Pertemuan 9 — *Linear regression*, Evaluasi Hipotesis, dan *Gradient Descent*

Dari rumus bentuk tertutup ke mesin *optimization*

Hendri Karisma

Semester Ganjil 2026/2027

Program Studi Teknik Informatika, STMIK Tazkia

Tujuan dan alur pertemuan

Alur

1. *Linear regression*: *least squares*, persamaan normal, R^2
2. Evaluasi hipotesis: *confidence interval*, *metric*, *cross-validation*, *paired t-test*
3. Mengevaluasi model *regression*
4. *Loss function* dan *gradient descent*
5. *Learning rate*, SGD, *mini-batch*
6. *Logistic regression* singkat
7. Praktikum, Ringkasan, kuis

Setelah pertemuan ini Anda dapat

- menghitung $\hat{\beta}$, RSS, dan R^2 lewat persamaan normal;
- menyusun *confidence interval* untuk *error* dan memilih *metric* yang jujur;
- merancang protokol evaluasi dan *k-fold cross validation* tanpa kebocoran, lalu membandingkan dua model dengan *paired t-test*;
- mengevaluasi model *regression* dengan RMSE, MAE, dan R^2 uji terhadap *baseline*;
- menjalankan *gradient descent* dengan tangan dan menjelaskan pengaruh *learning rate* serta ukuran *batch*.

Sumber: buku ajar Bab 10 dan Bab 11.

Linear regression

Dari *classification* ke *regression*

Pada 1886 Francis Galton mengukur tinggi 205 pasang orang tua dan sekitar 930 anak dewasanya. Simpangan tinggi anak dari rata-rata kira-kira dua pertiga simpangan orang tuanya, dan Galton menyebut gejala itu *regression towards mediocrity*. Metode *least squares*-nya lebih tua: Legendre memakainya pada 1805 untuk orbit komet.

Yang berubah dari *classification*

- *Output* $y \in \mathbb{R}$: harga rumah, lama kirim, nilai kuis
- *Error* diukur sebagai **jarak**, dengan *metric* MSE, RMSE, MAE, R^2

Yang tetap

- Pembagian *training*, *validation*, dan *test* tetap wajib
- *Overfitting* dan *inductive bias* (di sini: asumsi linier) tetap ada

Untuk d *feature* dan kolom konstanta 1 di setiap baris:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d, \qquad \hat{\mathbf{y}} = X\beta, \quad X \in \mathbb{R}^{n \times (d+1)}$$

Kata "linier" mengacu pada **parameter**. Kolom x^2, $\log x$, atau $x_1 x_2$ boleh ditambahkan dan modelnya tetap linier dalam β; parameter di dalam $e^{\beta_1 x}$ tidak boleh.

Mengapa *error* kuadrat

Ukuran kesalahannya adalah jumlah kuadrat ***residual***:

$$\mathrm{RSS}(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \|\mathbf{y} - X\beta\|^2, \qquad \mathrm{MSE} = \frac{\mathrm{RSS}}{n}$$

Alasan praktis

Smooth dan *convex* dalam β: *derivative*-nya ada di mana-mana, minimumnya tunggal, dan ada bentuk tertutup. *Error* mutlak $|y - \hat{y}|$ patah di nol dan tidak punya bentuk tertutup.

Alasan prinsipil

Untuk $y_i = \mathbf{x}_i^\top \beta + \varepsilon_i$ dengan $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ saling bebas,

$$\log L(\beta) = -\tfrac{n}{2} \log(2\pi\sigma^2) - \tfrac{1}{2\sigma^2} \textstyle\sum_i (y_i - \mathbf{x}_i^\top \beta)^2$$

Suku pertama tidak memuat β, jadi memaksimalkan $\log L$ sama dengan meminimalkan RSS.

Least squares adalah *maximum likelihood estimator* bila *error*-nya Gaussian. Bila *error*-nya berekor tebal atau penuh *outlier*, *error* mutlak atau Huber lebih tepat; keduanya muncul lagi di bagian *loss function*.

Menurunkan persamaan normal

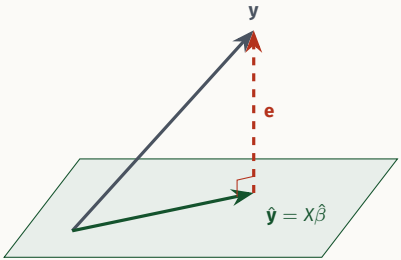
Kembangkan RSS (ingat $\beta^\top X^\top \mathbf{y} = \mathbf{y}^\top X\beta$ karena skalar):

$$\mathrm{RSS}(\beta) = \mathbf{y}^\top\mathbf{y} - 2\beta^\top X^\top \mathbf{y} + \beta^\top X^\top X\beta$$
$$\frac{\partial\,\mathrm{RSS}}{\partial\beta} = -2X^\top\mathbf{y} + 2X^\top X\beta$$

Samakan dengan nol, lahir ***normal equation*** (persamaan normal):

$$X^\top X\beta = X^\top\mathbf{y} \implies \boxed{\hat{\beta} = (X^\top X)^{-1}X^\top\mathbf{y}}$$

Hessian $2X^\top X$ semi-definit positif, karena $\mathbf{v}^\top X^\top X\mathbf{v} = \|X\mathbf{v}\|^2 \geq 0$. RSS cembung, sehingga titik stasionernya minimum global.



Persamaan normal dapat ditulis $X^\top(\mathbf{y} - X\hat{\beta}) = \mathbf{o}$: *residual* **e** tegak lurus (*normal*) terhadap setiap kolom X. Jadi $\hat{\mathbf{y}}$ adalah **proyeksi ortogonal y** ke ruang kolom X.

