machines (SSVM)
4. Robust Support Vector Machines (RSVM)
5. Bayesian Support Vector Machines (BSVM)

Masih banyak lagi variasi SVM yang telah dikembangkan, namun jelas bahwa masing-masing variasi SVM tersebut memiliki kelemahan dan keunggulan tersendiri.

Kernel Support Vector Machines

Pada machine learning, kernel adalah sebuah kelas algoritma untuk menganalisa pola. Tugas utama kernel adalah untuk menemukan dan mempelajari tipe-tipe hubungan pada dataset seperti pengelompokan, klasifikasi, pengurutan, dan korelasi. Umumnya, kernel menggunakan data yang telah direpresentasikan dalam vector.

Pada SVM, untuk memisahkan kelas algoritma ini akan berusaha menemukan hyperplane yang terbaik pada input space, Hyperplane pemisah terbaik antara kedua kelas dapat ditemukan dengan mengukur margin hyperplane tsb. dan mencari titik maksimalnya. Margin adalah jarak antara hyperplane tersebut dengan pattern terdekat dari masing-masing class. Pattern yang paling dekat ini disebut sebagai support vector. Berikut adalah ilustrasi dari pemisahan kelas pada SVM.



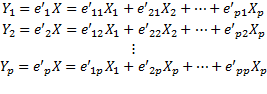
Pada kenyataannya tidak semua data dapat dipisahkan secara linier, untuk mengatasi permasalahan tersebut SVM dapat dimodifikasi dengan menambahkan fungsi kernel didalamnya. ide dasar dari metode kernel ini adalah dengan memetakan data 𝑥 ke ruang vektor yang berdimensi lebih tinggi dengan fungsi Φ(𝑥) sehingga pada ruang vektor yang baru ini, hyperplane dapat dikonstruksikan. Selanjutnya, perhitungan untuk menemukan titik-titik support vectornya bergantung pada dot product dari data yang sudah ditransformasikan pada ruang baru dimensi yang baru. Karena sulitnya menemukan fungsi transformasi dari Φ, maka menurut Mercer, perhitungan dot product tersebut dapat digantikan dengan fungsi kernel 𝐾(𝑥𝑖,𝑥𝑗) dimana fungsi tersebut mendefinisikan transformasi Φ secara implisit. Inilah yang disebut dengan “kernel trick”.

**Principal Component Analysis**

*Principal Component Analysis* adalah analisis *multivariate* yang mentransformasi variabel-variabel asal yang saling berkorelasi menjadi variabel baru yang tidak saling berkorelasi dengan mereduksi sejumlah variabel tersebut sehingga mempunyai dimensi yang lebih kecil namun dapat menerangkan sebagian besar keragaman variabel aslinya.

Banyaknya komponen utama yang terbentuk sama dengan banyaknya variabel asli. Pereduksian (penyederhanaan) dimensi dilakukan dengan kriteria persentase keragaman data yang diterangkan oleh beberapa komponen utama pertama. Apabila beberapa komponen utama pertama telah menerangkan lebih dari 75% keragaman data asli, maka analisis cukup dilakukan sampai dengan komponen utama tersebut.

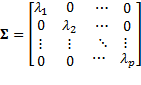
Bila komponen utama diturunkan dari populasi multivariat normal dengan random vektor **X**= (*X*1, *X*2,… , *Xp*) dan vektor rata-rata *μ* = (*μ*1, *μ*2, … , *μp*) dan matriks kovarians **Σ** dengan akar ciri (*eigenvalue*) yaitu *λ*1 ≥ *λ*2 ≥ ⋯ ≥ *λp* ≥ 0 didapat kombinasi linier komponen utama yaitu sebagai berikut.

[](https://1.bp.blogspot.com/-NqTt4ENv-3s/VPjSBAnA-VI/AAAAAAAAATI/m7FPSO8G-w0/s1600/kombinasi+linier.png)

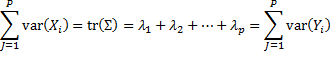
Maka Var(*Yi* ) = *ei*'Σ*ei* dan Cov(*Yi*,*Yk* ) = *ei*'Σ*ei* dimana *i*,*k* = 1, 2, ... , *p*.  
Syarat untuk membentuk komponen utama yang merupakan kombinasi linear dari variabel **X** agar mempunyai varian maksimum adalah dengan memilih vektor ciri (*eigen vector*) yaitu *e* = (*e*1, *e*2, …, *ep*) sedemikian hingga Var(*Yi*) = *ei*'Σ*ei* maksimum dan *ei*'*ei*=1.

* Komponen utama pertama adalah kombinasi linear *e*1'*X* yang memaksimumkan Var(*e*1'*X*) dengan syarat *e*1'*e*1= 1.
* Komponen utama kedua adalah kombinasi linear *e*2'*X* yang memaksimumkan Var(*e*2'*X*) dengan syarat *e*2'*e*2 = 1.
* Komponen utama ke-*i* adalah kombinasi linear *ei*'*X* yang memaksimumkan Var(*ei*'*X*) dengan syarat *ei*'*ek* = 1 dan Cov(*ei*'*ek*)=0 untuk *k* < 1.

Antar komponen utama tersebut tidak berkorelasi dan mempunyai variasi yang sama dengan akar ciri dari **Σ**. Akar ciri dari matriks ragam peragam **Σ** merupakan varian dari komponen utama **Y**, sehingga matriks ragam peragam dari **Y** adalah:

[](https://2.bp.blogspot.com/-hGe9ku8sZJA/VPjV35Jt6tI/AAAAAAAAATU/1KopO1H6sbQ/s1600/matriks+ragam+peragam.png)

Total keragaman variabel asal sama dengan total keragaman komponen utama yaitu:

[](https://1.bp.blogspot.com/-n5QbuhV30o4/VPjWWNAQ2oI/AAAAAAAAATc/t8LDDKObT-8/s1600/total+keragaman.png)

Penyusutan dimensi dari variabel asal dilakukan dengan mengambil sejumlah kecil komponen yang mampu menerangkan bagian terbesar keragaman data. Apabila komponen utama yang diambil sebanyak *q* komponen, dimana *q* < *p*, maka proporsi dari keragaman total yang bisa diterangkan oleh komponen utama ke-*i* adalah:

[https://4.bp.blogspot.com/-BJHhKQB0gqc/VPjW2b8qymI/AAAAAAAAATk/eCvV8OkpZsY/s1600/total%2Bkeragaman%2Bkomponen%2Butama%2Bke%2Bi.png](https://4.bp.blogspot.com/-BJHhKQB0gqc/VPjW2b8qymI/AAAAAAAAATk/eCvV8OkpZsY/s1600/total+keragaman+komponen+utama+ke+i.png)

Penurunan komponen utama dari matriks korelasi dilakukan apabila data sudah terlebih dahulu ditransformasikan ke dalam bentuk baku **Z**. Transformasi ini dilakukan terhadap data yang satuan pengamatannya tidak sama. Bila variabel yang diamati ukurannya pada skala dengan perbedaan yang sangat lebar atau satuan ukurannya tidak sama, maka variabel tersebut perlu dibakukan (*standardized*).  
Variabel baku (**Z**) didapat dari transformasi terhadap variabel asal dalam matriks berikut:

[https://2.bp.blogspot.com/-0ad7yqPeGNc/VPjXZBZLdZI/AAAAAAAAATs/60Ze30outEw/s1600/variabel%2Bbaku.png](https://2.bp.blogspot.com/-0ad7yqPeGNc/VPjXZBZLdZI/AAAAAAAAATs/60Ze30outEw/s1600/variabel+baku.png)

**V**1/2 adalah matriks simpangan baku dengan unsur diagonal utama adalah (*αii*)1/2 sedangkan unsur lainnya adalah nol. Nilai harapan E(**Z**) = 0 dan keragamannya adalah

[https://4.bp.blogspot.com/-FEGufpkrqUQ/VPjXod_Wx8I/AAAAAAAAAT0/FfKXv2RIQAU/s1600/keragaman.png](https://4.bp.blogspot.com/-FEGufpkrqUQ/VPjXod_Wx8I/AAAAAAAAAT0/FfKXv2RIQAU/s1600/keragaman.png)

Dengan demikian komponen utama dari Z dapat ditentukan dari vektor ciri yang didapat melalui matriks korelasi variabel asal *ρ*. Untuk mencari akar ciri dan menentukan vektor pembobotnya sama seperti pada matriks **Σ**. Sementara *trace* matriks korelasi *ρ* akan sama dengan jumlah *p* variabel yang dipakai.  
Pemilihan komponen utama yang digunakan didasarkan pada nilai akar cirinya, yaitu komponen utama akan digunakan jika akar cirinya lebih besar dari satu.

Algoritma PCA :

1. Menghitung rata-rata vektor .

1. Menghitung matriks normalisasi dengan mengurangi rata-rata vektor Ψ dari setiap X.

1. Menghitung matriks *covariance* dengan mengalikan transpose matriks ΦT dengan matriks baru Φ.

1. Menghitung *eigenvector* dan *eigenvalue.*

**Train/Test and cross validation**

Cross validation adalah metode statistik yang dapat digunakan untuk mengevaluasi kinerja model atau algoritma dimana data dipisahkan menjadi dua subset yaitu data proses pembelajaran dan data validasi / evaluasi. Model atau algoritma dilatih oleh subset pembelajaran dan divalidasi oleh subset validasi. Selanjutnya pemilihan jenis cross validation dapat didasarkan pada ukuran dataset. Biasanya cross validation K-fold digunakan karena dapat mengurangi waktu komputasi dengan tetap menjaga keakuratan estimasi.

Kelebihan dari metode ini adalah tidak adanya masalah dalam pembagian data. Setiap data akan menjadi test set sebanyak satu kali dan akan menjadi training set sebanyak K-1 kali. Kekurangan dari metode ini adalah algoritma pembelajaran harus dilakukan sebanyak K kali yang berarti menggunakan K kali waktu komputasi.

Langkah dalam k fold cross validation adalah :

1. Bagi dataset ke dalam partisi yang sama (atau "lipatan")

Contoh jadi jika k = 5 dan dataset memiliki 150 observasi

Masing-masing dari 5 lipatan akan memiliki 30 pengamatan

1. Gunakan fold 1 sebagai set pengujian dan penyatuan sebagai set pelatihan

Pengujian set = 30 observasi (fold 1)

Training set = 120 observasi (lipatan 2-5)

1. Hitung akurasi pengujian
2. Ulangi langkah 2 dan 3 kali K, menggunakan lipatan yang berbeda saat pengujian ditetapkan setiap kali
3. Gunakan akurasi pengujian rata-rata sebagai perkiraan akurasi di luar sampel

Dalam k-fold cross validation, data awal dibagi secara acak menjadi k subset saling eksklusif (berdiri sendiri) atau disebut dengan fold, dari ukuran yang kirakira sama.

Proses training dan testing dilakukan sebanyak k kali eksperimen. Pada setiap iterasi, dimana satu partisi digunakan sebagai data testing dan memanfaatkan sisa partisi lainnya sebagai data training. Sebagai contoh apabila terdapat subset D1, D2,..., Dk, maka untuk iterasi pertama, subset D1 digunakan sebagai data testing sedangkan sisanya D2, ..., Dk digunakan sebagai data training untuk memperoleh model pertama, begitu juga untuk iterasi kedua, maka D2 digunakan untuk data testing dan sisanya D1, D3, ...,Dk digunakan untuk data training, begitu seterusnya sampai subset terakhir Dk.

Hasil perkiraan akurasi cross validation diperoleh dari jumlah keseluruhan klasifikasi yang benar dari iterasi k, dibagi dengan jumlah total tuple dalam data awal. Secara umum, stratified 10-fold cross validation dianjurkan untuk memperkirakan akurasi (meskipun daya komputasi memungkinkan untuk menggunakan fold yang lebih banyak) dikarenakan bias dan variansi yang relatif rendah.

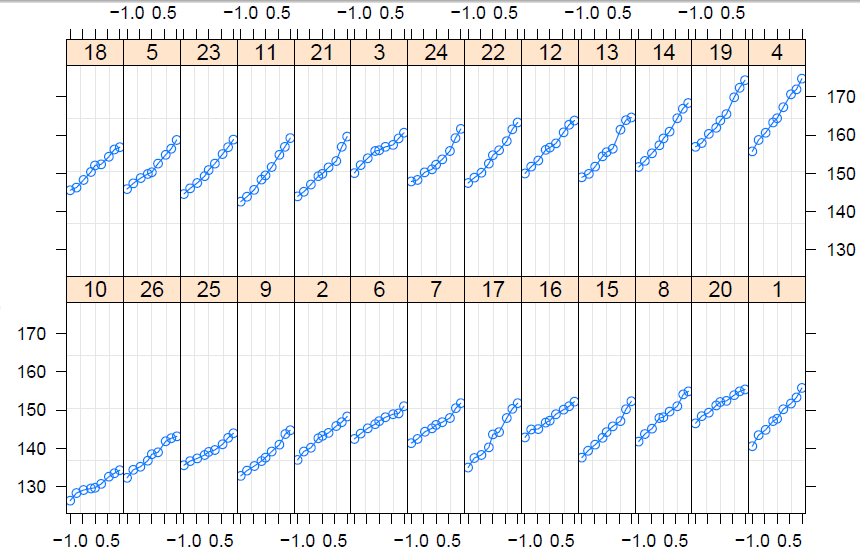
**Multi-Level Models**

Pemodelan multilevel adalah pendekatan yang dapat digunakan untuk menangani data yang dikelompokkan atau dikelompokkan. Pemodelan multilevel juga dapat digunakan untuk menganalisis data pengukuran berulang. Sebagai contoh, jika kita mengukur tekanan darah dari sekelompok pasien pada interval mingguan, kita bisa pikirkan pengukuran berturut-turut sebagai dikelompokkan dalam subyek individu. Satu kelebihan dari pendekatan pemodelan multilevel adalah bahwa ia dapat menangani data di mana waktu pengukuran bervariasi dari satu subjek ke subjek lainnya.

Dalam model multilevel, digunakan variabel acak untuk memodelkan variasi antar kelompok. Sebuah pendekatan alternatif adalah menggunakan model regresi biasa, tetapi untuk memasukkan satu set variabel besar untuk mewakili perbedaan antara kelompok-kelompok. Pendekatan bertingkat memiliki beberapa kelebihan.

1. Dapat menggeneralisasi ke populasi yang lebih luas
2. Lebih sedikit parameter yang dibutuhkan
3. Informasi dapat dibagi antar kelompok

Contoh ide pemodelan multilevel dengan serangkaian pengukuran ulang pengukuran data pola pertumbuhan untuk sampel 26 anak laki-laki - lihat Gambar 1. Ketinggian masing-masing anak diukur pada sembilan ukuran berbeda. Plot untuk masing-masing anak laki-laki berjalan dari kiri bawah ke kanan atas dan disusun dalam urutan tinggi maksimum.



Untuk setiap anak laki-laki, pola pertumbuhan tampaknya linier, jadi dicoba pemodelan dengan model regresi linier sederhana dari :

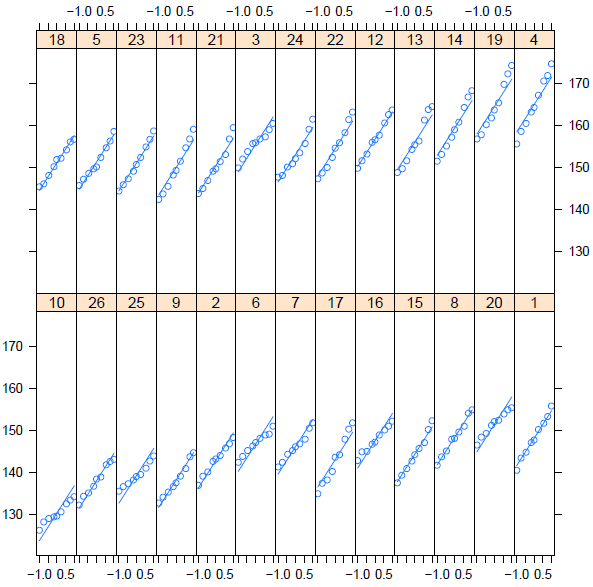
di mana H dan A mewakili tinggi dan usia dan mewakili variasi ketinggian itu

tidak dapat dijelaskan oleh hubungan linear dengan usia

Untuk membuat model lebih realistis, dipakai intercept dalam Model 1 untuk bervariasi dari subjek. Hij untuk pengukuran ke-dua pada subjek ke-j,

Perhatikan bahwa intercept Boj sekarang memiliki subscript j, menunjukkan bahwa ia akan bervariasi pada subjek.

Model 2 memberikan variasi dalam pengukuran individual pada satu subjek, sedangkan Model 3 memberikan variasi dari satu subjek ke yang lain. Kombinasi dari dua model ini memberikan apa yang dikenal sebagai model multilevel



**Reinforcement Learning**

Reinforcement Learning adalah salah satu paradigma baru di dalam learning theory. Reinforcement Learning dibangun dari proses mapping (pemetaan) dari situasi yang ada di environment (states) ke bentuk aksi (behavior) agar dapat memaksimalkan reward. Agent yang bertindak sebagai sang learner tidak perlu diberitahukan behavior apakah yang akan sepatutnya dilakukan, atau dengan kata lain, biarlah sang learner belajar sendiri dari pengalamannya. Ketika ia melakukan sesuatu yang benar berdasarkan rule yang kita tentukan, ia akan mendapatkan reward, dan begitu juga sebaliknya.

Reinforcement Learning secara umum terdiri dari 4 komponen dasar, yaitu :

1. Policy : kebijaksanaan

2. Reward function

3. Value function

4. Model of environment

Policy adalah fungsi untuk membuat keputusan dari agent yang menspesifikasikan tindakan apakah yang mungkin dilakukan dalam berbagai situasi yang ia jumpai. Policy inilah yang bertugas memetakan perceived states ke dalam bentuk aksi. Policy bisa berupa fungsi sederhana, atau lookup table. Policy ini merupakan inti dari RL yang sangat menentukan behavior dari suatu agent.

Reward function mendefinisikan tujuan dari kasus atau problem yang dihadapi. Ia mendefinisikan reward and punishment yang diterima agent saat ia berinteraksi dengan environment. Tujuan utama dari rewar d function ini adalah memaksimalkan total reward pada kurun waktu tertentu setelah agent itu berinteraksi.

Value function menspesifikasikan fungsi akumulasi dari total reward yang didapatkan oleh agent. Jika reward function berbicara pada masing-masing partial time dari prosesinteraksi, value function berbicara pada long-term dari proses interaksi.

Model of environment adalah sesuatu yang menggambarkan behavior dari environment. Model of environment ini sangat berguna untuk mendesain dan merencanakan behavior yang tepat pada situasi mendatang yang memungkinkan sebelum agent sendiri mempunyai pengalaman dengan situasi itu. Saat masa-masa awal RL dikembangkan, model of environment yang ada berupa trial and error. Namun modern RL sekarang sudah mulai menjajaki spektrum dari low -level, trial and error menuju high-level, deliberative planning.

**K-NN**

KNN adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Data pembelajaran diproyeksikan k ruang berdimensi banyak, dimana masing – masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian – bagian berdasarkan klasifikasi data pembelajaran. Sebuah titik pada ruangan ini ditandai dengan kelas c, jika kelas c merupakan klasifikasi yang paling banyak ditemui pada k buah tetangga terdekat titik tersebut.

Prinsip sederhana yang diadopsi oleh algoritma K-NN adalah: “*Jika suatu hewan berjalan seperti bebek, bersuara kwek-kwek seperti bebek, dan penampilannya seperti bebek, maka hewan itu mungkin bebek*”.

Pada algoritma K-NN, data berdimensi *q*, dapat dihitung jarak dari data tersebut ke data yang lain,

* + Nilai jarak ini yang digunakan sebagai nilai kedekatan/kemiripan antara data uji dengan data latih.

z = (x’,y’), adalah data uji dengan vektor x’ dan label kelas y’ yang belum diketahui

Hitung jarak d(x’,x), jarak diantara data uji z ke setiap vektor data latih, simpan dalam D

Pilih Dz ⊆ D, yaitu K tetangga terdekat dari z

KNN di matlab

Class = knnclassify(Sample, Training, Group, k, distance, rule)

K-Nearest Neighbor merupakan metode yang bersifat supervised, dimana hasil dari query instance yang baru diklasifikasikan berdasarkan mayoritas kategori pada KNN.

Pada fase training, algoritma ini hanya melakukan penyimpanan vektor – vektor fitur dan klasifikasi data training sample. Pada fase klasifikasi, fitur – fitur yang sama dihitung untuk testing data (klasifikasinya belum diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor training sample dihitung, dan sejumlah k buah yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari titik – titik tersebut.

**Bias/Variance Tradeoff**

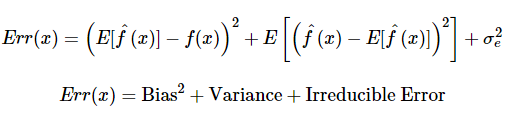
Penggolongan nonlinier lebih kuat daripada pengklasifikasi linier. Untuk beberapa masalah, terdapat classifier nonlinear dengan kesalahan klasifikasi nol, tetapi tidak ada classifier linier seperti itu.

Tradeoff membantu menjelaskan mengapa tidak ada metode pembelajaran yang optimal secara universal. Pertama, banyak model nonlinier memasukkan model linear sebagai kasus khusus. Misalnya, metode pembelajaran nonlinier seperti kNN akan dalam beberapa kasus menghasilkan classifier linier. Kedua, ada model nonlinier yang kurang kompleks daripada model linier. Misalnya, polinomial kuadrat dengan dua parameter kurang kuat daripada classifier linier 10.000 dimensi. Ketiga, kompleksitas pembelajaran bukanlah benar-benar milik penggolong karena ada banyak aspek pembelajaran yang membuat metode pembelajaran lebih baik kuat atau kurang kuat tanpa mempengaruhi jenis penggolong yang merupakan hasil akhir pembelajaran - terlepas dari linear atau tidak linier. Pengklasifikasi linier dan nonlinier hanya akan berfungsi sebagai proksi untuk metode pembelajaran yang lebih lemah dan lebih kuat dalam klasifikasi.

Bias adalah perbedaan kuadrat antara probabilitas kondisional yang sebenarnya dari d berada di c, dan ΓD (d) prediksi dari classifier yang dipelajari, rata-rata lebih dari set pelatihan. Bias besar jika metode pembelajaran menghasilkan pengklasifikasi yang secara konsisten salah. Bias kecil jika (i) pengklasifikasi secara konsisten benar atau (ii) perangkat pelatihan yang berbeda menyebabkan kesalahan pada dokumen yang berbeda atau (iii) perangkat pelatihan yang berbeda menyebabkan kesalahan positif dan negatif pada dokumen yang sama, tetapi rata-rata itu mendekati 0. Jika salah satu dari ketiga kondisi ini berlaku, maka ED ΓD(d) mendekati

Jika kita menunjukkan variabel yang kita coba prediksi sebagai YY dan kovariat kita sebagai XX, kita dapat mengasumsikan bahwa ada hubungan yang berkaitan satu sama lain seperti Y = f (X) + ϵY = f (X) + ϵ di mana error term ϵϵ terdistribusi normal dengan rata-rata nol seperti jadi ϵ∼N (0, σϵ) ϵ∼N (0, σϵ).

Dapat diperkirakan model f ^ (X) f dari f (X) f (X) menggunakan regresi linier atau teknik pemodelan lainnya. Dalam hal ini, kesalahan prediksi kuadrat yang diharapkan pada titik x adalah:



**Term Frequency / Inverse Document Frequency**

Algoritma Term Frequency Inverse-Document Frequency merupakan suatu algoritma yang menggalikan antara Term frequency dengan Inverse Document Frequency. Term frequency yaitu jumlah kemunculan sebuah term pada sebuah dokumen. Inverse Document Frequency yaitu pengurangan dominasi term yang sering muncul diberbagai dokumen, dengan memperhitungkan kebalikan frekuensi dokumen yang mengandung suatu kata.

Metode TF-IDF merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada information retrieval. Metode ini akan menghitung nilai Term Frequency (TF) dan Inverse Document Frequency (IDF) pada setiap token (kata) di setiap dokumen dalam korpus. Metode ini akan menghitung bobot setiap token t di dokumen d dengan rumus:

Wdt = tfdt \* IDFt

Dimana :

d : dokumen ke-d

t : kata ke-t dari kata kunci

W : bobot dokumen ke-d terhadap kata ke-t

tf : banyaknya kata yang dicari pada sebuah dokumen

IDF : Inversed Document Frequency

Nilai IDF didapatkan dari IDF : log2 (D/df) dimana :

D : total dokumen

df : banyak dokumen yang mengandung kata yang dicari

Setelah bobot (W) masing-masing dokumen diketahui, maka dilakukan proses pengurutan dimana semakin besar nilai W, semakin besar tingkat similaritas dokumen tersebut terhadap kata kunci, demikian sebaliknya.

