Bagian I

Pengetahuan Dasar

1

Pengenalan

> "People worry that computers will get too smart and take over the world, but the real problem is that they're too stupid and they've already taken over the world."
>
> Pedro Domingos

Penulis yakin istilah *machine learning* atau *deep learning* sudah tidak asing di telinga pembaca. *Machine learning* dan *deep learning* adalah salah satu materi kuliah pada jurusan Teknik Informatika atau Ilmu Komputer. Selain mengenal kedua istilah tersebut dari perkuliahan, pembaca mungkin mengenal istilah tersebut karena digunakan untuk pemasaran (*marketing*). Sebagai permulaan, *machine learning* dan *deep learning* bukanlah kedua hal yang berbeda.[1] Perlu diingat, *deep learning* adalah bagian dari *machine learning*. *Machine learning* sudah diaplikasikan pada banyak hal, baik untuk klasifikasi gambar, mobil tanpa pengemudi, klasifikasi berita, dsb. Bab ini menjelaskan konsep paling dasar dan utama *machine learning*.

1.1 Kecerdasan Buatan

Pada bagian pembukaan (*kata pengantar*) telah dijelaskan bahwa kami menganggap kamu sudah memiliki pengetahuan dasar tentang *artificial intelligence* (kecerdasan buatan), kami akan memberikan sedikit ikhtisar apa hubungan kecerdasan buatan dan *machine learning*. Saat pertama kali kamu mendengar istilah "kecerdasan buatan", mungkin kamu akan terpikir robot yang memiliki raga fisik. Tetapi, kecerdasan buatan tidak hanya terbatas pada sesuatu yang memiliki raga fisik. Raga fisik berguna untuk interaksi

[1] Walau istilah *deep learning* belakangan ini lebih populer.

yang ramah bagi manusia. Tidak mesti memiliki raga fisik, kecerdasan buatan sesungguhnya adalah program[2] yang memiliki bentuk matematis (instruksi); kita sebut sebagai **agen**. Berbeda dengan program biasa yang menghasilkan aksi berdasarkan instruksi, tujuan kecerdasan buatan adalah menciptakan program yang mampu mem-program (*output* program adalah sebuah program). Secara teori, program adalah ***automaton***[3] yang menjalankan suatu instruksi. Sama halnya dengan program pada umumnya, agen kecerdasan buatan juga menjalankan suatu instruksi. Yang menjadikanya beda dengan program biasa adalah **kemampuan untuk belajar**.[4] "Belajar" yang dimaksud tidaklah sama dengan proses manusia belajar. Mesin mampu belajar apabila ia mampu meng-*update* parameter (dijelaskan lebih detil kemudian), dimana parameter tersebut kurang-lebih merepresentasikan "pengetahuan" mesin.

Pada bidang keilmuan kecerdasan buatan, kita ingin menciptakan agen yang mampu melakukan pekerjaan yang membutuhkan kecerdasan manusia. Perhatikan, disini disebut kecerdasan manusia; hewan pun cerdas, tapi kecerdasan manusia dan hewan berbeda; yang kita ingin aproksimasi adalah kecerdasan manusia. Akan tetapi, kecerdasan manusia susah didefinisikan karena memiliki banyak aspek misalnya nalar (logika), kemampuan berbahasa, seni, dsb. Karena kecerdasan manusia memiliki banyak dimensi, kita dapat mencoba menyelesaikan masalah pada sub bidang lebih kecil–spesifik domain (*divide and conquer*). Sampai saat ini pun, peneliti belum juga mengetahui secara pasti apa yang membuat manusia cerdas, apa itu sesungguhnya cerdas, dan bagaimana manusia dapat menjadi cerdas. Dengan demikian, keilmuan kecerdasan buatan adalah interdisiplin, memuat: psikologis, linguistik, ilmu komputer, biologi, dsb. Bila kamu bertanya apakah program deterministik dapat disebut *kecerdasan buatan*, jawabannya "iya", *to some extent* (sampai pada level tertentu) karena memenuhi dimensi *acting rationally* (dijelaskan pada subbab 1.2).

Permasalahan utama bidang kecerdasan buatan terdiri dari (dari klasik sampai lebih modern),[5] yaitu:

1. ***Planning***. Diberikan *start state* dan *goal state*, agen harus merencanakan sekuens aksi untuk merubah *start state* menjadi *goal state*. Contoh permasalahan *planning* adalah merencanakan rute perjalanan dari kota A ke kota B. Bisa jadi, saat merencanakan sekuens aksi, ada kendala (*constraints*) yang harus dioptimisasi.

2. ***Representasi pengetahuan***, yaitu merepresentasikan pengetahuan dalam bentuk formal. Dengan representasi formal tersebut, kita dapat melakukan inferensi dengan operasi logika berbentuk simbolik, misal logika preposisi,

[2] Secara sederhana, program adalah kumpulan atau sekuens instruksi.
[3] Kami sarankan untuk membaca buku [2] untuk materi automata.
[4] Perlu diperhatikan, definisi ini adalah pandangan modern.
[5] Silahkan merujuk Association for the Advancement of Artificial Intelligence (AAAI).

logika orde pertama (*first-order logic*), teori Fuzzy, *abductive reasoning*, ontologi, maupun jaringan semantik (*semantic web*) [3].

3. ***Machine learning***, yaitu teknik untuk melakukan inferensi terhadap data dengan pendekatan matematis. Inti *machine learning* adalah untuk membuat model (matematis) yang merefleksikan pola-pola data (seiring kamu membaca buku ini, kamu akan lebih mengerti). Ini adalah bahasan utama buku ini. Pada abad ke-21 ini, *machine learning* banyak memanfaatkan statistika dan aljabar linier.

4. ***Multi-agent system***, yaitu sistem yang memiliki banyak agen berinteraksi satu sama lain untuk menyelesaikan permasalahan. Agen satu mengerjakan suatu hal tertentu, kemudian bekerja bersama untuk menyelesaikan masalah yang lebih besar (tidak dapat diselesaikan sendiri).

5. Dan lain sebagainya, silahkan mengacu pada topik konferensi *Association for the Advancement of Artificial Intelligence* (AAAI) .

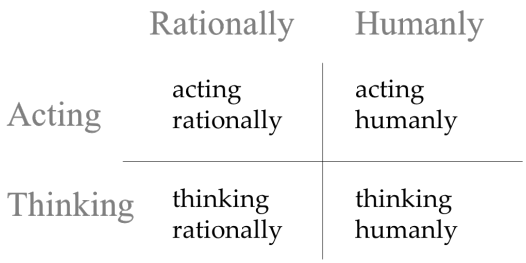
Perhatikan, sub keilmuan representasi pengetahuan dan *machine learning* sama-sama melakukan inferensi, tetapi pada representasi yang berbeda. Inferensi pada bidang keilmuan representasi pengetahuan mencakup tentang bagaimana cara (langkah dan proses) mendapatkan sebuah keputusan, diberikan premis. Sebagai contoh, a adalah anak b, dan c adalah anak b, maka apakah hubungan a dan c? Jawab: c adalah cucu a. Pada *machine learning*, inferensi yang dimaksud lebih menitikberatkan ranah hubungan variabel. Misalnya, *apakah penjualan akan meningkat apabila kita meningkatkan biaya marketing.* Bila kamu ingat dengan mata pelajaran matematika SMA (logika preposisi), kamu sadar bahwa membuat sistem cerdas menggunakan representasi pengetahuan simbolik itu susah. Kita harus mendefinisikan *term*, aturan logika, dsb. Belum lagi kita harus mendefinisikan aturan-aturan secara manual. Disamping itu, pengetahuan manusia sangatlah kompleks, dan translasi pengetahuan menjadi aturan-aturan formal tidaklah mudah. Representasi pengetahuan secara tradisional dianggap relatif kurang *scalable*. Artinya, kita tidak dapat merubah basis pengetahuan dengan mudah (meng-*update* "parameter"). Sementara itu, *machine learning* berada pada daerah representasi data/ilmu/pengetahuan dalam bentuk matematis karena keilmuan *machine learning* diturunkan dari matematika dan statistika. Teknik *machine learning* juga menjadi semakin populer akibat kemampuan komputasi yang terus meningkat.

Pada masa sekarang, kita dianugrahi dengan data yang banyak (bahkan tidak terbatas), teknik *machine learning* menjadi intuitif untuk melakukan inferensi pada data yang besar. Hal ini yang menyebabkan *machine learning* menjadi populer karena konstruksi model inferensi dapat dilakukan secara otomatis. *Machine learning* ibarat sebuah "alat", sama seperti rumus matematika. Bagaimana cara menggunakannya tergantung pada domain per-

masalahan. Dengan demikian, kamu harus paham betul bahwa memahami teknik-teknik *machine learning* saja tidak cukup. Kamu juga harus mengetahui domain aplikasi yang bersesuaian karena pemanfaatan teknik-teknik *machine learning* dapat berbeda pada domain yang berbeda. Sedikit cerita, sub keilmuan *data science* mempelajari banyak domain, misalnya data pada domain sosial, ekonomi, bahasa, maupun visual. Seiring kamu membaca buku ini, kami harap kamu semakin mengerti hal ini.

1.2 Intelligent Agent

Agen cerdas memiliki empat kategori berdasarkan kombinasi dimensi cara inferensi (*reasoning*) dan tipe kelakuan (*behaviour*) [4, 5]. Kategori agen dapat dilihat pada Gambar 1.1 dengan penjelasan sebagai berikut:



Gambar 1.1: Dimensi kecerdasan.

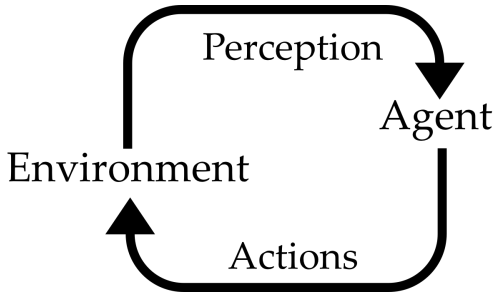
1. ***Acting Humanly***. Pada dimensi ini, agen mampu bertingkah dan berinteraksi layaknya seperti manusia. Contoh terkenal untuk hal ini adalah *turing test*. Tujuan dari *turing test* adalah untuk mengevaluasi apakah suatu sistem mampu "menipu" manusia. Disediakan seorang juri, kemudian juri berinteraksi dengan sesuatu di balik layar. Sesuatu di balik layar ini bisa jadi manusia atau program. Program dianggap mampu bertingkah (berinteraksi) seperti layaknya manusia apabila juri tidak dapat membedakan ia sedang berkomunikasi dengan manusia atau program.

2. ***Acting Rationally***. Pada dimensi ini, agen mampu bertingkah dengan optimal. Tindakan optimal belum tentu menyerupai tindakan manusia, karena tindakan manusia belum tentu optimal. Misalnya, agen yang mampu memiliki rute terpendek dari suatu kota A ke kota B untuk mengoptimalkan penggunaan sumber daya. Sebagai manusia, bisa saja kita mencari jalan sesuka hati.

3. ***Thinking Humanly***. Pada dimensi ini, agen mampu berpikir seperti manusia dalam segi kognitif (e.g. mampu mengerti apa itu kesedihan atau kesenangan). Dapat dibilang, meniru bagaima proses berpikir di otak terjadi (pemodelan otak).

4. ***Thinking Rationally***. Pada dimensi ini, agen mampu berpikir secara rasional. Sederhananya sesuai dengan konsep logika matematika. *Thinking Humanly* lebih cenderung pada pemodelan kognitif secara umum, sementara dimensi *thinking rationally* cenderung pada pemodelan proses berpikir dengan prinsip optimisasi (apa yang harus dilakukan agar hasil optimal).

Perlu dicatat, "*acting*" berarti agen mampu melakukan aksi. Sementara "*thinking*" adalah pemodelan proses. Untuk mewujudkan interaksi manusia-komputer seperti manusia-manusia, tentunya kita ingin *intelligent agent* bisa mewujudkan dimensi *acting humanly* dan *thinking humanly*. Sayangnya, manusia tidak konsisten [6]. Sampai saat ini, konsep kecerdasan buatan adalah meniru manusia; apabila manusia tidak konsisten, peneliti susah untuk memodelkan cara berpikir atau tingkah laku manusia. Berhubung agen dilatih menggunakan contoh-contoh data buatan manusia, ketidak-konsistensi-an agen semata-mata mencerminkan ketidak-konsistensi-an pembuat data. Dengan hal itu, saat ini kita paling mungkin menciptakan agen yang mempunyai dimensi *acting rationally*.



Gambar 1.2: *Agent* vs *environment* [7].

Perhatikan Gambar 1.2! Agen mengumpulkan informasi dari lingkungannya, kemudian memberikan respon berupa aksi. Lingkungan (*environment*) yang dimaksud bisa jadi macam-macam, misal: rumah, papan catur, agen lain, dsb. Kita ingin agen melakukan aksi yang benar. Tentu saja kita perlu mendefinisikan secara detail, teliti, tepat (*precise*), apa arti "aksi yang benar." Dengan demikian, lebih baik apabila kita mengukur kinerja agen, menggunakan ukuran kinerja yang objektif (disebut *performance measure*). Misal-

nya untuk robot pembersih rumah, *performance measure*-nya adalah seberapa persen debu yang dapat ia bersihkan. *Performance measure* adalah ukuran bagaimana model pembelajaran mesin dievaluasi secara eksternal. Di sisi internal, model harus mengoptimalkan suatu fungsi utiiltas (***utility function***), yaitu fungsi apa yang harus dimaksimalkan atau diminimalkan oleh agen tersebut khususnya pada tahap latihan (*training*). Misalnya, robot pembersih rumah memiliki *utility function* untuk mengukur kotoran di tempat terbatas tempat ia berada atau rute paling efisien untuk membersihkan rumah. Setiap tindakan yang dilakukan agen rasional harus mengoptimalkan baik nilai *utility function* (internal) dan *performance measure* (eksternal). Pada buku ini, istilah *performance measure* dan *utility function* merujuk pada hal yang berbeda. Tetapi, pada beberapa kasus, *utility function* dan *performance measure* dapat diukur dengan fungsi yang sama. Mungkin kamu merasa penjelasan ini agak abstrak pada saat ini. Tetapi, kamu akan semakin mengerti perbedaan keduanya setelah membaca buku ini.

1.3 Konsep Belajar

Bayangkan kamu berada di suatu negara asing! Kamu tidak tahu norma yang ada di negara tersebut. Apa yang kamu lakukan agar bisa menjadi orang "normal" di negara tersebut? Tentunya kamu harus **belajar**! Kamu mengamati bagaimana orang bertingkah laku di negara tersebut dan perlahan-lahan mengerti norma yang berlaku. Belajar adalah usaha memperoleh kepandaian atau ilmu; berlatih; berubah tingkah laku atau tanggapan yang disebabkan oleh pengalaman.[6] Pembelajaran adalah proses, cara, perbuatan atau menjadikan orang atau makhluk hidup belajar. Akan tetapi, pada *machine learning*, yang menjadi siswa bukanlah makhluk hidup, tapi mesin.