Contoh terhitung: jam belajar dan nilai kuis

Jam belajar mandiri per minggu (x) dan nilai kuis skala 10 (y) untuk lima mahasiswa.

x_i	y_i	$x_i - \bar{x}$	$y_i - \bar{y}$
1	2	-2	-2
2	4	-1	0
3	5	0	1
4	4	1	0
5	5	2	1
$\bar{x} = 3$,	$\bar{y} = 4$		

Garis selalu melewati $(\bar{x}, \bar{y}) = (3, 4)$, akibat langsung rumus $\hat{\beta}_0$.

Cara 1: rumus satu variabel.

$$S_{xy} = \sum (x_i - \bar{x})(y_i - \bar{y}) = 4 + 0 + 0 + 0 + 2 = 6, \quad S_{xx} = \sum (x_i - \bar{x})^2 = 10$$

$$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}} = 0{,}6, \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} = 4 - 0{,}6 \times 3 = 2{,}2 \;\Rightarrow\; \hat{y} = 2{,}2 + 0{,}6\,x$$

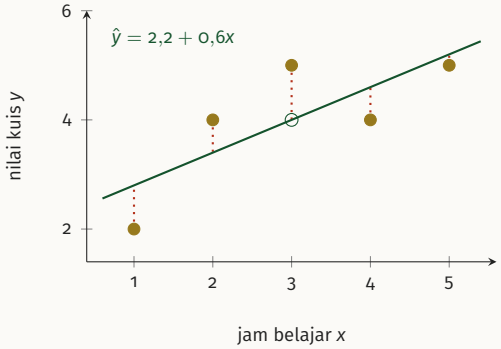
Cara 2: bentuk matriks. $\sum x_i = 15$, $\sum x_i^2 = 55$, $\sum y_i = 20$, $\sum x_i y_i = 66$:

$$X^\top X = \begin{bmatrix} 5 & 15 \\ 15 & 55 \end{bmatrix}, \quad X^\top \mathbf{y} = \begin{bmatrix} 20 \\ 66 \end{bmatrix}, \quad \det = 275 - 225 = 50$$

$$\hat{\beta} = \frac{1}{50} \begin{bmatrix} 55 & -15 \\ -15 & 5 \end{bmatrix} \begin{bmatrix} 20 \\ 66 \end{bmatrix} = \frac{1}{50} \begin{bmatrix} 110 \\ 30 \end{bmatrix} = \begin{bmatrix} 2{,}2 \\ 0{,}6 \end{bmatrix}$$

Bentuk matriks tetap berlaku untuk sepuluh atau seratus *feature*; yang membesar hanya ukuran $X^\top X$.

Garis, *residual*, dan R^2



Garis titik-titik merah adalah *residual*; lingkaran kosong adalah $(\bar{x}, \bar{y})$.

x_i	y_i	$\hat{y}_i$	$e_i = y_i - \hat{y}_i$	e_i^2
1	2	2,8	$-0{,}8$	0,64
2	4	3,4	$+0{,}6$	0,36
3	5	4,0	$+1{,}0$	1,00
4	4	4,6	$-0{,}6$	0,36
5	5	5,2	$-0{,}2$	0,04
		jumlah	0,0	2,40

$\mathrm{RSS} = 2{,}40$, $\mathrm{MSE} = 0{,}48$, $\mathrm{RMSE} = \sqrt{0{,}48} = 0{,}69$.
$\mathrm{TSS} = \sum(y_i - \bar{y})^2 = 4 + 0 + 1 + 0 + 1 = 6{,}00$.

$$R^2 = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{2{,}40}{6{,}00} = 0{,}60$$

$$R^2_{\text{adj}} = 1 - (1 - R^2)\tfrac{n-1}{n-p-1} = 1 - 0{,}4 \cdot \tfrac{4}{3} = 0{,}47$$

Jumlah *residual* selalu nol bila model memuat konstanta β_0.

Membaca R^2, dan seberapa yakin kita pada koefisien

$R^2 = 0{,}60$: 60% variasi nilai kuis terjelaskan oleh jam belajar, dibandingkan dengan menebak $\bar{y}$ untuk semua orang.

- Menambah *feature* apa pun, termasuk kolom bilangan acak, tidak pernah menurunkan R^2 latih. R^2_{adj} dapat turun; di sini $0{,}60 \to 0{,}47$ karena $n = 5$.
- Pada data perilaku manusia $R^2 = 0{,}3$ sudah bermanfaat; garis lurus pada data melengkung bisa memberi 0,9.
- Laporkan RMSE bersama R^2: "salah rata-rata sekitar 0,69 poin pada skala 10" lebih mudah ditimbang.

Standard error kemiringan

$$\hat{\sigma}^2 = \frac{\text{RSS}}{n-p-1} = \frac{2{,}40}{3} = 0{,}80$$

$$\text{SE}(\hat{\beta}_1) = \sqrt{\hat{\sigma}^2 / S_{xx}} = \sqrt{0{,}80/10} = 0{,}283$$

$$t = \frac{\hat{\beta}_1}{\text{SE}(\hat{\beta}_1)} = \frac{0{,}6}{0{,}283} = 2{,}12$$

Derajat kebebasan $n - p - 1 = 3$, nilai kritis dua sisi 95% $= 3{,}182$. Karena $2{,}12 < 3{,}182$, lima mahasiswa belum cukup menjadi bukti bahwa jam belajar berhubungan dengan nilai kuis, walaupun garisnya tampak naik.

Semua angka ini dihitung pada *training data*. Kinerja pada data baru ditaksir dengan protokol evaluasi di bagian berikutnya.