**K-Means Clustering**

Data Clustering merupakan salah satu metode Data Mining yang bersifat tanpa arahan (unsupervised). Ada dua jenis data clustering yang sering dipergunakan dalam proses pengelompokan data yaitu hierarchical (hirarki) data clustering dan non-hierarchical (non hirarki) data clustering. K-Means merupakan salah satu metode data clustering non hirarki yang berusaha mempartisi data yang ada ke dalam bentuk satu atau lebih cluster/kelompok. Metode ini mempartisi data ke dalam cluster/kelompok sehingga data yang memiliki karakteristik yang sama dikelompokkan ke dalam satu cluster yang sama dan data yang mempunyai karakteristik yang berbeda dikelompokkan ke dalam kelompok yang lain. Adapun tujuan dari data clustering ini adalah untuk meminimalisasikan objective function yang diset dalam proses clustering, yang pada umumnya berusaha meminimalisasikan variasi di dalam suatu cluster dan memaksimalisasikan variasi antar cluster.

Algoritma pada K-Means

1. Tentukan jumlah cluster

2. Alokasikan data ke dalam cluster secara random

3. Hitung centroid/rata-rata dari data yang ada di masing-masing cluster

4. Alokasikan masing-masing data ke centroid/rata-rata terdekat

5. Kembali ke Step 3, apabila masih ada data yang berpindah cluster atau apabila perubahan nilai centroid, ada yang di atas nilai threshold yang ditentukan atau apabila perubahan nilai pada objective function yang digunakan di atas nilai threshold yang ditentukan.

Beberapa alternatif penerapan K-Means dengan beberapa pengembangan teori-teori penghitungan terkait telah diusulkan. Hal ini termasuk pemilihan:

1.Distance space untuk menghitung jarak di antara suatu data dan centroid

Beberapa distance space telah diimplementasikan dalam menghitung jarak (distance) antara data dan centroid termasuk di antaranya L1 (Manhattan/City Block) distance space, L2 (Euclidean) distance space, dan Lp (Minkowski) distance space.

2. Metode pengalokasian data kembali ke dalam setiap cluster

Secara mendasar, ada dua cara pengalokasian data kembali ke dalam masing-masing cluster pada saat proses iterasi clustering. Kedua cara tersebut adalah pengalokasian dengan cara tegas (hard), dimana data item secara tegas dinyatakan sebagai anggota cluster yang satu dan tidak menjadi anggota cluster lainnya, dan dengan cara fuzzy, dimana masing-masing data item diberikan nilai kemungkinan untuk bisa bergabung ke setiap cluster yang ada. Kedua cara pengalokasian tersebut diakomodasikan pada dua metode Hard K-Means dan Fuzzy K-Means. Perbedaan di antara kedua metode ini terletak pada asumsi yang dipakai sebagai dasar pengalokasian.

Pengalokasian kembali data ke dalam masing-masing cluster dalam metode Hard K-Means didasarkan pada perbandingan jarak antara data dengan centroid setiap cluster yang ada. Data dialokasikan ulang secara tegas ke cluster yang mempunyai centroid terdekat dengan data tersebut.

Metode Fuzzy K-Means (atau lebih sering disebut sebagai Fuzzy C-Means) mengalokasikan kembali data ke dalam masing-masing cluster dengan memanfaatkan teori Fuzzy. Teori ini mengeneralisasikan metode pengalokasian yang bersifat tegas (hard) seperti yang digunakan pada metode Hard K-Means. Dalam metode Fuzzy K-Means dipergunakan variabel membership function, ik u , yang merujuk pada seberapa besar kemungkinan suatu data bisa menjadi anggota ke dalam suatu cluster.

3.Objective function yang digunakan

Objective function yang digunakan khususnya untuk Hard K-Means dan Fuzzy K-Means ditentukan berdasarkan pada pendekatan yang digunakan dalam poin Distance space untuk menghitung jarak di antara suatu data dan centroid dan poin Metode pengalokasian data kembali ke dalam setiap cluster

Beberapa permasalahan yang sering muncul pada saat menggunakan metode K-Means untuk melakukan pengelompokan data adalah:

1. Ditemukannya beberapa model clustering yang berbeda

2. Pemilihan jumlah cluster yang paling tepat

3. Kegagalan untuk converge

4. Pendeteksian outliers

5. Bentuk masing-masing cluster

6. Masalah overlapping

K-Means ini dapat disebut sebagai metode semi-supervised classification, karena metode ini mengalokasikan data items ke masing-masing cluster secara unsupervised dan menentukan jumlah cluster yang paling sesuai dengan data yangdianalisa secara supervised.

Untuk keperluan seperti itu, beberapa peneliti telah mengusulkan pengembangan metode K-Means yang secara khusus memanfaatkan kernel trik, dimana data spac untuk data awal di-mapping ke feature space yang berdimensi tinggi. Beberapa hal yang perlu diperhatikan dalam pengembangan metode K-Means dengan kernel trik ini adalah bahwa data pada feature space tidak lagi dapat didefinisikan secara eksplisit, sehingga penghitungan nilai membership function dan centroid tidak dapat dilakukan secara langsung.

**Gradient Boosting algorithms GBM, XGBoost, LightGBM, CatBoost**

Gradient descent (ascent) adalah algoritma optimasi orde pertama. Untuk menemukan minimum lokal dari fungsi menggunakan gradien descent, diambil langkah sebanding dengan negatif dari gradien (atau perkiraan gradien) dari fungsi pada titik sekarang.

Jika diambil langkah sebanding dengan gradien positif, maka akan didapatkan maksimum lokal fungsi tersebut; prosedur ini kemudian dikenal sebagai gradient ascent

Gradient descent juga dikenal sebagai steepest descent, sedangkan gradient ascent dikenal dengan steepest ascent.Mencari maksimum suatu fungsi sama dengan menemukan minimumnya, hanya dengan mengalikan nilai tersebut dengan negatif satu atau dengan kata lain maksimum adalah negatifnya dari minimumnya suatu fungsi.

Sebagai contoh andaikan fungsi tujuan kita adalah

Secara umum f(x) dapat berupa fungsi nonlinear. Metode-metode descent adalah metode iteratif untuk memperoleh solusi pendekatan dari problem awal. Misalkan diberikan suatu titik awal x(1) maka kita mencari x(2), sehingga pergerakan xk pada setiap iterasi haruslah memenuhi dimana pergerakan dari x(k) ke x(k+1) dapat dituliskan sebagai berikut :

dengan menyatakan langkah (step) pada setiap iterasi dan sk adalah arah (direction). Kita bermaksud membuat xk konvergen ke solusi optimal x\*.

Pencarian arah adalah hal yang penting dalam metode-metode descent, dan pencarian ini yang membedakan antara algoritma descent yang satu dengan lainnya. Begitu pencarian arah telah ditentukan maka tahap selanjutnya adalah melakukan pencarian langkah (line search, step search), yaitu

Bila pencarian arah ditentukan berdasarkan gradient, maka metode descent ini disebut metode descent berbasis gradient (gradient based descent methods).

Algoritma Maksimal :

Mulai dari titik awal v0

Bergerak dari v0 ke v1 dengan arah f(v0) :

v1 = v0 + t0 f(v0)

dengan t0 adalah solusi dari masalah optimisasi

berikut:

max f(v0 + t0 f(v0) )

s.t t0 ≥ 0

Langkah – langkah tersebut diulangi sampai didapat nilai vi dan vi+1 yang cukup dekat

Algoritma Minimum :

Mulai dari titik awal v0

Bergerak dari v0 ke v1 dengan arah f(v0) :

v1 = v0 - t0 f(v0)

dengan t0 adalah solusi dari masalah optimisasi

berikut:

min f(v0 - t0 f(v0) )

s.t t0 ≥ 0

Langkah – langkah tersebut diulangi sampai didapat nilai vi dan vi+1 yang cukup dekat

**Unsupervised ML:**

**Clustering**

Algoritma clustering merupakan algoritma pengelompokkan sejumlah data ( N ) menjadi kelompok – kelompok data tertentu ( cluster ). Objek data yang terletak didalam satu cluster harus mempunyai kemiripan. Sedangkan yang tidak berada didalam satu cluster tidak mempunyai kemiripan.

Jumlah kemungkinan peng-clusteran – an. Misalnya, data X dimana :

X = {x1,x2,……….,xn}

Rumus yang digunakan untuk menentukan jumlah cluster adalah :

S(N,m) = (-1)m-1 iN

Clustering adalah proses pengelompokan kumpulan data menjadi beberapa kelompok sehingga objek di dalam satu kelompok memiliki banyak kesamaan dan memiliki banyak perbedaan dengan objek di kelompok lain. Perbedaan dan persamaannya biasanya berdasarkan nilai atribut dari objek tersebut dan dapat juga berupa perhitungan jarak. Clustering sendiri juga disebut unsupervised classification, karena clustering lebih bersifat untuk dipelajari. Cluster analysis merupakan proses partisi satu set objek data ke dalam himpunan bagian. Setiap himpunan bagian adalah cluster, sehingga objek yang ada di dalam cluster mirip satu sama dengan lainnya, dan mempunyai perbedaan dengan objek dari cluster yang lain. Partisi tidak dilakukan dengan manual algoritma clustering. Oleh karena itu, clustering sangat berguna dan bisa menemukan grup yang tidak dikenal dalam data.

Konsep Dasar Clustering

Proses clustering akan menghasilkan cluster yang baik apabila:

a) Tingkat kesamaan yang tinggi dalam satu kelas.

b) Tingkat kesamaan yang rendah antar kelas.

Kesamaan yang dimaksud merupakan pengukuran secara numerik terhadap dua buah objek. Nilai kesamaan ini akan semakin tinggi apabila memiliki kemiripan yang tinggi. Perbedaan kualitas hasil clustering tergantung pada metode yang dipakai.

Tipe data pada clustering:

a) Variabel berskala interval.

b) Variabel biner.

c) Variabel nominal, ordinal, dan rasio.

d) Variabel dengan tipe lainnya.

Metode clustering juga harus dapat mengukur kemampuannya dalam usaha untuk menemukan suatu pola tersembunyi pada data yang tersedia.

Penjelasan Matlab :

Algoritma kluster mengatur satu set objek N ke dalam grafik langsung yang mengungkap kelompok objek. Secara khusus, algoritma memilih objek induk untuk setiap objek. “Parent” objek juga merupakan salah satu objek di set dan mungkin objek itu sendiri. “Parent” ini dapat dinyatakan sebagai pemetaan P (n) dari indeks objek ke dalam diri mereka sendiri. Sebuah objek itu “Parent” sendiri adalah yatim piatu dan menjadi akar dari sebuah kelompok. “Parent” dari suatu objek dua properti:Ini adalah tetangga dari objek, yaitu jarak antara objek dan induknya kurang dari atau sama dengan nilai tertentu r.

Jadi memiliki lebih banyak tetangga daripada tetangga lain dari objek (yaitu, itu adalah tetangga dengan sebagian besar tetangga) atau setidaknya memiliki banyak tetangga seperti tetangga lainnya dan indeks tertinggi di antara tetangga dengan sebagian besar tetangga.

Set data iris terdiri dari data yang diukur dari 150 tanaman iris; 50 masing-masing setosa, versicolor dan virginika. Ada empat ukuran untuk setiap tanaman: panjang dan lebar sepal dan panjang dan lebar kelopak. Kolom ke-f dari kumpulan data menunjukkan tipe iris sebagai 1, 2 atau 3. Informasi ini tidak disalin ke matriks X.

Dalam contoh ini radius r diatur ke 1 = 10 jarak antar-objek maksimum menggunakan jarak Euclidean tanpa normalisasi.

**Hierarchical Clustering**

Hierarchical clustering adalah metode clustering yang mengelompokkan data dengan urutan partisi berkala. Metode ini dikelompokkan menjadi dua metode yaitu agglomerative dan divisive, metode agglomerative berawal dari obyek-obyek individual dimana pada awalnya banyaknya cluster sama dengan banyaknya obyek. Pertama-tama obyek-obyek yang paling mirip dikelompokkan, dan kelompok-kelompok awal ini digabungkan sesuai dengan kemiripannya. Akhirnya sewaktu kemiripan berkurang, semua subkelompok digabungkan menjadi satu cluster tunggal. Sementara Metode Hierarchical divisive merupakan proses kebalikan dari agglomerative, keduanya mengorganisasi data ke dalam struktur hirarki berbasis matrix proximity, hasil dari dari Hierarcichal Clustering digambarkan dalam bentuk binary tree ataupun dendogram, root merupakan keseluruhan dataset dan tiap cabang merupakan data point, clustering akhir dapat diperoleh dari pemotongan dendogram pada level-level yang sesuai.

Berbeda dengan klastering hirarki yang menghasilkan suatu tingkatan berurutan klaster dengan cara penggabungan secara iterative atau pemisahan, partitional clustering mengelompokkan datapoint kedalam k klaster tanpa struktur hirarki. Metode ini membagi set data ke dalam sejumlah kelompok yang tidak saling overlap antara satu kelompok dengan kelompok lainnya, artinya setiap data hanya menjadi satu kelompok, termasuk dalam metode ini adalah K-Means dan DBSCAN.

Pengelompokan data berdasarkan hierarkinya. Langkah-langkah melakukan hierarchical clustering:

a) Identifikasi item dengan jarak terdekat.

b) Gabungkan item itu ke dalam satu cluster.

c) Hitung jarak antar cluster.

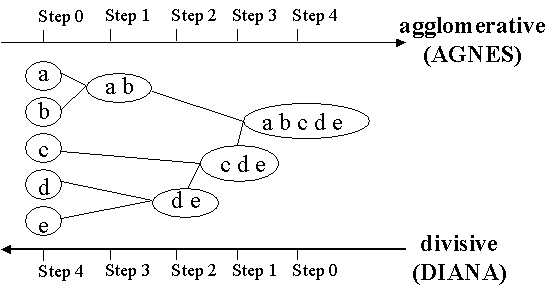
d) Ulangi dari awal, sampai semua terhubung.

Dari teknik hierarchical clustering, dapat dihasilkan suatu kumpulan partisi yang berurutan, dimana dalam kumpulan tersebut terdapat:

a.Cluster – cluster yang mempunyai poin – poin individu. Cluster – cluster ini berada di level yang paling bawah.

b.Sebuah cluster yang didalamnya terdapat poin – poin yang dipunyai semua cluster didalamnya. Single cluster ini berada di level yang paling atas.

Hasil keseluruhan dari algoritma hierarchical clustering secara grafik dapat digambarkan sebagai tree, yang disebut dengan dendogram. Tree ini secara grafik menggambarkan proses penggabungan dari cluster – cluster yang ada, sehingga menghasilkan cluster dengan level yang lebih tinggi



Gambar. Dendogram

**Dimension Reduction**

Konsep ini mengacu pada proses pengurangan jumlah variabel acak dipertimbangkan dan dapat dibagi lagi dalam pemilihan fitur dan ekstraksi fitur. Kuncinya adalah untuk mengurangi jumlah dimensi, tetapi tetap mempertahankan sebagian besar informasi. Ketidaknyamanan yang terkait dengan analisis data multivariat ketika dimensi, itulah jumlah variabel terlalu besar.

Ada beberapa algoritma yang ditawarkan untuk mereduksi dimensi dari sebuah gambar. Algoritma tersebut antara lain:

A. Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) telah berhasil digunakan sebagai teknik mereduksi dimensi untuk banyak masalah, seperti speech recognition, face recognition, dan pengambilan informasi multimedia. LDA mereduksi gambar dengan cara mencari matriks W untuk memaksimalkan nilai perbandingan antara matriks penyebaran antar kelas (Sb) dan matriks penyebaran dalam kelas (Sw).

B. Principal Component Analysis

Principal Component Analysis merupakan metode statistika yang digunakan untuk mereduksi / mengurangi dimensi input tanpa kehilangan informasi penting dari input. Dalam PCA, seluruh dataset yang dimiliki akan diproyeksikan kedalam subspace yang berbeda untuk mendapatkan arah (component) yang memaksimalkan variance dari dataset yang dimiliki.

Secara umum penyelesaian PCA dilakukan dalam 4 tahap yaitu :

1. Input data (m x n), dimana m : jumlah pixel dan n : jumlah citra atau sampel.

2. Preprocessing PCA dengan melakukan standarisasi dan mencari covariance atau correlation matrix.

3. Proses PCA dengan menggunakan EVD (Eigen Value Decomposition) atau SVD (Singular Value Decomposition).

4. Output yang merupakan data hasil transformasi (m x k), dimana m : jumlah pixel dan k : jumlah principal component.

Tiga dimensi utama dari data set :

1. Kolom (Fitur)
2. Baris (kasus/contoh)
3. Nilai fitur

Tiga operasi dasar dari reduksi data :

1. Delete kolom
2. Delete baris
3. Pengurangan jumlah nilai kolom

Pendekatan yang digunakan yaitu mereduksi dengan menggantikan sekumpulan fitur awal dengan fitur campuran yang baru dimana merupakan proporsi hasil bagi dari dua fitur awal.

Parameter analisis dasarnya berdasarkan :

1. Waktu komputasi : data yang lebih sederhana dapat mereduksi waktu untuk proses data mining.
2. Keakuratan prediksi : mengukur seberapa baik data dapat disimpulkan dan mengeneralisasi ke dalam suatu model.
3. Penyajian dari model data mining : menjelaskan kesederhanaan representasi menjadi model yang dapat dimengerti.

**Feature Selection and Transformation**

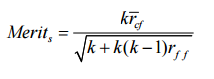
Seleksi fitur adalah salah teknik terpenting dan sering digunakan dalam pre-processing data mining, khususnya untuk knowledge discovery maupun discovery science. Teknik ini mengurangi jumlah fitur yang terlibat dalam menentukan suatu nilai kelas target, mengurangi fitur irelevan, berlebihan dan data yang menyebabkan salah pengertian terhadap kelas target yang membuat efek segera bagi aplikasi. Hasilnya, aplikasi data mining bisa dipercepat, mempertinggi kinerja mining seperti akurasi peramalan. Seleksi atau seleksi fitur merupakan proses yang memilih subset dari ukuran fitur asalnya. Tugas utama dalam seleksi fitur adalah menentukan fitur mana yang dipilih dan dipakai dalam rangka peramalan atribut atribut fitur. Fitur dianggap relevan bila nilainya bervariasi secara sistematis dengan keanggotaan kategori. Dengan demikian algoritma seleksi fitur mempunyai peran kritis dalam banyak aplikasi machine learning, CRM, data mining secara umum dan tentu saja dalam analisis genomic. Dengan sedemikian luasnya bidang aplikasi yang memerlukan keunggulan algoritma seleksi fitur, sangat penting untuk membuat kerangka evaluasi peringkat (rank) berbasiskan pada kesamaan (similarity) antara klasifier yang dipakai sebagai inti algoritma seleksi fitur.

Pada dasarnya subset yang akan didapat dari seleksi fitur merupakan urutan berperingkat dari semua fitur yang dipilih oleh masing masing algoritma. Terdapat dua pendekatan dalam proses pemilihan fitur, yaitu pendekatan filter dan pendekatan wrapper. Pendekatan filter dapat didefinisikan, dimana metrik digunakan pada kriteria lokal atau secara independen dan berbeda dengan model target. Sedangkan pendekatan wrapper didefinisikan, dimana metrik ditentukan oleh performa atau akurasi dari model target. Penggunaan metode dalam pemilihan fitur sangat berpengaruh terhadap akurasi klasifikasi. Metode wrapper secara sistem sangat mahal dari segi komputasi karena metode ini menggunakan learning algorithm didalam melakukan pencarian fitur yang paling memenuhi kriteria (fungsi objektif) terhadap target. Pendekatan filter tidak memerlukan waktu dan biaya yang lama dalam komputasinya, sehingga pada penelitian ini sangat efisien jika menggunakan pendekatan filter.

Pendekatan wrapper akan melakukan feedback dalam proses pemilihan fitur, ini terjadi karena algoritma pembelajaran yang digunakan harus mengevaluasi kembali subset fitur yang terpilih, apakah sudah fit atau belum. Sedangkan dalam pendekatan filter, fitur subset akan dipilih langsung secara sekuensial, korelasi ataupun dengan metode lain tanpa ada proses evaluasi fitur yang dipilih. sistematis dengan keanggotaan kategori kelas target.

Teknik seleksi fitur yang dipakai dalam contoh matlab adalah metode ILFS, InfF, ECFS, mrmr, relieff, mutinffs, fsv, laplacian, mcfs, rfe, L0, fisher, UDFS, llcfs, cfs Correlation-based Feature.

Sebagai inti dalam CFS adalah teknik heuristic untuk mengevaluasi nilai atau harga subset fitur. Teknik ini mempertimbangkan kegunaan fitur individual bagi prakiraan label kelas dengan level interkorelasi di antara fitur-fitur. Fitur secara individual menguji mana ukuran yang berkaitan dengan variable yang diamati (sebagai kelas target). Persamaan berikut adalah formalisasi nilai harga heuristic yang dimaksud:



Dimana Merits merupakan harga heuristic subset fitur S yang berisi k fitur rcf yang merupakan rata-rata korelasi fitur-kelas, rff adalah rata-rata interkorelasi fitur ke fitur. Pada kenyataannya semua variable distandardisasi sesuai rumus korelasi Pearson. Numerator dianggap telah dipahami sebagai indikasi bagaimana sifat prediksi suatu fitur kelompok, sedangkan denominator menunjukkan bagaimana redundansi data antara fitur

**Artificial Intelligence:**

**Natural Language Processing**

“Natural Language Processing (NLP) atau Pengolahan Bahasa Alami (PBA) merupakan salah satu bidang ilmu kecerdasan buatan (Artificial Intelligence) yang mempelajari komunikasi antara manusia dengan komputer”. Pengolahan Bahasa Alami tidak bertujuan untuk mentransformasikan bahasa yang diterima dalam bentuk suara menjadi data digital dan/atau sebaliknya pula; melainkan bertujuan untuk memahami arti dari teks/tulisan yang diberikan dalam format bahasa alami dan memberikan respon yang sesuai, misalnya dengan melakukan suatu aksi tertentu atau menampilkan data tertentu.

Pemrosesan bahasa alami (NLP) mengacu pada kelas luas teknik komputasi untuk memasukkan data ucapan dan teks, bersama dengan jenis data rekayasa lainnya, ke dalam pengembangan sistem cerdas. Data bahasa manusia mentah dapat berasal dari berbagai sumber, termasuk sinyal audio, web dan media sosial, dokumen dan database yang berisi informasi berharga seperti perintah suara, sentimen publik pada topik, data operasional, dan laporan pemeliharaan. Pemrosesan bahasa alami dapat digunakan untuk menggabungkan dan menyederhanakan sumber data besar ini, mengubahnya menjadi wawasan yang bermakna dengan visualisasi, model topik, dan pengklasifikasi pembelajaran mesin.

Pemrosesan bahasa alami digunakan dalam keuangan, manufaktur, elektronik, perangkat lunak, teknologi informasi, dan industri lain untuk aplikasi seperti:

Mengotomasi pengklasifikasian ulasan berdasarkan sentimen, baik positif atau negatif

Menghitung frekuensi kata atau frasa dalam dokumen dan melakukan pemodelan topik

Mengembangkan jadwal pemeliharaan peralatan prediktif berdasarkan data sensor dan log teks

Mengotomasi pelabelan dan penandaan rekaman ucapan

Natural Language Processing dalam recurrent neural network memiliki Sebuah jaringan saraf berulang/ RNN yang tidak mengerti teks, dan karenanya setiap kata dalam teks harus memiliki beberapa bentuk angka perwakilan. Vektor word-embeddings adalah pilihan yang tepat karena kata-kata dapat diwakili oleh banyak konsep yang diberikan oleh komponen vektor kata-embeddings. Jaringan syaraf rekuren dapat dibuat untuk bekerja dua arah, baik dengan menyediakan vektor kata-embeddings sebagai input atau dengan membiarkan jaringan belajar vektor-vektor embeddings itu sendiri. Dalam kasus terakhir, vektor kata-embeddings akan lebih selaras menuju masalah utama yang dipecahkan melalui jaringan saraf berulang RNN. Namun, pada saat itu jaringan saraf berulang mungkin memiliki banyak parameter lain untuk dipelajari, atau jaringan mungkin memiliki sangat sedikit data untuk dilatih. Dalam kasus seperti itu, harus mempelajari vektor kata-embeddings sebagai parameter mungkin mengarah ke overfitting atau hasil yang optimal. Menggunakan embedding kata-kata yang telah dilatih sebelumnya mungkin merupakan pilihan yang lebih baik.

**Multi-Agent System**

Sistem multi agent dibangun atas 3 kategori yang umum, yaitu, provider agent, service requester agent, dan middle agent. Provider agent penyedia layanan, seperti pencarian informasi, mememcahkan masalah yang spesifik. Requester agent adalah agent yang membutuhkan layanan tersebut. Sedangkan Middle Agent digunakan untuk mencari kecocokan antar agent. Proses pencocokan yang dilakukan oleh Middle agent ini disebut matchmaking.

Dalam pendistribusian job dapat saja terjadi ketidakseimbangan workload diantara agen. Dalam menangani masalah ketidak seimbangan sistem ada dua cara teknik. Metode load balancing pada saat agen mengalami overload sehingga agen tersebut harus dipindahkan ke komputer lain untuk mengurangi workload. Dengan hanya melakukan migrasi agen dari suatu komputer ke komputer lain masih memungkinkan terjadinya overload karena adanya penambahan agen dan task pada komputer tujuan sehingga dapat mengakibatkan terjadi proses reload balancing pada sistem yang pada akhirnya akan membuat sistem mengalami overload dan menjadi lambat. Teknik lain yaitu mengalokasikan resource komputasi berdasarkan kebutuhan komputasi agen. Metode load balancing yang diusulkan membutuhkan waktu preprocessing yang lama karena harus mengestimasi terlebih dahulu waktu penyelesaian job.