Definsi sebelumnya mungkin sedikit "abstrak", kita harus mengkonversi definisi tersebut sebagai definsi operasional (bentuk komputasi). Secara operasional, belajar adalah perubahan tingkah laku berdasarkan pengalaman (*event*/data) untuk menjadi lebih baik. Pada konteks *machine learning*, belajar adalah menyesuaikan konfigurasi parameter (tingkah laku) terhadap *utility function* sesuai dengan data (lingkungan).

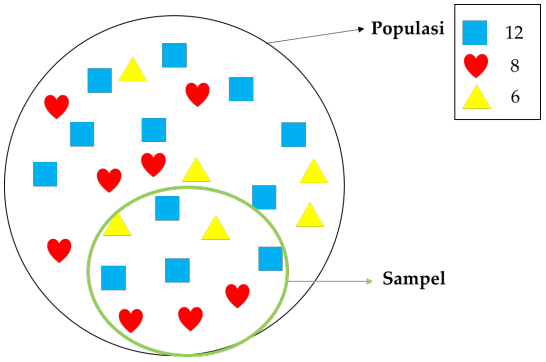
1.4 Statistical Learning Theory

Pada masa sekarang ini data bertebaran sangat banyak dimana-mana. Pemrosesan data secara manual tentu adalah hal yang kurang bijaksana. Beberapa pemrosesan data yang dilakukan seperti kategorisasi (kategorisasi teks berita), peringkasan dokumen, ekstraksi informasi (mencari subjek, objek, dan relasi diatara keduanya pada teks), rekomendasi produk berdasarkan

[6] KBBI Web, diakses pada 10 Oktober 2016

catatan transaksi, dll [7]. Tujuan *machine learning* minimal ada dua: **memprediksi masa depan** (*unobserved event*); dan/atau **memperoleh ilmu pengetahuan** (*knowledge discovery/discovering unknown structure*). Kedua hal ini berkaitan sangat erat. Sebagai contoh, manusia tahu bahwa cara menggunakan pensil dan pulpen sama, walaupun saat kita belum pernah menggunakan pulpen (penulis berasumsi kamu belajar menulis menggunakan pensil). Memprediksi masa depan berarti kita tahu bahwa pulpen adalah alat tulis. *Knowledge discovery* berarti kita tahu bahwa cara menggunakan pulpen dan pensil itu sama, walaupun belum pernah menggunakan pulpen sebelumnya.[7]

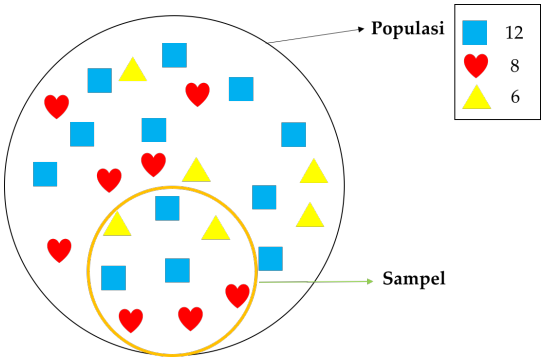
Untuk mencapai tujuan tersebut, kita menggunakan data (sampel), kemudian membuat model untuk menggeneralisasi "aturan" atau "pola" data sehingga kita dapat menggunakannya untuk mendapatkan informasi/membuat keputusan [8, 9]. *Statistical learning theory* (yang diaplikasikan pada *machine learning*) adalah teknik untuk memprediksi masa depan dan/atau menyimpulkan/mendapatkan pengetahuan dari data **secara rasional dan non-paranormal**. Hal ini sesuai dengan konsep *intelligent agent*, yaitu bertingkah berdasarkan lingkungan. Dalam hal ini, yang bertindak sebagai lingkungan adalah data. *Performance measure*-nya adalah seberapa akurat prediksi agen tersebut atau seberapa mirip "pola" data yang ditemukan terhadap data asli. Disebut *statistical* karena basis pembelajarannya memanfaatkan banyak teori statistik untuk melakukan inferensi (misal memprediksi *unobserved event*).[8]



Gambar 1.3: Ilustrasi makanan pesta 1.

Perhatikan Gambar 1.3 (permasalahan yang disederhanakan). Misalkan kamu diundang ke suatu pesta. Pada pesta tersebut ada 3 jenis kue yang disajikan. Kamu ingin mengetahui berapa rasio kue yang disajikan dibandingkan masing-masing jenisnya (seluruh populasi). Tetapi, karena susah untuk menganalisis seluruh data atau keseluruhan data tidak tersedia, kamu mengam-

[7] Baca *zero-shot learning*
[8] Selain itu, *machine learning* juga banyak memanfaatkan teori aljabar linier.



Gambar 1.4: Ilustrasi makanan pesta 2.

bil beberapa sampel. Dari sampel tersebut, kamu mendapati bahwa ada 4 buah kue segi empat, 3 buah kue hati dan 2 buah kue segitiga. Lalu kamu menyimpulkan (model) bahwa perbandingan kuenya adalah 4:3:2 (segiempat:hati:segitiga). Perbandingan tersebut hampir menyerupai kenyataan seluruh kue yaitu 4:2.67:2. Cerita ini adalah kondisi ideal.

Perhatikan Gambar 1.4, temanmu Haryanto datang juga ke pesta yang sama dan ingin melakukan hal yang sama (rasio kue). Kemudian ia mengambil beberapa sampel kue. Dari sampel tersebut ia mendapati bahwa ada 3 buah segiempat, 3 buah hati dan 2 buah segitiga, sehingga perbandingannya adalah 3:3:2. Tentunya hal ini sangat melenceng dari populasi.

Dari dua sampel yang berbeda, kita menyimpulkan, menginferensi (*infer*) atau mengeneralisasi dengan berbeda. Kesimpulan yang kita buat berdasarkan sampel tersebut, kita anggap merefleksikan populasi, kemudian kita menganggap populasi memiliki aturan/pola seperti kesimpulan yang telah kita ciptakan [10]. Baik pada statistika maupun *statistical machine learning*, pemilihan sampel (selanjutnya disebut ***training data***) adalah hal yang sangat penting. Apabila *training data* tidak mampu merepresentasikan populasi, maka model yang dihasilkan pembelajaran (*training*) tidak bagus. Untuk itu, biasanya terdapat juga ***validation data*** dan ***test data***. Mesin dilatih menggunakan *training data*, kemudian diuji kinerjanya menggunakan *validation data*[9] dan *test data*. Seiring dengan membaca buku ini, konsep *training data*, *validation data*, dan *test data* akan menjadi lebih jelas. Cara mengevaluasi apakah sampel (data) yang kita miliki cukup "bagus" untuk merepresentasikan populasi sangat bergantung pada domain dan aplikasi.

Seperti halnya contoh sederhana ini, persoalan *machine learning* sesungguhnya menyerupai persoalan *statistical inference* [10]. Kita berusaha mencari tahu populasi dengan cara menyelidiki fitur (*features* atau sifat-sifat) yang dimiliki sampel. Kemudian, menginferensi aksi yang harus dilakukan

[9] Pada umumnya bertindak sebagai *stopping criterion* saat proses *training*.

terhadap *unobserved data* berdasarkan kecocokan fitur-fitur *unobserved data* dengan model/aturan yang sudah ada.

Dari sisi metode pembelajaran, algoritma *machine learning* dapat dikategorikan sebagai: *supervised learning* (subbab 1.6), *semi-supervised learning* (subbab 1.8), *unsupervised learning* (subbab 1.9), dan *reinforcement learning*. Masing-masing metode akan dibahas pada subbab berikutnya (kecuali *reinforcement learning*, karena diluar cakupan buku ini).

1.5 Training, Validation, Testing Set

Terdapat dua istilah penting dalam pembangunan model *machine learning* yaitu: ***training*** dan ***testing***. *Training* adalah proses konstruksi model dan *testing* adalah proses menguji kinerja model pembelajaran. *Dataset* adalah kumpulan data (sampel dalam statistik). Sampel ini adalah data yang kita gunakan untuk membuat model maupun mengevaluasi model *machine learning*. Umumnya, *dataset* dibagi menjadi tiga jenis yang tidak beririsan (suatu sampel pada himpunan tertentu tidak muncul pada himpunan lainnya):

1. ***Training set*** adalah himpunan data yang digunakan untuk melatih atau membangun model. Pada buku ini, istilah *training data(set)* mengacu pada *training set*.
2. ***Development set*** atau ***validation set*** adalah himpunan data yang digunakan untuk mengoptimisasi saat melatih model. Model dilatih menggunakan *training set* dan pada umumnya kinerja **saat latihan** diuji dengan *validation set*. Hal ini berguna untuk generalisasi (agar model mampu mengenali pola secara generik). Pada buku ini, istilah *development* dan *validation data(set)* mengacu pada hal yang sama.
3. ***Testing set*** adalah himpunan data yang digunakan untuk menguji model setelah **proses latihan selesai**. Pada buku ini, istilah *testing data(set)* atau *test set* mengacu pada *testing set*. Perlu kami tekankan, *testing set* adalah *unseen data*. Artinya, model dan manusia tidak boleh melihat sampel ini saat proses latihan. Banyak orang yang tergoda untuk melihat *testing set* saat proses latihan walaupun itu adalah tingkah laku yang buruk karena menyebabkan *bias*.

Suatu sampel pada himpunan data kita sebut sebagai ***data point*** atau ***instance*** yang merepresentasikan suatu kejadian statistik (*event*). Perlu diingat, *training*, *validation*, dan *testing data* secara ideal diambil (*sampled*) dari distribusi yang sama dan memiliki karakteristik yang sama (*independently and identically distributed*). Distribusi pada masing-masing dataset ini juga sebaiknya seimbang (*balanced*) dan memuat seluruh kasus. Misal, sebuah dataset *binary classification* sebaiknya memuat 50% kasus positif dan 50% kasus negatif.

Pada umumnya, rasio pembagian *dataset* (*training*: *validation*: *testing*) adalah (80% : 10% : 10%) atau (90% : 5% : 5%). *Validation set* pada umumnya bisa tidak digunakan apabila *dataset* berukuran kecil (hanya dibagi menjadi *training* dan *testing set* saja). Dalam kasus ini, pembagian *dataset* menjadi *training* dan *testing* set pada umumnya memiliki rasio (90% : 10%), (80% : 20%), (70% : 30%), atau (50% : 50%). Pada kasus ini, kinerja saat *training* diuji menggunakan *training set* (dikenal sebagai ***closed testing***).

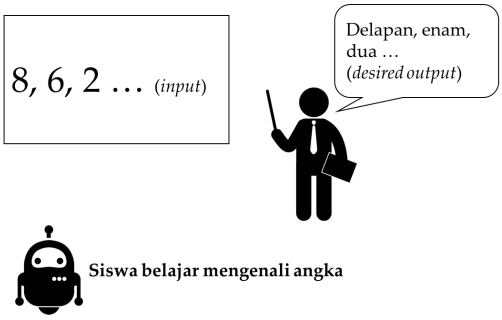
Saat tidak menggunakan *validation set* (hanya ada *training* dan *testing set*), kita juga memiliki opsi untuk mengevaluasi model dengan metode *K-cross-validation*.[10] Artinya, kita membagi *training dataset* (atau keseluruhan *dataset*) menjadi K bagian. Kita menggunakan $K-1$ bagian untuk *training*, kemudian menguji kinerja model saat latihan (*validation*) menggunakan satu bagian. Hal ini diulangi sebanyak K kali dimana sebuah bagian data digunakan sebagai *testing set* sebanyak sekali (bergilir). Mungkin sekarang kamu merasa pusing karena membaca banyak istilah. Jangan khawatir! Kamu akan lebih meresapi arti istilah-istilah tersebut seiring membaca buku ini.

1.6 Supervised Learning

Jika diterjemahkan secara literal, *supervised learning* adalah pembelajaran terarah/terawasi. Artinya, pada pembelajaran ini, ada guru yang mengajar (mengarahkan) dan siswa yang diajar. Kita disini berperan sebagai guru, kemudian mesin berperan sebagai siswa. Perhatikan Gambar 1.5 sebagai ilustrasi! Pada Gambar 1.5, seorang guru menuliskan angka di papan "8, 6, 2" sebagai contoh untuk siswanya, kemudian gurunya memberikan cara membaca yang benar untuk masing-masing angka. Contoh angka melambangkan ***input***, kemudian cara membaca melambangkan ***desired output*** (sering disebut "***gold standard***"). Pasangan ***input–desired output*** ini disebut sebagai *instance* (untuk kasus *supervised learning*). Pembelajaran metode ini disebut *supervised* karena ada yang memberikan contoh jawaban (*desired output*).

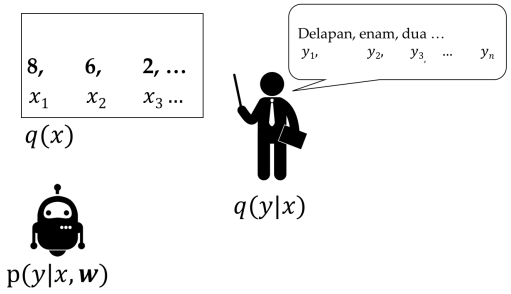
Perhatikan Gambar 1.6 dan Gambar 1.7, x adalah kejadian (*event–random variable*), untuk *event* tertentu dapat dinotasikan sebagai $\{x_1, x_2, x_3, \cdots, x_N\}$. x dapat berupa vektor, teks, gambar, dan lain sebagainya (perhatikan konteks pembahasan buku). Demi pembahasan yang cukup generik, pada bab ini kita membicarakan x yang merepresentasikan *event*, *data point*, atau *input*. Seorang guru sudah mempunyai jawaban yang benar untuk masing-masing contoh dengan suatu fungsi distribusi probabilitas kondisional (***conditional probability density function***) $q(y \mid x)$ baca: *function q for y given x*, melambangkan hasil yang benar/diharapkan untuk suatu event. Siswa (mesin) **mempelajari tiap pasang pasangan *input-desired output (training data)*** dengan mengoptimalkan *conditional probability density function* $p(y \mid x, \mathbf{w})$, dimana y adalah target (*output*), x adalah input dan vektor $\mathbf{w}$ adalah

[10] https://www.openml.org/a/estimation-procedures/1



Gambar 1.5: *Supervised learning.*

learning parameters. Proses belajar, yaitu mengoptimalkan **w** disebut sebagai training. Semakin kamu membaca buku ini, konsep ini akan menjadi semakin jelas. Proses *training* bertujuan untuk mengaproksimasi $q(y \mid x)$ melalui $p(y \mid x, \mathbf{w})$. Proses pembelajaran bertujuan untuk mengubah-ubah **w** sedemikian sehingga $p(y \mid, x, \mathbf{w})$ mirip dengan $q(y \mid x)$.