Kode: persamaan normal, R^2, dan *standard error*

```python
import numpy as np

x = np.array([1, 2, 3, 4, 5], dtype=float)  # jam belajar
y = np.array([2, 4, 5, 4, 5], dtype=float)  # nilai kuis
X = np.c_[np.ones_like(x), x]               # kolom konstanta

# selesaikan, JANGAN invers
beta = np.linalg.solve(X.T @ X, X.T @ y)
# lebih stabil (QR/SVD)
beta_l, *_ = np.linalg.lstsq(X, y, rcond=None)
e = y - X @ beta
rss, tss = e @ e, ((y - y.mean()) ** 2).sum()
n, p = len(y), X.shape[1] - 1
print(beta, 1 - rss / tss)        # [2.2 0.6] 0.6
sigma2 = rss / (n - p - 1)
se = np.sqrt(np.diag(sigma2 * np.linalg.inv(X.T @ X)))
print(se, beta[1] / se[1])        # [0.938 0.283] t=2.12
```

Kode lengkap: `sample-code-buku`, folder `bab10-evaluasi-regresi-linier`, `regresi_normal_equation.py`.

Jangan membalik matriks langsung

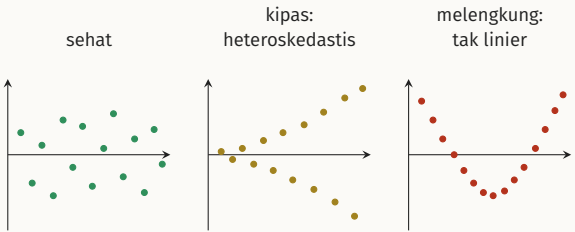
Rumus $(X^\top X)^{-1} X^\top \mathbf{y}$ cocok untuk penurunan. Membentuk $X^\top X$ mengkuadratkan *condition number*, sehingga *error* pembulatan membesar ketika *feature* saling berkorelasi. Pakai `np.linalg.solve`, atau lebih baik `np.linalg.lstsq` (QR/SVD). `LinearRegression` di scikit-learn juga memakai *least squares* yang stabil.

Invers pada baris `se` hanya dipakai untuk matriks 2×2 demi membaca diagonalnya: $\sqrt{0{,}8 \times 1{,}1} = 0{,}938$ dan $\sqrt{0{,}8 \times 0{,}1} = 0{,}283$.

Asumsi *linear regression* dan plot *residual*

- **Linearitas**: hubungan kira-kira berupa garis atau bidang.
- ***Error* saling bebas**: sering dilanggar pada runtun waktu; *standard error* jadi terlalu kecil.
- **Homoskedastisitas**: *variance error* tetap di seluruh $\hat{y}$.
- **Normalitas *error***: perlu untuk uji t dan selang pada sampel kecil.
- **Multikolinearitas rendah**: periksa $\mathrm{VIF}_j = 1/(1 - R_j^2)$; di atas sekitar 5–10 patut dicurigai.

Ketiga asumsi pertama (dengan rata-rata *error* nol) membuat *least squares* menjadi BLUE menurut teorema Gauss–Markov.



Plot *residual* e_i terhadap $\hat{y}_i$ (ilustrasi). Yang diharapkan adalah awan tak berpola di sekitar nol. Kipas: coba $\log y$ atau *standard error* tangguh. Lengkung: tambah x^2 atau ganti model.

Korelasi tidak membuktikan sebab

$\hat{\beta}_1 = 0{,}6$ berarti satu jam belajar tambahan *disertai* kenaikan nilai rata-rata 0,6 poin. Motivasi bisa menjadi *confounder* yang menaikkan keduanya. Klaim sebab-akibat memerlukan eksperimen teracak.

Evaluasi hipotesis

Sample error dan *true error*

Decision tree tanpa *pruning* pada 200 baris data mudah mencapai *training accuracy* 100% dengan satu *leaf* per baris. Angka latih dihitung pada data yang membentuk model, dan setiap keputusan yang diambil sambil melihatnya ikut menempelkan informasi data itu ke dalam model.

Sample error: yang bisa dihitung

$$\mathrm{error}_S(h) = \frac{1}{n} \sum_{x \in S} \delta\big(f(x) \neq h(x)\big)$$

Proporsi salah pada sampel S berukuran n. Angka ini yang keluar dari `1 - model.score(X_uji, y_uji)`.

True error: yang kita inginkan

$$\mathrm{error}_{\mathcal{D}}(h) = \Pr_{x \sim \mathcal{D}} \big[f(x) \neq h(x)\big]$$

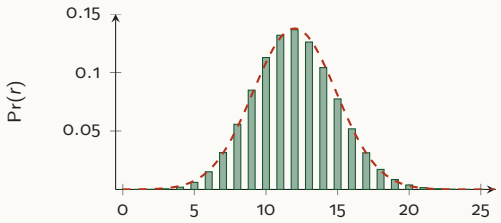
Peluang h salah pada satu contoh acak dari populasi $\mathcal{D}$. Angka ini tidak pernah kita ketahui; ia hanya dapat ditaksir.

Dua pertanyaan evaluasi hipotesis (Mitchell Bab 5): seberapa dekat $\mathrm{error}_S(h)$ kepada $\mathrm{error}_{\mathcal{D}}(h)$, dan bila h_1 lebih baik daripada h_2 pada satu sampel, seberapa yakin kita bahwa h_1 lebih baik pada populasi?