Sebuah skema load balancing dimana sistem melakukan alokasi job keseluruh agent worker pada setiap komputer secara dinamis. Sistem multi agen terdiri dari agent monitor dan agent worker. Agent worker bertugas sebagai pekerja yang melakukan proses eksekusi job. Agent monitor bertanggung jawab dalam mengawasi kondisi load agent worker. Load balancing pengalokasian job ditentukan berdasarkan pertimbangan kondisi load agent worker, antrian job dan resource komputasi komputer dimana agent worker berada.

Dalam menperkirakan kondisi agent worker, agent monitor akan mengirimkan sebuah pesan ACL secara periodik ke setiap agent worker yang berada pada setiap komputer. Agent monitor akan segera mencatat waktu forwarding F(t) pesan dari agent monitor ke agent worker dan waktu receiving R(t) yaitu pesan balasan dari agent worker. Agent monitor akan mengkalkulasi nilai round trip time (RTT) pesan berdasarkan persamaan berikut ini:

RTT=receiving time R(t) – forwarding time F(t)

Nilai Round Trip Time (RTT) dari setiap agent worker digunakan untuk menentukan kondisi agent worker. Nilai RTT yang didapatkan oleh agent monitor akan mendeskripsikan kondisi load agent worker yang juga merupakan kondisi penggunaaan daya komputasi oleh agent worker saat melakukan proses enkripsi job.

Dalam menperkirakan antrian job pada agent worker, Agent monitor akan mengirimkan sebuah pesan ACL (Agent Communication Language) secara periodik ke setiap agent worker yang berada pada setiap komputer. Pesan ACL akan dikirimkan ke seluruh agent worker dalam rentang waktu lima detik. Agent worker akan mencatat secara real time antrian job yang sedang terjadi.

Pada saat agent worker menerima pesan dari agent monitor maka jumlah antrian job yang sedang terjadi pada saat itu akan dimasukkan kedalam pesan ACL dan selanjutnya pesan tersebut dikirimkan agent monitor. Agent monitor akan menerima pesan ACL dari setiap agent worker yang berisi jumlah antrian job yang sedang terjadi pada masing-masing agent worker. Dalam penelitian ini antrian job yang terjadi pada agent worker diklasifikasikan kedalam tiga kelompok yaitu: rendah, sedang, dan padat.

**Automated Planning**

Perencanaan dan penjadwalan otomatis adalah cabang kecerdasan buatan yang menyangkut realisasi strategi atau urutan tindakan, biasanya untuk eksekusi oleh agen cerdas, robot otonom dan kendaraan tanpa awak. Tidak seperti kontrol klasik dan masalah klasifikasi, solusinya kompleks dan harus ditemukan dan dioptimalkan dalam ruang multidimensi. Perencanaan juga terkait dengan teori keputusan.

Di lingkungan yang dikenal dengan model yang tersedia, perencanaan dapat dilakukan secara offline. Solusi dapat ditemukan dan dievaluasi sebelum pelaksanaan. Dalam lingkungan yang tidak diketahui secara dinamis, strategi sering perlu direvisi secara online. Model dan kebijakan harus disesuaikan. Solusi biasanya menggunakan proses trial and error berulang yang biasa terlihat pada kecerdasan buatan. Ini termasuk pemrograman dinamis, pembelajaran penguatan dan optimasi kombinatorial. Bahasa yang digunakan untuk menggambarkan perencanaan dan penjadwalan sering disebut bahasa aksi.

Sistem transisi state (atau domain perencanaan klasik)

Ø Σ = (S, A, γ, biaya); biaya adalah opsional

• S - himpunan status terbatas yang mungkin ada dalam sistem

• A - serangkaian tindakan terbatas: hal-hal yang dapat dilakukan agen

• γ: S x A → S - fungsi prediksi (atau fungsi transisi-state)

▸ fungsi parsial: γ (s, a) tidak ditentukan kecuali berlaku dalam s

▸ Dom (a) = {s∈S | γ (s, a) didefinisikan} = {s ∈ S | a berlaku}

▸ Rentang (a) = {γ (s, a) | s∈Dom (a)}

biaya: S x A → R + -atau biaya: A → R +

▸ dapat berupa biaya moneter, waktu yang diperlukan, hal lain

▸ sering dihilangkan dari Σ; default adalah biaya (a) ≡ 1

Bidang perencanaan AI dimulai sebagai pembuktian teorema otomatis dan masih banyak menggunakan notasi. Representasi klasik setara dengan representasi status-variabel dan mewakili sifat yang kaku dan bervariasi menggunakan predikat logis

• adjacent (l, m) - lokasi l berbatasan dengan lokasi m

• loc (r, l) - robot r berada di lokasi l

• loc (c, l), loc (c, r) - container c berada di lokasi l atau pada robot r

• cargo (r) - ada wadah di robot r

Ide dasar dari Planning with control rules diberikan keadaan dan tindakan a, kemudian lakukan pengujian domain khusus pada γ (s, a) untuk menemukan kasus. Dimana a tidak mengarah ke solusi, a didominasi (ada solusi yang lebih baik di sepanjang jalur lainnya) dan a tidak mengarah ke solusi yang dapat diterima menurut domain tertentu.

**Fuzzy Systems**

Pengajuan yang diberikan adalah MATLAB yang dikhususkan untuk optimasi genetika dari sistem inferensi fuzzy tipe-Mamdani (FIS) dan memiliki batasan / pembatasan sebagai berikut:

 FIS harus memiliki satu output.

 Hanya fungsi keanggotaan segitiga yang harus digunakan untuk menentukan input dari FIS.

 Proses optimasi hanya menyetel masukan dari FIS.

Toolbox menggunakan fungsi aplikasi Optimasi, dibangun di MATLAB, dan terdiri dari 6 skrip dan 3 fungsi, yang deskripsi singkatnya dapat ditemukan di dalam kode. Toolbox dapat digunakan dengan menjalankan hanya 2 skrip utama: "GA\_FIS\_Tuning" dan "FIS\_Genetic\_Design".

Skrip "GA\_FIS\_Tuning" digunakan untuk mendapatkan parameter yang disetel dari FIS pada beberapa set data tuning. Skrip "FIS\_Genetic\_Design" digunakan untuk memodifikasi FIS awal dengan menerapkan parameter yang dicari.

FIS yang akan dioptimalkan dalam bagian ini adalah FIS "skindetect" dan mampu mengenali piksel manusia-kulit. FIS yang dianggap memiliki 3 input kontinu (R, G, B) dengan kisaran [0,255] dan output kategoris (Kelas) dengan dua kemungkinan nilai (Skin, Non-Skin), dan basis aturannya berisi 147 aturan.

Dataset yang digunakan untuk mengoptimalkan "skindetect" FIS adalah "Skin Segmentation" dataset. Dataset adalah satu set piksel RGB milik salah satu dari 2 kategori yang mungkin: warna kulit dan warna non-kulit. Dimensi dari dataset adalah 245057 × 4.

Seluruh proses optimasi dapat dianggap sebagai urutan dari mengikuti dua langkah.

Langkah 1. Mengeksekusi skrip “GA\_FIS\_Tuning” :

1. Pilih FIS "skindetect" yang akan dioptimalkan.

2. Satu dapat meningkatkan jangkauan atribut input dari FIS: misalnya, di kisaran dari R, G, B atribut meningkat 10, yang berarti bahwa rentang [0, 255] telah diubah ke rentang [−10, 265] 2.

3. Tentukan urutan yang tepat dari atribut dataset: dalam "skindetect" FIS urutan

adalah {R, G, B, Class}, tetapi dalam dataset "Skin Segmentation" urutannya adalah {B, G, R, Kelas}.

4.Karena proses pengoptimalan genetik bisa terlalu padat sumber daya dilakukan pada seluruh dataset "Skin Segmentation", yang dapat dibuat dan disimpan subset acak dari "Segmentasi Kulit" untuk digunakan sebagai set data tuning.

5.Pada akhirnya, ditetapkan beberapa parameter dari algoritma genetika dan memilih mode konsol atau jendela untuk menampilkan proses optimasi3.

Langkah 2 Mengeksekusi skrip “FIS\_Genetic\_Design” :

1. Pilih FIS “skindetect” sebagai FIS awal.

2. Parameter yang disetel harus dimasukkan: jika mode konsol telah dipilih sebelumnya, maka array tuned-parameter adalah variabel "x"; dalam kasus mode jendela, hasil optimasi harus diekspor ke ruang kerja, dan variabel "x" yang diperlukan akan menjadi elemen dari susunan struktur yang diekspor.

3. Pilih nama FIS yang dioptimalkan.

Pada bagian ini, "skindetect" dan "skindetect\_optimized" dibandingkan dalam hal kesalahan root-mean-square (RMSE) pada keseluruhan "Segmentasi Kulit" dataset dengan menggunakan skrip “FIS\_Running”. Pengoptimalan genetik secara signifikan mengakibatkan penurunan kesalahan kinerja inferensi fuzzy

**Artificial Neural Networks**

Jaringan Saraf Tiruan (JST) adalah paradigma pemrosesan informasi yang terinspirasi oleh cara sistem saraf biologis, seperti otak, memproses informasi. Elemen kunci dari paradigma ini adalah struktur baru dari sistem pemrosesan informasi. Ini terdiri dari sejumlah besar elemen pemrosesan yang sangat saling berhubungan (neuron) yang bekerja serempak untuk memecahkan masalah-masalah tertentu. JST dikonfigurasi untuk aplikasi tertentu, seperti pengenalan pola atau klasifikasi data, melalui proses pembelajaran. Belajar dalam sistem biologis melibatkan penyesuaian pada koneksi sinaptik yang ada di antara neuron. Ini juga berlaku untuk ANN.

Jenis jaringan syaraf tiruan yang paling umum terdiri dari tiga kelompok, atau lapisan dari unit: sebuah lapisan unit "input" dihubungkan ke lapisan unit "tersembunyi", yang terhubung ke lapisan unit "output".

Aktivitas unit input mewakili informasi mentah yang dimasukkan ke dalam jaringan. Aktivitas setiap unit tersembunyi ditentukan oleh aktivitas unit input dan bobot pada koneksi antara input dan unit tersembunyi. Perilaku unit output tergantung pada aktivitas unit tersembunyi dan bobot antara unit tersembunyi dan keluaran.

Jenis jaringan sederhana ini menarik karena unit-unit tersembunyi bebas untuk membangun representasi mereka sendiri dari input. Bobot antara input dan unit tersembunyi menentukan kapan setiap unit tersembunyi aktif, dan dengan memodifikasi bobot ini unit tersembunyi dapat memilih apa yang diwakilinya.

Perilaku ANN (Artificial Neural Network) bergantung pada bobot dan fungsi input-output (fungsi transfer) yang ditentukan untuk unit. Fungsi ini biasanya termasuk dalam salah satu dari tiga kategori:

-linear (atau ramp)

-ambang

-sigmoid

Untuk unit linear, aktivitas output sebanding dengan total output tertimbang. Untuk unit ambang, output ditetapkan pada salah satu dari dua level, tergantung pada apakah total input lebih besar dari atau kurang dari beberapa nilai ambang.

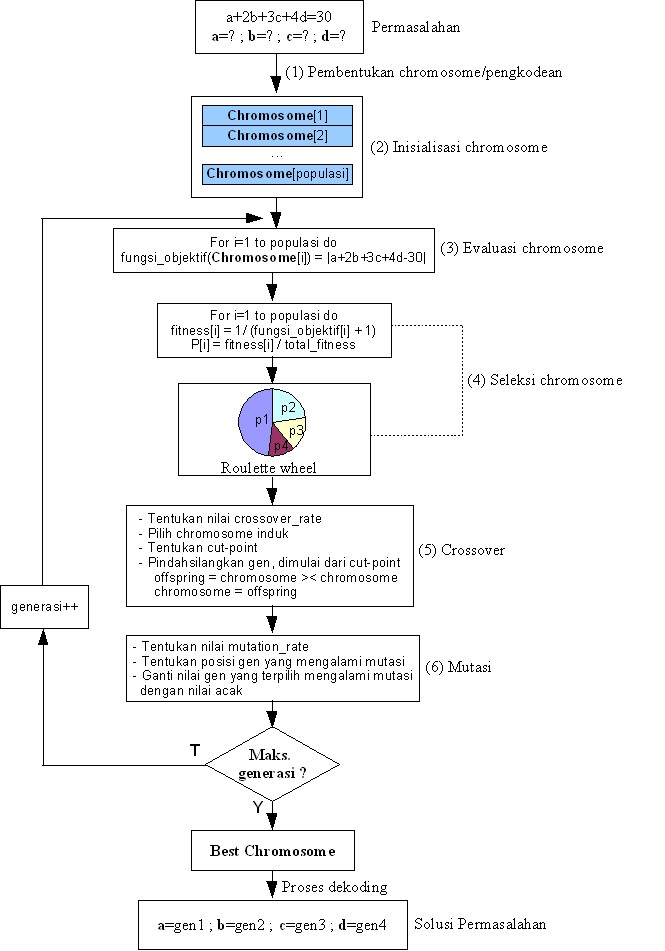
Untuk unit sigmoid, output bervariasi secara kontinyu tetapi tidak linear saat input berubah. Unit Sigmoid menanggung kemiripan yang lebih besar dengan neuron nyata daripada unit linear atau ambang batas, tetapi ketiganya harus dianggap perkiraan kasar.

Untuk melatih jaringan saraf untuk melakukan beberapa tugas, kita harus menyesuaikan bobot setiap unit sedemikian rupa sehingga kesalahan antara output yang diinginkan dan output aktual berkurang. Proses ini mengharuskan jaringan saraf menghitung turunan kesalahan dari bobot (EW). Dengan kata lain, ia harus menghitung bagaimana kesalahan berubah karena setiap bobot dinaikkan atau diturunkan sedikit. Algoritma propagasi balik adalah metode yang paling banyak digunakan untuk menentukan EW.

Algoritma propagasi balik paling mudah untuk dipahami jika semua unit dalam jaringan adalah linier. Algoritma ini menghitung setiap EW dengan terlebih dahulu menghitung EA, tingkat di mana kesalahan berubah ketika tingkat aktivitas unit diubah. Untuk unit output, EA hanyalah perbedaan antara output aktual dan yang diinginkan. Untuk menghitung EA untuk unit tersembunyi di lapisan sebelum layer output, pertama-tama kita mengidentifikasi semua bobot antara unit tersembunyi dan unit output yang terhubung. Kemudian mengalikan bobot tersebut dengan EA dari unit output tersebut dan menambahkan produk. Jumlah ini sama dengan EA untuk unit tersembunyi yang dipilih. Setelah menghitung semua EA di lapisan tersembunyi tepat sebelum lapisan output, kita dapat menghitung dalam mode seperti EA untuk lapisan lain, bergerak dari lapisan ke lapisan dalam arah yang berlawanan dengan cara kegiatan merambat melalui jaringan. Inilah yang memberi propagasi balik namanya. Setelah EA dihitung untuk unit, maka lurus ke depan untuk menghitung EW untuk setiap koneksi masuk unit. EW adalah produk dari EA.

**Genetic Algorithm**

Misalkan ada persamaan a+2b+3c+4d = 30, kita mencari nilai a, b, c, dan d yang memenuhi persamaan diatas. Kita mencoba menggunakan algoritma genetika untuk menyelesaikan permasalahan diatas. Diagram alir dari algoritma genetika untuk menyelesaikan permasalahan diatas dapat dibawah ini.



Penjelasan mengenai langkah-langkah penyelesaian permasalahan diatas menggunakan algoritma genetika adalah sebagai berikut:

1. Pembentukan chromosome

Karena yang dicari adalah nilai a, b, c, d maka variabel a, b, c, d dijadikan sebagai gen-gen pembentuk chromosome. Batasan nilai variabel a adalah bilangan integer 0 sampai 30. Sedangkan batasan nilai variabel b, c, dan d adalah bilangan integer 0 sampai 10.

2. Inisialisasi

Proses inisialisasi dilakukan dengan cara memberikan nilai awal gen-gen dengan nilai acak sesuai batasan yang telah ditentukan.

Misalkan kita tentukan jumlah populasi adalah 6, maka:

Chromosome[1] = [a;b;c;d] = [12;05;03;08]

Chromosome[2] = [a;b;c;d] = [02;01;08;03]

Chromosome[3] = [a;b;c;d] = [10;04;03;04]

Chromosome[4] = [a;b;c;d] = [20;01;10;06]

Chromosome[5] = [a;b;c;d] = [01;04;03;09]

Chromosome[6] = [a;b;c;d] = [20;05;07;01]

3. Evaluasi Chromosome

Permasalahan yang ingin diselesaikan adalah nilai variabel a, b, c, dan d yang memenuhi persamaan a+2b+3c+4d = 30, maka fungsi\_objektif yang dapat digunakan untuk mendapatkan solusi adalah fungsi\_objektif(chromosome) = | (a+2b+3c+4d) – 30 |

Kita hitung fungsi\_objektif dari chromosome yang telah dibangkitkan:

fungsi\_objektif(chromosome[1]) = Abs(( 12 + 2\*5 + 3\*3 + 4\*8 ) - 30)

= Abs((12 + 10 + 9 + 32 ) - 30)

= Abs(63 - 30)

= 33

fungsi\_objektif(chromosome[2]) = Abs(( 2 + 2\*1 + 3\*8 + 4\*3 ) - 30)

= Abs(( 2 + 2 + 24 + 12 ) - 30)

= Abs(40 - 30)

= 10

fungsi\_objektif(chromosome[3]) = Abs(( 10 + 2\*4 + 3\*3 + 4\*4 ) - 30)

= Abs(( 10 + 8 + 9 + 16 ) - 30)

= Abs(43 - 30)

= 13

fungsi\_objektif(chromosome[4]) = Abs(( 20 + 2\*1 + 3\*10 + 4\*6 ) - 30)

= Abs(( 20 + 2 + 30 + 24 ) - 30)

= Abs(76 - 30)

= 46

fungsi\_objektif(chromosome[5]) = Abs(( 1 + 2\*4 + 3\*3 + 4\*9 ) - 30)

= Abs(( 1 + 8 + 9 + 36 ) - 30)

= Abs(54 - 30)

= 24

fungsi\_objektif(chromosome[6]) = Abs(( 20 + 2\*5 + 3\*7 + 4\*1 ) - 30)

= Abs(( 20 + 10 + 21 + 4) - 30)

= Abs(55 - 30)

= 25

Rata-rata dari fungsi objektif adalah:

rata-rata = (33+10+13+46+24+25)/6

= 151 / 6

= 25.167

4. Seleksi Chromosome

Proses seleksi dilakukan dengan cara membuat chromosome yang mempunyai fungsi\_objektif kecil mempunyai kemungkinan terpilih yang besar atau mempunyai nilai probabilitas yang tinggi. Untuk itu dapat digunakan fungsi fitness = (1/(1+fungsi\_objektif)), fungsi\_objektif perlu ditambah 1 untuk menghindari kesalahan program yang diakibatkan pembagian oleh 0.

fitness[1] = 1 / (fungsi\_objektif[1]+1)

= 1 / 34

= 0.0294

fitness[2] = 1 / (fungsi\_objektif[2]+1)

= 1 / 11

= 0.0909

fitness[3] = 1 / (fungsi\_objektif[3]+1)

= 1 / 14

= 0.0714

fitness[4] = 1 / (fungsi\_objektif[4]+1)

= 1 / 47

= 0.0212

fitness[5] = 1 / (fungsi\_objektif[5]+1)

= 1 / 25

= 0.0400

fitness[6] = 1 / (fungsi\_objektif[6]+1)

= 1 / 26

= 0.0385

total\_fitness = 0.0294 + 0.0909 + 0.0714 + 0.0212 + 0.04 + 0.0385

= 0.2914

Rumus untuk mencari probabilitas: P[i] = fitness[i] / total\_fitness

P[1] = 0.0294 / 0.2914

= 0.1009

P[2] = 0. 0909 / 0.2914

= 0.3119

P[3] = 0. 0714 / 0.2914

= 0.2450

P[4] = 0. 0212 / 0.2914

= 0.0728

P[5] = 0.04 / 0.2914

= 0.1373

P[6] = 0.0385 / 0.2914

= 0.1321

Dari probabilitas diatas dapat kita lihat kalau chromosome ke 2 yang mempunyai fitness paling besar maka chromosome tersebut mempunyai probabilitas untuk terpilih pada generasi selanjutnya lebih besar dari chromosome lainnya. Untuk proses seleksi kita gunakan roulete wheel, untuk itu kita harus mencari dahulu nilai kumulatif probabilitasnya:

C[1] = 0.1009

C[2] = 0.1009+ 0.3119

= 0.4128

C[3] = 0.1009+ 0.3119 + 0.2450

= 0.6578

C[4] = 0.1009+ 0.3119 + 0.2450 + 0.0728

= 0.7306

C[5] = 0.1009+ 0.3119 + 0.2450 + 0.0728 + 0.1373

= 0.8679

C[6] = 0.1009+ 0.3119 + 0.2450 + 0.0728 + 0.1373 + 0.1321

= 1

Setelah dihitung cumulative probabilitasnya maka proses seleksi menggunakan roulete-wheel dapat dilakukan. Prosesnya adalah dengan membangkitkan bilangan acak R dalam range 0-1.

Jika R[k] < C[1] maka pilih chromosome 1 sebagai induk, selain itu pilih chromosome ke-k sebagai induk dengan syarat C[k-1] < R < C[k]. Kita putar roulete wheel sebanyak jumlah populasi yaitu 6 kali (bangkitkan bilangan acak R) dan pada tiap putaran, kita pilih satu chromosome untuk populasi baru. Misal:

R[1] = 0.201

R[2] = 0.284

R[3] = 0.009

R[4] = 0.822

R[5] = 0.398

R[6] = 0.501

Angka acak pertama R[1] adalah lebih besar dari C[1] dan lebih kecil daripada C[2] maka pilih chromosome[2] sebagai chromosome pada populasi baru, dari bilangan acak yang telah dibangkitkan diatas maka populasi chromosome baru hasil proses seleksi adalah:

chromosome[1] = chromosome[2]

chromosome[2] = chromosome[2]

chromosome[3] = chromosome[1]

chromosome[4] = chromosome[5]

chromosome[5] = chromosome[2]

chromosome[6] = chromosome[3]

Chromosome baru hasil proses seleksi:

chromosome[1] = [02;01;08;03]

chromosome[2] = [02;01;08;03]

chromosome[3] = [12;05;03;08]

chromosome[4] = [01;04;03;09]

chromosome[5] = [02;01;08;03]

chromosome[6] = [10;04;03;04]

5. Crossover

Setelah proses seleksi maka proses selanjutnya adalah proses crossover. Metode yang digunakan salah satunya adalah one-cut point, yaitu memilih secara acak satu posisi dalam chromosome induk kemudian saling menukar gen. Chromosome yang dijadikan induk dipilih secara acak dan jumlah chromosome yang mengalami crossover dipengaruhi oleh parameter crossover\_rate ( ρc ).

Pseudo-code untuk proses crossover adalah sebagai berikut:

begin

k← 0;

while(k<populasi) do

R[k] ← random(0-1);

if (R[k] < ρc ) then

select Chromosome[k] as parent;

end;

k = k + 1;

end;

end;

Misal kita tentukan crossover probability adalah sebesar 25%, maka diharapkan dalam satu generasi ada 50% Chromosome (3 chromosome) dari satu generasi mengalami proses crossover. Prosesnya adalah sebagai berikut:

Pertama kita bangkitkan bilangan acak R sebanyak jumlah populasi

R[1] = 0.191

R[2] = 0.259

R[3] = 0.760

R[4] = 0.006

R[5] = 0.159

R[6] = 0.340

Maka Chromosome ke k akan dipilih sebagai induk jika R[k] < ρc, dari bilangan acak R diatas maka yang dijadikan induk adalah chromosome[1], chromosome[4] dan chromosome[5].