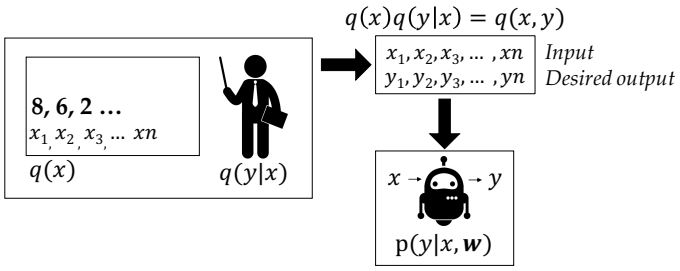


Gambar 1.6: *Supervised learning - mathematical explanation.*

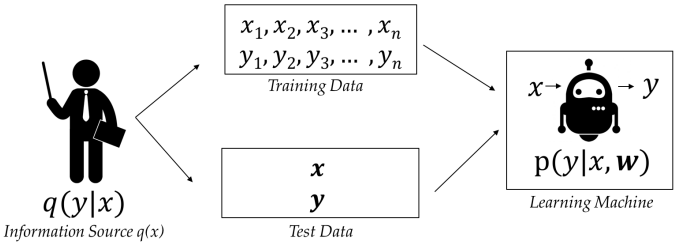
Perhatikan Gambar 1.8! model memiliki panah ke *training data* dan *test data*, artinya model hasil *training* sangat bergantung pada **data dan guru**. Model yang dihasilkan *training* (hasil pembelajaran kemampuan siswa) untuk data yang sama bisa berbeda untuk guru yang berbeda.[11]

Tujuan *supervised learning*, secara umum untuk melakukan klasifikasi (*classification*). Misalkan mengklasifikasikan gambar buah (apa nama buah pada gambar), diilsutrasikan pada Gambar 1.9. Apabila hanya ada dua kategori, disebut ***binary classification***. Sedangkan bila terdapat lebih dari dua kategori, disebut ***multi-class classification***. Contoh *multi-class classifica-*

[11] Penulis rasa hal ini sangat intuitif berhubung hal serupa terjadi pada manusia.



Gambar 1.7: *Supervised learning - mathematical explanation* 2.



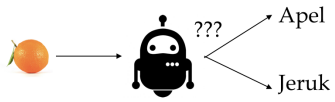
Gambar 1.8: *Supervised learning framework.*

tion adalah mengklasifikasikan gambar buah ke dalam himpunan kelas: *apel*, *mangga* atau *sirsak*.

Ada tipe klasifikasi lain disebut ***multi-label classification*** yaitu ketika kita ingin mengklasifikasikan suatu sampel ke dalam suatu himpunan kelas. Perhatikan! Perbedaan *multi-class* dan *multi-label classification* agak *tricky*. Pada *multi-class classification*, suatu sampel hanya bisa berkorespondensi dengan satu kelas. Sedangkan pada *multi-label classification*, satu sampel dapat berkorespondensi dengan lebih dari satu kelas. Misalnya, suatu berita dapat masuk ke kategori *agama* dan *politik* pada waktu bersamaan. Artinya, label pada *multi-class classification* bersifat *mutually exclusive*, sedangkan label tidak bersifat *mutually exclusive* pada *multi-label classification*.[12] Perhatikan Gambar 1.1 sebagai ilustrasi, dimana setiap baris merepresentasikan kelas yang berkorespondensi dengan setiap *input*, nilai "1" melambangkan TRUE dan nilai "0" melambangkan FALSE. *Multi-label classification* dapat didekomposisi menjadi beberapa *Binary classification*, yaitu mengklasifikasikan apakah *input* dapat ditetapkan ke suatu kelas atau tidak (dijelaskan lebih lanjut pada bab-bab berikutnya).

Pemahaman *supervised learning* adalah mengingat persamaan 1.1. Ada tiga hal penting pada *supervised learning* yaitu *input*, *desired output*, dan

[12] https://scikit-learn.org/stable/modules/multiclass.html



Gambar 1.9: Ilustrasi *binary classification*.

Instance	**Apel**	**Mangga**	**Sirsak**
Gambar-1	1	0	0
Gambar-2	0	1	0
Gambar-3	0	0	1

(a) Multi-class classification.

Instance	**Apel**	**Mangga**	**Sirsak**
Gambar-1	1	1	0
Gambar-2	0	1	1
Gambar-3	1	0	1

(b) Multi-label classification.

Tabel 1.1: Ilustrasi *multi-label* dan *multi-class classification*. Nilai "1" melambangkan TRUE dan "0" melambangkan FALSE (*class assignment*).

learning parameters. Perlu ditekankan *learning parameters* berjumlah lebih dari satu, dan sering direpresentasikan dengan vektor (*bold*) atau matriks. Berdasarkan model yang dibuat, kita dapat melakukan klasifikasi (misal simbol yang ditulis di papan adalah angka berapa). Secara konseptual, klasifikasi didefinisikan sebagai persamaan 1.2 yaitu memilih label (kelas/kategori y) paling optimal dari sekumpulan label C, diberikan (*given*) suatu sampel (*instance*) data tertentu.

$$p(y \mid x, \mathbf{w}) \tag{1.1}$$

$$\hat{y_i} = \arg\max_{y_i \in C} p(y_i \mid x_i, \mathbf{w}) \tag{1.2}$$

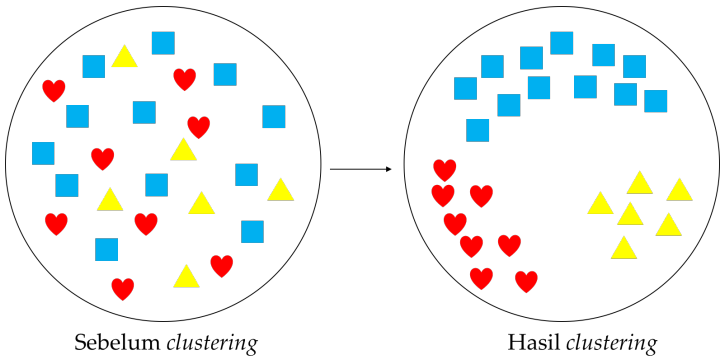
1.7 Regresi

Pada persoalan regresi, kita ingin memprediksi *output* berupa bilangan kontinu. Misalnya pada regresi suatu fungsi polinomial, kita ingin mencari tahu fungsi $f(x)$ diberikan data $\{(x_1, y_1), \cdots, (x_N, y_N)\}$. Setelah itu, kita gunakan fungsi aproksimasi untuk mencari tahu nilai y_{N+1} dari data baru x_{N+1}. Perbedaan regresi dan klasifikasi adalah pada tipe *output*. Untuk regresi, tipe *output* adalah nilai kontinu; sementara tipe *output* pada persoalan klasifikasi adalah suatu objek pada himpunan (i.e., memilih opsi pada himpunan jawaban). Tetapi, kita dapat mengkonversi fungsi regresi menjadi fungsi klasifikasi (dijelaskan pada bab 5).

1.8 Semi-supervised Learning

Semi-supervised learning mirip dengan *supervised learning*, bedanya pada proses pelabelan data. Pada *supervised learning*, ada "guru" yang harus membuat "kunci jawaban" *input-output*. Sedangkan pada *semi-supervised learning* tidak ada "kunci jawaban" eksplisit yang harus dibuat guru. Kunci jawaban ini dapat diperoleh secara otomatis (misal dari hasil *clustering*). Pada kategori pembelajaran ini, umumnya kita hanya memiliki sedikit data. Kita kemudian menciptakan data tambahan baik menggunakan *supervised* ataupun *unsupervised learning*, kemudian membuat model belajar dari data tambahan tersebut.

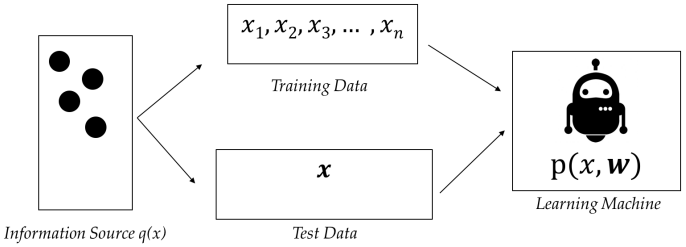
1.9 Unsupervised Learning



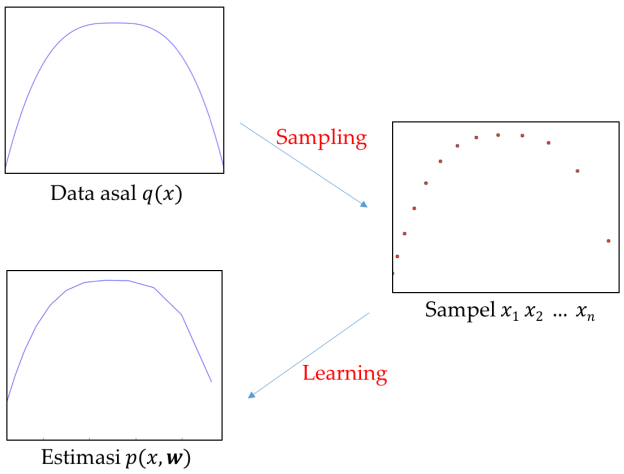
Gambar 1.10: Ilustrasi *clustering*

Jika pada *supervised learning* ada guru yang mengajar, maka pada *unsupervised learning* tidak ada guru yang mengajar. Contoh permasalahan unsupervised learning adalah *clustering*. Mengingat contoh kue sebelumnya, kita ingin mengelompokkan kue-kue yang sama, diilustrasikan oleh Gambar 1.10. Yang kamu lakukan adalah membuat kelompok-kelompok berdasarkan karakteristik kue, misal kelompok kue biru, kelompok kue kuning, atau kelompok kue merah. Teknik-teknik mengelompokkan ini akan dibahas pada bab-bab berikutnya. Contoh algoritma *unsupervised learning* sederhana adalah *K-means* (bab 4).

Perhatikan Gambar 1.11 dan Gambar 1.12! Berbeda dengan supervised learning yang memiliki *desired output*, pada *unsupervised learning* tidak ada *desired output* (jelas, tidak ada gurunya, tidak ada yang memberi contoh).



Gambar 1.11: *Unsupervised learning framework*



Gambar 1.12: *Generalization error of unsupervised learning*

Kita ingin mencari tahu distribusi asli data $q(x)$, berdasarkan beberapa sampel data. *Learning* dilakukan dengan mengoptimalkan $p(x \mid \mathbf{w})$ yang mengoptimasi parameter $\mathbf{w}$. Perbedaan antara estimasi dan fungsi asli disebut sebagai ***generalization loss*** (atau ***loss*** saja – dijelaskan pada bab 5). Kunci pemahaman *unsupervised learning* adalah mengingat persamaan 1.3, yaitu ada *input* dan parameter.

$$p(x \mid \mathbf{w}) \tag{1.3}$$

Perlu kami tekankan, *unsupervised learning* $\neq$ *clustering*! *Clustering* adalah salah satu bentuk *unsupervised learning*; yaitu salah satu hasil inferensi persamaan 1.3. *Unsupervised learning* adalah mencari sifat-sifat (*properties*) data. Kita ingin aproksimasi $p(x \mid \mathbf{w})$ semirip mungkin dengan $q(x)$, dimana $q(x)$ adalah distribusi data yang asli. Dataset di-sampel dari distribusi $q(x)$, kemudian kita ingin mencari tahu $q(x)$ tersebut.

1.10 Proses Belajar

Seperti yang sudah dijelaskan pada subbab sebelumnya, pada *supervised* maupun *unsupervised learning*, kita ingin mengestimasi sesuatu dengan teknik *machine learning*. Kinerja model pembelajaran berubah-ubah sesuai dengan parameter $\mathbf{w}$ (parameter pembelajaran). Kinerja model diukur oleh fungsi tujuan (*utility function*–saat latihan, *performance measure*–saat melakukan prediksi), yaitu mengoptimalkan nilai fungsi tertentu; misalnya meminimalkan nilai *error* (dijelaskan kemudian). Secara intuitif, *learning machine* mirip seperti saat manusia belajar. Kita awalnya membuat banyak kesalahan, tetapi kita mengetahui atau diberi tahu mana yang benar. Untuk itu kita menyesuaikan diri secara perlahan agar menjadi benar (iteratif). Inilah yang juga dilakukan *learning machine*, yaitu mengubah-ubah parameter $\mathbf{w}$ untuk mengoptimalkan suatu fungsi tujuan. Akan tetapi, *machine learning* membutuhkan sangat banyak data. Sementara, manusia dapat belajar dengan contoh yang sedikit. Perhatikan, dengan adanya parameter $\mathbf{w}$, kamu mungkin akan menganggap bahwa teknik *machine learning* adalah fungsi-parametrik. Sebenarnya, ada juga algoritma *machine learning* non-parametrik, tetapi algoritham parametrik yang lebih sering digunakan di masa buku ini ditulis.

Secara bahasa lebih matematis, kami beri contoh *supervised learning*. Kita mempunyai distribusi klasifikasi asli $q(y \mid x)$. Dari distribusi tersebut, kita diberikan beberapa sampel pasangan *input-output* $\{z_1, z_2, z_3, \ldots, z_n\}; z_i = (x_i, y_i)$. Kita membuat model $p(y \mid x, \mathbf{w})$. Awalnya diberi (x_1, y_1), model mengestimasi fungsi asli dengan mengoptimalkan parameter $\mathbf{w}$ sesuai dengan data yang ada. Seiring berjalannya waktu, ia diberikan data observasi lainnya, sehingga *learning machine* menyesuaikan dirinya (konvergen) terhadap observasi yang baru $(x_2, y_2), (x_3, y_3), \ldots$. Semakin lama, kita jadi makin percaya bahwa model semakin optimal (mampu memprediksi fungsi aslinya). Apabila kita diberikan lebih banyak data data sampai sejumlah tak hingga, aproksimasi kita akan semakin mirip dengan distribusi aslinya.

1.11 Tips

Jujur, pengarang sendiri belum menguasai bidang ini secara penuh, tetapi berdasarkan pengalaman pribadi (+membaca) dan beberapa rekan; ada beberapa materi wajib yang harus dipahami untuk mengerti bidang *machine learning*. Sederhananya, kamu harus menguasai banyak teori matematika dan probabilitas agar dapat mengerti *machine learning* sampai tulang dan jeroannya. Kami tidak menyebutkan bahwa mengerti *machine learning* secara intuitif (atau belajar dengan pendekatan deskriptif) itu buruk, tetapi untuk mengerti sampai dalam memang perlu mengerti matematika (menurut pengalaman kami). Disarankan untuk belajar materi berikut:

1. Matematika Diskrit dan Teori Bilangan

2. Aljabar Linier dan Geometri (vektor, matriks, skalar, dekomposisi, transformasi, tensor, dsb)
3. Kalkulus (diferensial dan integral)
4. Teori Optimasi (*Lagrange multiplier, Convex Iptimization, Gradient Descent, Integer Linear Problem,* dsb)
5. Probabilitas dan Statistika (probabilitas, *probability densities*, *hypothesis testing*, *inter-rater agreement*, Bayesian, *sampling*)
6. Teori Fuzzy

Mungkin kamu sudah tahu, tetapi penulis ingin mengingatkan ada dua buku yang sangat terkenal ("kitab") sebagai materi belalajar *machine learning* dan *deep learning*:

1. Patten Recognition and Machine Learning, oleh Christopher M. Bishop [8]
2. Deep Learning, oleh Ian Goodfellow, Yoshua Bengio, dan Aaron Courville [11]

Apabila pembaca memiliki kesempatan, penulis sangat menyarankan membaca kedua buku tersebut.