Error sebagai variabel acak

Ambil 40 *test example*, hitung yang salah, ulangi dengan 40 contoh lain: hasilnya berbeda. Setiap *test example* adalah percobaan Bernoulli dengan peluang salah $p = \mathrm{error}_{\mathcal{D}}(h)$, sehingga jumlah kesalahan r berdistribusi **binomial**:

$$\Pr(r) = \binom{n}{r} p^r (1-p)^{n-r}$$



jumlah kesalahan r dari $n = 40$

- $E[r/n] = p$: error_S adalah ***unbiased estimator***.
- *Standard deviation*-nya $\sqrt{p(1-p)/n} \approx \sqrt{\mathrm{err}(1-\mathrm{err})/n}$.

$p = 0{,}3$, $n = 40$: *true error* tetap 30%, tetapi kesalahan teramati berkisar sekitar 6 sampai 18. Garis putus-putus: hampiran normal dengan rata-rata 12 dan *variance* 8,4.

Bias dan *variance* penaksir

Bias: meleset sistematis, terjadi bila *error* diukur pada *training data*. Bias hilang begitu *test data* dipisahkan. ***Variance***: berayun dari sampel ke sampel bila *test data* sedikit; mengecil sebanding $1/\sqrt{n}$.

Confidence interval untuk *error*

Untuk n cukup besar (patokan: $n \geq 30$ dan $n \cdot \mathrm{err}(1-\mathrm{err}) \geq 5$), binomial dihampiri normal, dan selang kepercayaan $N\%$ untuk $\mathrm{error}_{\mathcal{D}}(h)$ adalah

$$\mathrm{err} \pm z_N \sqrt{\frac{\mathrm{err}\,(1-\mathrm{err})}{n}}$$

Tingkat kepercayaan $N\%$	50%	68%	80%	90%	95%	98%	99%
Pengali z_N	0,67	1,00	1,28	1,64	1,96	2,33	2,58

Cara membaca selang 95%: bila percobaannya (ambil sampel uji, hitung selang) diulang berkali-kali, sekitar 95% selang yang terbentuk memuat *error* sebenarnya. *True error* adalah angka tetap dalam statistika frekuentis.

Batas hampiran normal

Bila n kecil atau err dekat 0 atau 1, batas bawah bisa negatif. Pakai selang Wilson atau Clopper–Pearson (`statsmodels`), atau tambah *test data*.

Contoh terhitung: 12 salah dari 40 sampel uji

1. $\mathrm{err} = 12/40 = 0{,}30$
2. $\sqrt{0{,}30 \times 0{,}70/40} = \sqrt{0{,}00525} = 0{,}0725$
3. Untuk 95%, $1{,}96 \times 0{,}0725 = 0{,}142$
4. Selang: $0{,}30 \pm 0{,}142 = [\,0{,}158\,;\ 0{,}442\,]$

Error sebenarnya bisa 16%, bisa juga 44%; ujung atas hampir tiga kali ujung bawah. "*Accuracy* 70%" tanpa selang ini menyembunyikan rentang 56%–84%. Selang ini hanya sah bila 40 sampel itu tidak pernah dipakai melatih atau menyetel model.

Fungsi `selang_error(12, 40)` di `bab10-evaluasi-regresi-linier/selang_kepercayaan.py` mencetak `err=0.300 sd=0.0725 CI95=[0.158; 0.442]`.

n	Selang 95%	Lebar
10	[0,016 ; 0,584]	0,568
40	[0,158 ; 0,442]	0,284
100	[0,210 ; 0,390]	0,180
400	[0,255 ; 0,345]	0,090
1.000	[0,272 ; 0,328]	0,057
10.000	[0,291 ; 0,309]	0,018

Semua baris memakai $\mathrm{err} = 0{,}30$; yang berubah hanya n. Lebar turun sebanding $1/\sqrt{n}$, sehingga memotong lebar menjadi setengah memerlukan *test data* empat kali lebih banyak. Klaim "model kami 2% lebih akurat" yang diuji pada 50 baris data sebaiknya tidak dipercaya.

Metric di luar *accuracy*: matriks konfusi

		***Prediction* model**	
		Positif	Negatif
Kenyataan	Positif	TP	FN
	Negatif	FP	TN

- TP: penipuan yang tertangkap; FN: penipuan yang lolos
- FP: transaksi wajar yang ditahan sia-sia; TN: transaksi wajar yang dilewatkan

Data kartu kredit Dal Pozzolo dkk.: 492 penipuan dari 284.807 transaksi (0,17%). Model yang selalu menjawab "sah" berakurasi 99,83% dengan *recall* nol.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}, \qquad Recall = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Precision: dari yang saya tuduh, berapa yang benar bersalah? ***Recall***: dari yang benar bersalah, berapa yang saya tangkap? Rata-rata harmonik pada F_1 menghukum ketimpangan: bila salah satunya mendekati nol, F_1 ikut mendekati nol.

Contoh terhitung: penipuan satu dari seribu

Dari 10.000 transaksi, 10 penipuan. Tiga model diuji pada data yang sama.

Model	TP	FN	FP	TN	***Accuracy***	***Precision***	***Recall***	$\mathbf{F_1}$
A, selalu menjawab “aman”	0	10	0	9.990	**99,90%**	—	0,00	0,000
B, *threshold* longgar	8	2	90	9.900	99,08%	0,082	**0,80**	0,148
C, *threshold* ketat	5	5	10	9.980	99,85%	0,333	0,50	**0,400**

Precision B $= 8/98 = 0{,}082$. *Precision* C $= 5/15 = 0{,}333$ dan $F_1 = 2(0{,}333)(0{,}5)/0{,}833 = 0{,}400$. Model A berakurasi tertinggi dan tidak pernah menangkap satu pun penipuan.

Memilih *metric* menurut *cost*

Recall: melewatkan positif mahal (skrining kanker). *Precision*: tuduhan salah mahal (penyaring *spam*). F_1: belum ada alasan memihak. ROC-AUC: menilai urutan skor. PR-AUC: kelas positif sangat langka.