Setelah melakukan pemilihan induk proses selanjutnya adalah menentukan posisi crossover. Ini dilakukan dengan cara membangkitkan bilangan acak dengan batasan 1 sampai (panjang chromosome-1), dalam kasus ini bilangan acak yang dibangkitkan adalah 1 – 3. Misalkan didapatkan posisi crossover adalah 1 maka chromosome induk akan dipotong mulai gen ke 1 kemudian potongan gen tersebut saling ditukarkan antar induk.

chromosome[1] >< chromosome[4]

chromosome[4] >< chromosome[5]

chromosome[5] >< chromosome[1]

Posisi cut-point crossover dipilih menggunakan bilangan acak 1-3 sebanyak jumlah crossover yang terjadi, misal

C[1] = 1

C[2] = 1

C[3] = 2

offspring[1] = chromosome[1] >< chromosome[4]

= [02;01;08;03] >< [01;04;03;09]

= [02;04;03;09]

offspring[4] = Chromosome[4] >< Chromosome[5]

= [01;04;03;09] >< [02;01;08;03]

= [01;01;08;03]

offspring[5] = Chromosome[5] >< Chromosome[1]

= [02;01;08;03] >< [02;01;08;03]

= [02;01;08;03]

Dengan demikian populasi Chromosome setelah mengalami proses crossover menjadi:

chromosome[1] = [02;04;03;09]

chromosome[2] = [02;01;08;03]

chromosome[3] = [12;05;03;08]

chromosome[4] = [01;01;08;03]

chromosome[5] = [02;01;08;03]

chromosome[6] = [10;04;03;04]

6. Mutasi

Jumlah chromosome yang mengalami mutasi dalam satu populasi ditentukan oleh parameter mutation\_rate. Proses mutasi dilakukan dengan cara mengganti satu gen yang terpilih secara acak dengan suatu nilai baru yang didapat secara acak. Prosesnya adalah sebagai berikut. Pertama kita hitung dahulu panjang total gen yang ada dalam satu populasi. Dalam kasus ini panjang total gen adalah total\_gen = (jumlah gen dalam chromosome) \* jumlah populasi

= 4 \* 6

= 24

Untuk memilih posisi gen yang mengalami mutasi dilakukan dengan cara membangkitkan bilangan integer acak antara 1 sampai total\_gen, yaitu 1 sampai 24. Jika bilangan acak yang kita bangkitkan lebih kecil daripada variabel mutation\_rate (ρm) maka pilih posisi tersebut sebagai sub-chromosome yang mengalami mutasi. Misal ρm kita tentukan 10% maka diharapkan ada 10% dari total\_gen yang mengalami populasi:

jumlah mutasi = 0.1 \* 24

= 2.4

= 2

Misalkan setelah kita bangkitkan bilangan acak terpilih posisi gen 12 dan 18 yang mengalami mutasi. Dengan demikian yang akan mengalami mutasi adalah chromosome ke-3 gen nomor 4 dan Chromosome ke-5 gen nomor 2. Maka nilai gen pada posisi tersebut kita ganti dengan bilangan acak 0-30.

Misalkan bilangan acak yang terbangkitkan adalah 2 dan 5. Maka populasi chromosome setelah mengalami proses mutasi adalah:

chromosome[1] = [02;04;03;09]

chromosome[2] = [02;01;08;03]

chromosome[3] = [12;05;03;02]

chromosome[4] = [01;01;08;03]

chromosome[5] = [02;05;08;03]

chromosome[6] = [10;04;03;04]

Setelah proses mutasi maka kita telah menyelesaikan satu iterasi dalam algoritma genetika atau disebut dengan satu generasi. Maka fungsi\_objective setelah satu generasi adalah:

chromosome[1] = [02;04;03;09]

fungsi\_objektif[1] = Abs(( 2 + 2\*4 + 3\*3 + 4\*9 ) - 30)

= Abs(( 2 + 8 + 9 + 36 ) - 30)

= Abs( 55 - 30)

= 25

chromosome[2] = [02;01;08;03]

fungsi\_objektif[2] = Abs(( 2 + 2\*1 + 3\*8 + 4\*3 ) - 30)

= Abs(( 2 + 2 + 24 + 12 ) - 30)

= Abs(40 - 30)

= 10

chromosome[3] = [12;05;03;02]

fungsi\_objektif[3] = Abs(( 12 + 2\*5 + 3\*3 + 4\*2 ) - 30)

= Abs(( 12 + 10 + 9 + 8 ) - 30)

= Abs(39 - 30)

= 9

chromosome[4] = [01;01;08;03]

fungsi\_objektif[4] = Abs(( 1 + 2\*1 + 3\*8 + 4\*3 ) - 30)

= Abs(( 1 + 2 + 24 + 12 ) - 30)

= Abs(39 - 30)

= 9

chromosome[5] = [02;05;08;03]

fungsi\_objektif[5] = Abs(( 2 + 2\*5 + 3\*8 + 4\*3 ) - 30)

= Abs(( 2 + 10 + 24 + 12 ) - 30)

= Abs(48 - 30)

= 18

chromosome[6] = [10;04;03;04]

fungsi\_objektif[6] = Abs(( 10 + 2\*4 + 3\*3 + 4\*4 ) - 30)

= Abs(( 10 + 8 + 9 + 16 ) - 30)

= Abs(43 - 30)

= 13

Rata-rata fungsi objektif setelah satu generasi adalah:

rata-rata = ( 25 + 10 + 9 + 9 + 18 + 13) / 6

= 84 / 6

= 14.0

Dapat dilihat dari hasil perhitungan fungsi objektif diatas bahwa setelah satu generasi, nilai hasil rata-rata fungsi\_objektif lebih menurun dibandingkan hasil fungsi\_objektif pada saat sebelum mengalami seleksi, crossover dan mutasi. Hal ini menunjukkan bahwa chromosome atau solusi yang dihasilkan setelah satu generasi lebih baik dibandingkan generasi sebelumnya. Maka pada generasi selanjutnya chromosome-chromosome yang baru adalah:

chromosome[1] = [02;04;03;09]

chromosome[2] = [02;01;08;03]

chromosome[3] = [12;05;03;02]

chromosome[4] = [01;01;08;03]

chromosome[5] = [02;05;08;03]

chromosome[6] = [10;04;03;04]

Chromosome-chromosome ini akan mengalami proses yang sama seperti generasi sebelumnya yaitu proses evaluasi, seleksi, crossover dan mutasi yang kemudian akan menghasilkan chromosome-chromosome baru untuk generasi yang selanjutnya. Proses ini akan berulang sampai sejumlah generasi yang telah ditetapkan sebelumnya.

Setelah 50 generasi didapatkan chromosome yang terbaik adalah:

Chromosome = [07;05;03;01]

Jika didekode maka:

a=7 ; b=5 ; c=3 ; d=1

Jika dihitung terhadap persamaan f = a+2b+3c+4d:

7 + (2\*5) + (3\*3) + (4\*1) = 30

**Evolutionary Algorithm**

Ciri utama evolution strategies (ES) adalah penggunaan vektor bilangan pecahan (real-vector) sebagai representasi solusi. Berbeda dengan GAs yang menggunakan crossover sebagai operator reproduksi utama dan mutasi sebagai operator penunjang, ES lebih bertumpu pada operator mutasi. Mekanisme self-adaptation digunakan untuk mengontrol perubahan nilai parameter pencarian. GAs dan ES bisa digunakan untuk menyelesaikan permasalahan yang sama. Tetapi mana yang terbaik di antara kedua metode tersebut sangat tergantung pada permasalahan yang dihadapi.

Beberapa notasi digunakan oleh ES. μ (miu) menyatakan ukuran populasi. λ (lambda) menyatakan banyaknya offspring yang dihasilkan pada proses reproduksi (sama seperti crossover rate dan mutation rate pada GAs). Beberapa penelitian menyarankan besarnya nilai λ sebesar 7μ. Perbedaan nyata ES dan GAs adalah pada operator yang digunakan. Perbedaan yang lain adalah mutasi pada GAs digunakan untuk menghasilkan individu baru (offspring) sebagai tambahan dari offspring yang diproduksi oleh operator crossover. Pada ES, mutasi diterapkan pada offspring yang dihasilkan proses rekombinasi. Rekombinasi pada ES mirip dengan operator crossover pada GAs tapi bisa menggunakan lebih dari satu induk.

ES(μ, λ) tidak menggunakan rekombinasi dalam proses reproduksi. Seleksi menggunakan elitism selection hanya melibatkan individu dalam offspring, individu induk dalam populasi tidak dilibatkan. ES(μ/r, λ) serupa dengan ES(μ, λ) dengan tambahan melibatkan proses rekombinasi. ES(μ+λ) tidak menggunakan rekombinasi dan proses seleksi menggunakan elitism selection melibatkan individu offspring dan induk.

Recombinasi dilakukan untuk menghasilkan offspring sebanyak λ dari sejumlah μ individu dalam populasi. Setiap satu individu offspring dihasilkan dari beberapa induk. Induk dipilih secara acak dari populasi. Metode rekombinasi paling sederhana adalah dengan menghitung rata-rata nilai elemen induk. ciri utama ES adalah penggunaan vektor bilangan pecahan (real-vector) sebagai representasi. Pada perkembangannya, ES juga diadopsi untuk permasalahan kombinatorial yang menggunakan representasi permutasi. Cara paling mudah adalah dengan menggunakan struktur ES yang hanya menggunakan mutasi tanpa rekombinasi. Mekanisme self-adaptation juga tidak digunakan. Pendekatan ini pada hakekatnya menghasilkan siklus yang sama seperti GAs tanpa crossover. Adopsi mekanisme self-adaptation pada representasi permutasi bisa dilakukan dengan cara sederhana jika yang digunakan adalah reciprocal exchange mutation atau insertion mutation.

**Differential Evolution**

Differential Evolution merupakan metode metaheuristik yang handal dalam menemukan titik global optima. Hal ini dikarenakan adanya proses seleksi stokastik dalam langkahlangkahnya dan kemampuan untuk memproses sejumlah individu secara parallel. Algoritma awal DE akan dikembangkan sehingga dapat digunakan sebagai alternatif penyelesaian estimasi parameter customer utility dan maksimasi keuntungan.

Adapun langkah-langkah dalam pengembangan model adalah sebagai berikut:

1. Menentukan parameter yang digunakan untuk menjalankan iterasi.

a. Parameter customer utility yaitu jumlah pelanggan (n) yang akan digunakan,

periode (t), pembelian (d), keputusan harga (p), dan waktu pengiriman katalog

(mms).

b. Parameter DE yaitu crossover rate (CR), factor skala pembeda vektor (F) dan jumlah iterasi maksimum.

2. Mengevaluasi nilai tujuan sesuai dengan model matematis yang telah dibuat.

3. Membangkitkan tiga vektor acak r0,r1,r2 dari populasi awal sehingga didapatkan vector mutan.

4. Membentuk populasi turunan dengan mengkombinasi vektor turunan dan vektor mutan.

5. Melakukan update vektor populasi dengan cara seleksi. Apabila populasi turunan membentuk nilai yang lebih baik maka akan dijadikan populasi baru, dan begitu pula sebaliknya.

6. Jika stopping criteria terpenuhi maka iterasi akan berhenti. Jika tidak, maka iterasi akan berlanjut dengan kembali pada proses ketiga.

Pada tahap ini dilakukan penentuan parameter yang mendukung berjalannya proses optimasi. Beberapa parameter tersebut antara lain:

a. F atau skala pembeda vektor. Parameter ini digunakan pada saat mutasi vektor. Nilai F mulai dari 0 hingga 1, namun Price dan Storn (2005) merekomendasikan untuk menggunakan F mulai dari 0.5 hingga 1 dan yang digunakan adalah 0.9.

b. Crossover rate (CR). Parameter ini digunakan pada saat tahapan kawin silang untuk menghasilkan populasi turunan. Nilai CR yang memiliki kemampuan menyelesaikan beberapa fungsi adalah 0≤CR≤0.2 atau 0.9≤CR≤1.

c. Upper dan lower Bound. Batas atas nilai vektor sesuiai distribusi normal adalah 3.49. Jika ada nilai vektor yang melanggar batas ini maka secara otomatis akan berubah menjadi 3.49. dan begitu pula sebaliknya yang terjadi pada btas bawah sebesar -3.4.

d. Membangkitkan populasi vektor yang berukuran NxD, yaitu jumlah pelanggan yang akan diperhitungkan dikalikan banyaknya parameter yang dicari.

e. Populasi vektor yang telah dibangkitkan kemudian dievaluasi dengan cara menghitung nilai fungsinya.

Pembentukan vektor mutan (vi) dilakukan dengan mengkombinasikan 3 vektor (r0,r1,r2) yang dipilih secara acak dari populasi yang telah ada dengan F sebagai factor skala pembeda. Tahap crossover merupakan tahap yang dikenal dengan istilah kawin silang, yaitu menggabungkan vektor dari populasi parents (xi) dengan vektor mutan (vi). Diawali dengan membangkitkan bilangan random sebanyak jumlah pelanggan (randi). Apabila nilai CR yang telah ditentukan pada inisialisasi lebih besar dari nilai randi maka vektor i akan tetap berisi xi dan begitu pula sebaliknya jika CR kurang dari atau sama dengan nilai randi maka vektor i akan berisi vi.

Tahap seleksi yaitu membandingkan nilai fungsi objektif dari populasi parents dengan populasi turunan. Populasi turunan yang dihasilkan dari crossover kemudian dimasukkan kedalam fungsi objektif yang telah dibentuk, apabila nilainya lebih tinggi disbanding nilai parents maka populasi baru akan menggantikan parents. Hasil dari seleksi ini kemudian akan disimpan sebagai nilai baru untuk iterasi berikutnya, antara lain: nilai vektor, nilai fungsi objektif, dan nilai expected valuation.

**Neuro-Fuzzy (Fuzzy Neural Network)**

Untuk mengaktifkan sistem untuk menangani kognitif ketidakpastian dengan cara yang lebih seperti manusia, seseorang dapat menggabungkan konsep fuzzy logika ke dalam jaringan saraf. Hasilnya sistem hibrida disebut neural fuzzy, neural jaringan fuzzy, neuro-fuzzy atau fuzzy-neuro. Proses komputasi yang dibayangkan sistem neural fuzzy adalah sebagai berikut. Itu dimulai dengan pengembangan "neuron fuzzy" berdasarkan pemahaman neuronal biologis morfologi, diikuti dengan pembelajaran mekanisme. Ini mengarah pada hal-hal berikut tiga langkah dalam komputasi saraf fuzzy proses.

• pengembangan model saraf fuzzy termotivasi oleh neuron biologis,

• model koneksi sinaptik yang menggabungkan ketidakjelasan dalam jaringan syaraf

• pengembangan algoritma pembelajaran (itu adalah metode menyesuaikan sinaptik bobot)

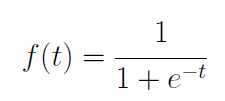
Dua model yang mungkin dari sistem neural fuzzy adalah

• Menanggapi pernyataan linguistik, yang blok antarmuka fuzzy menyediakan input vektor ke jaringan saraf multi-layer. Jaringan saraf dapat diadaptasi (dilatih) untuk menghasilkan output perintah atau keputusan yang diinginkan.

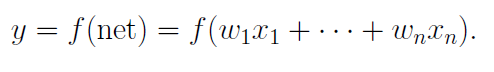
* Sebuah jaringan saraf multi-layered menggerakkan mekanisme inferensi fuzzy.

Unsur-unsur pemrosesan dasar jaringan saraf disebut neuron buatan, atau sederhananya neuron. Aliran sinyal dari neuron input, xj, dianggap unidirectionalas ditunjukkan oleh panah, seperti neuron aliran sinyal keluaran. Neuron input tidak mengubah input sinyal sehingga output mereka sama dengan mereka memasukkan. Sinyal xi berinteraksi dengan berat wi ke menghasilkan produk pi = wixi, i = 1,. . . , n. Informasi input pi dikumpulkan, oleh Selain itu, untuk menghasilkan input ke neuron. Neuron menggunakan transfernya fungsi f, yang bisa menjadi sigmoidal fungsi,





Untuk menghitung output adalah :



Jaringan saraf sederhana ini, yang menggunakan perkalian, Selain itu, dan sigmoidal f, akan disebut sebagai jaringan saraf biasa (atau standar).

**Evolving ANN**

ANN evolusi menjelaskan kepada artificial neural network yang spesifik dimana terdapat evolusi dalam adaptasi pembelajarannya. Evolusi ANN ini digunakan untuk performansi tugas diantara pelatihan weight connection, arsitektur design, adaptasi aturan pembelajaran, seleksi input fitur, inisialisasi berat, aturan ekstraksi dari ANN. Satu perbedaan fitur EANN adalah adaptasi mereka terhadap lingkugan yang dinamis. Dua adaptasi yaitu evolusi dan pembelajaran kepada lingkungan dinamik lebih efisien dan efektif, EANN bisa menganggap sebagai framework umum untuk sistem adaptif, sistem dapat berubah arstitektur mereka dan aturan pembalajaran tanpa intervensi manusia.

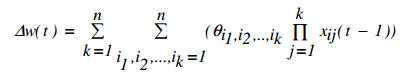
EA menjelaskan kelas dengan berbasis populasi stokastik yang mengembangkan ide dan juga prinsi dari natura evolusi. Termasuk didalamnya strategi evolusi, pemrograman evolusi dan algoritma genetika. Satu hal terpenting adalah strategi populasi. Individu dalam populasi dan mengubah informasi masing-masing untuk performa tugas mereka. Evolusi ditampilkan sampai ANN dengan 3 level : connection weights, archiectures, dan learning rules. EVolusi koneksi bobot menjelaskan adaptif dan pengenalan global untuk pelatihan, terutama reinforcement learning dan recurrent network dimana berbasis gradient. Evolusi arsitektur menjalankan ANN untuk adaptasi topologi untuk tugas masing-masing yag berbeda tanpa intervensi manusia. Evolusi learning rules menjalankan proses learning to learn di ANN dimana adaptasi learning rules selama evolusi. Bisa dianggap sebagai proses adaptif otomatis selama pembelajaran aturan.

Banyak dari ANN konvensional sekarang sedang dirancang secara statistik cukup akurat tetapi mereka masih meninggalkan rasa tidak enak pengguna yang mengharapkan komputer untuk menyelesaikan masalah mereka secara akurat. Kelemahan penting adalah bahwa perancang harus tentukan jumlah neuron, distribusinya atas beberapa lapisan dan interkoneksi di antara mereka. Beberapa metode telah diusulkan untuk secara otomatis membangun ANNs untuk mengurangi kompleksitas jaringan yang ditentukan jumlah unit, lapisan, dll yang sesuai. Algoritme optimasi topologi seperti Extentron, Upstart , Pruning dan Cascade Correlation, dll memiliki keterbatasannya sendiri.

Dalam evolusi EANN dapat diperkenalkan di berbagai tingkatan. Pada tingkat terendah, evolusi dapat dimasukkan ke dalam berat pelatihan, di mana bobot ANN berevolusi. Pada tingkat yang lebih tinggi berikutnya, evolusi dapat dimasukkan ke dalam jaringan saraf adaptasi arsitektur, di mana arsitektur (jumlah lapisan tersembunyi, tidak ada neuron tersembunyi dan transfer node fungsi) berevolusi. Pada tingkat tertinggi, evolusi dapat dimasukkan ke dalam mekanisme pembelajaran.

Adaptasi arsitektur evolusioner dapat dicapai dengan algoritma konstruktif dan destruktif. Konstruktif algoritma yang menambah kompleksitas ke jaringan mulai dari arsitektur yang sangat sederhana sampai seluruh jaringandapat mempelajari tugas . Algoritma destruktif dimulai dengan arsitektur besar dan menghapus node dan interkoneksi sampai JST tidak lagi dapat melakukan tugasnya. Kemudian pemindahan terakhir dibatalkan. Untuk jaringan yang optimal, fungsi transfer node yang diperlukan (Gaussian, sigmoidal, dll.) Dapat diformulasikan sebagai global masalah pencarian, yang berevolusi secara bersamaan dengan pencarian arsitektur.

Agar jaringan saraf benar-benar optimal, aturan pembelajaran harus disesuaikan secara dinamis sesuai dengan arsitekturnya dan masalah yang diberikan. Memutuskan tingkat pembelajaran dan momentum dapat dianggap sebagai upaya belajar pertama aturan. Aturan pembelajaran dasar dapat digeneralisasikan oleh fungsi



**Evolving Fuzzy Systems**

Integrasi jaringan saraf dan sistem fuzzy menjadi satu JST menarik banyak peneliti. Integrasi aturan fuzzy ke dalam model neuron tunggal dan kemudian ke dalam struktur jaringan saraf yang lebih besar, pembelajaran yang erat dan aturan penalaran fuzzy ke dalam struktur koneksionis. Banyak model jaringan saraf fuzzy dikembangkan berdasarkan prinsip-prinsip ini. Dalam sistem koneksionis yang berkembang ide-ide ini dikembangkan lebih lanjut, di mana bukannya melatih struktur koneksiist tetap, struktur dan fungsionalitas berevolusi dari data yang masuk.

Prinsip Evolving Fuzzy Systems didasarkan pada pembelajaran lokal - neuron dialokasikan sebagai pusat kluster data dan sistem menciptakan model lokal dalam kluster ini. Fuzzy pengelompokan sebagai sarana untuk menciptakan sistem berbasis pengetahuan lokal. Evolving Fuzzy Systems adalah sistem berbasis sambungan modular yang berevolusi struktur dan fungsionalitas dengan cara yang berkesinambungan, mandiri, online, adaptif, dan interaktif dari informasi yang masuk. Mereka dapat memproses data dan pengetahuan dalam pengawasan dan / atau cara yang tidak diawasi. Evolving Fuzzy Systems mempelajari model lokal dari data melalui pengelompokan data dan menghubungkan fungsi output lokal untuk setiap cluster yang diwakili dalam struktur koneksionis. Mereka dapat belajar secara bertahap item data tunggal atau potongan data dan juga secara bertahap mengubah fitur masukan mereka. Elemen Evolving Fuzzy Systems telah diusulkan sebagai bagian dari model jaringan saraf klasik, seperti Self Organizing Peta, Fungsi Basis Radial, Fuzzy ARTMap, pertumbuhan gas saraf, sistem neuro-fuzzy, Jaringan Alokasi Sumber .

Berikut ini adalah prinsip utama Evolving Fuzzy Systems sebagaimana dinyatakan :

(1) Pembelajaran yang cepat dari sejumlah besar data, mis. menggunakan pelatihan "one-pass", dimulai dengan sedikit pengetahuan sebelumnya;

(2) Adaptasi secara real-time dan dalam mode on-line di mana data bar diakomodasi karena didasarkan pada pembelajaran lokal;

(3) "Buka", struktur yang berkembang, di mana variabel input baru (relevan dengan tugas), keluaran baru (misalnya kelas), koneksi baru dan neuron ditambahkan / berevolusi "dengan cepat";

(4) Baik pembelajaran data dan representasi pengetahuan difasilitasi dalam

cara yang komprehensif dan fleksibel, misalnya, pembelajaran yang diawasi, tidak diawasi belajar, berkembang pengelompokan, "tidur" belajar, melupakan / pemangkasan, aturan fuzzy penyisipan dan ekstraksi;

(5) Interaksi aktif dengan ECOS lain dan dengan lingkungan dalam multimodal mode;

(6) Mewakili ruang dan waktu dalam skala yang berbeda, misalnya, kelompok data, memori jangka pendek dan panjang, usia data dll

**Fuzzy EA (Evolutionary Algorithm)**

EA adalah pencarian stokastik dan heuristik optimasi yang berasal dari teori evolusi klasik, yang diimplementasikan pada komputer di sebagian besar kasus. Ide dasarnya adalah jika hanya orang-orang itu populasi mereproduksi, yang memenuhi kriteria seleksi tertentu, dan individu lain dari populasi mati, populasi akan bertemu dengan orang-orang yang paling memenuhi kriteria seleksi.

Jika reproduksi tidak sempurna ditambahkan populasi dapat mulai menjelajahi ruang pencarian dan akan pindah ke individu yang memiliki probabilitas seleksi yang meningkat dan mewarisi properti ini kepada mereka keturunan. Dinamika populasi ini mengikuti aturan dasar teori evolusi Darwin, yang dapat dijelaskan secara singkat sebagai "survival of the fittest" [DC59].

Untuk memecahkan masalah optimisasi dengan heuristik evolusioner, para individu dari suatu populasi harus merupakan solusi yang mungkin dari masalah yang diberikan dan probabilitas seleksi diatur sebanding dengan kualitas solusi yang diwakili. Kualitas solusi yang diwakili juga disebut kebugaran Φ individu. Kami menulis modal huruf A, B, C untuk set individu atau populasi. Generasi terkini dari proses evolusi akan ditunjukkan oleh huruf s. Satu Individu dengan indeks i dari populasi A (s) akan memiliki shortcut ai (s). Kualitas solusi yang diwakili oleh seorang individu juga disebut i Φ dari individu. Probabilitas seleksi dari individu ai (s) akan pi. Ketika deskripsi solusi yang mungkin untuk yang diberikan terdiri dari n elemen, elemen ke-i yang membentuk solusi yang memungkinkan disebut atribut xi, terlepas dari tipe data yang diperlukan. Oleh karena itu, seorang individu terdiri dari beberapa atribut x dan akan mewakili solusi yang mungkin di seluruh atribut ini, yang harus dioptimalkan.

Dalam kasus sederhana ini kami menggunakan satu parameter strategi σi untuk setiap parameter keputusan xi σi menugaskan a standar deviasi terhadap mutasi fenotipe untuk setiap parameter keputusan, yang sama dengan mean ukuran langkah mutasi. Jika pengkodean dihilangkan, fungsi target dapat dievaluasi tanpa perhitungan lebih lanjut:



**Ant Colony Optimization**

Ant Colony Optimization ACO atau Algoritma Koloni Semut adalah sebuah probabilistik komputasi teknik untuk memecahkan masalah yang dapat dikurangi untuk menemukan jalur yang baik melalui grafik. Mekanisme untuk menyelesaikan masalah terlalu kompleks untuk ditangani oleh satu semut adalah contoh yang baik dari diri-sistem terorganisir. Sistem ini didasarkan pada umpan balik positif (yang deposit menarik feromon semut lain yang akan memperkuat sendiri) dan negatif (disipasi dari rute oleh sistem mencegah penguapan dari labrakan). Secara teoritis, jika jumlah feromon tetap sama dari waktu ke waktu pada semua sisi, tidak ada rute yang akan dipilih. Namun, karena umpan balik, sedikit variasi pada pinggir akan diperkuat dan dengan demikian memungkinkan pilihan dari tepi.