1.12 Contoh Aplikasi

Sebenarnya, aplikasi pemanfaatan *machine learning* sudah terasa dalam kehidupan sehari-hari. Contoh mudahnya adalah produk-produk Google, misalnya google translate (machine translation, handwritten recognition, speech recognition, Alpha Go). Berikut adalah beberapa artikel menarik:

1. techcrunch google AI beats go world champion
2. http://www-formal.stanford.edu/jmc/whatisai/node3.html
3. https://www.google.com/selfdrivingcar/
4. http://www.osnews.com/story/26838/Palm_I_m_ready_to_wallow_now/page2/

Soal Latihan

1.1. Aplikasi

(a) Carilah contoh-contoh penerapan *machine learning* pada kehidupan sehari-hari selain yang telah disebutkan!
(b) Mengapa mereka menggunakan teknik *machine learning* untuk menyelesaikan permasalahan tersebut?
(c) Apakah tidak ada opsi teknik lainnya? Jelaskan bila ada!
(d) Apa kelebihan dan kekurangan teknik *machine learning* daripada teknik lainnya (yang kamu jelaskan pada soal (c))?

1.2. Kecerdasan

Jelaskan tahapan perkembangan kecerdasan manusia berdasarkan kategori usia! Dari hal ini, kamu akan mengerti kenapa beberapa peneliti membuat agen cerdas berdasarkan kategori usia tertentu.

2

Fondasi Matematis

> "He uses statistics as a drunken man uses lamp posts – for support rather than for illumination."
>
> Andrew Lang

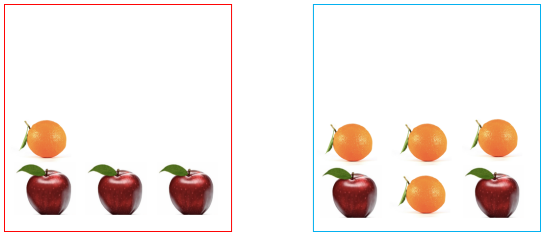
Mungkin saat pertama kali membaca bab ini, kamu merasa bab ini tidak masuk akal atau kurang dibutuhkan. Seiring membaca buku ini, mungkin bab ini akan sering dikunjungi kembali. Bab ini hanyalah pengingat materi yang sudah kamu pernah pelajari saja (semacam *cheatsheet*). Kamu boleh melewati bab ini apabila sudah familiar dengan materi probabilitas, statistika, serta aljabar linier. Seharusnya, bab ini memang diberikan paling awal. Tetapi penulis khawatir pembaca mungkin merasa buku ini tidak menarik apabila bab ini diberikan paling awal.

Bab ini memuat sekilas tentang probabilitas, statistika, dan operasi matriks. Tentunya untuk mengerti materi tersebut sebaiknya kamu mengambil kuliah khusus berkaitan karena kamu diharapkan sudah memiliki cukup latar pengetahuan, bab ini sebenarnya hanyalah sekilas pengingat. Kami akan banyak memakai contoh-contoh dari buku Bishop [8] untuk materi probabilitas. Penulis merekomendasikan untuk menonton kuliah *statistical inference* (coursera) oleh Professor Brian Caffo.

2.1 Probabilitas

Di dunia ini, ada banyak hal yang tidak pasti (*uncertain*). Ssungguhnya, *machine learning* berurusan dengan ketidakpastian (*uncertainty*). Dengan begitu, *machine learning* memiliki kaitan yang sangat erat dengan statistika. Probabilitas menyediakan *framework* untuk kuantifikasi dan manipulasi ketidakpastian [8]. Mari kita lihat contoh sederhana. Terdapat dua buah kotak

berwarna merah dan berwarna biru. Pada kotak merah terdapat 3 *apel* dan 1 *jeruk*. Pada kotak biru, terdapat 2 *apel* dan 4 *jeruk*, kita ingin mengambil buah dari salah satu kotak tersebut. Ilustrasi persoalan dapat dilihat pada Gambar 2.1. Dalam hal ini, kotak adalah *random variable*. *Random variable* k (melambangkan kotak) dapat bernilai *merah* atau *biru*. Begitu pula dengan buah, dilambangkan dengan variabel b, dapat bernilai *apel* atau *jeruk*.



Gambar 2.1: Kotak apel dan jeruk.

Saat kita mengambil buah dari kotak biru, peluang untuk memilih *apel* bernilai $\frac{2}{6}$, sedangkan peluang untuk memilih *jeruk* bernilai $\frac{4}{6}$; kita tulis probabilitas ini sebagai $P(b = apel) = \frac{2}{6}$; dan $P(b = jeruk) = \frac{4}{6}$. Artinya, jika kita mengambil buah dari kotak biru, mungkin kita mendapatkan jeruk. Dengan mengkuantifikasi ketidakpastian, kita bisa mengatur ekspektasi. Nilai suatu probabilitas harus lebih besar sama dengan nol sampai kurang dari atau sama dengan satu ($0 \leq P \leq 1$). Nilai nol berarti suatu kejadian tidak mungkin muncul, sementara nilai satu berarti suatu kejadian pasti terjadi.

Lalu sekarang ada pertanyaan baru; pada suatu percobaan, berapakah probabilitas mengambil sebuah *apel* dari kotak biru **atau** sebuah *jeruk* dari kotak merah. Hal ini dituliskan sebagai $P((k = biru, b = apel)$ **atau** $(k = merah, b = jeruk))$. Nilai probabilitas tersebut dapat dihitung dengan

$$\begin{aligned} &P((k = biru, b = apel) \vee (k = merah, b = jeruk)) \\ &= P(k = biru, b = apel) + P(k = merah, b = jeruk) \end{aligned} \tag{2.1}$$

- $P(k = biru, b = apel)$ disebut *joint probability*, yaitu probabilitas kejadian yang dipengaruhi oleh beberapa variabel (kondisi untuk kedua variabel terpenuhi).
- $P(k = biru, b = apel) + P(k = merah, b = jeruk)$ disebut **aturan tambah**.

Penting untuk diingat bahwa hasil operasi apapun terhadap probabilitas (baik tambah, kurang, kali, atau bagi) haruslah lebih besar sama dengan nol sampai kurang dari atau sama dengan satu ($0 \leq P \leq 1$).

Misalkan terdapat percobaan lain, kali ini kamu mengambil 1 buah. Kamu ingin mengetahui berapakah probabilitas untuk mengambil buah *apel* kotak mana saja. Hal ini dihitung dengan persamaan 2.2.

$$P(b = apel) = \sum_{i=1}^{I} P(k = k_i, b = apel) \tag{2.2}$$

Aturan tambah seperti ini disebut ***marginal probability*** karena hasilnya didapat dengan menjumlahkan probabilitas seluruh kemungkinan nilai pada variabel tertentu (buah) dengan mengontrol variabel lainnya (kotak).

Kemudian, kamu ingin melakukan percobaan lain. Kali ini kamu mengambil 2 buah sekaligus dari kedua kotak. Kamu ingin mengetahui berapakah probabilitas mengambil buah *apel* yang berasal dari kotak biru **dan** buah *jeruk* yang berasal dari kotak merah. Dalam kasus ini, kejadiannya adalah saling bebas (*independent*), artinya mengambil buah dari kotak biru tidak akan mempengaruhi hasil pengambilan dari kotak merah (dan sebaliknya). Apabila kedua *random variable* x dan y bersifat ***independent*** (tidak bergantung satu sama lain), maka $P(x = X, y = Y) = P(X) \times P(Y)$. Permasalahan mengambil buah dapat dihitung dengan persamaan 2.3.

$$\begin{aligned} &P((k = biru, b = apel) \wedge (k = merah, b = jeruk)) \\ &= P(k = biru, b = apel) \times P(k = merah, b = jeruk) \end{aligned} \tag{2.3}$$

Aturan ini disebut **aturan kali**.

Untuk *joint probability*, secara umum dapat ditulis sebagai $P(x, y)$, yaitu peluang x dan y muncul bersamaan. Apabila kedua variabel x dan y tidak saling bebas, maka keduanya disebut ***dependent***. Artinya x dan y saling mempengaruhi. Apabila suatu variabel x dikondisikan (*conditioned*) oleh variabel lain (misal y). Maka probabilitas x adalah *conditional probability function*, ditulis $P(x \mid y)$. Artinya probabilitas x yang dikondisikan oleh y. $P(x \mid y)$ dapat dihitung dengan persamaan 2.4, yaitu peluang kejadian x dan y muncul bersamaan dibagi dengan peluang kejadian y. Apabila x ternyata tidak dikondisikan oleh variabel y, maka $P(x \mid y) = P(x)$.

$$P(x \mid y) = \frac{P(x, y)}{P(y)} \tag{2.4}$$

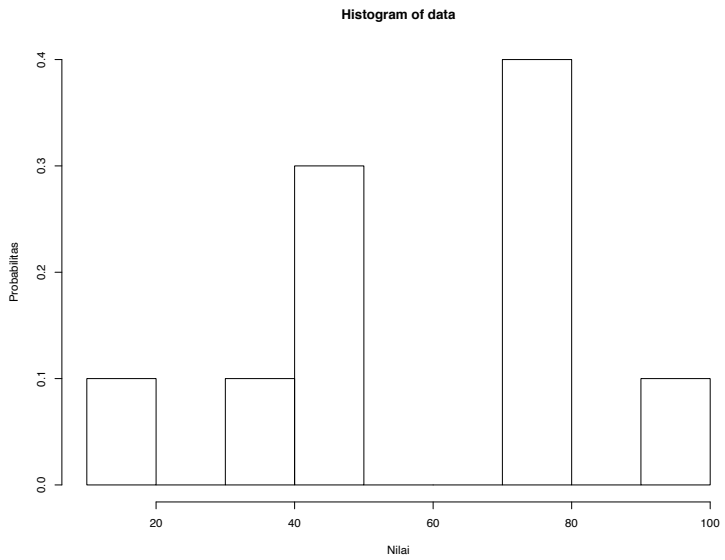
2.2 Probability Density Function

Probability density function dikenal juga dengan istilah distribusi, yaitu tentang persebaran nilai. Sebagai contoh, penulis menceritakan pelajaran di sekolah. Terdapat ujian mata pelajaran di kelas yang beranggotakan 10 siswa, diberikan pada Tabel 2.1. Terdapat 3 orang anak mendapatkan nilai 50, 2 orang anak mendapatkan nilai 75 dan 80, 1 orang anak mendapatkan nilai

id	nilai
1	50
2	75
3	80
4	100
5	50
6	50
7	75
8	80
9	40
10	10

Tabel 2.1: Contoh daftar nilai siswa.

100, 1 orang anak mendapat nilai 40, serta 1 orang anak mendapatkan nilai 10.



Gambar 2.2: Persebaran probabilitas nilai siswa Tabel 2.1.

Guru ingin mengetahui persebaran (distribusi) nilai ujian untuk menentukan batas kelas nilai menggunakan *quantile*. Grafik persebaran nilai mengkuantifikasi seberapa mungkin suatu siswa mendapat nilai tertentu, dapat dilihat pada Gambar 2.2. Grafik ini disebut sebagai distribusi. Fungsi yang menghasilkan distribusi tersebut disebut *probability density function*. Apabila kita

menjumlahkan probabilitas (probabilitas siswa mendapat nilai 0–100) nilainya adalah 1.

Ini adalah contoh untuk data diskrit, tetapi sering kali kita berurusan dengan data kontinu. Untuk mengetahui nilai probabilitas dari himpunan *event*/kejadian, kita dapat mengintegralkan kurva distribusi kejadian pada interval tertentu. Ciri *probability density function*, nilai dibawah kurva pada interval $-\infty$ sampai ∞ adalah 1, i.e., $p(x) \geq 0; \int_{-\infty}^{\infty} p(x)dx = 1$.

2.3 Expectation dan Variance

Salah satu operasi paling penting dalam probabilitas adalah menemukan nilai rata-rata (*average*) sebuah fungsi [8]. Hal ini disebut menghitung ekspektasi (*expectation*). Untuk sebuah fungsi $f(x)$ dengan distribusi probabilitas *random variable* adalah $p(x)$, nilai expectation diberikan pada persamaan 2.5.

$$E(f) = \begin{cases} \sum_x p(x)f(x); & diskrit \\ \int p(x)f(x)dx; & kontinu \end{cases} \tag{2.5}$$

Dalam kasus nyata, misalkan diberikan N buah sampel, *random variable* x dan fungsi $f(x)$, dimana sampel tersebut diambil dengan distribusi tertentu yang kita tidak ketahui, maka fungsi untuk menghitung nilai *expectation* menjadi persamaan 2.6, dimana x_i merepresentasikan data ke-i (*point*).

$$E(f) \simeq \frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{2.6}$$

Perhatikan, persamaan ini sama dengan persamaan untuk menghitung rata-rata (*mean* atau μ) seperti yang sudah kamu pelajari di SMA. Untuk mengetahui seberapa variasi nilai $f(x)$ di sekitar nilai rata-ratanya, kita menghitungnya mengunakan *variance*, disimbolkan dengan $var(f)$ atau σ^2 (persamaan 2.7).

$$\sigma^2 = var(f) = E\left((f(x) - E(f))^2\right) \tag{2.7}$$

Bila nilai *variance* tinggi, secara umum banyak variabel yang nilainya jauh dari nilai rata-rata. Interpretasi secara "geometris" mata, berarti distribusinya semakin "lebar" seperti pada Gambar 2.3. Untuk fungsi dengan lebih dari satu *random variable*, kita menghitung *covariance*. *Covariance* adalah *variance* untuk kombinasi variabel.

Akar dari *variance*, yaitu $\sqrt{\sigma^2} = \sigma$ disebut *standard deviation* (sering disingkat "SD"). SD melambangkan seberapa jauh jarak dari masing-masing data point x_i dengan rata-rata μ. SD cukup penting karena merupakan ukuran "persebaran" (*spread out*) data.

2.4 Bayesian Probability

Pada subbab sebelumnya, kita menghitung probabilitas dengan frekuensi kejadian yang dapat diulang. Pada pandangan Bayesian, kita ingin menguantifikasi ketidakpastian untuk kejadian yang mungkin sulit untuk diulang. Misalkan kita ingin tahu, seberapa peluang Mars dapat dihuni. Ini adalah sesuatu yang tidak dapat dihitung dengan frekuensi, maupun sebuah kejadian yang dapat diulangi (pergi ke mars, lihat berapa orang yang hidup). Akan tetapi, tentunya kita memiliki sebuah asumsi awal (*prior*). Dengan sebuah alat canggih baru, kita dapat mengumpulkan data baru tentang Mars. Dengan data tersebut, kita mengoreksi pendapat kita tentang Mars (*posterior*). Hal ini menyebabkan perubahan dalam pengambilan keputusan.