Bila *cost* diketahui

Misalnya penipuan lolos Rp5 juta dan transaksi wajar yang ditahan Rp50 ribu (angka ilustrasi): hitung $5\,000\,000 \cdot FN + 50\,000 \cdot FP$ untuk tiap *threshold*, lalu pilih yang termurah, pada *validation data*.

Protokol evaluasi: latih, validasi, uji, tanpa kebocoran



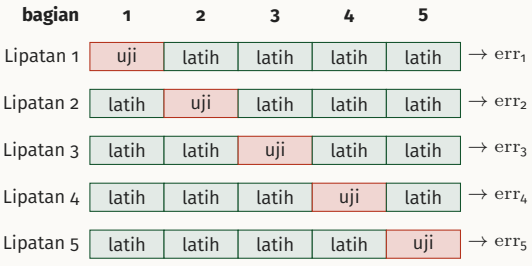
Validation data boleh dipakai berulang untuk membandingkan kandidat (kedalaman 5 atau 10, $C = 1$ atau 10). Sekali *test data* dipakai untuk mengambil keputusan, ia berubah menjadi data validasi dan angkanya cenderung terlalu optimistis.

Bentuk *data leakage* yang paling sering

- *Normalization*, pengisian nilai hilang, *feature selection*, atau SMOTE sebelum pembagian
- *Feature* yang mengandung jawaban (“tanggal pelunasan klaim”)
- Pembagian acak pada data berurut waktu
- Kelompok yang terpecah: satu pasien dengan lima kunjungan

Pencegahan utama: semua *preprocessing* masuk ke `Pipeline` yang di-*fit* hanya pada *training data*, ditambah pembagian yang menghormati waktu dan kelompok.

k-fold cross validation



$$\bar{\text{err}} = \frac{1}{k}\sum_{i=1}^{k} \text{err}_i, \qquad s = \sqrt{\frac{1}{k-1}\sum_{i=1}^{k}(\text{err}_i - \bar{\text{err}})^2}$$

Setiap baris data tepat sekali menjadi *test data*, dan *k* angka memperlihatkan ayunannya. *Cost*-nya: model dilatih *k* kali. Kohavi menyarankan *stratified* 10-fold sebagai pilihan baku.

Varian menurut struktur data

- *Stratified*: proporsi kelas dijaga; baku untuk *classification*
- *Group*: satu pasien atau pengguna dalam satu lipatan
- *Time series split*: latih di masa lalu, uji di masa depan
- *Nested*: CV dalam untuk menyetel, CV luar untuk menilai
- *Leave-one-out*: $k = n$, untuk data sangat sedikit

Membandingkan dua algoritma: *paired t-test*

Kedua algoritma diuji pada **lipatan yang sama**, sehingga selisihnya dapat dibandingkan lipatan demi lipatan; variasi “lipatan ini kebetulan sulit” dialami keduanya.

Lipatan	Pohon tunggal	*Random forest*	d_i
1	0,22	0,16	0,06
2	0,26	0,21	0,05
3	0,19	0,15	0,04
4	0,24	0,20	0,04
5	0,21	0,18	0,03
Rata-rata	0,224	0,180	**0,044**

$$t = \frac{\bar{d}}{s_{\bar{d}}}, \qquad s_{\bar{d}} = \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} (d_i - \bar{d})^2}$$

1. Simpangan $d_i - \bar{d}$: 0,016; 0,006; −0,004; −0,004; −0,014
2. Jumlah kuadratnya 0,00052
3. $s_{\bar{d}} = \sqrt{0{,}00052/(5 \cdot 4)} = \sqrt{0{,}000026} = 0{,}0051$
4. $t = 0{,}044/0{,}0051 \approx 8{,}6$
5. Derajat kebebasan 4; nilai kritis dua sisi 95% = 2,776

Karena $8{,}6 > 2{,}776$, selisihnya **nyata**. *Random forest* menang di setiap lipatan dengan jarak hampir sama (0,03 sampai 0,06), sehingga simpangan selisihnya kecil.

Lipatan tidak saling bebas

Pada 10-fold, dua lipatan berbagi sekitar 89% *training data*, sehingga uji t cenderung terlalu mudah menyatakan “nyata”. Alternatif yang lebih aman: uji 5×2 CV (Dietterich) atau koreksi Nadeau–Bengio.

Selisih kecil, *p-value*, dan kodenya

Selisih kecil yang tidak terbukti

Model X: 0,182; 0,201; 0,177; 0,194; 0,188.
Model Y: 0,174; 0,200; 0,181; 0,182; 0,190.
$d_i = 0{,}008;\ 0{,}001;\ -0{,}004;\ 0{,}012;\ -0{,}002$,
$\bar{d} = 0{,}003$. Jumlah kuadrat simpangan 0,000184,
$s_{\bar{d}} = \sqrt{0{,}000184/20} = 0{,}00303$, dan
$t \approx 0{,}99 < 2{,}776$: **tidak ada bukti** Y lebih baik.

p-value: andaikan kedua algoritma sama baik (hipotesis nol), p adalah peluang melihat selisih sebesar ini atau lebih besar karena keberuntungan pembagian data. p tidak menyatakan peluang model B lebih baik dan tidak mengukur besar selisihnya.

```python
from scipy import stats
from sklearn.datasets import load_breast_cancer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import (StratifiedKFold,
                                     cross_val_score)
from sklearn.tree import DecisionTreeClassifier

X, y = load_breast_cancer(return_X_y=True)
cv = StratifiedKFold(n_splits=10, shuffle=True,
                     random_state=0)  # lipatan SAMA
e_pohon = 1 - cross_val_score(
    DecisionTreeClassifier(random_state=0), X, y, cv=cv)
e_rf = 1 - cross_val_score(
    RandomForestClassifier(300, random_state=0),
    X, y, cv=cv)
t, p = stats.ttest_rel(e_pohon, e_rf)
print(f"selisih={(e_pohon - e_rf).mean():.4f} "
      f"t={t:.2f} p={p:.4f}")
# selisih=0.0405 t=3.73 p=0.0047
```