Algoritma optimisasi koloni semut telah diterapkan ke banyak permasalahan optimasi kombinatorial, mulai dari penugasan kuadrat melipat protein atau routing kendaraan dan banyak metode yang diturunkan telah disesuaikan untuk masalah dinamis dalam variabel-variabel riil, masalah-masalah stokastik, multi-target dan implementasi paralel. Pada setiap tahap, semut memilih untuk berpindah dari satu kota ke yang lain menurut beberapa aturan:

1. Harus mengunjungi setiap kota tepat satu kali;

2. Sebuah kota yang jauh memiliki lebih sedikit kesempatan untuk dipilih (visibilitas);

3. Semakin kuat jejak feromon diletakkan pada tepi antara dua kota, semakin besar kemungkinan bahwa tepi akan dipilih;

4. Setelah menyelesaikan perjalanannya, deposit semut lebih feromon pada semua sisi itu dilalui, jika perjalanan pendek;

5. Setelah setiap iterasi, jejak feromon menguap.

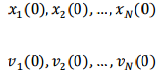
**Particle Optimization (PSO)**

Dalam Particle Swarm Optimization (PSO), kawanan diasumsikan mempunyai ukuran tertentu dengan setiap partikel posisi awalnya terletak di suatu lokasi yang acak dalam ruang multidimensi. Setiap partikel diasumsikan memiliki dua karakteristik: posisi dan kecepatan. Setiap partikel bergerak dalam ruang/space tertentu dan mengingat posisi terbaik yang pernah dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi terbaiknya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi tersebut.

Pada algoritma PSO ini, pencarian solusi dilakukan oleh suatu populasi yang terdiri dari beberapa partikel. Populasi dibangkitkan secara randomdengan batasan nilai terkecil dan terbesar. Setiap partikel merepresentasikan posisi atau solusi dari permasalahan yang dihadapi. Setiap partikel melakukan pencarian solusi yang optimal dengan melintasi ruang pencarian (search space). Hal ini dilakukan dengan cara setiap partikel melakukan penyesuaian terhadap posisi terbaik dari partikel tersebut (local best) dan penyesuaian terhadap posisi partikel terbaik dari seluruh kawanan (global best) selama melintasi ruang pencarian. Jadi, penyebaran pengalaman atau informasi terjadi di dalam partikel itu sendiri dan antara suatu partikel dengan partikel terbaik dari seluruh kawanan selama proses pencarian solusi. Setelah itu, dilakukan proses pencarian untuk mencari posisi terbaik setiap partikel dalam sejumlah iterasi tertentu sampai didapatkan posisi yang relatif steady atau mencapai batas iterasi yang telah ditetapkan. Pada setiap iterasi, setiap solusi yang direpresentasikan oleh posisi partikel, dievaluasi performansinya dengan cara memasukkan solusi tersebut kedalam fitness function.

Langkah-langkah penyelesaiannya :

1. Asumsikan bahwa ukuran kelompok atau kawanan (jumlah partikel) adalah N. Untuk mengurangi jumlah evaluasi fungsi yang diperlukan untuk menemukan solusi, sebaiknya ukuran N tidak terlalu besar, tetapi juga tidak terlalu kecil,agar ada banyak kemungkinan posisi menuju solusi terbaik atau optimal. Jika terlalu kecil sedikit kemungkinan menemukan posisi partikel yang baik. Terlalu besar juga akan membuat perhitungan jadi panjang. Biasanya digunakan ukuran kawanan adalah 20 sampai 30 partikel.
2. Bangkitkan populasi awal 𝑥 dengan rentang 𝑥 (𝐵) dan 𝑥(𝐴) secara random sehingga didapat 𝑥1, 𝑥2, … , 𝑥𝑁. Partikel j dan kecepatannya pada iterasi i dinotasikan sebagai 𝑥𝑗 (𝑖) dan 𝑣𝑗(𝑖), sehingga partikel-partikel awal ini dinotasikan



1. Hitung kecepatan dari semua partikel. Semua partikel bergerak menuj titik optimal dengan suatu kecepatan tertentu. Awalnya semua kecepatan dari partikel diasumsikan sama dengan nol. Set iterasi i = 1.
2. Pada iterasi ke-i, temukan 2 parameter penting untuk setiap partikel j yaitu:a. Nilai terbaik sejauh ini dari 𝑥𝑗 (𝑖) (koordinat partikel j pada iterasi i) dan nyatakan sebagai Pbest,j, dengan nilai fungsi tujuan paling rendah (kasus minimasi), 𝑓[𝑥𝑗 (𝑖)], yang ditemui sebuah partikel j pada semua iterasi sebelumnya. Nilai terbaik untuk semua partikel 𝑥𝑗 (𝑖) yang ditemukan sampai iterasi ke-i, Gbest, dengan nilai fungsi tujuan paling kecil/minimum diantara semua partikel untuk semua iterasi sebelumnya, 𝑓[𝑥𝑗 (𝑖)],

b.Hitung kecepatan partikel j pada iterasi ke i dengan rumus sebagai berikut



**Bee Colony**

Algoritma BCO (Bee Colony Optimization) adalah salah satu algoritma yang digunakan untuk pencarian jalur. Contoh yang dibahas kali ini adalah mengenai pencarian jalur yang melalui semua titik tujuan dengan jarak paling rendah. Bee Colony Optimization adalah algoritma optimasi yang berdasarkan pada tingkah laku kumpulan lebah madu dalam sebuah koloni untuk menemukan sumber makanan. Kemungkinan solusi dilambangkan dengan posisi sumber makanan, sedangkan nilainya dilambangkan dengan jumlah nektar yang terdapat dalam sumber makanan tersebut.

Sebelum masuk kedalam langkah-langkah pembahasan algoritma, ada beberapa konstanta atau parameter yang harus diketahui, yaitu:

\* Tentukan jumlah lebah yang digunakan dalam perhitungan

\* Tentukan jumlah lebah aktif, lebah nonaktif, dan lebah pencari

Jumlah ketiga jenis lebah ini harus sama dengan variabel jumlah lebah diatas

\* Tentukan jumlah maksimal perjalanan yang dapat dilakukan lebah sebelum harus kembali ke sarangnya

\* Ini mencegah seekor lebah keluar terlalu lama karena tidak mendapatkan perjalanan yang lebih baik dari perjalanan yang sudah ditempuhnya.

\* Tentukan jumlah iterasi yang dilakukan untuk masing-masing lebah

Semakin besar angkanya, semakin optimal hasil perhitungannya, tetapi semakin lama perhitungannya.

Langkah-langkah penyelesaian masalah ini :

1. Inisialisasi sarang lebah beserta semua obyek yang ada di dalamnya

Buat lebah sebanyak parameter total Lebah. Beri status pada masing-masing lebah sesuai dengan parameter total lebah aktif, total lebah nonaktif, total lebah pencari. Tentukan jalur acak pada masing-masing lebah, kemudian cari jejak terpendek sementara yang diperoleh secara acak. Simpan jalur terbaik sementara berdasarkan jalur dengan jarak terpendek.

2. Lakukan proses perhitungan sebanyak jumlah perulangan

\* Lakukan proses perulangan sebanyak jumlah iterasi untuk semua lebah yang berada pada daftar lebah. Lakukan proses perhitungan sesuai dengan status lebah yang sedang diproses

3. Jika lebah ini lebah aktif, maka lakukan proses perhitungan untuk lebah aktif

3a. Tentukan jalur baru untuk lebah ini

Dim jalurBaru() = TentukanJalurBaru(daftarLebah(i).jalur)

3b. Hitung nilai jalur untuk jalur yang baru ditemukan

3c. Jika jalur yang baru ternyata lebih baik dari jalur lebah tersebut

Tentukan apakah nilai acak kurang dari nilai probabilitas kesalahan

Jika benar maka Lebah melakukan kesalahan, sehingga menolak solusi yang lebih baik

3d1. Tentukan apakah nilai acak kurang dari nilai probabilitas kesalahan

Jika benar maka lebah melakukan kesalahan, sehingga menerima solusi yang lebih buruk

3d2. Jika tidak maka lebah tidak melakukan kesalahan, sehingga menolak solusi yang baru

3e. Setelah perhitungan tersebut, maka lebah akan kembali ke sarang

ada 3 kemungkinan keadaan pada saat lebah sudah berada di sarang:

3e1. Jika jumlah perjalanan lebah tersebut sudah melebihi maksimum perjalanan, maka lebah ini akan berubah menjadi lebah nonaktif

3e2. Jika jalur baru diterima oleh lebah tersebut, maka lakukan pengecekan apakah jalur baru ini lebih baik dari solusi umum. Kemudian lakukan tarian waggle

Dalam dunia nyata, tarian waggle dilakukan oleh lebah untuk mengirimkan informasi kepada teman lebahnya mengenai sumber makanan yang lebih baik

Untuk menyimpan data titik dan jarak antar titik, diperlukan sebuah class tersendiri yang tidak berhubungan dengan class sarang lebah, yaitu class daftarTitik.

**Firefly Algorithm (FA)**

Algoritma Kunang-kunang adalah sebuah algoritma metaheuristik yang terinspirasi dari perilaku berkedip kunang-kunang. Tujuan utama berkedipnya kunang-kunang adalah untuk menarik kunang-kunang yang lain. Xin-She Yang merumuskan algoritma kunang-kunang ini dengan asumsi :

1. Semua kunang-kunang itu unisex, jadi suatu kunang-kunang akan tertarik pada kunang-kunang yang lain.

2. Daya tarik sebanding dengan tingkat kecerahan kunang-kunang, kunang-kunang dengan tingkat kecerahan lebih rendah akan tertarik dan bergerak ke kunang-kunang dengan tingkat kecerahan lebih tinggi, kecerahan dapat berkurang seiring dengan bertambahnya jarak.

3. Jika tidak ada kunang-kunang yang lebih terang dari kunang-kunang yang diberikan, maka kunang-kunang ini akan bergerak secara random.

FA adalah swarm-intelligence-based, sehingga memiliki keunggulan serupa yang berbasis kecerdasan lainnya

memiliki algoritma. Bahkan, analisis sederhana menunjukkan parameter

bahwa beberapa varian PSO seperti Accelerated PSO [78] adalah kasus khusus kunang-kunang

algoritma ketika γ = 0 [65].

Namun, FA memiliki dua keunggulan utama dibandingkan algoritma lain: pembagian otomatis dan kemampuan menangani multimodality. Pertama, FA didasarkan pada daya tarik dan daya tarik menurun dengan jarak. Ini mengarah pada fakta bahwa seluruh populasi dapat secara otomatis membagi menjadi beberapa subkelompok, dan setiap kelompok dapat berkerumun setiap mode atau optimal lokal. Di antara semua mode ini, solusi global terbaik bisa digunakan ditemukan. Kedua, subdivisi ini memungkinkan kunang-kunang untuk dapat menemukan semua optima secara bersamaan jika ukuran populasi cukup tinggi daripada jumlah mode. Secara matematis, 1 / √γ mengontrol jarak rata-rata sekelompok kunang-kunang yang bisa dilihat oleh kelompok-kelompok yang berdekatan. Oleh karena itu, seluruh populasi dapat dibagi menjadi beberapa subkelompok dengan jarak rata-rata yang diberikan. Dalam kasus ekstrim ketika γ = 0, seluruh populasi tidak akan membagi lagi. Kemampuan pembagian otomatis ini membuatnya sangat cocok untuk masalah optimasi multimodal yang sangat nonlinier.

Selain itu, parameter dalam FA dapat disetel untuk mengontrol keacakan sebagai iterasi berlanjut, sehingga konvergensi juga dapat dipercepat dengan menyetel parameter ini. Keuntungan di atas membuatnya fleksibel untuk menangani masalah yang berkelanjutan, pengelompokan dan klasifikasi, dan optimasi kombinatorial juga.

Intensitas cahaya I dari kunang-kunang yang berkedip berkurang sebagai jarak dari sumber r menurun dalam hal I ∝ 1 / r 2. Selain itu, penyerapan udara menyebabkan cahaya menjadi lebih lemah dan lebih lemah karena jarak dari sumber meningkat. Di sini, intensitas cahaya sebanding dengan fungsi objektif dari masalah yang sedang terjadi dioptimalkan (yaitu, I (s) ∝ f (s), di mana s = S (x) mewakili solusi kandidat). Untuk memformulasikan FA, beberapa karakteristik kunang-kunang yang memukau diidealkan, sebagai berikut:

• Semua kunang-kunang adalah unisex.

• Daya tarik mereka sebanding dengan intensitas cahayanya.

• Intensitas cahaya kunang-kunang dipengaruhi atau ditentukan oleh lansekap dari

fungsi objektif.

Perhatikan bahwa intensitas cahaya saya dan daya tarik dalam beberapa hal sama. Sementara intensitas disebut sebagai pengukuran mutlak cahaya yang dipancarkan oleh kunang-kunang, daya tarik adalah ukuran relatif dari cahaya yang harus dilihat dalam mata pemakainya dan dinilai oleh kunang-kunang lainnya

**Cuckoo Search**

CS adalah algoritma pencarian metaheuristik yang baru-baru ini dikembangkan oleh Yang dan Deb pada tahun 2009. Terinspirasi oleh parasit induk obligat perilaku beberapa spesies cuckoo, dikombinasikan dengan penerbangan Lévy yang menggambarkan pola mencari makan yang diadopsi oleh banyak hewan dan serangga, telah terbukti efektif dalam menyelesaikan masalah pengoptimalan berkelanjutan. Ini memberikan hasil yang efektif untuk fungsi multimodal dibandingkan dengan kedua algoritma genetik (GA) dan partikel Optimasi swarm (PSO). Dalam konteks ini, Discrete meningkatkan Cuckoo Search algorithm (DCS) dibandingkan dengan discrete particle swarm optimization (DPSO) simulasi genetik sistem koloni semut annealing dengan teknik optimasi partikel swarm (GSA-ACS-PSOT).

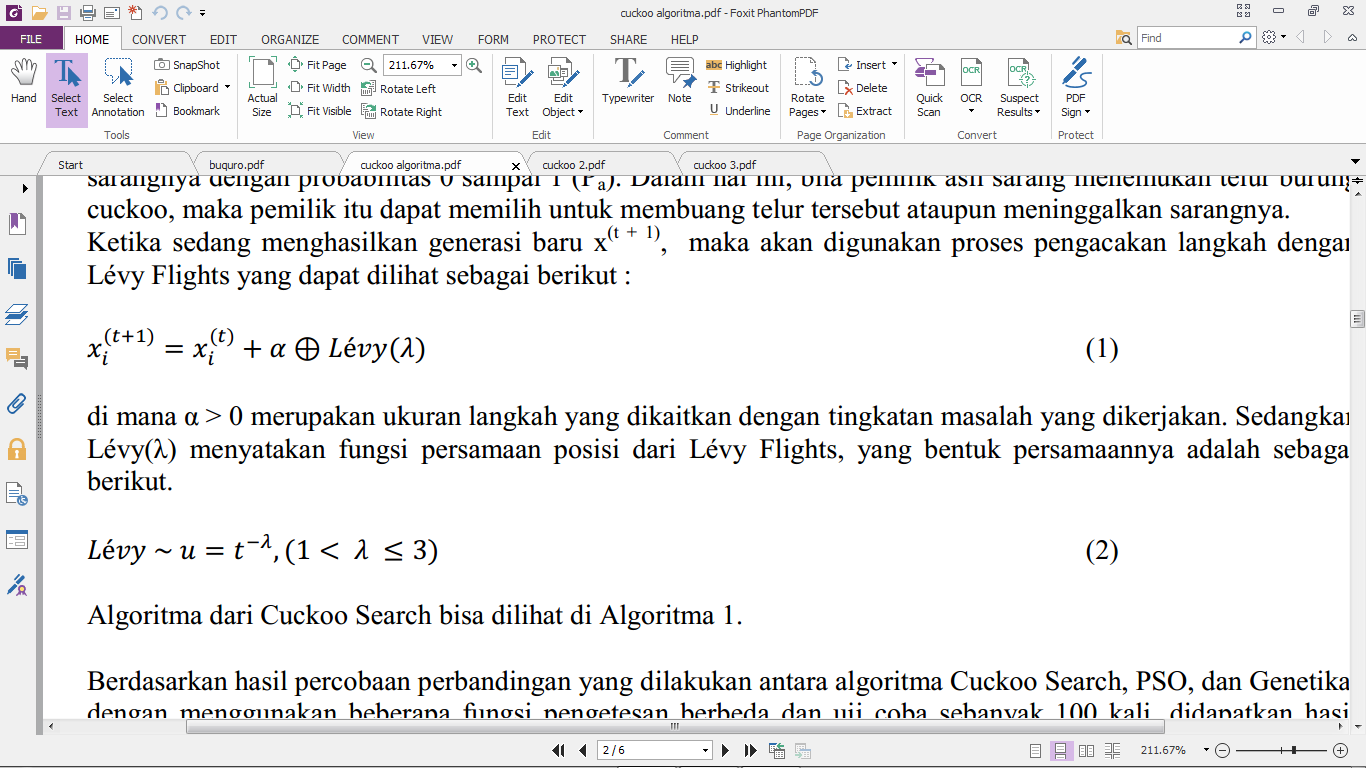
Di antara banyak fitur menarik dari cuckoos, yang disebut fitur parasitisme induk diadopsi oleh beberapa spesies yang paling banyak dipelajari dan didiskusikan. Cuckoos betina bertelur di sarang yang diamati sebelumnya dari jenis burung lain untuk membiarkan unggas inang menetas dan merenung cuckoo cewek muda. Untuk meningkatkan kemungkinan memiliki cuckoo baru dan mengurangi kemungkinan meninggalkan telur oleh burung inang, cuckoo menggunakan beberapa strategi.

Perilaku cuckoos digabungkan dalam algoritma pencarian cuckoo dengan Lévy penerbangan untuk mencari sarang baru secara efektif. Penerbangan Lévy, dinamai oleh Prancis matematikawan Paul Lévy, mewakili model perjalanan acak yang dicirikan oleh mereka panjang langkah yang mematuhi distribusi kekuatan hukum. Beberapa penelitian ilmiah telah dilakukan menunjukkan bahwa pencarian mangsa oleh pemburu mengikuti biasanya model yang sama dari Lévy penerbangan. Model ini biasanya diwakili oleh langkah acak kecil yang diikuti dijangka panjang dengan lompatan besar.

Berdasarkan tiga aturan tersebut langkah dasar dari Cuckoo Search (CS) dapat diringkas sebagai berikut :

xi (t+1) = xi (t) + α ⊕ L´evy (λ), (1)

Dimana xi (t+1) adalah generasi solusi baru, i adalah burng kukuk ke-i. α > 0 adalah ukuran langkah yang seharusnya berhubungan dengan skala kepentingan masalah tersebut. Dalam kebanyakan kasus dapat digunakan α = O(L/10), L adalah karakteristik skala kepentingan masalah. Sedangkan Lévy(λ) menyatakan fungsi persamaan posisi dari Lévy Flights, yang bentuk persamaannya adalah sebagai berikut.

 (2)

Persamaan 1 merupakan persamaan stokastik untuk langkah acak (random walk). Secara umum, langkah acak (random walk) adalah rantai Markov yang status / lokasi berikutnya hanya tergantung pada lokasi saat ini dan probabilitas transisi. Lambang ⊕ berarti perkalia entrywise.

**Bat Algorithm**

Algoritma kelelawar pertama kali diperkenalkan oleh Xin – She Yang pada tahun 2010 dalam jurnalnya yang berjudul A New Metaheuristic Bat – Inspired Algorithm. Kemampuan kelelawar dalam menangkap mangsanya merupakan hal yang sangat unik. Kelelawar mempunyai kemampuan handal dalam echolocation.Echolocation adalah kemampuan menentukan suatu lokasi dengan menggunakan echo atau gelombang. Kemampuan echolocation ini digunakan untuk mencari mangsa, menghindari penghalang dan menentukan kelelawar. Semua kelelawar menggunakan echolocation lokasi pada celah untuk bertengger dalam kegelapan (Yang, 2010).

Kelelawar memancarkan kadar pulsa suara yang sangat nyaring dan mendengar gelombang yang memantul dari objek di sekelilingnya. Kadar pulsa dapat bervariasi dan berkorelasi dengan strategi memburu segerombolan kelelawar. Berikut adalah representasi kemampuan kelelawar dalam algoritma kelelawar :

1. Kemampuan kelelawar dalam echolocation dapat dikembangkan secara variasi menjadi bat – inspiredalgorithm atau algoritma kelelawar. Sebagai contoh dapat digunakan beberapa rule atau aturan dari kemampuan untuk menentukan jarak dan kelelawar ‘mengetahui’ perbedaan mangsa dan penghalang.

2. Kelelawar terbang secara acak dengan kecepatan (Vi) pada posisi (Xi), dengan frekuensi (f), panjang gelombang (λ) dan kenyaringan (A) dalam mencari mangsa. Kelelawar mempunyai kemampuan untuk menyesuaikan panjang gelombang (λ) dari dari pancaran pulsa dan mengaturkadar dari dari emisi pulsa (r) ∈ [0,1] yang sangat penting menentukan target terdekat.

3. Walaupun kenyaringan dapat bervariasi berapapun itu, tapi diasumsikan kenyaringan dapat bervariasi dari bilang positif yang maksimum sampai bilang konstan minimum.

Adapun implementasi algoritma kelelawar sebagai feature selector pada kasus klasifikasi direpresentasikan sebagai berikut :

a. Posisi

Masing – masing posisi kelelawar direpresentasikan sebagai string biner dengan panjang N dimana N merupakan total fitur dalam dataset. Bit ‘1’ mengindikasikan fitur tersebut dipilih dan bit ‘0’ tidak dipilih

b.Kenyaringan

Kenyaringan (A) merupakan perubahan jumlah fitur dalam suatu iterasi selama pencarian lokal di sekitar kelelawar terbaik global dan pencarian lokal pada setiap kelelawar



Dimana:

Xnew: posisi kelelawar baru

Xold: posisi kelelawar sebelumnya

At: kenyaringan rata-rata semua kelelawar dalam satu iterasi

E: bilangan acak diantara -1 dan 1

Kenyaringan berada di rentang nilai maksimum dan minimum yang telah ditentukan.Penentuan nilai maksimum dan minimun bergantung pada domain aplikasi dan ukuran dataset. Secara empiris, penentuan nilai maksimum yaitu 1/5 N dimana N merupakan total fitur.

1. Frekuensi

Frekuensi merupakan elemen bilangan real yang akan mempengaruhi nilai kecepatan. Pemilihan nilai minimum dan maksimum disesuaikan dengan domain aplikasi.



Dimana:

fi : frekuensi kelelawar ke - i

β: variabel konstan diantara 0 dan 1

fmax : nilai maksimum frekuensi

fmin : nilai minimum frekuensi

1. Kecepatan

Kecepatan masing-masing kelelawar direpresentasikan dengan bilangan integer positif. Kecepatan akan mempengaruhi berapa jumlah fitur yang harus berubah dalam suatu kurun waktu. Kelelawar berkomunikasi satu dengan yang lainnya melalui solusi global terbaik dan bergerak menuju solusi terbaik global.



Dimana:

: kecepatan kelelawar ke – i pada iterasi ke-t

: kecepatan kelelawar ke – i pada iterasi ke-( − 1)

( x\*- )fi: merupakan perbedaan posisi kelelawar global (\*) dengan kelelawar ke – i pada iterasi ke – t

1. Pulse rate

Pulse rate (r) mempunyai peran untuk menentukan kapan pencarian lokal dari kelelawar terbaik global dilewati. Besar nilai pulse rateakan mengurangi kemungkinan melakukan pencarian lokal dan begitu pula sebaiknya. Maka, ketika kelelawar mendekati solusi terbaik, pulse rate akan perlahan – perlahan berkurang.



Dimana merupakan kadar pulsa kelelawar dengan solusi baru,

R0 merupakan kadar pulsa kelelawar dengan solusi sebelumnya, t adalah kurun waktu, γ adalah nilai konstan.

**Krill Herd**

Untuk mengatasi standar kinerja yang terbatas KH pada masalah kompleks, sebuah krill pencarian bebas baru algoritma herd diusulkan. Pencarian gratis strategi telah diperkenalkan ke dalam KH standar untuk menghindari semua individu krill terjebak ke dalam optima lokal.

Algoritma yang diusulkan dapat sangat memperkaya keragaman populasi krill dan meningkatkan akurasi perhitungan, yang mengarah ke kinerja pengoptimalan yang baik. Krill herd (KH) adalah metaheuristik baru metode optimasi kecerdasan swarm untuk memecahkan masalah optimasi, yang didasarkan pada simulasi perilaku menggembalakan individu krill. Waktu tergantung posisi krill individu dalam permukaan dua dimensi adalah ditentukan oleh tiga aksi utama berikut:

(1) gerakan yang diinduksi oleh individu krill lainnya;

(2) aktivitas mencari makan;

(3) difusi fisik.