Pada keadaan ini, kita ingin mampu menguantifikasi ekspresi ketidakpastian; dan membuat revisi tentang ketidakpastian menggunakan bukti baru [8]. Dalam Bayesian, nilai probabilitas digunakan untuk merepresentasikan derajat kepercayaan/ketidakpastian.

$$P(x \mid y) = \frac{P(y \mid x)P(x)}{P(y)} \tag{2.8}$$

$P(x)$ disebut *prior*, yaitu pengetahuan/asumsi awal kita. Setelah kita mengobservasi fakta baru y (dapat berupa sekumpulan data atau satu *data point*/*event*), kita mengubah asumsi kita. $P(y \mid x)$ disebut ***likelihood function***. *Likelihood function* mendeskripsikan peluang data, untuk asumsi/ pengetahuan tentang x yang berubah-ubah (x sebagai parameter yang dapat diatur). Dengan *likelihood function* tersebut, kita mengoreksi pendapat akhir kita yang dapat digunakan untuk mengambil keputusan (*posterior*). Secara umum probabilitas Bayesian mengubah *prior* menjadi *posterior* akibat adanya kepercayaan baru (*likelihood*).

$$posterior \propto likelihood \times prior \tag{2.9}$$

Teori ini hebat karena kita dapat mentransformasi $P(x \mid y)$ dimana x dependen terhadap y menjadi bentuk $P(y \mid x)$ yang mana y dependen terhadap x. Transformasi ini sangat berguna pada berbagai macam persoalan.

Pada umumnya, untuk mengestimasi *likelihood*, digunakan *maximum likelihood estimator*; yang berarti mengatur nilai x untuk memaksimalkan nilai $P(y \mid x)$. Dalam literatur *machine learning*, banyak menggunakan *negative log of likelihood function* [8]. Ingat kembali nilai probabilitas $0 \leq P \leq 1$. Kadangkala, nilai dibelakang koma (0.xxxx) sangatlah panjang, sehingga dapat terjadi $underflow$ pada komputer. Kita menggunakan nilai logaritma probabilitas untuk menghindari $underflow$. Nilai probabilitas $0 \leq P \leq 1$ membuat nilai logaritmanya sebagaian besar negatif, secara monotonik bertambah, maka memaksimalkan nilai *likelihood* ekuivalen dengan meminimalkan negatif logaritma probabilitas.

Perhatikan kembali persamaan 2.8, secara intuitif, *posterior* dipengaruhi *prior*, artinya bergantung pada sampel yang kita punya, karena *prior* didapatkan atau diestimasi berdasarkan sampel. Hal ini berlaku pada *machine learning*, kualitas model yang dihasilkan bergantung pada kualitas training data.

Pada umumnya, kita tidak mengetahui seluruh informasi tentang situasi tertentu dan tidak mengetahui seluruh informasi probabilitas. Sebagai contoh, probabilitas $P(x \mid y)$ dapat dihitung dengan $\frac{P(x,y)}{P(x)}$. Tetapi, kita tidak tahu seberapa banyak kejadian (x, y) pada saat bersamaan. Oleh sebab itu, kita bisa menggunakan teori bayes untuk menghitung probabilitas dengan informasi lain yang kita tahu.

2.5 Gaussian Distribution

Distribusi adalah fenomena acak atau deskripsi matematis suatu *random variable*. Distribusi paling terkenal (mungkin juga terpenting) adalah *bell curve* atau distribusi normal. Distribusi normal adalah bentuk khusus dari *Gaussian distribution*. Ada beberapa macam distribusi yang akan dibahas pada bab ini, yaitu: *Univariate Gaussian*, *Multivariate Gaussian*, dan *Gaussian Mixture Model*. Pertama kita bahas ***Univariate Gaussian*** terlebih dahulu.

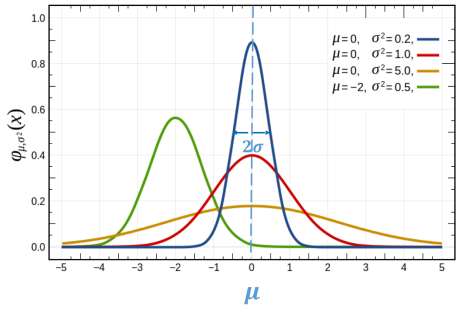
Disebut *univariate* karena distribusinya bergantung pada **satu** input variabel, misalkan x. Distribusi univariate Gaussian dikarakteristikkan oleh variabel x, *mean* (μ) dan *variance* (σ^2) diberikan pada persamaan 2.10. μ dan σ^2 adalah rata-rata dan *variance* untuk kumpulan data. Karena nilai μ dan σ^2 bergantung pada x, maka kita dapat menganggap bahwa *univariate gaussian* bergantung pada satu *random variable* saja yaitu x.

$$N(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)} \tag{2.10}$$

Perhatikan Gambar 2.3, x adalah absis dan nilai N untuk x tertentu (persamaan 2.10) adalah ordinat pada kurva ini. Bentuk distribusi berubah-ubah sesuai dengan nilai rata-rata (*mean*), serta *variance*. Semakin besar *variance*-nya, maka kurva distribusi semakin lebar (seperti yang dijelaskan sebelumnya). Untuk menggeser-geser kurva ke kiri maupun ke kanan, dapat dilakukan dengan menggeser nilai *mean*. Untuk mencari nilai pada suatu interval tertentu, cukup mengintegralkan fungsi pada interval tersebut. Nilai integral fungsi dari $-\infty$, hingga ∞ adalah satu.

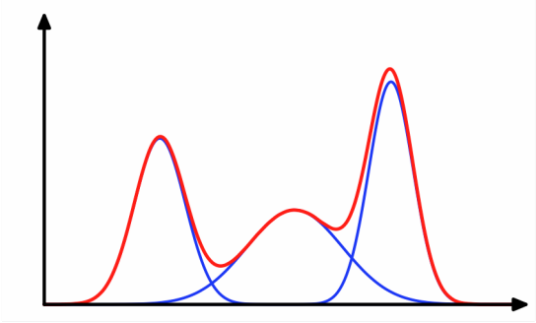
Multivariate Gaussian adalah distribusi gaussian yang bergantung pada lebih dari satu *random variable*. Sedangkan *Gaussian Mixture Model* (GMM) adalah gabungan dari satu atau lebih distribusi Gaussian. Masing-masing distribusi Gaussian memiliki bobot yang berbeda di GMM. Konon katanya,

[1] source: `wikimedia.org by Inductiveload`



Gambar 2.3: Univariate Gaussian.[1]

GMM dapat memodelkan fungsi apapun [12]. Ilustrasinya diberikan pada Gambar 2.4 yang tersusun dari 3 buah *Univariate gaussian*. Distribusi populasi berwarna merah, sedangkan GMM berwarna biru.

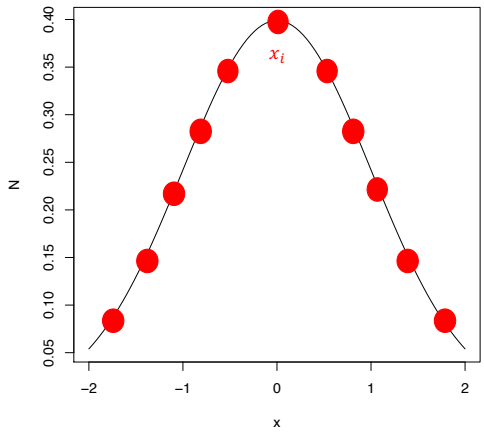


Gambar 2.4: Gaussian Mixture Model.[2]

2.6 Apakah Karakteristik Sampel Mencerminkan Populasi?

Sekarang bayangkan kita diberikan M buah data (diskrit) hasil observasi. Diasumsikan observasi dihasilkan oleh distribusi univariate Gaussian N dengan rata-rata μ dan *variance* σ^2. Ilustrasi diberikan pada Gambar 2.5. Setiap data diambil secara independen dari distribusi yang sama, disebut ***independent and identically distributed*** (iid). Kita tahu bahwa data yang in-

[2] http://dirichletprocess.weebly.com/clustering.html



Gambar 2.5: Contoh *sampling* dari distribusi normal. Titik berwarna merah melambangkan sampel (*instance*).

dependen, apabila dihitung probabilitasnya maka tersusun atas probabilitas masing-masing data, seperti pada persamaan 2.11.

$$p(x \mid \mu, \sigma^2) = \prod_{i=1}^{M} N(x_i \mid \mu, \sigma^2) \tag{2.11}$$

Kita ingin mencari tahu bagaimana distribusi yang sebenarnya. Untuk itu, kita harus mencari nilai μ dan σ^2 pada $p(x \mid \mu, \sigma^2)$. Kita dapat menggunakan teknik *maximum likelihood estimation*, untuk mengestimasi parameter yang memberikan distribusi (silahkan dieksplorasi lebih lanjut). Secara empiris, ternyata *mean* dari sampel dan *variance* dari sampel (diberikan sampel dengan M yang cukup banyak) cukup baik untuk mengestimasi *mean* dan *variance* dari distribusi sebenarnya (persamaan 2.12).

$$\mu \approx \mu_{sample} = \frac{1}{M}\sum_{i=1}^{M} x_i; \qquad \sigma^2 \approx \sigma^2_{sample} = \frac{1}{M}\sum_{i=1}^{M}(x_i - \mu_{sample})^2 \tag{2.12}$$

Pada statistik, apabila kita mengetahui (atau mengasumsikan) bahwa suatu kumpulan sampel diambil dari suatu distribusi q, maka kita bisa mengestimasi distribusi q menggunakan sampel yang ada sebagai p. Apabila sampel yang diberikan berjumlah sangat banyak ($M \to \infty$), maka p akan semakin mirip dengan q (konvergen). Kami menyarankan pembaca untuk menonton kuliah *statistical inference* oleh Professor Brian Caffo, dimana konsep mengestimasi *mean* dan *variance* populasi diestimasi menggunakan *mean* dan *variance* sampel dijelaskan dengan sangat baik (dengan banyak ilustrasi).

Hal ini memiliki interpretasi (secara kasar) yang cukup penting pada *machine learning*. Masih ingat materi bab 1? Pada *machine learning*, kita juga mengestimasi sesuatu yang kita tidak ketahui (populasi) dengan sampel data yang kita miliki. Kita anggap populasi memiliki karakteristik yang sama dengan sampel data. Semakin banyak sampel data yang kita miliki, maka semakin bagus estimasi kita terhadap populasi (jumlah dataset menjadi sangat penting).

Dengan bahasa lebih manusiawi, misal kamu diminta untuk membuat model klasifikasi berita otomatis untuk outlet berita tertentu. Populasi data yang ada yaitu: (1) berita yang sudah di-*publish* selama ini dan (2) berita di masa akan datang. Sampel yang kita punya adalah (1), yaitu berita masa lalu. Kita asumsikan bahwa sampel ini dapat mencerminkan karakteristik berita yang akan datang. Sehingga, model yang dilatih dengan data (1) dapat digunakan untuk memproses data (2).

Walau demikian, asumsi bahwa sampel berita yang dimiliki memiliki karakteristik yang sama dengan berita yang akan muncul di masa akan datang belum tentu tepat di dunia nyata. Sebagai contoh, kata *data science* belum digunakan pada tahun 1990an. Kosa kata selalu berkembang sesuai jaman. Bagaimana jika editor outlet berita bersangkutan mengundurkan diri dan diganti dengan editor lainnya? Gaya penulisan bisa saja berubah. Artinya, karaketeristik data sampel yang kita miliki hanya dapat mencerminkan data masa depan apabila banyak kondisi tetap sama, i.e., (1) dan (2) diambil dari distribusi yang sama. Sedangkan pada kenyataan, (1) dan (2) bisa jadi diambil dari distribusi yang berbeda. Pada kasus ini, model pembelajaran mesin yang dihasilkan menjadi *obsolete*, dan kita harus membuat model yang baru. Melatih atau membuat model baru menjadi peling untuk merefleksikan perubahan pada dunia.

2.7 Teori Keputusan

Diberikan himpunan pasangan data *input-output* $(x_i, y_i); x = input, y = output/target$; walaupun tidak pasti, kita ingin mengestimasi hubungan antara *input* dan *output*. Untuk itu kita melakukan estimasi $p(y \mid x, \mathbf{w})$, dimana $\mathbf{w}$ adalah *learning parameters*. Pada bab pertama, kamu telah mempelajari bahwa kita mampu melakukan hal ini dengan teknik *machine learning*. Lebih jauh lagi, kita juga harus mampu untuk membuat keputusan berbasiskan perkiraan nilai y, aspek ini disebut *decision theory* [8].

Dalam *machine learning* kita dapat membangun model dengan tujuan untuk meminimalkan *error* (secara umum meminimalkan *loss*); konsep meminimalkan *error* dijelaskan pada materi model linear (bab 5). Ibaratnya untuk sebuah robot, kita ingin robot tersebut tidak melakukan tindakan yang salah. Tetapi, kadang kala meminimalkan *error* belum tentu membuat model menjadi lebih baik. Kami ilustrasikan menggunakan contoh dari Bishop [8].

Misalkan kita diminta untuk membuat model klasifikasi kanker. Kita dapat mengklasifikasikan pasien menjadi dua kelas $\{kanker, normal\}$.

Apabila kita ingin meminimalkan *error* saja maka kita ingin mengklasifikasikan secara tepat orang yang kanker dianggap memiliki kanker dan yang tidak dianggap sebagai tidak. Akan tetapi, terdapat *tradeoff* yang berbeda saat salah klasifikasi. Apabila kita mengklasifikasikan orang yang normal sebagai kanker, konsekuensi yang mungkin adalah membuat pasien menjadi stres atau perlu melakukan pemeriksaan ulang. Tetapi bayangkan, apabila kita mengklasifikasikan orang kanker sebagai normal, konsekuensinya adalah penanganan medis yang salah. Kedua kasus ini memiliki beban yang berbeda. Secara sederhana, kesalahan klasifikasi bisa memiliki bobot berbeda untuk tiap kelasnya. Untuk penyederhanaan pembahasan, pada buku ini, kita anggap kesalahan klasifikasi memiliki bobot yang sama.

Fungsi tujuan pembelajaran (secara umum untuk merepresentasikan *error* atau *loss*) dituangkan dalam ***utility function***. Sekali lagi kami tekankan, tujuan *machine learning* adalah memaksimalkan kinerja. Kinerja diukur berdasarkan *utility function*. ***Loss* adalah ukuran seberapa dekat atau berbeda model yang dihasilkan dengan konsep asli**, sementara *error* adalah salah satu fungsi untuk mengukur *loss*.

Untuk mengukur nilai *loss*; dapat diekspresikan dengan *loss function*. Secara umum, ada dua macam *loss*, yaitu ***generalization loss/error*** dan ***training loss/error***. *Generalization loss/error* adalah ukuran sejauh mana algoritma mampu **memprediksi** *unobserved data* dengan tepat, karena kita hanya membangun model dengan data yang terbatas, tentunya bisa saja terdapat ketidakcocokan dengan data yang asli. Sedangkan *training loss/error* seperti namanya, ukuran *loss* saat *training*. Misalkan $q(x)$ adalah distribusi data asli. Menggunakan sampel data dengan distribusi $p(x)$, *generalization loss* dan *training loss* dapat dihitung dengan persamaan 2.13. Fungsi ini disebut sebagai ***cross entropy loss***. Perhatikan, masih banyak fungsi lain yang bisa digunakan untuk mengukur *loss*.

$$H = -\sum_{i=1}^{N} q(x)\log\left(p(x)\right) \tag{2.13}$$

Tentunya sekarang kamu bertanya-tanya. Kita tidak mengetahui bagaimana $q(x)$ aslinya, bagaimana cara menghitung *generalization loss*? Nah, untuk itulah ada teknik-teknik pendekatan distribusi populasi $q(x)$, misalnya **maximum likelihood method, maximum posterior method** dan *Bayesian method* (silahkan dieksplorasi). Ekspektasi terhadap biasanya $q(x)$ diaproksimasi dengan menggunakan data sampel yang kita miliki. Bentuk persamaan 2.13 memiliki kaitan dengan *confidence*. Konsep ini akan dijelaskan lebih detil pada bab 5.