Mengevaluasi model *regression*: RMSE, MAE, dan R^2 uji

Untuk m titik uji yang tidak ikut melatih:

$$\mathrm{RMSE} = \sqrt{\tfrac{1}{m}\sum (y_i - \hat{y}_i)^2}, \quad \mathrm{MAE} = \tfrac{1}{m}\sum |y_i - \hat{y}_i|$$

$$R^2_{test} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_{test})^2}$$

RMSE selalu $\geq$ MAE; bila jauh di atas MAE, segelintir titik menyumbang *error* besar. R^2_{test} bisa negatif: model lebih buruk daripada menebak rata-rata *test data* itu sendiri.

Tiga mahasiswa baru

Garis $\hat{y} = 2{,}2 + 0{,}6\,x$ dipakai untuk tiga mahasiswa yang tidak ikut melatih.

x_i	y_i	$\hat{y}_i$	e_i	e_i^2
1,5	3	3,1	$-0{,}1$	0,01
3,5	5	4,3	$+0{,}7$	0,49
6,0	5	5,8	$-0{,}8$	0,64

MSE $= 1{,}14/3 = 0{,}38$, RMSE $= 0{,}62$, MAE $= 1{,}6/3 = 0{,}53$. Rata-rata nilai uji 4,33, $\sum (y_i - \bar{y}_{test})^2 = 2{,}67$, sehingga $R^2_{test} = 1 - 1{,}14/2{,}67 = 0{,}57$, sedikit di bawah R^2 latih 0,60.

Baseline yang menebak rata-rata latih (4) memberi *error* $-1, +1, +1$: MSE $= 1{,}00$, MAE $= 1{,}00$, $R^2_{test} = 1 - 3/2{,}67 = -0{,}125$. Di scikit-learn *baseline* ini adalah `DummyRegressor`.

Cross-validation untuk *regression*: kodenya

Data diabetes scikit-learn: 442 pasien, 10 *feature*, target perkembangan penyakit setahun kemudian. Kode: `bab10-evaluasi-regresi-linier/evaluasi_regresi.py`

```
import numpy as np
from sklearn.datasets import load_diabetes
from sklearn.dummy import DummyRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import KFold, cross_validate, train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

X, y = load_diabetes(return_X_y=True)
X_lt, X_uj, y_lt, y_uj = train_test_split(X, y, test_size=0.2, random_state=0)
model = make_pipeline(StandardScaler(), LinearRegression())
cv = KFold(n_splits=5, shuffle=True, random_state=0)
skor = ["neg_mean_squared_error", "neg_mean_absolute_error", "r2"]
for nama, m in [("rata-rata", DummyRegressor()), ("linier", model)]:
    h = cross_validate(m, X_lt, y_lt, cv=cv, scoring=skor, return_train_score=True)
    print(nama, np.sqrt(-h["test_neg_mean_squared_error"]).mean(),
          -h["test_neg_mean_absolute_error"].mean(),
          h["train_r2"].mean(), h["test_r2"].mean(), h["test_r2"].std(ddof=1))
print(model.fit(X_lt, y_lt).score(X_uj, y_uj))   # R2 uji = 0.332
```

Cross-validation untuk *regression*: membaca hasilnya

	RMSE	MAE	R^2_{train}	R^2_{CV}	sd
rata-rata	78,6	67,8	0,000	−0,030	0,032
linier	53,9	43,9	0,556	0,506	0,113

Rata-rata 5 lipatan, dibulatkan. R^2 pada *test data* (89 pasien): 0,332.

Scikit-learn menamai skornya `neg_mean_squared_error` karena semua fungsi pemilihan model memaksimalkan skor; MSE diberi tanda negatif agar "lebih besar lebih baik" tetap berlaku. Pada *regression* dipakai `KFold`, karena `StratifiedKFold` memerlukan label kelas.

- RMSE turun dari 78,6 (*baseline*) ke 53,9, dengan $R^2_{\text{CV}} = 0{,}506$: model ini jelas lebih berguna daripada menebak rata-rata.
- R^2 latih 0,556 dan CV 0,506 berdekatan. Dengan 10 *feature* dan 353 baris latih, *linear regression* hampir tidak *overfitting*; kelemahannya bias.
- R^2 uji 0,332 jauh di bawah rata-rata CV. Simpangan baku antarlipatan 0,113 menunjukkan satu pembagian kecil memang bisa berayun sejauh itu, gejala yang sama dengan lebar *confidence interval*.

Membandingkan *linear regression* dengan *ridge*: pakai `KFold` yang sama untuk keduanya, lalu *paired t-test* atas MSE per lipatan.

Batas bentuk tertutup

Cost-nya tumbuh cepat

Menyusun $X^\top X$ butuh $O(nd^2)$ operasi dan menyelesaikan sistemnya $O(d^3)$. Untuk $d = 100$ ringan. Untuk $d = 10.000$, tahap $O(d^3)$ saja sekitar 10^{12} operasi, dan matriks $10^4 \times 10^4$ harus muat di memori. Data yang tidak muat di memori tidak dapat disusun menjadi satu X.

$X^\top X$ bisa singular

Terjadi bila ada *feature* yang merupakan kombinasi linier *feature* lain, atau $d > n$, keadaan biasa pada data teks dan genomik.