Algoritma KH digunakan dengan Lagrangian model seperti ini :



Dimana Ni adlah motion dari individu krill lainnya; 𝐹𝑖 adalah foraging motion; dan 𝐷𝑖 adalah difusi fisik. Dalam KH, krill dipengaruhi oleh "tetangga" dan krill optimal, dan jarak penginderaan setiap krill adalah tetap. Namun di alam, aksi setiap krill gratis dan tidak pasti. DZ untuk mensimulasikan kebebasan ini, perilaku individu yang tidak pasti dari krill, makalah ini memperkenalkan strategi pencarian gratis ke dalam algoritma kawanan krill dan mengusulkan krill pencarian bebas baru algoritma kawanan (FSKH).

**SI for dimension reduction (swarm intelligence)**

Pengurangan dimensionalitas yang efisien dan kuat diperlukan metode untuk memproses pola dimensi tinggi, misalnya, untuk visualisasi atau memposting pemrosesan dengan algoritme simbolik. Dengan meningkatkan set data menjadi kelas masalah penting dalam pembelajaran mesin, dan berbagai macam metode telah diperkenalkan. Anehnya, tidak banyak algoritma berbasis swarm untuk diketahui. Metode ini menghitung pemetaan dari data berdimensi tinggi ruang ke ruang laten dari dimensi yang lebih rendah. Titik laten di ruang ini seharusnya melestarikan karakteristik topologi liontin berdimensi tinggi mereka seperti hubungan lingkungan dan jarak.

Pola dimensi kehilangan informasi sekecil mungkin. Banyak metode DR mencari pemetaan F: Rd → Rq dari ruang data berdimensi tinggi Rd ke ruang laten dari dimensi bawah Rq dengan q <d. Non-parametrik metode pengurangan dimensi menghitung satu set representasi dimensi rendah X = (x1, ..., xN) ∈ Rq × N untuk N pola pengamatan berdimensi tinggi Y = (y1, ..., yN) ∈ Rd × N.

Keputusan, informasi mana yang bisa hilang, dan mana yang harus dipertahankan dalam pemetaan F tergantung pada tujuan proses DR(dimension reduction), dan kesalahan fungsi yang ditentukan untuk metode yang digunakan. Banyak metode DR menggunakan implisit definisi masalah optimisasi yang mereka pecahkan. Namun, masalah belajar model fungsional F dapat menjadi masalah optimisasi yang sulit, karena laten variabel X tidak diketahui. Mempelajari pemetaan rekonstruksi f: Rq → Rd kembali dari laten ke ruang data juga bisa diinginkan. Beberapa metode mempelajari ini memetakan secara otomatis. Metode DR terkenal adalah PCA yang dibatasi untuk manifold linier. Penyisipan linear lokal (LLE), dan ISOMAP adalah terkenal untuk pengurangan dimensi non-linear.

Dalam sistem alam dapat diamati, di mana unit-unit yang relatif sederhana mengatur dalam kelompok. Bentuk organisasi kolektif dan terkoordinasi ini dikenal sebagai swarm intelligence. Kerugian perilaku sederhana dikompensasi oleh jumlah besar mereka, dan paralelisme masif. Kawanan terdiri dari sejumlah besar entitas sederhana yang bekerja sama untuk bertindak berorientasi pada tujuan. Alami dan buatan sistem telah menunjukkan menerapkan strategi solusi yang sukses.

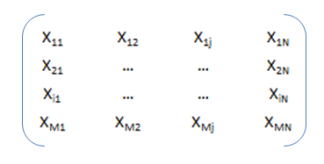
Algoritma PSO (Particle Swarm Optimization) adalah salah satu algoritma optimasi yang dapat digunakan untuk pengambilan keputusan. Tetapi bisa juga digunakan untuk pencarian jalur. Contoh yang dibahas kali ini adalah mengenai pencarian posisi dengan pengembalian nilai fungsi minimal. .

Particle Swarm Optimization adalah teknik optimasi dengan cara menghitung secara terus menerus calon solusi dengan menggunakan suatu acuan kualitas. Algoritma ini mengoptimasi permasalahan dengan cara menggerakan partikel / calon solusi di dalam ruang permasalahan menggunakan fungsi tertentu untuk posisi dan kecepatan dari partikel. Pergerakan partikel dipengaruhi oleh solusi terbaik partikel tersebut, dan solusi terbaik secara umum yang didapatkan dari partikel lain. Sekumpulan partikel ini dinamakan swarm, dan pada akhirnya swarm ini akan bergerak menuju kepada solusi terbaik.

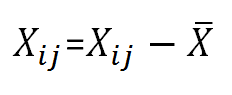
Ada banyak metode data mining yang dapat bekerja dengan baik pada data berdimensi rendah, sedangkan set data dalam data mining umum nya berdimensi tinggi yang memiliki banyak fitur sehingga selain di perlukan pemrosesan awal seperti binerisasi dan aggregasi, reduksi dimensi data juga penting dilakukan. jika kita mengurangi dimensionalitas data secara langsung maka kemungkinan ada karakteristik data yang hilang bisa saja fitur yang di buang adalah termasuk fitur yang berpengaruh.

Metode reduksi dimensionalitas data bekerja dengan cara tertentu untuk menangkap karakteristik data dengan memetakan set data dari dimensi semula ke dimensi lain yang relatif rendah. Pemetaan ini menghasilkan prinsipal componen yang kemudian dapat diambil komponen atau fitur dari dimensi baru yang mempunyai pengaruh yang besar pada set data dan membuang data yang tidak berpengaruh. salah satu metode yang sudah digunakan secara luas adalah Principal Component Analysis

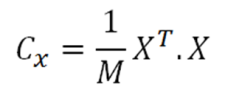
Metode ini melakukan pemetaan/tranformasi set data dari dimensi lama kedimensi yang baru dengan memanfaatkan tehnik Aljabar Linier. PCA memerlukan masukan data yang mempunyai sifat zero-mean pada setiap fitur nya. sifat zero-mean didapat dengan mengurangkan semua nilai dengan nilai rata-rata nya. Set data X dengan dimensi MxN dimana M adalah jumlah data dan N adalah jumlah Fitur, akan tampak seperti berikut :



Untuk fitur ke-j, semua nilai pada kolom tersebut dikurangi dengan nilai rata-rata nya, diformulasikan dengan.



Selanjutnya dilakukan perhitungan matrix kovarian dari matrix X, yaitu Cx. formula yang digunakan adalah dot product untuk setiap fitur nya.



Sifat-sifat Cx :

1.Cx adalah matrix simetris bujur sangkar berukuran NxN

2.Bagian Diagonal Utama (dari kiri atas ke bawah) adalah nilai varian masing-masing fitur sesuai index kolom nya.

3.Bagian selain diagonal utama adalah kovarian diantara pasangan dua fitur yang berkesesuaian.

Matrix Cx Menangkap kovarian diantara semua pasangan yang mungkin dari fitur data set data matrix x.

Nilai kovarian merefleksikan noise dan redudansi pada fitur

Tujuan PCA :

1.Meminimalkan redudansi yang di ukur oleh nilai jarak dari kovarian

2.Memaksimalkan nilai keluaran pemetaan, diukur dengan varian

Jika Y adalah matrix set data hasil pemetaan dan Cy adalah matrix covarian dari Y. maka :

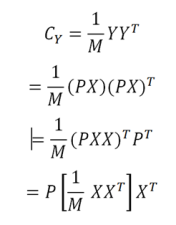
Semua elemen selain diagonal Utama Haruslah nol, maka Cy harus matrix diagonal(Cy Matrix terkorelasi).

Peletakan dimensi dalam Y dari kiri ke kanan di urutkan menurun.

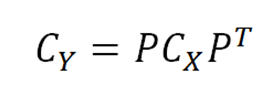
Cara yang umum digunakan adalah dengan mencari Eigen Value dan Eigen Vector

eigenVectordanValue

dengan mencari matrix ortonormal P dimana Y=PX dan Cy =(1/M)\*YYT adalah matrix diagonal, dan kolom dari P adalah komponen utama dari X, Persamaan Cy Bisa di jabarkan :



Dengan mensubstidusikan persamaan tersebut kita mendapatkan persamaan Cy berdimensi NxN :



**SI for Decision Trees Optimization**

Gagasan utama di balik Decision Trees diciptakan lebih dari 70 tahun yang lalu, dan saat ini mereka adalah salah satu alat Machine Learning yang paling kuat. Pengembang menggunakannya untuk kompetisi ilmu data, sistem peringkat, manajemen operasi, dan banyak masalah lain yang memerlukan pembuatan keputusan otomatis.

Keuntungan utama DT adalah bahwa mereka adalah metode “kotak putih”. Ini berarti bahwa kita dapat dengan mudah menjelaskan keputusan mereka, berbeda dengan Neural Networks yang kompleksitasnya biasanya terlalu tinggi.

Pelatihan Pohon Keputusan

Algoritma singkat belajar bekerja seperti ini:

Pada awalnya kita perlu menemukan beberapa fitur untuk memulai. Biasanya kami menghitung berapa banyak masing-masing fitur memengaruhi keputusan akhir secara individual dan memilih yang paling berpengaruh.

Selanjutnya, kita perlu menemukan cara membagi keputusan berdasarkan fitur ini. Jika variabel ini mungkin hanya memiliki 2 nilai (Ya / Tidak atau Benar / Salah dll), gunakan saja. Namun, jika fitur ini dapat mengambil nilai dalam rentang yang luas, kita perlu menghitung ambang mana yang akan memberi kita prediksi terbaik.

Akhirnya, dibagi dataset berdasarkan aturan keputusan baru dan mengulangi proses untuk mendapatkan pohon yang lebih dalam.

Boosting

Setiap pohon tunggal akan selalu cenderung meningkatkan bias dengan lebih mendalam. Bisa menggunakan puluhan / ratusan / ribuan untuk membuat satu prediksi. Teknik yang paling luar biasa untuk itu adalah Gradient Boosting. Singkatnya, secara iteratif menciptakan lebih banyak model (seperti DT) untuk mengurangi kesalahan bersama setelah setiap langkah.

**SI for Decision Trees Optimization**

Simpul dalam pohon keputusan bertindak sebagai aturan, secara rekursif mempartisi ruang keputusan. Jika aturannya mencakup satu fitur, seperti contoh yang diberikan di atas, maka partisi adalah sumbu parallel2, meskipun ini menghasilkan batas keputusan yang sederhana, karena kedalaman pohonnya peningkatan ruang fitur dapat dipartisi menjadi lebih banyak dan lebih berbeda bagian, dan batas keputusan yang dihasilkan menjadi lebih kompleks (meskipun piecewise linear, dan sumbu paralel di bagian ini).

Heuristik PSO awalnya diusulkan untuk optimalisasi terus menerus fungsi non-linear. Pekerjaan berikutnya di lapangan telah mengembangkan beberapa metode untuk operasinya dalam domain diskrit namun terus menerus domain tetap bidang prinsip penyebarannya.

Dalam PSO standar, populasi tetap dari solusi potensial M, {si} M i = 1, adalah dipertahankan, di mana masing-masing solusi ini (atau partikel) diwakili oleh titik di ruang P-dimensi (di mana P adalah jumlah parameter yang akan dioptimalkan).

Masing-masing solusi ini mempertahankan pengetahuan tentang 'terbaik' sebelumnya posisi yang dievaluasi (pi terbaik pribadi), dan juga memiliki akses ke 'terbaik' solusi yang ditemukan sejauh ini oleh populasi secara keseluruhan, g, yang menurut definisi adalah juga salah satu pribadi terbaik anggota swarm. Tingkat perubahan posisi sebuah partikel / solusi dari satu iterasi / generasi ke generasi berikutnya bergantung pada posisi terbaik lokal sebelumnya, posisi terbaik global, dan lintasan sebelumnya (kecepatannya, vi). Rumus umum untuk menyesuaikan parameter jth

Kecepatan partikel adalah:



Di mana w, c1, c2, χ ≥ 0. w adalah inersia partikel (berapa banyak partikelnya) kecepatan sebelumnya mempengaruhi lintasan berikutnya), c1 dan c2 adalah kendala pada kecepatan menuju global dan lokal terbaik dan χ merupakan kendala pada keseluruhan pergeseran posisi (sering kecepatan mutlak maksimum, Vmax juga diterapkan).

r1 dan r2 adalah penarikan acak dari distribusi seragam kontinyu, yaitu r1,

r2 ∼ U (0, 1). Dalam [10] model terakhir yang disajikan memiliki w dan χ tetap pada 1, dan c1 dan c2 diperbaiki pada 2. Pekerjaan selanjutnya cenderung ke berbagai istilah inersia ke bawah selama pencarian untuk membantu konvergensi terakhir.

Heuristik PSO telah terbukti menjadi optimasi yang sangat populer teknik, dengan reputasi untuk konvergensi yang relatif cepat, dan dengan demikian adalah miliknya.

**SI for Artificial Neural Network Optimization**

Properti dari Sworm Intelligence utk ANN terdiri dari banyak individu, yang relatif homogen (identik atau mereka milik beberapa tipologi)

● interaksi didasarkan pada perilaku sederhana aturan yang mengeksploitasi hanya informasi lokal yang individu bertukar secara langsung atau melalui lingkungan Hidup

● perilaku keseluruhan sistem hasil dari interaksi, yaitu perilaku kelompok berorganisasi

● skalabilitas - sistem dapat mempertahankan fungsinya sambil meningkatkan ukurannya tanpa perlu mendefinisikan kembali cara bagian-bagiannya berinteraksi

● tindakan paralel - individu menyusun segerombolan dapat melakukan tindakan yang berbeda dalam tempat yang berbeda pada saat yang bersamaan

● toleransi kesalahan - individu yang gagal dapat dengan mudah diberhentikan dan diganti oleh yang lain itu berfungsi penuh

masalah optimasi P → triple (S, Ω, f)

●S adalah ruang pencarian yang ditentukan di atas himpunan terbatas

variabel keputusan Xi, i = 1,. . . , n

Ω adalah seperangkat batasan di antara variabel

f: S → R+ adalah fungsi obyektif yang menugaskan a nilai biaya positif untuk setiap solusi S

tujuan - temukan solusi s that S sedemikian rupa sehingga:

f (s) ≤ f (s), s ∀ ∈ S (f minimisasi)

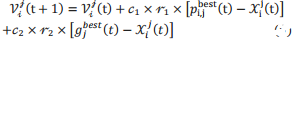
f (s) ≥ f (s), s ∀ ∈ S (f maksimalisasi)

Metode komputasi ini meningkatkan dan mengoptimalkan a masalah melalui solusi iteratif dengan terbang di sekitar cari ruang untuk menemukan solusi terbaik.

Dalam banyak aspek, PSO mirip dengan evolusi komputasi, tetapi pada dasarnya tidak ada operator evolusi di PSO. Setiap solusi kandidat diwakili sebagai partikel yang memiliki kecepatan, posisi dan pencarian solusi ruang secara iteratif. Setiap partikel dikaitkan dengan dua properti (posisi X dan kecepatan V), misalkan X dan V dari saya partikel th diberikan sebagai:



dimana N merepresentasikan dimensi masalah



**K-Means Clustering based on Swarm Intelligence**

Algoritma standar itu pertama kali diperkenalkan oleh Stuart Lloyd pada tahun 1957 sebagai teknik modulasi kode pulsa. Pengelompokan K-Means algorithm adalah metode analisis cluster berbasis partisi. Menurut algoritma ini, pertama-tama kita memilih objek k sebagai pusat klaster awal, kemudian hitung jarak antara setiap pusat cluster dan setiap objek dan menetapkannya ke cluster terdekat, perbarui rata-rata semua kluster, ulangi proses ini sampai fungsi kriteria bertemu. Kotak kriteria kesalahan untuk pengelompokan dijelaskan dibawah ini:



xij adalah contoh j dari i-class, mi adalah pusat dari i-class, ni adalah jumlah sampel i-class, Algoritma langkah ditampilkan ara (1).

Algoritma K-means pengelompokan hanya digambarkan sebagai

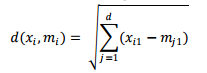
berikut:

Input: N objek menjadi klaster {x1, x2 ........ xn}, angka dari kluster k;

Output: k cluster dan jumlah ketidaksamaan antara setiap objek dan pusat gugus terdekatnya adalah yang terkecil;

• Memilih objek k sebagai pusat klaster awal secara arbitrer (m1, m2 ... mk);

• Hitung jarak antara setiap objek xi dan masing-masing pusat klaster, lalu tetapkan setiap objek ke yang terdekat klaster, rumus untuk menghitung jarak sebagai:



i = 1,2 ........ N

j = 1,2 ........ k

d (xi, mj) adalah jarak antara data i dan cluster j;

Hitung rata-rata objek di setiap cluster sebagai pusat klaster baru,



i = l, 2. . . k; Ni adalah jumlah sampel klaster arus i;

Keuntungan dari pengelompokan K-mean

• Pengelompokan K-mean sederhana dan fleksibel.

• Algoritma pengelompokan K-mean mudah dimengerti dan mengimplementasikan.

jaminan untuk solusi optimal

**DENCLUE clustering based on Swarm Intelligence**

Density based clustering bekerja dengan mendeteksi area di mana titik terkonsentrasi dan di mana mereka dipisahkan oleh area yang kosong atau jarang. Titik yang bukan bagian dari gugus diberi label sebagai derau.

Density based clustering menggunakan algoritma pengelompokan pembelajaran mesin tanpa pengawasan yang secara otomatis mendeteksi pola berdasarkan murni pada lokasi spasial dan jarak ke sejumlah tetangga tertentu. Algoritma ini dianggap tidak diawasi karena mereka tidak memerlukan pelatihan apa pun tentang apa artinya menjadi sebuah kelompok.

Density based clustering menyediakan tiga Metode Clustering yang berbeda untuk menemukan klaster dalam data titik:

**Defined distance** (DBSCAN) - Menggunakan jarak tertentu untuk memisahkan kelompok padat dari derau yang lebih jarang. Algoritma DBSCAN adalah yang tercepat dari metode pengelompokan, tetapi hanya cocok jika ada Jarak Penelusuran yang sangat jelas untuk digunakan, dan itu berfungsi dengan baik untuk semua kluster potensial. Ini mengharuskan semua kelompok yang berarti memiliki kepadatan yang sama.

**Self-adjusting** (HDBSCAN) —Menggunakan berbagai jarak untuk memisahkan kelompok dengan keragaman yang bervariasi dari kebisingan yang lebih jarang. Algoritma HDBSCAN adalah yang paling digerakkan oleh data dari metode pengelompokan, dan dengan demikian membutuhkan input pengguna yang paling sedikit.

**Multi-scale** (OPTICS) —Menggunakan jarak antar fitur yang berdekatan untuk membuat plot reachability yang kemudian digunakan untuk memisahkan cluster dengan keragaman yang bervariasi dari noise. Algoritma OPTICS menawarkan fleksibilitas yang paling dalam fine-tuning cluster yang terdeteksi, meskipun secara komputasi intensif, terutama dengan Jarak Pencarian yang besar.

Density based clustering, jalur untuk Keluaran Fitur dan nilai yang mewakili jumlah minimum fitur yang diperlukan untuk dianggap sebagai kluster. Bergantung pada Metode Klaster yang dipilih, mungkin ada parameter tambahan untuk ditentukan seperti yang dijelaskan di bawah ini.

Density based clustering juga penting dalam perhitungan jarak inti, yang merupakan pengukuran yang digunakan oleh ketiga metode untuk menemukan cluster. Secara konseptual, jarak inti atau setiap titik, adalah pengukuran jarak yang diperlukan untuk melakukan perjalanan dari setiap titik ke jumlah fitur minimum yang ditentukan. Jadi, jika Fitur Minimum per Cluster besar dipilih, maka jarak inti yang sesuai akan lebih besar. Jika fitur Minimum Minimum per Cluster dipilih, maka jarak inti yang sesuai akan lebih kecil. Pada batas-batas cluster, titik-titik akan memiliki jarak inti yang besar, (dan kemungkinan tidak termasuk dalam kelompok). Jarak-inti, terkait dengan parameter Jarak Pencarian, yang digunakan baik oleh metode Jarak Ditentukan (DBSCAN) dan Multi-skala (OPTICS).

**Operation Research & Optimization:**

**Simplex Method and Sensitivity Analysis**

Pada analisis sensitivitas LP berdasarkan solusi optimal yang ditemukan melalui kedua jenis metode. Dengan analisis sensitivitas, dipastikan dampak ketidakpastian terkait nilai parameter pada kualitas solusi optimal dan informasi terkait. Dalam LP parameter (input data) dari model dapat berubah dalam batas-batas tertentu mempertahankan optimalitas solusi atau beberapa parameter lain yang ditentukan oleh solusi optimal.

Dengan solusi optimal dari masalah LP, informasi ekonomi tambahan dapat diperoleh, yaitu harga ganda sumber daya terkait dan biaya aktivitas yang dikurangi (diperhitungkan).

• Harga ganda atau bayangan dari sumber daya adalah nama teknis dari unit senilai sumber daya itu. Nilai unit sumber daya adalah perubahan dalam nilai obyektif yang optimal per unit perubahan dalam ketersediaan sumber daya. Mengetahui kisaran sumber daya tertentu yang membuat harga ganda dari sumber daya itu tidak berubah memberikan dukungan untuk pengambilan keputusan ilmiah.

• Biaya yang dikurangi didefinisikan sebagai

Mengurangi biaya per unit = Biaya sumber daya yang dikonsumsi per unit - Pendapatan per unit.

Ini digunakan untuk mengevaluasi "daya tarik" dari "produk" dari sudut pandang optimasi yang bergantung pada nilai relatif dari pendapatan per unit dan biaya sumber daya yang dikonsumsi oleh satu unit.

Analisis kepekaan berhubungan dengan menentukan:

• harga ganda (sesuai dengan solusi optimal) dan ketentuan mengenai perubahan ketersediaan sumber daya yang akan menjaga harga ganda saat ini berlaku;

• biaya yang berkurang (sesuai dengan solusi optimal) dan kondisi mengenai perubahan koefisien fungsi obyektif yang akan menjaga solusi optimal saat ini (titik) tidak berubah.

• Analisis sensitivitas aljabar menggunakan (z-baris dalam) tablo optimal untuk menentukan harga ganda dan biaya yang dikurangi, dan dengan menghitung ulang tablo simpleks optimal dengan sisi kanan yang dimodifikasi (atau koefisien yang dimodifikasi dari fungsi tujuan) di ketentuan penyimpangan dari data asli menentukan:

(I) Kelayakan (kondisi dan) rentang untuk sumber daya yang memastikan bahwa harga ganda yang terkait dengan solusi optimal tetap berlaku. Nilai saat ini dari fungsi obyektif dan titik optimum saat ini akan berubah karena perubahan parameter, tetapi nilai baru mereka dapat dengan mudah menghitung berdasarkan ekspresi mereka dalam hal penyimpangan parameter saat ini, ditemukan dalam simpleks optimal yang direkomputasi. tablo.

(ii) Kondisi optimal dan rentang optimal untuk biaya unit yang akan membuat solusi optimal tidak berubah. Nilai saat ini dari fungsi obyektif akan berubah karena perubahan parameternya, tetapi nilai barunya dapat dengan mudah dihitung berdasarkan (barisnya dalam) simpang siur simplex optimal yang direkomputasi.

Bentuk dasar primal dari masalah LPdan menggunakan tablo yang terkait (primal). Pendekatan ini memberikan motivasi yang lebih baik untuk nama "variabel dasar", menggambarkan bahwa penggunaan label "−1" hanyalah konvensi, dan lebih nyaman untuk operasi pivot dan ruang lingkup analisis sensitivitas.

Dua kasus akan dipertimbangkan untuk analisis sensitivitas:

1. Sensitivitas solusi optimal untuk perubahan ketersediaan sumber daya (sisi kanan kendala).

2. Sensitivitas solusi optimal terhadap perubahan laba unit atau biaya unit (koefisien fungsi obyektif).

(I) Analisis Kepekaan Grafis

• Perubahan ketersediaan sumber daya akan memindahkan garis yang sesuai dengan paralel pembatasnya untuk dirinya sendiri. Harga ganda untuk sumber daya tertentu tetap berlaku untuk perubahan (bertambah atau berkurang) dalam sumber daya yang memindahkan garis yang sesuai dengan paralel pembatasnya sendiri ke titik mana pun pada segmen garis yang sesuai dengan kendala sumber daya lain yang terkait dengan solusi optimal (sudut titik). Titik ekstrim dari garis segmen ini menentukan kisaran penerapan harga ganda yang diberikan. Kami menyebut kisaran tersebut sebagai rentang kelayakan untuk (harga ganda yang berhubungan dengan) sumber daya tertentu.

    Harga ganda dan kisaran kelayakannya, ditentukan dengan menggunakan solusi optimal dari masalah LP, memungkinkan membuat keputusan ekonomi tentang masalah itu (ketika varian perubahan dalam sumber daya dipertimbangkan).

**Transportation Algorithm**

Algoritma transportasi mengikuti langkah-langkah yang tepat dari metode simpleks, Namun, daripada menggunakan tablo simpleks biasa, terdapat keuntungan dari struktur khusus dari model transportasi untuk mengatur perhitungan dalam bentuk yang lebih nyaman.

Ringkasan Algoritma Transportasi. Langkah-langkah dari algoritma transportasi

adalah paralel tepat dari algoritma simpleks.

Langkah 1. Tentukan solusi awal yang layak, dan lanjutkan ke langkah 2.