Secara lebih filosofis, berkaitan dengan meminimalkan *loss*; tugas *machine learning* adalah untuk menemukan struktur tersembunyi (*discovering hidden structure*). Hal ini sangat erat kaitannya dengan *knowledge discovery* dan *data*

mining. Bila kamu membuka forum di internet, kebanyakan akan membahas perihal *learning machine* yang memaksimalkan akurasi (meminimalkan *error*). Selain harus memaksimalkan akurasi (meminimalkan salah *assignment*), kita juga harus mampu membuat model yang cukup generik. Artinya tidak hanya memiliki kinerja tinggi pada *training data*, tapi juga mampu memiliki kinerja yang baik untuk *unseen data*. Hal ini dapat tercapai apabila model yang dihasilkan melakukan inferensi yang mirip dengan inferensi sebenarnya (konsep asli). Kami tekankan kembali, meminimalkan *loss* adalah hal yang lebih penting, dimana meminimalkan *error* dapat digunakan sebagai sebuah *proxy* untuk mengestimasi *loss* (pada banyak kasus).

2.8 Hypothesis Testing

Diberikan dua model pembelajaran mesin dengan kinerja A dan B. Kita ingin memutuskan model pembelajaran mesin mana yang "lebih baik". Perhatikan, kualitas model tidak hanya tergantung pada satu metrik (misal akurasi), tetapi kita juga mempertimbangkan kompleksitas model (*running time*) dan sebagainya. Hal ini membuat keputusan model mana yang "lebih baik" menjadi cukup kompleks. Demi penyederhanaan pembahasan, anggap kita hanya melihat dari sisi performa yang diukur menggunakan skor tertentu.

Banyak makalah penelitian pembelajaran mesin yang hanya melihat nilai kinerja secara mentah. Artinya, apabila kinerja $A > B$ maka model A memiliki "lebih baik" dari model B. Hal ini terkadang tidak sesuai dengan prinsip statistik, dimana suatu model bisa saja mendapatkan kinerja A secara kebetulan. Konsep *Statistical hypothesis testing* menjadi penting untuk menarik konklusi apakah memang benar $A > B$ secara tidak kebetulan.

Konsep ini menjadi semakin penting saat menggunakan *artificial neural network* (ANN). Parameter model ANN pada umumnya diinisiasi secara *random*. Artinya, apabila kita melatih arsitektur yang sama sebanyak dua kali, kita belum tentu mendapatkan model dengan kinerja sama persis. Reimers dan Gurevych [13] menjelaskan betapa pentingnya melatih model ANN berkali-kali.

Kita hanya dapat menyimpulkan dengan benar model mana yang "lebih baik" hanya setelah melakukan *statistical hypothesis testing*. Ada banyak cara untuk melakukan *hypothesis testing*, dan tes yang digunakan berbeda tergantung metrik *performance measure*. Buku ini tidak akan membahas dengan detil, dan kami merekomendasikan untuk membaca paper oleh Dror et al. [14, 15]. Kami harap pembaca dapat menangkap bahwa memutuskan apakah suatu model A lebih baik dari B tidak cukup hanya dengan melihat ukuran kinerja secara numerik, tetapi hal ini harus dibuktikan menggunakan *statistical hypothesis testing*.

2.9 Teori Informasi

Kami tidak akan membahas bagian ini terlalu detail, jika kamu membaca buku, topik ini sendiri bisa mencapai satu buku [16]. Mudah-mudahan bab ini dapat memberikan gambaran (serius, ini sekedar gambaran!). *Information Theory*/Teori Informasi menjawab dua pertanyaan fundamental, pertama: bagaimana cara kompresi data terbaik (jawab: *entropy*); kedua: apakah cara transmisi komunikasi terbaik (jawab: channel capacity) [16]. Dalam *statistical learning theory*, fokus utama adalah menjawab pertanyaan pertama, yaitu bagaimana melakukan kompresi informasi. Contoh aplikasi *entropy* adalah *decision tree learning*.

Pada *machine learning*, kita ingin fitur pembelajaran yang digunakan mampu melambangkan *information source properties*. Artinya, kita ingin memilih fitur yang memuat informasi terbanyak (relatif terhadap *information source*). Karena hal tersebut, mengerti entropy menjadi penting. Ada sebuah strategi pemilihan fitur (*feature selection*) dengan membangun *decision tree*. Awalnya kita bentuk *training data* dengan semua kemungkinan fitur, kemudian mengambil beberapa fitur yang dekat dengan *root*. Hal tersebut dimaksudkan untuk mencari fitur yang memuat banyak informasi. Kemudian, fitur tersebut dapat dicoba pada algoritma learning lainnya. Detil akan dijelaskan pada bab 6.

2.9.1 Entropy

Diberikan sebuah random variabel x, kita ingin mengetahui seberapa banyak informasi yang kita dapatkan ketika kita mengobservasi sebuah nilai spesifik x_i. Kuantitas informasi yang kita dapatkan bisa dipandang sebagai "*degree of surprise*" [8]. Misalkan kita mengetahui seorang teman A sering makan es krim. Suatu ketika kita diberitahu bahwa dia sedang makan es krim, tentu kita tidak heran lagi karena hal tersebut sudah lumrah. Tetapi, apabila kita diberitahu bahwa teman A tidak memakan es krim yang diberikan teman B (padahal kita tahu dia suka), maka akan ada efek "kaget". Kasus kedua memuat lebih banyak informasi karena suatu kejadian yang seharusnya tidak mungkin, terjadi. Hal ini dikuantifikasi dengan persamaan ***Shannon Entropy*** 2.14.

$$S(x) = -\sum_{i=1}^{N} p(x_i)\log(p(x_i)) \tag{2.14}$$

Mari kita ambil contoh dari Bishop [8]. Misalkan sebuah *random variable* x memiliki 8 kemungkinan kejadian yang kemungkinannya sama (yaitu $\frac{1}{8}$). *Entropy* untuk kasus ini adalah (log dalam basis 2) diberikan oleh

$$S = -8\frac{1}{8}\log(\frac{1}{8}) = 3 \tag{2.15}$$

Sekarang mari kita ambil contoh dari [16]. Misalkan sebuah *random variable* x memiliki 8 kemungkinan kejadian $\{a, b, c, d, ..., h\}$ dengan peluang $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}$
Maka *entropy*-nya adalah 2. Dari contoh ini, kita tahu bahwa distribusi yang uniform memiliki *entropy* yang lebih besar dibanding distribusi yang tidak uniform. Banyaknya informasi sebanding dengan **turunnya** nilai *entropy*.

Seperti yang telah diceritakan sebelumnya, *event* yang memiliki "efek kaget" memiliki banyak informasi. Dari sisi *information transmission*, dapat diinterpretasikan kita dapat mengirimkan data sebuah distribusi dengan jumlah bit lebih sedikit untuk distribusi yang uniform. Distribusi yang memberikan nilai *entropy* maksimal adalah distribusi Gaussian [8]. Nilai *entropy* bertambah seiring *variance* distribusi bertambah. Dari sisi fisika, kamu dapat mempelajari *entropy* pada *statistical mechanics* (*microstate*, *macrostate*).

Perhatikan, *entropy* (persamaan 2.14) dan *cross entropy* (persamaan 2.13) memiliki persamaan agak berbeda (adanya distribusi asli q). Tetapi interpretasi kedua formula adalah sama. Distribusi yang memiliki nilai cukup uniform memiliki nilai *entropy*/*cross entropy* yang tinggi, sementara memiliki nilai rendah untuk distribusi yang condong ke nilai tertentu (*skewed*).

2.9.2 Relative Entropy dan Mutual Information

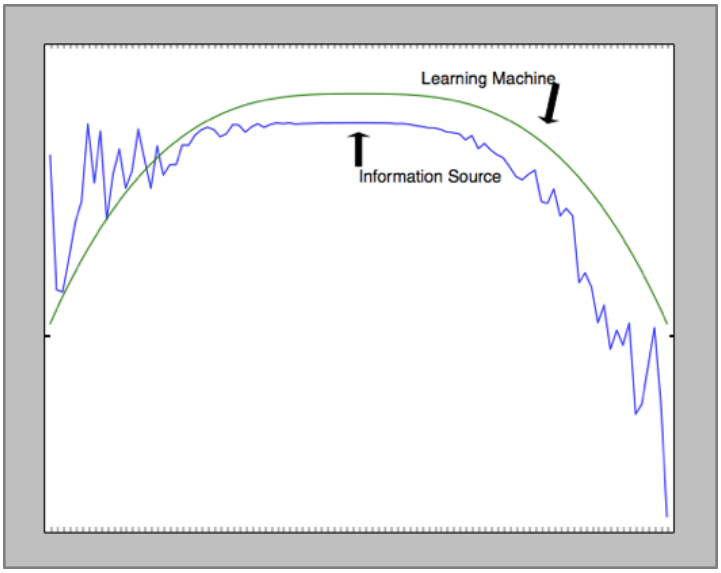
Kami harap kamu masih ingat materi bab 1, karena materi bagian ini juga menyinggung kembali materi tersebut. Misalkan kita mempunyai data dengan *probability density function* $q(x)$. Sebuah *learning machine* mengaproksimasi data tersebut dengan *probability density function* $p(x)$. Ingat! *Machine learning* adalah pendekatan (*approximation*). Ketika kita melakukan aproksimasi, seringkali aproksimasi yang dilakukan tidaklah tepat seperti pada Gambar 2.6.

Tentunya kita ingin tahu seberapa bagus aproksimasi kita, untuk mengukurnya terdapat sebuah perhitungan yang bernama **Kullback-Leibler Divergence** (KL-divergence). Secara konseptual, dirumuskan sebagai persamaan 2.16. Perlu diperhatikan $KL(q||p) \neq KL(p||q)$ (kecuali $p = q$).

$$KL(q||p) = -\int q(x)\log\left(\frac{q(x)}{p(x)}\right)dx \tag{2.16}$$

Persamaan 2.16 dapat diminimalkan jika dan hanya jika $q(x) = p(x)$. Kita dapat menganggap KL-divergence sebagai ukuran seberapa jauh aproksimasi dan distribusi populasi. Akan tetapi, kita tidak mengetahui $q(x)$. Karena itu, kita harus mengaproksimasi KL-divergence. Misalkan kita diberikan *training data* $\mathbf{x} = \{x_1, x_2, ...x_n\}$ yang kita asumsikan diambil (*drawn*) dari suatu distribusi $q(x)$. Lalu kita membuat *learning machine* $p(x \mid \mathbf{w})$. Ekspektasi terhadap $q(x)$ dapat diaproksimasi dengan menggunakan data sampel ini, dituangkan menjadi persamaan 2.17 [8].

$$KL(q||p) \approx \frac{1}{N}\sum_{i=1}^{N}\left(-\log(p(x_i \mid \mathbf{w})) + \log(q(x_i))\right) \tag{2.17}$$



Gambar 2.6: Information source vs learning machine.

KL-divergence disebut juga sebagai *relative entropy*.[3] Dari sisi pemrosesan informasi, KL-divergence dapat diinterpretasikan sebagai berapa informasi tambahan rata-rata untuk mengirimkan data distribusi dengan menggunakan fungsi aproksimasi dibanding menggunakan distribusi sebenarnya, seberapa pengurangan ketidakyakinan terhadap *posterior* seiring diberikannya data observasi yang baru. Dengan kata lain, **seiring diberikan observasi yang baru, kita semakin yakin terhadap nilai posterior** (semakin banyak jumlah sampel yang kita miliki maka model lebih dapat dipercaya). KL-Divergence = Cross Entropy - Entropy (buktikan!).[4]

2.10 Matriks

Subbab ini adalah pengingat untuk operasi perjumlahan, pengurangan, perkalian, dan transpose matriks karena banyak digunakan di buku ini. Diberikan dua buah matriks **U** dan **V**. **U** dan **V** dapat dijumlahkan jika dan hanya jika dimensi kedua matriks itu sama. Perjumlahan matriks dinotasikan dengan

[3] kamu dapat mencoba library entropy di scipy (python) untuk mendapat gambaran lebih detil
`https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.entropy.html`

[4] https://towardsdatascience.com/entropy-cross-entropy-and-kl-divergence-explained-b09cdae917a

$\mathbf{U} + \mathbf{V} = \mathbf{C}$. Matriks $\mathbf{C}$ memiliki dimensi yang sama dengan $\mathbf{U}$ dan $\mathbf{V}$. Nilai elemen baris ke-i dan kolom ke-j ($\mathbf{C}_{i,j}$) dihitung sebagai penjumlahan nilai elemen matriks $\mathbf{U}$ dan $\mathbf{V}$ pada baris dan kolom yang bersesuaian, seperti diilustrasikan pada persamaan 2.18. Pengurangan dua buah matriks dilakukan serupa.

$$\mathbf{C}_{i,j} = \mathbf{U}_{i,j} + \mathbf{V}_{i,j} \tag{2.18}$$

Dua buah matriks $\mathbf{U}$ dan $\mathbf{V}$ dapat dikalikan jika $\mathbf{U}$ memiliki kolom sebanyak baris pada $\mathbf{V}$. Misalkan matriks $\mathbf{U}$ berdimensi $N \times M$ dan $\mathbf{V}$ berdimensi $M \times O$, maka kedua matriks tersebut dapat dikalikan dan menghasilkan matriks $\mathbf{C}$ dengan dimensi $N \times O$ (dimensi baris $\mathbf{U}$ dan kolom $\mathbf{V}$), dimana tiap elemen pada matriks $\mathbf{C}$ dihitung dengan persamaan 2.19 (operasi antara vektor baris dan vektor kolom).

$$\mathbf{C}_{x,y} = \sum_{i=1}^{M} \mathbf{U}_{x,i} + \mathbf{V}_{i,y} \tag{2.19}$$

Selain perkalian antar dua buah matriks, sebuah matriks juga dapat dikalikan dengan skalar, dinotasikan dengan $a\mathbf{U}$. Hasil perkalian adalah sebuah matriks dengan dimensi yang sama dengan $\mathbf{U}$, dimana tiap elemen dikalikan dengan nilai skalar.

$$(a\mathbf{U})_{i,j} = a \times \mathbf{U}_{i,j} \tag{2.20}$$

Suatu matriks $\mathbf{U}$ berdimensi $N \times M$ apabila di transpose menghasilkan matriks $\mathbf{U}^T$ berdimensi $M \times N$, dimana elemen ke-i, j pada matriks $\mathbf{U}^T$ adalah elemen ke-j, i pada matriks $\mathbf{U}$, seperti diilustraiskan pada persamaan 2.19.

$$\mathbf{U}^T_{i,j} = \mathbf{U}_{j,i} \tag{2.21}$$

Ada satu istilah lagi yang perlu kamu ketahui yaitu **tensor**. Tensor adalah generalisasi untuk vektor (1 dimensi) dan matriks (2 dimensi) yang memiliki N dimensi. Tensor sering digunakan untuk notasi pada *artificial neural network*. Tetapi demi kemudahan pengertian, penulis menggunakan notasi matriks.