Dua jalan keluar

- ***Regularization***. *Ridge* menambah $\lambda\|\beta\|^2$: $\hat{\beta}_{ridge} = (X^\top X + \lambda I)^{-1} X^\top \mathbf{y}$ selalu dapat dibalik untuk $\lambda > 0$. *Lasso* memakai $\lambda\|\beta\|_1$ dan membuat sebagian koefisien tepat nol. λ dipilih dengan *cross-validation*.
- **Metode numerik**. Turuni permukaan *error* selangkah demi selangkah:

$$\beta_{t+1} = \beta_t - \eta \nabla \mathrm{RSS}(\beta_t)$$

Cukup *gradient*, yang dapat dihitung per *batch* data.

Jembatan

Logistic regression, *neural network*, dan *transformer* tidak punya bentuk tertutup sama sekali. Mesin numerik pada bagian berikut melatih semuanya.

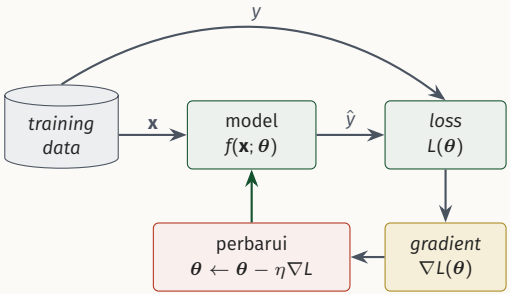
Loss function dan *gradient descent*

Belajar sebagai *optimization*

1. Pilih **model** $f(\mathbf{x}; \boldsymbol{\theta})$; pada *linear regression* $f(\mathbf{x}; \beta) = \mathbf{x}^\top \beta$.
2. Pilih ***loss function*** $\ell(\hat{y}, y)$ untuk *error* satu contoh.
3. Rata-ratakan di seluruh *training data*, **risiko empiris** atau *cost function*:

$$L(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \ell\big(f(\mathbf{x}_i; \boldsymbol{\theta}),\, y_i\big)$$

4. Cari $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$.



Cukup sediakan model dan *loss*; satu mesin *optimization* yang sama melatih *logistic regression* sampai *transformer*. Persamaan normal menyelesaikan $\nabla L = \mathbf{0}$ dengan aljabar; *gradient descent* mendekatinya bertahap.

Tiga *loss function* untuk *regression*

Mulai sekarang ***residual*** ditulis $e_i = \hat{y}_i - y_i$ (tanda dibalik agar rumus *gradient* lebih rapi; untuk *loss* simetris tidak berpengaruh).

MSE

$$\frac{1}{n}\sum_i e_i^2$$

Smooth, *derivative* $2e$. Minimum model konstan: rata-rata. Satu *outlier* ber-*error* 10 menyumbang 100.

MAE

$$\frac{1}{n}\sum_i |e_i|$$

Tahan *outlier*, minimumnya median. *Derivative* ± 1 dan patah di nol, sehingga GD berosilasi dekat minimum.

Huber

$$\ell_\delta(e) = \begin{cases} \frac{1}{2}e^2 & |e| \le \delta \\ \delta\left(|e| - \frac{1}{2}\delta\right) & |e| > \delta \end{cases}$$

Gradient-nya terpotong di $\pm\delta$.

Pengaruh satu *outlier* (*residual* jam belajar, Huber $\delta = 1$)

$e = (0{,}8;\ -0{,}6;\ -1{,}0;\ 0{,}6;\ 0{,}2)$: MSE $= 0{,}48$, MAE $= 3{,}2/5 = 0{,}64$, Huber $= (0{,}32 + 0{,}18 + 0{,}50 + 0{,}18 + 0{,}02)/5 = 0{,}24$. Nilai mahasiswa kelima salah dicatat sehingga $e_5 = 6{,}0$: MSE $= (2{,}36 + 36)/5 = 7{,}672$, MAE $= (3{,}0 + 6{,}0)/5 = 1{,}80$, Huber $= (1{,}18 + 1 \cdot (6 - 0{,}5))/5 = 1{,}336$. Sumbangan titik itu ke *gradient*: $2 \times 6 = 12$ pada MSE, hanya $\delta = 1$ pada Huber.

Gradient dan *gradient* MSE *linear regression*

Gradient

$$\nabla L(\boldsymbol{\theta}) = \left(\frac{\partial L}{\partial \theta_1}, \ \ldots, \ \frac{\partial L}{\partial \theta_d} \right)^\top$$

Menunjuk ke arah kenaikan L tercepat; panjangnya menyatakan kecuraman.

Hampiran Taylor orde satu: $L(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}) + \nabla L^\top \Delta\boldsymbol{\theta}$. Untuk langkah sepanjang tetap, $\nabla L^\top \Delta\boldsymbol{\theta}$ paling negatif bila $\Delta\boldsymbol{\theta}$ berlawanan arah dengan ∇L. Jadi kita melangkah ke $-\nabla L$.

Satu contoh, $\hat{y}_i = \mathbf{x}_i^\top \beta$, aturan rantai:

$$\frac{\partial}{\partial \beta_j} (\hat{y}_i - y_i)^2 = 2(\hat{y}_i - y_i)\, x_{ij}$$

Rata-ratakan dan tulis dalam bentuk matriks:

$$\nabla L(\beta) = \frac{2}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)\, \mathbf{x}_i = \frac{2}{n} X^\top (X\beta - \mathbf{y})$$

- Setiap contoh mendorong parameter searah *feature vector*-nya, sebanding dengan *error*-nya; contoh yang tepat ($e_i = 0$) tidak mendorong apa-apa.
- $\nabla L = \mathbf{0}$ memberi kembali persamaan normal $X^\top X\beta = X^\top \mathbf{y}$.