Langkah 2. Gunakan kondisi optimal dari metode simpleks untuk menentukan

memasukkan variabel dari antara semua variabel non-dasar. Kalau yang optimal

kondisi puas, berhenti. Jika tidak, lanjutkan ke langkah 3.

Langkah 3. Gunakan kondisi kelayakan metode simpleks untuk menentukan variabel meninggalkan dari antara semua variabel dasar saat ini, dan temukan solusi dasar baru.

Model transportasi umum dengan sumber m dan n tujuan memiliki persamaan m+ n kendala, satu untuk setiap sumber dan setiap tujuan. Namun, karena model transportasi selalu seimbang (jumlah pasokan = jumlah permintaan), salah satu dari persamaan ini adalah redundan. Dengan demikian, model memiliki m + n - 1 persamaan kendala independen, yang berarti bahwa solusi dasar awal terdiri dari m + n- 1 variabel dasar. Jadi, dalam Contoh di bawah ini, solusi awal memiliki 3 + 4 - 1 = 6 variabel dasar.

Struktur khusus masalah transportasi memungkinkan pengamanan non-artifisial mulai solusi dasar menggunakan salah satu dari tiga metode:

1. Metode sudut barat laut

2. Metode biaya rendah

3. Metode aproksimasi vogel

Iteratif Perhitungan Algoritma Transportasi

Setelah menentukan solusi awal, kami menggunakan algoritme berikut untuk menentukan solusi optimal:

Langkah 1: Gunakan kondisi optimal simplex untuk menentukan variabel yang masuk sebagai variabel nonbasic saat ini yang dapat meningkatkan solusi. Jika kondisi optimal puas, berhenti. Jika tidak, lanjutkan ke langkah 2.

Langkah 2: Tentukan variabel yang ditinggalkan menggunakan kondisi simplex probabilitas. Perubahan dasar, dan kembali ke langkah 1.

**Network Models (Minimal Spanning Trees and Shortest Route Problem)**

Mengingat satu set lokasi dan kemungkinan jalan yang akan dibangun di antara pasangan kota-kota dengan biaya terkait, perlu dinentukan biaya minimum jaringan jal an yang menghubungkan semua lokasi.

Langkah 1. Inisialisasi

Pilih simpul apa pun di jaringan, misalkan saya. Biarkan S = {i}, dan biarkan S menjadi set berisi semua node selain i. Secara formal 

Langkah 2. Atur pembaruan S

Di antara ujung-ujungnya {i, j} menyeberang dari S ke S, temukan ujung-ujungnya dengan biaya terkecil (ikatan rusak secara sewenang-wenang), yaitu, temukan 



Langkah 3. Tes pemutusan

Jika set S berisi semua node, hentikan.

Model jaringan adalah model database yang dirancang sebagai pendekatan fleksibel untuk merepresentasikan objek dan hubungannya. Fitur unik dari model jaringan adalah skemanya, yang dipandang sebagai grafik di mana jenis hubungan adalah busur dan jenis objek adalah simpul. Tidak seperti model database lainnya, skema model jaringan tidak terbatas pada kisi atau hierarki; pohon hirarkis digantikan oleh grafik, yang memungkinkan untuk koneksi yang lebih mendasar dengan simpul.

**CPM and PERT**

Metode Jalur Kritis (CPM) adalah suatu teknik perencanaan yang berdasarkan suatu diagram jaringan kerja yang berisi lintasan-lintasan kegiatan dan urutan-urutan peristiwa yang ada selama penyelenggaraan proyek yang digambarkan kedalam suatu simbol-simbol.

Didalam suatu kegiatan yang besar, seperti penyelesaian suatu proyek, yang mencakup kegiatan-kegiatan yang terpisah tetapi berkaitan satu sama lainnya senantiasa ada sejumlah kegiatan yang dianggap “Vital” bagi selesainya proyek waktu penyelesaiannya tidak dapat ditunda-tunda kalau kita tidak ingin terjadi keterlambatan secara menyeluruh dari penyelesaian proyek.

Pada umumnya kegiatan yang bersifat kritis dapat ditemukan pada suatu jalur atau lintasan sejak awal sampai akhir proyek.Kemungkinan untuk menetapkan adanya lintasan kritis dalam suatu jaringan digunakan salah satu atau metode jalur kritis. Jumlah simbol yang digunakan dalam sebuah jaringan kerja, minimum ada dua macam dan maksimum ada tiga macam. Macam-macam simbol tersebut adalah:

a. Anak Panah

Anak panah ini melambangkan sebuah kegiatan dari suatu proyek. Pada umumnya nama kegiatan dicantumkan diatas anak panah dan lama kegiatan dibawahnya. Ekor anak panah ditasirkan sebagai kegiatan dimulai dan kepalanya ditafsirkan sebagai kegiatan selesai. Lamanya kegiatan adalah jarak waktu antara kegiatan dimulai dengan kegiatan selesai. Pada lamanya kegiatan diberi kode huruf besar A,B,C dan seterusnya.



b. Lingkaran

Lingkaran yang melambangkan peristiwa selalu digambarkan lingkaran yang terbagi atas tiga bagian ruangan: Ruangan sebelah atas merupakan tempat bilangan atau huruf yang menyatakan peristiwa. Ruangan sebelah kiri bawah merupakan yang menyatakan lamanya hari (waktu satuan hari) yang merupakan saat paling awal peristiwa yang bersangkutan. Ruangan sebelah kanan bawah merupakan tempat bilangan yang menyatakan saat paling lambat peristiwa yang bersangkutan boleh terjadi. Selisih waktu dari kedua saat tersebut adalah tenggang waktu peristiwa (Slack) berharga positif. Ada kemungkinan tenggang waktu tersebut berharga nol, maka peristiwa yang bersangkutan merupakan peristiwa yang kritis, jika berharga negatif peristiwa tersebut adalah peristiwa super kritis dan ini bertanda bahwa proyek tidak akan selesai pada waktu yang telah ditetapkan.



Keterangan: NE = Number of Efent

EET = Earlist Event Time = Waktu paling awal

LET = Latest Event Time = Waktu paling akhir

c. Anak Panah Terputus-putus (Dummy)

Anak panah terputus-putus melambangkan hubungan antar peristiwa, sama halnya dengan anak panah yang melambangkan kegiatan. Hubungan antar kegiatan (Dummy) tidak membutuhkan waktu, sumber daya dan ruangan. Oleh karena itu hubungan antar peristiwa tidak perlu diperhitungkan. Dummy ini menyatakan logika ketergantungan yang patut diperhatikan.



Untuk dapat membaca diagram jaringan kerja sebuah proyek perlu dijelaskan pengertian dasar hubungan antara simbol yang ada dalam setiap diagram jaringan kerja. Notasi yang dipakai dalam penjelasan mengenai hubungan antar simbol ini adalah sebagai berikut:

D (x) = Durasi kegiatan X

ES (x) = Waktu mulai paling cepat untuk kegiatan X

EF (x) = Waktu selesai paling cepat untuk kegiatan X

LS (x) = Waktu mulai paling lambat untuk kegiatan X

LF (x) = Waktu selesai paling lambat untuk kegiatan X

TF (x) = Tenggang waktu total untuk kegiatan X

FF (x) = Tenggang waktu bebas untuk kegiatan X

S = Waktu mulai proyek

T = Waktu penyelesaian Proyek

Adapun perhitungan didalam Critical Path Method adalah sebagai berikut:

a. Perhitungan kedepan

ES (x) = S untuk kegiatan permulaan

ES (x) = Maksimum EF (semua pendahuluan kegiatan)

EF (x) = ES (x) + D (x)

b. Perhitungan kebelakang

LF (x) = T untuk kegiatan penyelesaian

LF (x) = Minimum LF (semua pengikut kegiatan X)

LS (x) = LF (x) + D (x)

c. TF (x) = LS (x) – (x).

= LF (x) – EF (x)

Dengan melakukan perhitungan ini maka bias diperoleh durasi proyek, dan lintasan kritis untuk proyek. Untuk lebih jelasnya lihat pada gambar berikut:



Contoh Perhitungan:

EF (A) = 0 + 5

= 5

LS (H) = 24 – 8

= 16

TF (D) = 13 – 13 = 0

= 16 – 13 = 0

2.6.4.1. Metode Penyusunan Jaringan Kerja

Unsur yang diperlukan dalam membuat jaringan kerja proyek adalah jenis kegiatan, logika ketergantungan, perkiraan waktunya dan metode pelaksanaan. Jika hal tersebut diatas diketahui maka tidak dapat menghitung setiap kegiatan yaitu waktu mulai paling cepat, waktu selesai paling lambat, tenggang waktu total dan tenggang waktu bebas. Adapun langkah-langkah didalam menyusun jaringan kerja adalah sebagai berikut:

a. Inventarisasi Kegiatan

Proses inventarisasi kegiatan dilakukan denagn memecah suatu proyek menjadi beberapa bagian komponen utama proyek. Selanjutnya komponen utama ini dipecah menjadi beberapa komponen lagi, dan pada tahapan akhir akan didapat paket-paket pekerjaan. Proses ini biasa disebut Work Break Down Structure (WBS).

b. Logika Ketergantungan Kegiatan

Setelah semua jenis kegiatan diketahui maka kita dapat membuat jaringan kerja berdasarkan logika ketergantungan ini akan menghasilkan berbagai bentuk jaringan kerja. Berikut adalah contoh jaringan kerja yang paling sederhana :



Pada gambar nampak bahwa setiap kegiatan tidak dapat dikerjakan apabila kegiatan pendahuluannya belum selesai dikerjakan.

c. Perkiraan Waktu

Perkiraan waktu yang dimaksud adalah jangka waktu yang dibutuhkan untuk menyelesaikan setiap kegiatan. Pada umumnya apabila waktu pelaksanaan bertambah panjang maka biaya pelaksanaannya akan bertambah besar, dan demikian pula sebaliknya. Hal ini disebabkan oleh biaya overhead yang besarnya tergantung dari waktu pelaksanaan. Ada beberapa faktor yang menentukan lamanya suatu kegiatan yaitu :

1. Volume Pekerjaan

Kegiatan yang volumenya lebih besar membutuhkan waktu penyelesaian lebih lama dibandingkan dengan volume pekerjaan yang lebih kecil.

2. Tenaga Kerja

Tenaga kerja yang terampil, terdidik dan berpengalaman akan mempunyai produktifitas yang tinggi sehingga dapat menyelesaikan pekerjaan dengan cepat dan mutu yang baik.

3. Cuaca

Faktor cuaca memegang peranan penting dalam pelaksanaan dilapangan. Apabila cuaca buruk akan menyebabkan terganggunya pelaksanaan pekerjaan.

4. Lokasi Proyek

5. Prosedur Perkiraaan Waktu

Alokasi Sumber

Untuk mengetahui penggunaan sumber daya dalam menyelenggarakan proyek perlu diketahui lebih dahulu dua hal yaitu:

a. Alokasi sumber tidak terbatas

Alokasi sumber tidak terbatas adalah mengatur jadwal aktivitas-aktivitas sedemikian rupa, sehingga tingkat kebutuhan sumber dari waktu ke waktu menjadi seimbang. Metode peralatan sumber ini diberikan dengan minimasi jumlah kuadrat terkecil sumber yang dibutuhkan setiap satuan wkatu dalam jadwal proyek. Prosedurnya adalah sebagai berikut:

1. Susunlah peta jadwal proyek bersangkutan berdasarkan network yang telah dibuat. Aktivitas-aktivitas ditabelkan menurut nomor.

2. Lakukan penjadwalan kembali mulai dari aktivitas yang paling bawah sampai aktivitas paling atas dalam network planning, sehingga diperoleh alokasi sumber yang paling rata untuk setiap penjadwalan aktivitas tersebut.

b. Alokasi sumber terbatas

Alokasi sumber terbatas adalah pengaturan jadwal aktivitas-aktivitas sedemikian sehingga kebutuhan sumber tidak melampaui tingkat kemampuan sumber.

Teknik Penilaian dan Peninjauan Program (Program Evaluation and Review Technique / PERT)

Bila CPM memperkirakan waktu komponen kegiatan proyek dengan pendekatan deterministik satu angka yang mencerminkan adanya kepastian, maka PERT direkayasa untuk menghadapi situasi dengan kadar ketakpastian (uncertainty) yang tinggi pada aspek kurun waktu kegiatan. PERT memakai pendekatan yang menganggap bahwa kurun waktu kegiatan tergantung pada banyak faktor dan variasi, sehingga lebih baik perkiraan diberi rentang, yaitu dengan memakai tiga angka estimasi. PERT juga memperkenalkan parameter lain yang mencoba mengukur ketakpastian tersebut secara kuantitatif seperti deviasi standard an varians. Metode ini memiliki cara yang spesifik untuk menghadapi hal tersebut yang memang hampir selalu terjadi pada kenyataannya dan mengakomodasinya dalam berbagai bentuk perhitungan.

PERT mula-mula diperkenalkan dalam rangka merencanakan dan mengendalikan proyek besar dan kompleks, yaitu pembuatan peluru kendali yang dapat diluncurkan dari kapal selam dibawah permukaan air. Proyek tersebut melibatkan beberapa kontraktor dan rekaman dimana pemilik arti penting dalam penyelenggaraan proyek, seperti milestone dapat dicapai oleh mereka, atau bila tidak, seberapa jauh penyimpangannya. Hal ini menunjukan PERT lebih berorientasi ke terjadinya peristiwa sedangkan CPM condong ke orientasi kegiatan.

Project Evaluation And Review Technique (PERT), metode ini dikembangkan pada saat angkatan laut Amerika sedang mengembangkan polaris missile system program. Tim pengembang metode ini adalah dari angkatan laut Amerika, Lockheed Aircraft Corporation sebagai kontraktor dan Boos konsultan, PERT direkayasa untuk menghadapi situasi dengan kadar ketidakpastian yangn tinggi pada aspek kurun waktu kegiatan. PERT juga memakai pendekatan yang menganggap bahwa kurun waktu kegiatan tergantung pada banyak factor dan variasi. Dalam perhitungannya PERT memasukkan unsure-unsur ketakpastian seperti gangguan dalam pelaksanaan proyek untuk data perhitungan duarasi masing-masing kegiatan, sehingga dalam perhitungannya, PERT menggunakan tiga macam waktu. Metode ini digunakan untuk memecahkan suatu masalah jaringan kerja yang menggunakan taksiran waktu untuk setiap aktivitas dengan cara probabilistik, karena setiap kegiatan waktu merupakan variabel acak.

Dalam penaksiran waktu, perencanaan menggunakan macam-macam taksiran waktu pelaksanaan, yaitu:

1. Waktu optimistic (a), yaitu kemungkinan bahwa kegiatan dapat diselesaikan dalam waktu yang lebih singkat.

2. Waktu paling banyak timbul (ml), yaitu taksiran waktu yang biasanya terjadi dalam keadaan normal.

3. Waktu pesimistik (b), yaitu kemungkinan bahwa kegiatan dapat diselesaikan dalam waktu yang lebih lama.

**Goal Programming**

Pemrograman tujuan adalah cabang dari optimasi multiobjective, yang pada gilirannya adalah cabang dari analisis keputusan multi-kriteria (MCDA). Ini adalah program pengoptimalan. Hal ini dapat dianggap sebagai perluasan atau generalisasi pemrograman linier untuk menangani berbagai ukuran objektif yang biasanya bertentangan. Masing-masing langkah ini diberi sasaran atau nilai target yang ingin dicapai. Penyimpangan yang tidak diinginkan dari set nilai target ini kemudian diminimalkan dalam fungsi pencapaian. Ini bisa berupa vektor atau jumlah tertimbang tergantung pada varian pemrograman tujuan yang digunakan. Karena kepuasan target dianggap memuaskan pengambil keputusan, diasumsikan filosofi kepuasan yang mendasari. Pemrograman tujuan digunakan untuk melakukan tiga jenis analisis:

Tentukan sumber daya yang dibutuhkan untuk mencapai serangkaian tujuan yang diinginkan.

Tentukan tingkat pencapaian tujuan dengan sumber daya yang tersedia.

Memberikan solusi memuaskan terbaik di bawah berbagai jumlah sumber daya dan prioritas tujuan.

Beberapa informasi identik dengan pemrograman linier tetapi ada beberapa perbedaan. Catatan pertama bahwa tidak ada fungsi obyektif. Kedua, perhatikan bahwa ada 4 kolom tambahan di awal (kiri) tabel sebelum variabel keputusan. Kolom ekstra ini digunakan untuk tujuan dan bukan untuk kendala (di mana Anda dapat melihat bahwa mereka nol).

Tujuan / Batasan. Di setiap baris tabel, kami memasukkan kendala atau sasaran. Dua baris pertama mewakili kendala. Karena ini adalah kendala, empat kolom pertama tidak digunakan (angka 0 dimasukkan). Kendalanya dimasukkan dalam mode biasa.

Tiga baris berikutnya mewakili tujuan dan ada dua aspek untuk tujuan ini. Karena ini adalah sasaran, perhatikan bahwa tanda di baris adalah '='. Nilai-nilai di bawah x1 dan x2 berfungsi untuk menciptakan tujuan bersama dengan variabel d + dan menunjukkan dengan seberapa banyak overachieve atau kurang mencapai tujuan. Misalnya, baris 3 dalam tabel berarti

x1 - (d1 +) + (d1-) = 14.

Jika x1 di bawah 14 maka d1- mewakili jumlah di bawah ini sementara jika x1 di atas 14 maka d1 + mewakili jumlah yang kita keluarkan.

Demikian pula, baris berikutnya (tujuan 4) mewakili

16x1 + 12x2 - (d2 +) + (d2 -) + = 270

Jadi d2 + dan d2- mewakili jumlah laba di atas 270 dan di bawah 270 masing-masing.

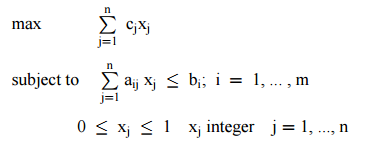
Secara matematis, tujuannya adalah d1 +, d1-, d2 +, d2-, d3 +, d3-. Pertanyaannya adalah bagaimana kita ingin memesan atau membobot sasaran-sasaran ini. Artinya, bagaimana kita membedakan pentingnya masing-masing dari 6 tujuan ini?

Prioritas dan bobot: Pertama, ada 6 gol (d1 +, d1-, d2 +, d2-, d3 +, d3-) dalam contoh ini, tetapi kami tidak peduli jika kami terlalu mengandalkan tujuan untung 270 juga tidak peduli jika kami memproduksi lebih banyak dari 5 unit produk 2. Oleh karena itu bobot dan prioritas kedua sasaran ini telah ditetapkan menjadi 0. Prioritas untuk empat sasaran lainnya berkisar dari 1 hingga 3. Arti prioritas yang berbeda adalah urutan di mana tujuan dipenuhi. . Dengan kata lain, gol dengan prioritas 1 harus dipenuhi sebelum gol dengan prioritas 2 yang harus dipenuhi sebelum gol dengan prioritas 3 dan seterusnya. Dalam contoh ini pertama kita ingin membuat tepat 14 unit produk 1, maka kami ingin menjamin tingkat keuntungan minimum kami sebesar 270, maka kami ingin mencoba menjamin tingkat batch minimum kami 5 untuk produk 2.

Dalam setiap prioritas adalah mungkin untuk menetapkan bobot yang berbeda untuk tujuan.

**Integer Linear Programming**

Sebuah perusahaan memiliki proyek yang ingin dilakukan tetapi karena keterbatasan anggaran tidak semuanya dapat dilakukan terpilih. Dalam proyek j tertentu diharapkan menghasilkan pendapatan c tetapi membutuhkan investasi pada waktu jj periode i untuk i 1, ... m. Modal yang tersedia dalam periode waktu i œ adalah b. Masalah memaksimalkan pendapatan tunduk pada i batasan anggaran dapat dirumuskan sebagai berikut: biarkan x 0 atau 1 sesuai dengan tidak melanjutkan atau j œ masing-masing melanjutkan dengan proyek j maka



Alasan di balik pendekatan ini adalah:

1. Selesaikan. masalah berkelanjutan sebagai LP yaitu mengabaikan integralitas

2. Jika kebetulan variabel dasar optimal semuanya bilangan bulat maka solusi optimal telah ditemukan.

Jika tidak:

3. yaitu kendala yang dipenuhi oleh semua solusi bilangan bulat untuk masalah tetapi tidak dengan Menghasilkan pemotongan

solusi L.P. saat ini.

4. dan pergi ke 1. Tambahkan kendala baru ini Ð Ñ

Ini adalah lurus ke depan untuk menunjukkan bahwa jika pada setiap tahap, solusi L.P. x saat ini adalah integer, itu adalah bilangan bulat optimal larutan. Ini karena x optimal di wilayah yang mengandung semua solusi bilangan bulat yang layak. Masalahnya adalah menentukan pemotongan yang memastikan konvergensi algoritma dalam sejumlah langkah yang terbatas.



**Travelling Salesman Problem**

Traveling Salesman Problem adalah salah satu masalah yang paling intensif dipelajari dalam matematika komputasional. Halaman-halaman ini dikhususkan untuk sejarah, aplikasi, dan riset terkini dari tantangan menemukan rute terpendek mengunjungi setiap anggota kumpulan lokasi dan kembali ke titik awal Anda.

Traveling Salesman Problem adalah masalah komputasi yang paling terkenal. Kita dapat menggunakan pendekatan brute force untuk mengevaluasi setiap kemungkinan tur dan memilih yang terbaik. Untuk n jumlah simpul dalam grafik, ada (n - 1)! sejumlah kemungkinan.

Alih-alih brute-force menggunakan pendekatan pemrograman dinamis, solusinya dapat diperoleh dalam waktu yang lebih singkat, meskipun tidak ada algoritma waktu polinomial.

Mari kita perhatikan grafik G = (V, E), di mana V adalah seperangkat kota dan E adalah seperangkat sisi tertimbang. Sebuah ujung e (u, v) menyatakan bahwa simpul u dan v terhubung. Jarak antara verteks u dan v adalah d (u, v), yang seharusnya tidak negatif.

Misalkan kita sudah mulai di kota 1 dan setelah mengunjungi beberapa kota sekarang kita berada di kota j. Oleh karena itu, ini adalah tur parsial. Kita tentu perlu tahu j, karena ini akan menentukan kota mana yang paling nyaman untuk dikunjungi berikutnya :

Untuk subkumpulan kota S Є {1, 2, 3, ..., n} yang mencakup 1, dan j Є S, misalkan C (S, j) adalah lamanya jalur terpendek yang mengunjungi setiap node dalam S tepat sekali , mulai dari 1 dan berakhir pada j.

Kapan | S | > 1, kita mendefinisikan C (S, 1) = ∝ karena jalur tidak dapat memulai dan berakhir pada 1.

Sekarang, biarkan ungkapkan C (S, j) dalam hal sub-masalah yang lebih kecil. Kita harus mulai dari 1 dan berakhir pada j. Kita harus memilih kota berikutnya sedemikian rupa

C (S, j) = minC (S− {j}, i) + d (i, j) wherei∈Sandi ≠ j

**Queuing System**

in.

1. First Come First Serve (FCFS)

FCFS merupakan algoritma penjadwalan yang paling sederhana yang digunakan CPU. Dengan menggunakan algoritma FCFS setiap proses yang berada pada status ready dimasukkan ke dalam antrian. FIFO sesuai dengan waktu kedatangannya. Proses yang tiba terlebih dahulu yang akan dieksekusi terlebih dahulu[2].

2. Shortest Job First (SJF)

Dengan algoritma ini maka setiap proses yang ada di antrian ready akan dieksekusi berdasarkan burst time terkecil. Hal ini mengakibatkan waiting time yang pendek untuk setiap proses dan karena hal tersebut maka waiting time rata-ratanya juga menjadi pendek, sehingga dapat dikatakan bahwa algoritma ini adalah algoritma yang optimal[1].

Ada beberapa kekurangan dari algoritma ini yaitu:

1. Kesulitan untuk memprediksi burst time proses yang akan dieksekusi selanjutnya .

2. Proses yang mempunyai burst time yang besar akan memiliki waiting time yang besar pula karena yang dieksekusi terlebih dahulu adalah proses dengan burst time yang lebih kecil

3. Round Robin

Algoritma ini didesin untuk sistem timesharing. Proses akan mendapat jatah sebesar time quantum dengan nilai quantum umumnya sebesar 10-100 ms. Jika time quantumnya habis atau proses sudah selesai CPU akan dialokasikan ke proses berikutnya. Tentu proses ini cukup adil karena tak ada proses yang diprioritaskan, semua proses mendapat jatah waktu yang sama dari CPU (1/n), dan tak akan menunggu lebih lama dari (n-1)/q. Algoritma ini sepenuhnya bergantung besarnya time quantum. Jika terlalu besar, algoritma ini akan sama saja dengan algoritma first-come first-served. Jika terlalu kecil, akan semakin banyak peralihan proses sehingga banyak waktu terbuang

4.Priority Schedulling (PS)

Priority Scheduling merupakan algoritma penjadwalan yan mendahulukan proses dengan nilai prioritas tertinggi. Setiap proses memiliki prioritasnya masing-masing. Prioritas suatu proses dapat ditentukan melalui beberapa karakteristik antara lain:

a. Batas waktu

b. Kebutuhan Memori

c. Akses file

d. Perbandingan antara I/O Burst dengan CPU Burst

e. Tingkat kepentingan proses

Penjadwalan dengan prioritas juga dapatdijalankan secara preemptive maupun nonpreemptive[2]. Pada preemptive, jika ada suatu proses yang baru datang memiliki prioritas yang lebih tinggi daripada proses yang sedang dijalankan, maka proses yang sedang berjalan tersebut dihentikan, lalu CPU dialihkan untuk proses yang baru datang tersebut. Sementara itu, pada non-preemptive, proses yang baru datang tidak dapat menganggu proses yang sedang berjalan, tetapi hanya diletakkan di depan antrian. Kelemahan pada penjadwalan prioritas adalah dapat terjadinya indefinite blocking (starvation) yaitu suatu proses dengan prioritas yang rendah memiliki kemungkinan untuk tidak dieksekusi jika terdapat proses lain yang memiliki prioritas lebih tinggi darinya. Solusi dari permasalahan ini adalah Paging, yaitu meningkatkan prioritas dari setiap proses yang menunggu dalam antrian secara bertahap [2].