2.11 Bacaan Lanjutan

Untuk lebih mengerti, silahkan membaca buku *statistical mechanis* oleh Hitoshi Nishimori [17], buku probabilitas dan statistika oleh Walpole et al. [18] atau Brian Caffo [10], buku aljabar linear oleh Gilbert Strang [19] dan buku *statistical learning theory* oleh James et al. [20].

Soal Latihan

2.1. KL-divergence
Cari tahu lebih lanjut apa itu Kullback-Leibler (KL) Divergence. Apa hubungan KL-divergence dengan *utility function*? Pada kasus apa saja kita dapat menggunakan KL-divergence sebagai *utility function*?

2.2. Utility Function
Selain *utility function* yang telah disebutkan, sebutkan dan jelaskan *utility function* lainnya!

2.3. Gaussian Mixture Model

(a) Sebutkan algoritma-algoritma *machine learning* yang (*in a sense*) dapat mengaproksimasi *Gaussian Mixture Model*!
(b) Apa yang begitu spesial pada GMM sehingga algoritma *machine learing* mencoba mengaproksimasi GMM?

3

Data Analytics

> "Hiding within those mounds of data is knowledge that could change the life of a patient, or change the world"
>
> Atul Butte

Bab ini memuat penjelasan tahapan-tahapan umum untuk analisis data, serta beberapa karakteristik tipe data. Materi pada bab ini dapat dianggap sebagai kerangka berpikir (*framework*) atau langkah kerja. Karena buku ini hanya bersifat pengantar, materi yang disampaikan mungkin kurang lengkap. Penulis menyarankan pembaca untuk membaca buku oleh Witten et al. [21] dan Jeff Leek [22].

3.1 Pengenalan Data Analytics

Secara umum, subbab ini adalah ringkasan dari buku Jeff Leek [22]. Untuk detailnya, kamu dapat membaca buku tersebut secara langsung. Penulis merekomendasikan buku tersebut karena ringkas dan mudah dipahami bahkan oleh pemula.

Kita tahu di dunia ini ada banyak masalah. Masalah adalah ketika tujuan yang diinginkan tidak tercapai (*current state* bukanlah *desired state*). Agar *current state* menjadi *desired state*, kita melakukan kegiatan yang disebut penyelesaian masalah (*problem solving*). Tiap bidang (domain) mendefinisikan permasalahan secara berbeda. Oleh karena itu, mengetahui teknik *machine learning* tanpa mengetahui domain aplikasi adalah sesuatu yang kurang baik (semacam buta). Kamu memiliki ilmu, tetapi tidak tahu ilmunya mau digunakan untuk apa. Contohnya, bidang keilmuan pemrosesan bahasa alami (*natural language processing*) menggunakan *machine learning* untuk mengklasifikasikan teks; bidang keilmuan pemrosesan suara menggunakan *machine*

learning untuk mentranskripsikan suara manusia menjadi teks. Tiap bidang merepresentasikan permasalahan ke dalam formulasi yang berbeda. Bisa jadi bentuk komputasi (representasi) pada bidang satu berbeda dengan bidang lainnya. Hal ini perlu kamu ingat karena interpretasi representasi sangat bergantung pada konteks permasalahan (domain). Buku ini adalah pengenalan teknik yang bersifat umum.

Seperti yang sudah dijelaskan pada bab-bab sebelumnya. *Machine learning* adalah inferensi berdasarkan data. *Raw data* atau data mentah adalah sekumpulan fakta (*record*, *event*) yang kemungkinan besar tidak memberikan penjelasan apapun. Sama halnya dengan kebanyakan data di dunia nyata, *raw data* bersifat tidak rapih, misalnya mengandung *missing value*, i.e., ada data yang tidak memiliki label padahal data lainnya memiliki label (ingat kembali materi bab 1). Agar mampu menganalisis *raw data* menggunakan teknik *machine learning*, pertama-tama kita harus merapikan data sesuai dengan format yang kita inginkan (*dataset*). Kemudian, mencari fitur-fitur yang dapat merepresentasikan data. Kedua kegiatan ini secara umum dikenal dengan istilah *pre-processing*. Setelah *pre-processing*, kita menggunakan teknik-teknik yang ada untuk menemukan pola-pola yang ada di data (membangun model). Pada beberapa bidang, *pre-processing* adalah tahapan yang memakan waktu paling banyak pada eksperimen *machine learning*, sedangkan proses melatih model mungkin memakan waktu jauh lebih singkat.

Dalam komunitas peneliti basis data, dipercaya bahwa data memiliki sangat banyak relasi yang mungkin tidak bisa dihitung. Teknik *machine learning* hanya mampu mengeksplorasi sebagian relasi yang banyak itu. Lalu, kita analisis informasi yang kita dapatkan menjadi pengetahuan yang digunakan untuk memecahkan permasalahan atau membuat keputusan.

Setelah kita menganalisis data dan mendapatkan pengetahuan baru, pengetahuan yang kita temukan dari data pada umumnya dipresentasikan (konferensi, rapat, dsb). Hal umum yang dipaparkan saat presentasi, yaitu:

1. *Performance measure*. Seberapa "bagus" model atau metode yang kamu usulkan, dibandingkan menggunakan model yang sudah ada. *Performance measure* biasa disajikan dalam bentuk tabel. Perhatikan, mengatakan model/metode kamu "lebih bagus" adalah suatu hal subjektif, untuk itu gunakanlah metode kuantitatif, seperti *p-value* dalam statistik (*hypothesis testing*)[1] untuk mengatakan bahwa memang metode kamu lebih baik (berbeda) dari *baseline*.

2. Tren. Bagaimana pola-pola umum yang ada di data, sesuai dengan tujuan analisis (permasalahan). Biasa disajikan dalam bentuk teks, kurva, atau grafik.

3. *Outlier*. Sajikan data-data yang "jarang" atau tidak sesuai dengan tren yang ada. Apa beda sifat data *outlier* ini dibanding data pada tren? Kamu

[1] https://onlinecourses.science.psu.edu/statprogram/node/138

harus menganalisis hal ini untuk meningkatkan *performance measure* pada penelitian atau analisis mendatang. Apakah *outlier* ini penting untuk diurus atau bisa dipandang sebelah mata tanpa membahayakan keputusan/sistem? Tidak jarang kamu mendapat inspirasi untuk meningkatkan kinerja sistem setelah menganalisis *outlier*.

Langkah kerja yang dijelaskan ini adalah pekerjaan rutin *data scientist*. Penulis ingin menekankan sekali lagi, bahwa memahami *machine learning* saja tidak cukup, **kamu harus memahami domain permasalahan** agar mampu melakukan analisis dengan tepat. Terdapat banyak hal yang hanya mampu kamu pahami dari menganalisis data, apabila kamu mengerti domain aplikasi.

3.2 Nilai Atribut dan Transformasi

Perhatikan Tabel 3.1 yang merupakan contoh *dataset* pada *machine learning*. **Dataset** adalah kumpulan sampel. Seorang anak ingin bermain tenis, tetapi keputusannya untuk bermain tenis (*play*) tergantung pada empat variabel {*outlook, temperature, humidity, windy*}. Keempat variabel ini disebut **fitur**. Setiap fitur memiliki **atribut** nilai dengan **tipe data** dan **range** tertentu. Keputusan untuk bermain (*play*) disebut sebagai label atau kelas (*class*). Pada bab 1 kamu telah diperkenalkan *supervised learning* dan *unsupervised learning*. Pada *supervised learning*, kita ingin mengklasifikasikan apakah seorang anak akan bermain atau tidak, diberikan fitur-fitur yang memuat kondisi observasi. Pada *unsupervised learning*, informasi kolom *play* tidak diberikan, kita harus mengelompokkan data tersebut sesuai dengan fitur-fiturnya (contoh lebih nyata diberikan pada bab 4).

Dari segi data statistik, terdapat beberapa tipe atribut [23]:

1. **Nominal**. Nilai atribut bertipe nominal tersusun atas simbol-simbol yang berbeda, yaitu suatu himpunan terbatas. Sebagai contoh, fitur *outlook* pada Tabel 3.1 memiliki tipe data **nominal** yaitu nilainya tersusun oleh himpunan {*sunny, overcast, rainy*}. Pada tipe nominal, tidak ada urutan ataupun jarak antar atribut. Tipe ini sering juga disebut **kategorial** atau **enumerasi**. Secara umum, tipe *output* pada *supervised learning* adalah data nominal.

2. **Ordinal**. Nilai ordinal memiliki urutan, sebagai contoh $4 > 2 > 1$. Tetapi jarak antar suatu tipe dan nilai lainnya tidak harus selalu sama, seperti $4-2 \neq 2-1$. Atribut ordinal kadang disebut sebagai **numerik** atau **kontinu**.

3. **Interval**. Tipe interval memiliki urutan dan *range* nilai yang sama. Sebagai contoh $1-5, 6-10, dst$. Kita dapat mentransformasikan/ mengkonversi nilai numerik menjadi nominal dengan cara merubahnya menjadi

id	outlook	temperature	humidity	windy	play (class)
1	sunny	hot	high	false	no
2	sunny	hot	high	true	no
3	overcast	hot	high	false	yes
4	rainy	mild	high	false	yes
5	rainy	cool	normal	false	yes
6	rainy	cool	normal	true	no
7	overcast	cool	normal	true	yes
8	sunny	mild	high	false	no
9	sunny	cool	normal	false	yes
10	rainy	mild	normal	false	yes
11	sunny	mild	normal	true	yes
12	overcast	mild	high	true	yes
13	overcast	hot	normal	false	yes
14	rainy	mild	high	true	no

Tabel 3.1: Contoh dataset *play tennis* (UCI machine learning repository).

interval terlebih dahulu. Lalu, kita dapat memberikan nama (simbol) untuk masing-masing interval. Misalkan nilai numerik dengan *range* $1-100$ dibagi menjadi 5 kategori dengan masing-masing interval adalah $\{1-20, 21-40, \ldots, 81-100\}$. Setiap interval kita beri nama, misal interval $81-100$ diberi nama *nilai A*, interval $61-80$ diberi nama *nilai B*.

4. **Ratio**. Tipe *ratio* (rasio) didefinisikan sebagai perbandingan antara suatu nilai dengan nilai lainnya, misalkan massa jenis (fisika). Pada tipe *ratio* terdapat *absolute zero* (semacam *ground truth*) yang menjadi acuan, dan *absolute zero* ini memiliki makna tertentu.

Secara umum, data pada *machine learning* adalah nominal atau numerik (ordinal). Variabel yang kita prediksi yaitu *play* disebut **kelas/*class*/label**. Untuk setiap baris pada Tabel 3.1, baris kumpulan nilai variabel non-kelas disebut **vektor fitur/*feature vector***. Contohnya pada Tabel 3.1, *feature vector*-nya untuk data dengan $id = 4$ adalah $x_4 = \{$*outlook=rainy, temperature=mild, humidity=high, windy=false*$\}$. *Feature vector* adalah representasi dari suatu observasi/data. Pada *machine learning*, kita melakukan operasi terhadap data pada representasi *feature vector*-nya. Kami serahkan pada pembaca untuk mencari contoh dataset dengan tipe numerik sebagai pekerjaan rumah.[2]

[2] https://www.cs.waikato.ac.nz/ml/weka/datasets.html

3.3 Ruang Konsep

Dengan data yang diberikan, kita ingin melakukan generalisasi aturan/ konsep yang sesuai dengan data. Hal ini disebut sebagai *inductive learning*. Cara paling sederhana untuk *inductive learning* adalah mengenumerasi seluruh kemungkinan kombinasi nilai sebagai *rule*, kemudian mengeleminasi *rule* yang tidak cocok dengan contoh. Metode ini disebut *list-then-eleminate*. Silahkan baca buku Tom Mitchell [4] untuk penjelasakn lebih rinci. Kemungkinan kombinasi nilai ini disebut sebagai ruang konsep (***concept space***). Sebagai contoh pada Tabel 3.1 himpunan nilai masing-masing atribut yaitu:

- $outlook = \{sunny, overcast, rainy\}$
- $temperature = \{hot, mild, cold\}$
- $humidity = \{high, normal\}$
- $windy = \{true, false\}$
- $play = \{yes, no\}$

sehingga terdapat $3 \times 3 \times 2 \times 2 \times 2 = 72$ kemungkinan kombinasi. Tentunya kita tidak mungkin mengenumerasi seluruh kemungkinan kombinasi nilai karena secara praktikal, atribut yang digunakan banyak. Terlebih lagi, apabila mengenumerasi kombinasi atribut bertipe numerik.

Ada algoritma lain yang mendaftar "seluruh kemungkinan kombinasi" bernama *candidate-elemination algorithm* yang lebih efisien dibanding *list-then-eliminate*. Akan tetapi, algoritma ini *computationally expensive* secara praktikal, dalam artian memiliki kompleksitas yang besar dan tidak bisa menyelesaikan permasalahan nyata. Kamu dapat membaca algoritma ini pada buku Tom Mitchell [4] juga. Selain *inductive learning*, kita juga dapat melakukan *deductive learning* yaitu melakukan inferensi dari hal general menjadi lebih spesifik. Walau demikian, secara praktis kita sebenarnya melakukan *inductive learning*.

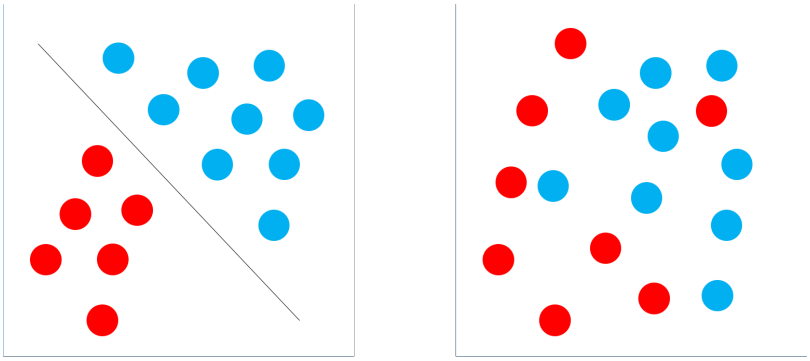
3.4 Linear Separability

id	humidity	windy	swim (class)
1	high	high	yes
2	normal	normal	no

Tabel 3.2: Contoh dataset *linearly separable*.

