2

Fondasi Matematis

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

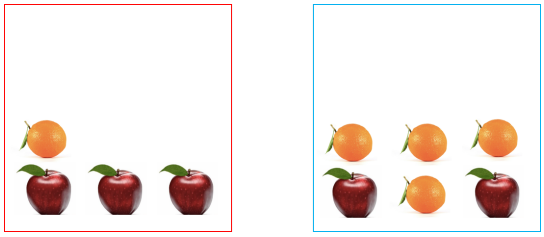
Mungkin saat pertama kali membaca bab ini, kamu merasa bab ini tidak masuk akal atau kurang dibutuhkan. Seiring membaca buku ini, mungkin bab ini akan sering dikunjungi kembali. Bab ini hanyalah pengingat materi yang sudah kamu pernah pelajari saja (semacam *cheatsheet*). Kamu boleh melewati bab ini 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 $P(x \mid y) = P(x)$.

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

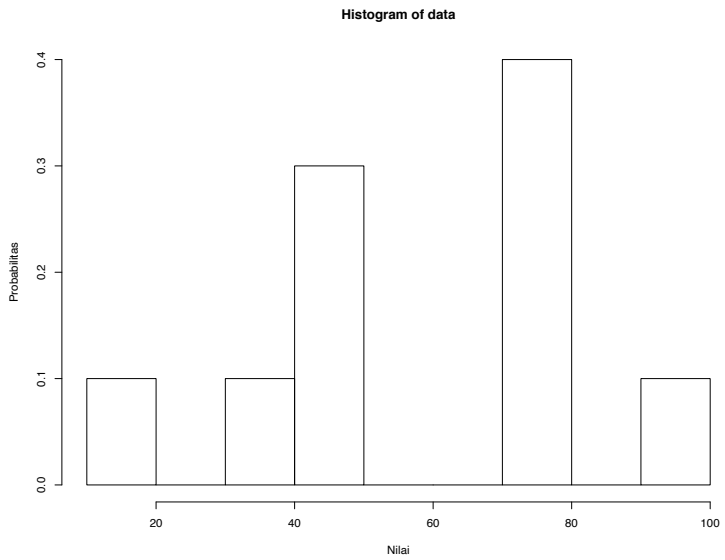
2.2 Probability Density Function

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.

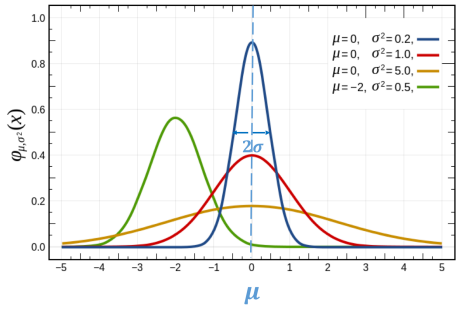
Disebut *univariate* karena distribusinya bergantung pada **satu** input variabel, misalkan x. Distribusi univariate Gaussian dikarakteristikkan oleh variabel x, *mean* (μ) dan *variance* (σ^2) diberikan pada persamaan 2.10. μ dan σ^2 adalah rata-rata dan *variance* untuk kumpulan data. Karena nilai 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.

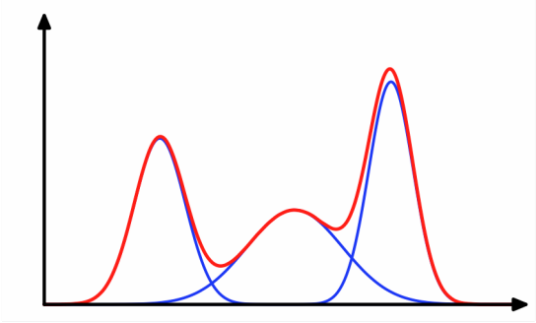
Multivariate Gaussian adalah distribusi gaussian yang bergantung pada lebih dari satu *random variable*. Sedangkan *Gaussian Mixture Model* (GMM) adalah gabungan dari satu atau lebih distribusi Gaussian. Masing-masing distribusi Gaussian memiliki bobot yang berbeda di GMM. Konon katanya,

[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.

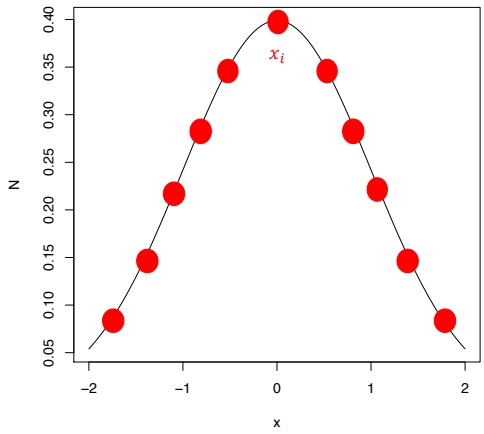


Gambar 2.4: Gaussian Mixture Model.[2]

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

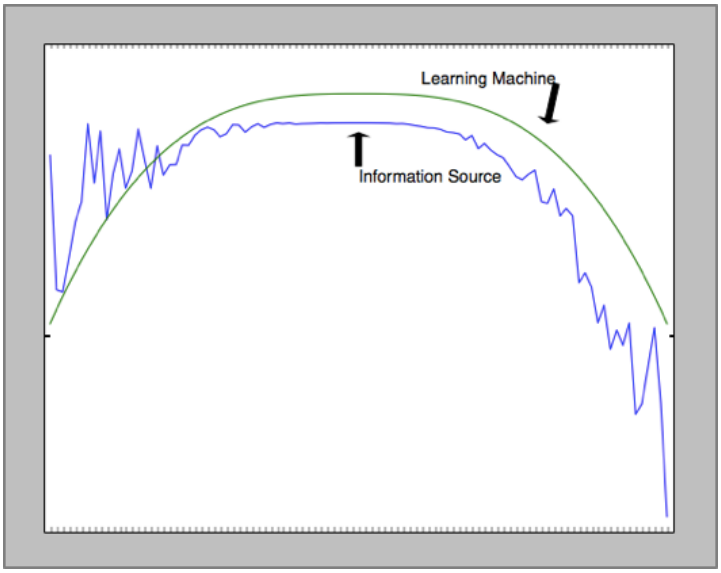
Kami harap kamu masih ingat materi bab 1, karena materi bagian ini juga menyinggung kembali materi tersebut. Misalkan kita mempunyai data dengan *probability density function* $q(x)$. Sebuah *learning machine* mengaproksimasi data tersebut dengan *probability density function* $p(x)$. Ingat! *Machine 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?