5. Multilevel Queue(MLQ)

Ide dasar dari algoritma ini adalah berdasarkan pada sistem prioritas proses. Prinsipnya adalah, jika setiap proses dapat dikelompokkan berdasarkan prioritasnya

**Monte Carlo Simulation**

Merupakan dasar dari semua algoritma untuk metode simulasi

􀂄 Didasari pada pemikiran penyelesaian suatu masalah untuk mendapatkan hasil lebih baik dengan cara memberi alternatif nilai sebanyak-banyaknya (nilai terbangkit) untuk mendapatkan tingkat ketelitian yang lebih tinggi

􀂄 Misal untuk memperoleh tingkat ketelitian sampai 0,01 maka diperlukan pembangkitan nilai sebanyak 10000, dsb.

􀂄 Teknik pembuatan program bersifat bebas hampir tidak ada rule yang terlalu mengikat

Algoritma :

1. Bangkitkan nilai 0/1 sebanyak 1000 kali (N=1000) dengan cara: n =(int)rand()%2

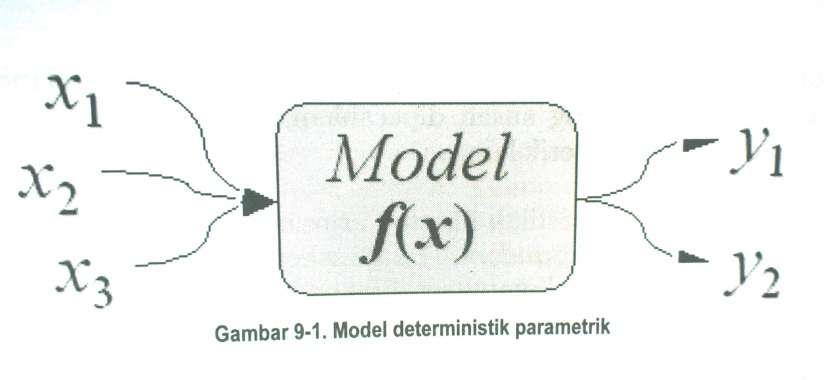
2. Klasifikasi

Jika n=0, maka M=M+1

Jika n=1, maka B=B+1

3. Hitung probabilitas M dengan cara M/N dan probabilitas B dengan cara B/N

Metode Monte Carlo adalah algoritma komputasi untuk mensimulasikan berbagai perilaku sistem fisika dan matematika. Metode ini terbukti efisien dalam memecahkan persamaan diferensial, integral medan radians. Metode monte carlo umumnya dilakukan meggunakan komputer dan memakai teknik simulasi komputer.



Menghitung luas di bawah kurva f(x) = 1 sin x.

jumlah titik di dalam kurva

Pa = --------------------------------------

jumlah titik yang dilempar

Maka luas daerah di bawah kurva :

Luas = Pa ( luas OABC)

**Nonliear Programming**

Non linear programing adalah berkebalikan dari linear programing.dana suatu cara pemecahan masalah dimana variabel- varbel didalamnya akan bersifat tidak linear, rel, atau bahkan terhubung satu sama lainnya. Masalah yg dialami tidak akan bisa dipecahkan dengan metode linear dimana justru biasanya akan timbul variabel baru atau fungsi-fungsibaru pada kondisi tertentu dan akan terus berlanjut.

Contoh permasalah pada non linear programming :

Minimumkan 

Terhadap 





Akan diselesaikan permasalahan tersebut, tetapisebelumnya akan diperiksa terlebih dahulu apakah  konveks atau tidak konveks.

Berdasarkan Definisi 2.28,  konveks jika matriks Hessiannya semidefinit positif.





Untuk menunjukkan bahwa **H** semidefinit positif atau tidak yaitu dengan menyelesaikan masalah berikut:

det (**H**-**I**) = 0

jika diperoleh  maka **H** semidefinit positif,

0 = det (**H**-**I**) = 



 dan 

Karena maka **H** semidefinit positif, sehingga  konveks.

Selanjutnya akan diselesaikan permasalahan awal.

dengan nilai awal 

Dari masalah meminimumkan tersebut didapatkan





**Iterasi 1:**

*Langkah 1*: 1. Mentukan 













2. Menentukan nilai dan 

.

.

3. Tampak bahwa 

Dari nilai  didapat , ini berarti .

Sehingga didapatkan 

4. Tentukan 



Akibatnya didapatkan 

*Langkah 2*: Dari langkah 1 telah diketahui bahwa:





Akibatnya 



Akan diselesaikan masalah meminimumkan terhadap .

1. Mencari nilai kritis dari 







( tidak memenuhi karena tidak terletak pada selang tertutup 0 dan 1 sehingga tidak ada nilai fungsi pada nilai kritis)

2. Nilai fungsi pada titik ujung selang





Jadi minimum  terhadap  adalah -4 dengan .

Dibentuk 

**State Space Search**

State Space Searchadalah proses yang digunakan dalam bidang ilmu komputer, termasuk kecerdasan buatan (AI), di mana konfigurasi berturut-turut atau keadaan suatu kejadian dipertimbangkan, dengan maksud untuk menemukan state tujuan dengan properti yang diinginkan.

Masalah sering dimodelkan sebagai ruang status, satu set menyatakan bahwa masalah bisa masuk. State Space Search membentuk grafik di mana 2 state terhubung jika ada operasi yang dapat dilakukan untuk mengubah negara pertama menjadi yang kedua.

State Space Search sering berbeda dari metode pencarian ilmu komputer tradisional karena ruang status tersirat: grafik ruang keadaan yang khas terlalu besar untuk dihasilkan dan disimpan dalam memori. Sebaliknya, node dibuat saat dieksplorasi, dan biasanya dibuang sesudahnya. Solusi untuk contoh pencarian kombinatorial dapat terdiri dari state tujuan itu sendiri, atau jalur dari beberapa state awal ke state tujuan.

Dalam pencarian ruang negara, ruang status secara resmi direpresentasikan sebagai tuple di mana:

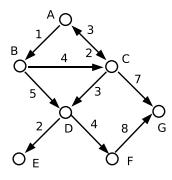
S adalah himpunan semua kemungkinan status;

A adalah sekumpulan tindakan yang mungkin, tidak terkait dengan keadaan tertentu tetapi mengenai semua ruang State;

Action (s) adalah fungsi yang menetapkan aksi mana yang mungkin dilakukan dalam keadaan tertentu;

Hasil (s, a) adalah fungsi yang mengembalikan state mencapai melakukan tindakan di keadaan s

Biaya (s, a) adalah biaya untuk melakukan tindakan adalam keadaan s. Di banyak ruang State adalah konstanta, tetapi ini tidak benar secara umum.



Pencarian ruang State berjumlah a cari melalui grafik yang diarahkan (dwihuruf) node grafik = menyatakan arc (tepi berarah) = transisi antar State Grafik dapat didefinisikan secara eksplisit atau secara implisi. Grafik dapat berisi siklus Jika kita juga membutuhkan biaya transisi, kita bekerja dengan grafik diarahkan tertimbang

**Dynamic Programming**

Pemrograman dinamis adalah metode optimasi matematis dan metode pemrograman komputer. Metode ini dikembangkan oleh Richard Bellman pada tahun 1950 dan telah menemukan aplikasi di berbagai bidang, mulai dari teknik penerbangan hingga ekonomi. Dalam kedua konteks itu mengacu pada menyederhanakan masalah rumit dengan memecahnya menjadi masalah sub sederhana dengan cara rekursif. Sementara beberapa masalah keputusan tidak dapat dipisah-pisahkan dengan cara ini, keputusan yang menjangkau beberapa titik dalam waktu sering terpecah secara rekursif. Demikian pula, dalam ilmu komputer, masalah yang dapat diselesaikan secara optimal dengan memecahnya menjadi sub-masalah dan kemudian secara rekursif menemukan solusi optimal untuk sub-masalah dikatakan memiliki substruktur yang optimal.

Jika sub-masalah dapat ditumpuk secara rekursif di dalam masalah yang lebih besar, sehingga metode pemrograman dinamis dapat diterapkan, maka ada hubungan antara nilai masalah yang lebih besar dan nilai sub-masalah. Dalam literatur optimasi hubungan ini disebut persamaan Bellman.

Dalam hal optimasi matematika, pemrograman dinamis biasanya mengacu pada menyederhanakan keputusan dengan memecahnya menjadi urutan langkah-langkah keputusan dari waktu ke waktu. Ini dilakukan dengan mendefinisikan urutan fungsi nilai V1, V2, ..., Vn, dengan argumen y mewakili status sistem pada waktu i dari 1 hingga n. Definisi Vn (y) adalah nilai yang diperoleh dalam keadaan y pada terakhir kali n. Nilai-nilai Vi pada jaman dulu i = n −1, n - 2, ..., 2, 1 dapat ditemukan dengan bekerja mundur, menggunakan hubungan rekursif yang disebut persamaan Bellman. Untuk i = 2, ..., n, Vi − 1 pada setiap negara y dihitung dari Vi dengan memaksimalkan fungsi sederhana (biasanya jumlah) keuntungan dari keputusan pada saat i - 1 dan fungsi Vi pada yang baru keadaan sistem jika keputusan ini dibuat. Karena Vi telah dihitung untuk negara yang dibutuhkan, operasi di atas menghasilkan Vi − 1 untuk negara-negara tersebut. Akhirnya, V1 pada status awal sistem adalah nilai solusi optimal. Nilai optimal dari variabel keputusan dapat dipulihkan, satu per satu, dengan melacak kembali perhitungan yang sudah dilakukan.

**Branch and Bound**

Metode Branch and Bound adalah sebuah teknik algoritma yang secara khusus mempelajari bagaimana caranya memperkecil Search Tree menjadi sekecil mungkin. Sesuai dengan namanya, metode ini terdiri dari 2 langkah yaitu :

· Branch yang artinya membangun semua cabang tree yang mungkin menuju solusi.

· Bound yang artinya menghitung node mana yang merupakan active node (E-node) dan node mana yang merupakan dead node (D-node) dengan menggunakan syarat batas constraint (kendala).

Langkah-langkah metode Branch dan Bound dapat dilakukan seperti berikut :

1. Selesaikan LP dengan metode simpleks biasa

2. Teliti solusi optimumnya. Jika variabel basis yang diharapkan bulat adalah bulat, solusi optimum bulat telah tercapai.

3. Nilai solusi pecah yang layak dicabangkan ke dalam sub-sub masalah. Tujuannya adalah untuk menghilangkan solusi kontinyu yang tidak memenuhi persyaratan bulat dalam masalah itu.

4. Untuk setiap sub-masalah, nilai solusi optimum kontinyu fungsi tujuan ditetapkan sebagai batas atas. Solusi bulat terbaik menjadi batas bawah (pada awalnya, ini adalah solusi kontinyu yang dibulatkan ke bawah). Sub-sub masalah yang memiliki batas atas kurang dari batas bawah yang ada, tidak diikut sertakan pada analisa selanjutnya. Suatu solusi bulat layak adalah sama baik atau lebih baik dari batas atas untuk setiap sub masalah yang dicari. Jika solusi yang demikian terjadi, suatu sub masalah dengan batas atas terbaik dipilih untuk dicabangkan. Kembali ke langkah 3.

Penetapan Batas (Bounding)

Pada algoritma branch and bound terdapat dua batas yaitu batas atas (upper bound) dan batas bawah (lower bound).

a. Pada masalah maksimisasi:

Batas atas merupakan solusi ILP relaksasi dari sub masalah tersebut sedangkan batas bawahnya adalah nilai dari sub masalah tersebut ataupun solusi dari sub masalah lain yang semua variabel keputusan yang harus bernilai integer sudah bernilai integer (solusi terbaik yang sejauh ini diperoleh).

b. Pada masalah minimisasi:

Batas bawah merupakan solusi ILP relaksasi dari sub masalah tersebut sedangkan batas atasnya adalah nilai dari sub masalah tersebut ataupun solusi dari sub masalah lain yang semua variabel keputusan yang harus bernilai integer sudah bernilai integer ( solusi terkecil (terbaik) yang sejauh ini diperoleh ).

Penghentian Pencabangan (Fathoming)

Pencabangan atau pencarian solusi pada suatu sub masalah dihentikan jika:

a. Infeasible atau tidak mempunyai daerah layak.

b. Semua variabel keputusan yang harus bernilai integer sudah bernilai integer

c. Pada masalah maksimisasi, penghentian pencabangan pada suatu sub masalah dilakukan jika batas atas dari sub masalah tersebut tidak lebih besar atau sama dengan batas bawah.

d. Sedangkan pada masalah minimisasi penghentian pencabangan pada suatu sub masalah dilakukan jika batas bawah tidak lebih lebih kecil atau sama dengan batas atas.

Kondisi Optimal

Kondisi optimal pada Branch and bound antara lain :

a. Jika tidak ada lagi sub masalah yang perlu dicabangkan lagi maka solusi optimal sudah diperoleh.

b. Pada masalah maksimisasi solusi optimal merupakan solusi submasalah yang saat ini menjadi batas bawah (lower bound)

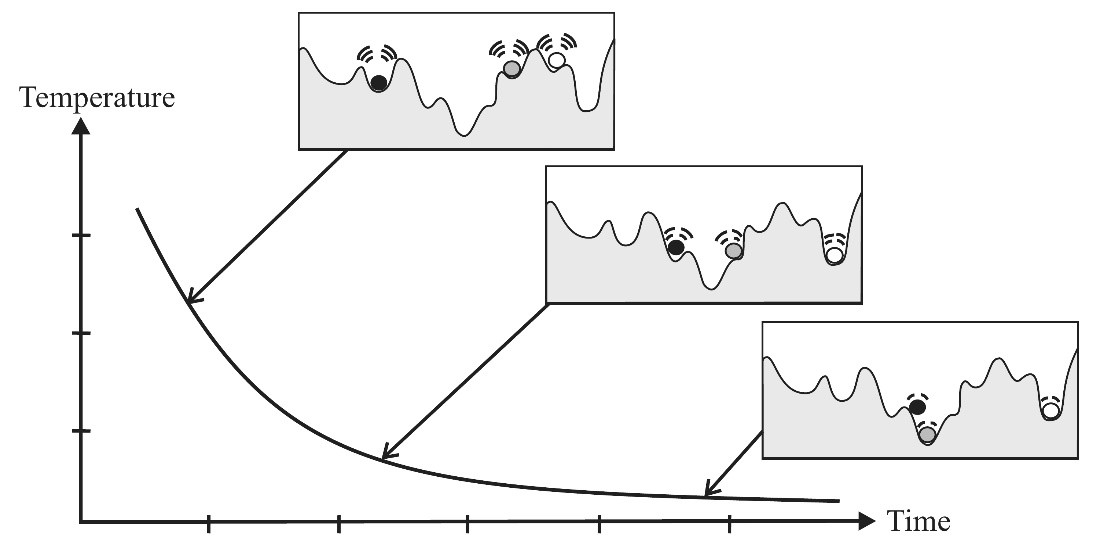
c. Pada masalah minimisasi solusi optimal merupakan solusi submasalah yang saat ini menjadi batas atas (upper bound).

**Simulated Annealing**

Simulated Annealing merupakan algoritma untuk optimisasi yang bersifat generik. Dengan menggunakan probabilitas dan mekanika statik, algoritma ini dapat digunakan untuk mencari pendekatan terhadapa solusi optimum global dari suatu permasalahan. Namun, SA tidak selamanya sempurna sebagai metode optimasi dalam pemecahan masalah, terdapat beberapa kekurangan yang mempengaruhi hasil akhir.

Penggunaan Simulated Annealing beride terkait dengan pemrosesan logam. Kata Annealing didefinisikan sebagai memanaskan kemudian mendinginkan. SA memanfaatkan analogi antara cara pendinginan dan pembekuan suatu bahan metal agar menjadi sebuah strujtur kristal dengan energi yang minimal (proses penguatan) dan juga untuk mencari state tujuan minimal dalam proses pencarian. Pada SA, terdapat beberapa parameter penting yang sangat menentukan, diantaranya Temperatur untuk terjadinya banyak perulangan serta perbandingan. Juga Pembangkitan bilangan random yang akan bersangkut paut dengan adanya probabilitas.

SA dianjurkan untuk masalah optimasi yang kompleks, Maka dari Itu algoritma dimulai pada suhu tertentu, Seiring berjalanya waktu, suhu secara bertahap akan menurun mengikuti jadwal pendinginan yang dapat dilihat pada gambar dibawah ini.



Terilihat dalam gambar, bahwa suatu temperatur yang tinggi (ilustrasi : pada saat logam panas) maka ada suatu titik yang dimana akan jadi batas penguatan/kristalisasi yang ditandai dengan nilai terendah. Karena Logam tersebut panas, maka akan semakin mudah untuk dibentuk. Seiring berjalannya waktu maka temperatur akan turun karena adanya perubahan suhu (di analogikan sebagai iterasi dalam algoritma), maka atom dan molekul logam akan saling mengikat kembali, dan mereka akan menemukan cara untuk saling mengikat dengan energi yang paling minimal.

Berikut merupakan prosedur pencarian dalam Simulated Annealing:

Inisialisasi minimal 2 buah variabel yang nilainya dipilih secara random.

Menetapkan suhu awal temperature.

Menetapkan nilai pendinginan temperature.

Menetapkan Fungsi hitung

Membuat Pengkondisian Looping ketika Temperatur Masih Tinggi, ada 2 kondisi :

– Jika proses sudah mencapai kriteria, program terhenti, dan keluar dari looping

– Jika Proses belum mencapai kriteria, maka program akan melakukan perulangan terus dan membandingkan nilai terus.

Menginisialisasi kembali 2 buah variabel yang bernilai random (ini akan menjadi nilai baru pembanding selama perulangan).

Menetapkan Fungsi hitung

Membuat pengkondisian hasil fungsi terkecil, terdapat 3 kondisi :

– Jika Nilai Fungsi Baru ternyata Lebih Kecil dari nilai Fungsi Sekarang,maka Nilai Baru di masukkan ke variabel Fungsi Sekarang. (Mendapatkan nilai terkecil).

– Jika Nilai Fungsi Baru ternyata Lebih Besar dari nilai Fungsi Sekarang, maka Nilai Baru akan dihitung Probabiltias nya, apakah dia nilai yang optimum atau tidak dengan kondisi apakah exp(-ΔE) > |0…1|\* . Jika Benar, ,maka Nilai Baru di masukkan ke variabel Fungsi Sekarang. (Mendapatkan nilai optimum).

– Jika Nilai Fungsi Baru ternyata Lebih Besar dari nilai Fungsi Sekarang dan Tidak Optimum, maka Nilai Terkecil dan teroptimum tetap di pegang olehFungsi Sekarang.

Menghitung penentu Iterasi Temperature, dengan mendinginkan Temperatur (temperature \* pendingin).

Mengeluarkan nilai terkecil/teroptimum.

\* Keterangan = ΔE adalah (Nilai Terbesar – Nilai Terkecil/Temp)

**Tabu Search**

Didalam menerapkan Tabu search, ada beberapa hal yang harus diperhatikan, yaitu:

Tabu list : tujuan dari penggunaan Tabu list (memori jangka pendek) adalah untuk menghindari mengevaluasi kandidat solusi yang pernah dikunjungi sebelumnya. Akan tetapi, menampung semua kandidat solusi yang pernah dievaluasi kedalam Tabu list akan membuat Tabu search menjadi tidak efektif (memerlukan ukuran memori yang besar dan wakt yang lama untuk mengecek apakah suatu candidate solusi pernah dievaluasi atau belum).

Aspiration criteria: umumnya diterapkan jika pergerakan tabu tersebut menghasilkan kandidat solusi yang memiliki nilai yang lebih baik daripada solusi terbaik yang telah dihasilkan.

Intensifikasi (memori jangka menengah): memori jangka menengah menyimpan sejumlah solusi yang berkualitas (elite solution) yang dihasilkan selama proses pencarian. Memori jangka menengah ini bertujuan untuk memberikan prioritas kepada atribut dari solusi berkualitas tersebut.

Diversifikasi (memori jangka panjang): memori jangka panjang menyimpan infromasi tentang kandidat solusi yang pernah dikunjungi. Berdasarkan infromasi tersebut, memori ini dapat mengeksplorasi area dalam ruang pencarian yang belum dikunjungi.

Tabu Search memiliki lima parameter utama, yaitu :

a. Prosedur local search,

b. Struktur neighbourhood, adalah suatu fungsi yang memetakan setiap solusi layak S ke solusi-solusi yang lainnya. Jumlah solusi layak dalam neighborhood biasanya dibatasi dengan menggunakan berbagai kriteria untuk mengurangi waktu proses pencarian solusi.

c. Kondisi tabu, adalah pelarangan penggunaan solusi yang telah ditemukan sebelumnya.

d. Kondisi aspirasi, adalah pengecualian pengambilan solusi yang telah masuk dalam tabu

e. Kriteria penghentian.

Algoritma Tabu Search bisa dihentikan berdasarkan kriteria tertentu, misalnya sejumlah iterasi yang ditentukan user, sejumlah waktu CPU tertentu, atau sejumlah iterasi berurutan tanpa peningkatan nilai fungsi objektif terbaik.

Tabu Search bekerja secara iteratif menggunakan algoritma local search pada setiap iterasi untuk mencari solusi terbaik di antara sebagian tetangga dari solusi terbaik saat ini. Pada setiap iterasi, algoritma local search memilih solusi tetangga yang memberikan peningkatan kualitas tertinggi. Tetapi, jika semua solusi tetangga tidak memberikan peningkatan kualitas, maka local search akan memilih solusi yang penurunan kualitasnya paling rendah. Kualitas di sini bergantung pada masalah yang dihadapi. Untuk masalah minimasi, semakin rendah nilai fungsi objektifnya berarti semakin tinggi kualitasnya. Sebaliknya, untuk masalah maksimasi, solusi dengan nilai fungsi objektif yang tinggi berarti kualitasnya tinggi.

**Parallelization & Distribution**

"Model paralelisme" mengacu pada paralelisasi dari algoritma ML atas ruang bersama parameter model, bukan ruang (biasanya i.i.d.) contoh data. Pada tingkat yang tinggi, parameter model adalah perantara yang berubah jumlah yang merupakan algoritma ML secara iteratif pembaruan, hingga konvergensi tercapai. Kunci

Keuntungan dari pendekatan model-paralel adalah bahwa secara eksplisit partisi parameter model menjadi himpunan bagian, memungkinkan masalah ML dengan ruang model besar untuk ditangani pada mesin dengan memori terbatas.

pendekatan model-paralel global menghasilkan rencana eksekusi optimal untuk fragmen input permintaan dengan membuat keputusan mengenai operasi pemesanan, pergerakan data antara situs dan pilihan algoritma terdistribusi dan lokal untuk operasi basis data. Ada sejumlah masalah terkait langkah ini. Mereka harus melakukan dengan pembatasan yang dikenakan pada model biaya, fokus pada subset dari query bahasa, trade-off antara biaya optimasi dan biaya eksekusi, dan optimalisasi / pengoptimalan ulang selang.

Ada trade-off yang diperlukan antara biaya optimasi dan kualitas rencana eksekusi yang dihasilkan. Biaya optimasi yang lebih tinggi mungkin dapat diterima untuk menghasilkan rencana "lebih baik" untuk kueri berulang, karena ini akan mengurangi biaya eksekusi query dan mengamortisasi biaya optimasi atas banyak eksekusi. Namun, biaya pengoptimalan tinggi tidak dapat diterima untuk kueri ad hoc yang hanya dijalankan sekali. Optimasi biaya terutama terjadi dengan mencari ruang solusi untuk rencana pelaksanaan alternatif. Didistribusikan sistem, ruang solusi dapat cukup besar karena berbagai strategi eksekusi terdistribusi.

Oleh karena itu, sangat penting untuk mempelajari penerapan strategi pencarian yang efisien yang menghindari pencarian lengkap pendekatan. Pengoptimalan kueri global biasanya dilakukan sebelum pelaksanaan kueri, oleh karena itu disebut statis. Masalah utama dengan pendekatan ini adalah model biaya yang digunakan untuk optimasi dapat menjadi tidak akurat karena perubahan ukuran fragmen atau reorganisasi basis data yang penting untuk pemuatan menyeimbangkan. Oleh karena itu, masalahnya adalah menentukan interval optimal dari kompilasi ulang / reoptimisasi dari queri yang memperhitungkan trade-off antara optimalisasi dan biaya eksekusi.