Perhatikan Tabel 3.2. Data pada tabel tersebut kita sebut ***linearly separable***. Sederhananya, untuk suatu nilai tertentu, fitur hanya berkorespondensi dengan kelas tertentu. Ambil contoh pada Tabel 3.2, saat *humidity=high* maka

swim=yes. Secara geometris, bila kita proyeksikan *feature vector* ke suatu ruang dimensi, memisahkan kelas satu dan kelas lainnya dapat diperoleh dengan cara menciptakan garis linier (*linear line*)–secara lebih umum, menggunakan *hyperplane*.[3] Ilustrasi dapat dilihat pada Gambar 3.1. Sementara pada Tabel 3.1, bila kita hanya melihat fitur *humidity* saja, ketika *humidity=high* bisa jadi *play=yes* atau *play=no*. Kasus ini disebut ***non-linearly separable***. Hidup kita tentu akan mudah apabila seluruh data bersifat *linearly separable*, sayangnya kebanyakan data yang ada bersifat *non-linearly separable*.



Gambar 3.1: *Linearly* vs *Non-Linearly Separable*.

Untuk memudahkan proses pada data *non-linearly separable*, kita pertamatama mentransformasikan data menjadi *linearly-separable*. Kita dapat menggunakan teknik transformasi data menggunakan *kernel function* seperti *radial basis function*.[4] Pada umumnya, *kernel function* mentransformasi data menjadi lebih tinggi (semacam menambah fitur). Misal dari data yang memiliki dua fitur, ditransformasi menjadi memiliki tiga fitur. Akan tetapi, hal ini tidak praktikal untuk banyak kasus (dijelaskan pada bab 11). Cara lainnya adalah memisahkan data menggunakan model non-linear, contoh: *artificial neural network*. Hal ini penting dipahami karena data yang bersifat *linearly separable* mudah dipisahkan satu sama lain sehingga mudah untuk melakukan *classification* atau *clustering*.

3.5 Seleksi Fitur

Pada subbab sebelumnya, telah dijelaskan bahwa kita dapat mentransformasi data *non-linearly separable* menjadi *linearly separable* dengan cara menam-

[3] https://en.wikipedia.org/wiki/Hyperplane
[4] Silakan baca teknik transformasi lebih lanjut pada literatur lain.

bah dimensi data. Pada bab ini, penulis akan menjelaskan justru kebalikannya! Pada permasalahan praktis, kita seringkali menggunakan banyak fitur (*computationally expensive*). Kita ingin menyederhanakan fitur-fitur yang digunakan, misalkan dengan memilih subset fitur awal, atas dasar beberapa alasan [22, 4]:[5]

1. Menyederhanakan data atau model agar lebih mudah dianalisis.
2. Mengurangi waktu *training* (mengurangi kompleksitas model dan inferensi).
3. Menghindari *curse of dimensionality*. Hal ini dijelaskan lebih lanjut pada bab 12.
4. Menghapus fitur yang tidak informatif.
5. Meningkatkan generalisasi dengan mengurangi *overfitting*.

Salah satu contoh cara seleksi fitur adalah menghapus atribut yang memiliki *variance* bernilai 0. Berdasarkan *information theory* atau *entropy*, fitur ini tidak memiliki nilai informasi yang tinggi. Dengan kata lain, atribut yang tidak dapat membedakan satu kelas dan lain bersifat tidak informatif. Kamu dapat membaca beberapa contoh algoritma seleksi fitur pada library sklearn.[6]

3.6 Classification, Association, Clustering

Pada *supervsied learning*, kita memprediksi kelas berdasarkan *feature vector* yang merepresentasikan suatu sampel (data/observasi). *Feature vector* bisa diibaratkan sebagai sifat-sifat atau keadaan yang diasosiasikan dengan kelas. Pada *supervised learning*, setiap *feature vector* berkorespondisi dengan kelas tertentu. Mencari kelas yang berkorespondensi terhadap suatu *input* disebut **klasifikasi** (***classification***). Contoh klasifikasi adalah mengkategorikan gambar buah (e.g. apel, jeruk, dsb). Sementara itu, apabila kita ingin mencari hubungan antara satu atribut dan atribut lainnya, disebut **association**. Sebagai contoh pada Tabel 3.1, apabila $outlook = sunny$, maka sebagian besar $humidity = high$. Di lain pihak, pada *unsupervised learning* tidak ada kelas yang berkorespondensi; kita mengelompokkan data dengan sifat-sifat yang mirip, disebut ***clustering***. Contoh *clustering* adalah pengelompokkan barang di supermarket. Perlu kamu catat bahwa *unsupervised learning* $\neq$ *clustering*. *Clustering* adalah salah satu *task* pada *unsupervised learning*.

Pada Tabel 3.1, hanya ada dua kelas, klasifikasi data ini disebut ***binary classification***. Apabila kelas klasifikasi lebih dari dua (*mutually exclusive*), disebut ***multi-class classification***. Apabila kelas-kelas tersebut tidak bersifat *mutually exclusive*, maka kita melakukan ***multi-label classification***. Mohon bedakan antara ***multi-label classification*** dan ***multi-level/hierarchical classification***. Pada *multi-level/hierarchical classification*, pertama-tama kita melakukan klasifikasi untuk suatu kelas generik, lalu

[5] https://en.wikipedia.org/wiki/Feature_selection
[6] http://scikit-learn.org/stable/modules/feature_selection.html

dilanjutkan mengklasifikan data ke kelas yang lebih spesifik. Contoh *multi-level classification* adalah *kingdom* (biologi), pertama diklasifikasikan ke *kingdom animalia*, lalu lebih spesifiknya ke *phylum Vertebrata*, dst. *Multi-label classification* hanya proses klasifikasi ke dalam banyak "kelas" tanpa tinjauan hirarkis.

Multi-class classification yang telah dijelaskan sebelumnya disebut juga sebagai *hard classification*, artinya apabila data diklasifikasikan ke kelas tertentu, maka tidak mungkin data berada di kelas lainnya (ya atau tidak). *Multi-label classification* bersifat lebih *soft*, karena dapat mengklasifikasikan ke beberapa kelas, misal data X memiliki masuk ke kategori kelas A, B dan C sekaligus (dengan nilai probabilitas masing-masing).

3.7 Mengukur Kinerja

Pada bab 1, sudah dijelaskan bahwa kita harus mengukur kinerja model dengan cara yang kuantitatif. Pada saat proses latihan, kita ingin model mengoptimalkan suatu nilai *utility function*, misal meminimalkan nilai *error* atau *entropy*. Pada saat latihan, model pembelajaran mesin dapat mengoptimalkan *utility function* yang berbeda-beda (tergantung algoritma).

Kita juga dapat mengevaluasi model secara eksternal dengan melewatkan data pada model (umumnya *validation* dan *testing data*), kemudian mengevaluasi prediksi final model tersebut dengan ukuran *performance measure*. *Performance measure* dapat mengukur kinerja model yang dikonstruksi oleh algoritma yang berbeda (akan lebih jelas sembari kamu membaca buku ini).

Sebagai contoh, kamu dapat membandingkan kinerja prediksi model dengan menggunakan metrik seperti akurasi, presisi, *recall*, F1-measure, BLEU [24], ROUGE [25], *intra-cluster similarity*, dsb. Masing-masing *performance measure* mengukur hal yang berbeda. Perlu kamu ketahui bahwa memilih ukuran kinerja tergantung pada domain permasalahan. Misalkan pada translasi otomatis, peneliti menggunakan ukuran BLEU; pada peringkasan dokumen, menggunakan ROUGE. Sementara itu, pada *information retrival*/sistem temu balik informasi menggunaan presisi, *recall*, F1-measure, atau *mean average precision* (MAP). Pada domain klasifikasi gambar, menggunakan akurasi. Masing-masing *performance measure* dapat memiliki karakteristik nilai optimal yang berbeda. Kamu harus mengerti domain permasalahan untuk mengerti cara mencapai titik optimal. Sebagai pengantar, diktat ini tidak dapat membahas seluruh domain. Dengan demikian, kamu harus membaca lebih lanjut literatur spesifik domain, misal buku pemrosesan bahasa alami atau sistem temu balik informasi, dsb. Sebagai contoh, untuk permasalahan klasifikasi, akurasi sering digunakan. Akurasi didefinisikan pada persamaan 3.1. Buku ini akan membahas *performance measure* dengan lebih lengkap pada bab 9.

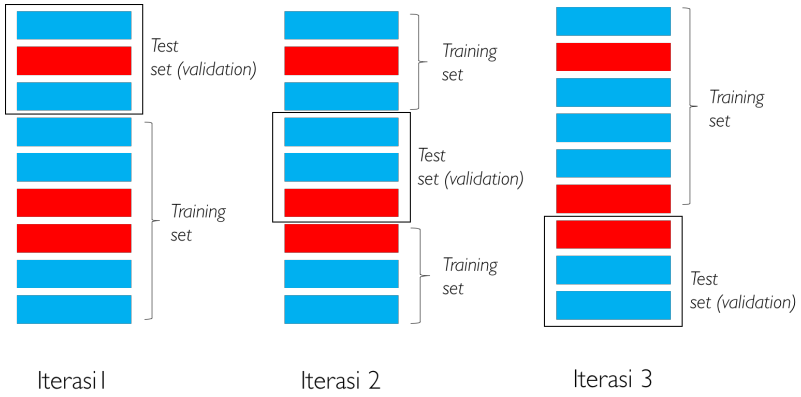
$$\text{akurasi} = \frac{\#input \text{ diklasifikasikan dengan benar}}{\text{banyaknya data}} \tag{3.1}$$

3.8 Evaluasi Model

Ada beberapa hal yang perlu kamu catat tentang proses evaluasi suatu model pembelajaran mesin:

1. ***Data splitting***. Seperti yang sudah dijelaskan pada subbab 1.5, pada umumnya kita memiliki *training*, *validation*/*validation*, dan *testing data*. Mesin dilatih menggunakan *training data*, saat proses *training*, *performance measure* diukur berdasarkan kemampuan mengenali/ mengeneralisasi *validation data*. Perlu diketahui, *performance measure* diukur menggunakan *validation data* untuk menghindari *overfitting* dan *underfitting*. Setelah selesai dilatih, maka model hasil pembelajaran dievaluasi dengan *testing data*. *Training*, *validation*, dan *testing* data tersusun oleh data yang independen satu sama lain (tidak beririsan) untuk memastikan model yang dihasilkan memiliki generalisasi cukup baik.
2. ***Overfitting dan Underfitting***. *Overfitting* adalah keadaan ketika model memiliki kinerja baik hanya untuk *training data*/*seen examples* tetapi tidak memiliki kinerja baik untuk *unseen examples*. *Underftting* adalah keadaan ketika model memiliki kinerja buruk baik untuk *training data* dan *unseen examples*. Hal ini akan dibahas lebih detil pada subbab 5.8.
3. ***Cross validation***. *Cross validation* adalah teknik untuk menganalisis apakah suatu model memiliki generalisasi yang baik (mampu memiliki kinerja yang baik pada *unseen examples*). Seperti yang kamu ketahui, kita dapat membagi data menjadi *training*, *validation*, dan *testing data*. Saat proses *training*, kita latih model dengan *training data* serta dievaluasi menggunakan *validation data*. Teknik *cross validation* bekerja dengan prinsip yang sama, yaitu membagi sampel asli menjadi beberapa subsampel dengan partisi sebanyak K (K-*fold*). Ilustrasi diberikan oleh Gambar 3.2. Persegi panjang melambangkan suatu sampel. Saat proses *training*, kita bagi data menjadi *training data* dan *test data* (i.e., *validation data*). Hal ini diulang sebanyak K kali. Kita evaluasi kemampuan generalisasi model dengan merata-ratakan kinerja pada tiap iterasi. Setelah itu, model dengan kinerja terbaik (pada iterasi teretentu) digunakan lebih lanjut untuk proses *testing* atau dipakai secara praktis. Perlu diperhatikan, setiap subsampel sebaiknya memiliki distribusi yang sama dengan sampel aslinya (keseluruhan sampel); i.e., pada contoh, proporsi warna biru dan merah adalah sama tiap partisi tiap iterasi. Konsep tersebut lebih dikenal dengan ***stratified sampling***.[7]

[7] https://en.wikipedia.org/wiki/Stratified_sampling



Gambar 3.2: Ilustrasi 3-*fold cross validation*. Warna merah dan biru melambangkan sampel (*instance*) yang tergolong ke dua kelas berbeda.

3.9 Kategori Jenis Algoritma

Algoritma pembelajaran mesin dapat dibagi menjadi beberapa kategori. Dari sudut pandang apakah algoritma memiliki parameter yang harus dioptimasi, dapat dibagi menjadi:[8]

1. Parametrik. Pada kelompok ini, kita mereduksi permasalahan sebagai optimisasi parameter. Kita mengasumsikan permasalahan dapat dilambangkan oleh fungsi dengan bentuk tertentu (e.g., linier, polinomial, dsb). Contoh kelompok ini adalah model linier.
2. Non parametrik. Pada kelompok ini, kita tidak mengasumsikan permasalahan dapat dilambangkan oleh fungsi dengan bentuk tertentu. Contoh kelompok ini adalah *Naive Bayes*, *decision tree* (ID3) dan *K-Nearest Neighbors*.

Dari sudut pandang lainnya, jenis algoritma dapat dibagi menjadi:

1. Model linear, contoh regresi linear, regresi logistik, *support vector machine*.
2. Model probabilistik, contoh *Naive Bayes*, *hidden markov model*.
3. Model non-linear, yaitu (*typically*) *artificial neural network*.

Selain kedua skema pengkategorian ini, terdapat skema pengkategorian lain (silahkan eksplorasi sendiri).

[8] *Artificial Neural Network* dapat dikategorikan ke dalam keduanya.

3.10 Tahapan Analisis

Bagian ini adalah ringkasan bab ini. Untuk menganalisis data, terdapat langkah yang perlu kamu perhatikan

1. Memutuskan tujuan analisis data (*defining goal*)
2. Mendapatkan data
3. Merapihkan data
4. Merepresentasikan data sebagai *feature vector*
5. Melakukan transformasi dan/atau *feature selection* (mengurasi dimensi *feature vector*)
6. Melatih model (*training*) dan menganlisis kinerjanya pada *validation* (*development*) *data*.
7. Melakukan *testing* dan analisis model baik secara kuantitatif dan kualitatif
8. Menyajikan data (presentasi)

Saat ini, mungkin kamu merasa bab ini kurang berguna. Hal ini disebabkan karena konsep yang dipaparkan oleh bab ini bersifat *high-level*. Kamu mungkin harus membaca bab ini kembali setelah selesai membaca seluruh buku agar mendapatkan pemahaman lebih mendalam.

Soal Latihan

3.1. Konversi atribut
Sebutkan dan jelaskan macam-macam cara untuk mengkonversi atribut! Sebagai contoh, numerik-nominal dan nominal-numerik.

3.2. Transformasi data
Sebutkan dan jelaskan macam-macam cara transformasi data (e.g. merubah *non-linearly separable* menjadi *linearly separable*)

3.3. Seleksi fitur
Bacalah algoritma seleksi fitur pada *library* sklearn. Jelaskan alasan (*rationale*) dibalik penggunaan tiap algoritma yang ada!

3.4. Inductive Learning
Jelaskanlah algoritma *list-then-eliminate* dan *candidate-elimination*!

3.5. Tahapan analisis
Agar mampu memahami tahapan analisis data dan pembelajaran mesin secara lebih praktikal, kerjakanlah tutorial berikut!
https://scikit-learn.org/stable/tutorial/basic/tutorial.html