Aturan pembaruan

Cauchy (1847) mengusulkan langkah berulang

$$\boxed{\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)}$$

dengan $\eta > 0$ disebut ***learning rate*** atau *step size*.

Setiap langkah memakai **seluruh** *training data* untuk menghitung *gradient*, sehingga versi ini disebut ***batch gradient descent***.

Batch gradient descent

***Input*:** data, η, batas iterasi T, toleransi ε

1. $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$ (nol atau acak kecil)
2. untuk $t = 1, \ldots, T$:
   - $\mathbf{g} \leftarrow \frac{1}{n} \sum_i \nabla_{\boldsymbol{\theta}} \, \ell\big(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i\big)$
   - bila $\|\mathbf{g}\| < \varepsilon$: berhenti
   - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \, \mathbf{g}$
3. kembalikan $\boldsymbol{\theta}$

Untuk $\hat{y} = \beta_0 + \beta_1 x$ *gradient*-nya tinggal dua angka:

$$g_0 = \tfrac{2}{n} \textstyle\sum_i e_i, \qquad g_1 = \tfrac{2}{n} \textstyle\sum_i e_i x_i, \qquad e_i = \beta_0 + \beta_1 x_i - y_i$$

Contoh terhitung: satu langkah *gradient descent*

Data jam belajar $x = (1, 2, 3, 4, 5)$, $y = (2, 4, 5, 4, 5)$, mulai dari $\beta_0 = \beta_1 = 0$, $\eta = 0{,}05$.

x_i	y_i	$\hat{y}_i$	e_i	$e_i x_i$
1	2	0	-2	-2
2	4	0	-4	-8
3	5	0	-5	-15
4	4	0	-4	-16
5	5	0	-5	-25
		jumlah	-20	-66

MSE awal $= (4 + 16 + 25 + 16 + 25)/5 = 17{,}2$.

***Gradient*:**

$$g_0 = \tfrac{2}{5}(-20) = -8, \qquad g_1 = \tfrac{2}{5}(-66) = -26{,}4$$

Langkah: $\beta_0 \leftarrow 0 - 0{,}05\,(-8) = 0{,}4$ dan $\beta_1 \leftarrow 0 - 0{,}05\,(-26{,}4) = 1{,}32$.

Periksa: $\hat{y} = 0{,}4 + 1{,}32x = (1{,}72;\ 3{,}04;\ 4{,}36;\ 5{,}68;\ 7{,}00)$, *error* $e = (-0{,}28;\ -0{,}96;\ -0{,}64;\ 1{,}68;\ 2{,}00)$, MSE turun dari 17,2 ke **1,6464**.

Gradient negatif berarti *loss* turun bila parameter dinaikkan. Langkah $-\eta\mathbf{g}$ menaikkan keduanya dan mengoreksi *prediction* awal yang terlalu rendah.

Iterasi berikutnya: melompat lalu merangkak

t	β_0	β_1	$(g_0;\ g_1)$	MSE
0	0	0	$(-8;\ -26{,}4)$	17,2000
1	0,400	1,320	$(0{,}72;\ 5{,}04)$	1,6464
2	0,364	1,068	$(-0{,}864;\ -0{,}72)$	1,1047
3	0,407	1,104		1,067
10	0,607	1,041		0,9418
50	1,395	0,823		0,5981
100	1,857	0,695		0,5015
500	2,200	0,600		0,4800

Iterasi 1: $\sum e_i = 1{,}8$ dan $\sum e_i x_i = 12{,}6$, sehingga $g = (0{,}72;\ 5{,}04)$:

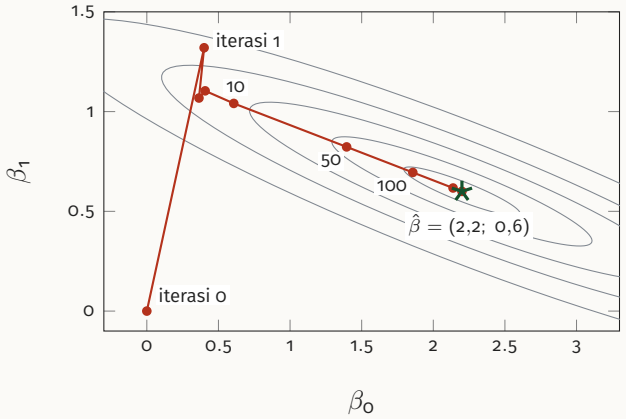
$$\beta_0 \leftarrow 0{,}4 - 0{,}036 = 0{,}364$$

$$\beta_1 \leftarrow 1{,}32 - 0{,}252 = 1{,}068$$

- Langkah pertama memangkas MSE dari 17,2 ke 1,65; 99 langkah berikutnya hanya ke 0,50.
- β_1 melompat melewati nilai akhirnya (0,6) lalu bolak-balik; β_0 merangkak dari 0,4 menuju 2,2.

Setelah 500 iterasi $\beta = (2{,}200;\ 0{,}600)$ dengan MSE 0,48, sama dengan persamaan normal dan RSS 2,40 di awal pertemuan.

Mengapa merangkak: lembah yang pipih



Kontur MSE berbentuk elips yang sangat pipih. Hessian $H = \frac{2}{n}X^\top X = \begin{bmatrix} 2 & 6 \\ 6 & 22 \end{bmatrix}$ ber-*eigenvalue* $\lambda_{\min} \approx 0{,}34$ dan $\lambda_{\max} \approx 23{,}66$; rasionya, yaitu ***condition number***, sekitar 70.

Gradient selalu tegak lurus garis kontur. Pada lembah sempit ia menunjuk ke dinding lembah, sehingga GD memantul di antara dinding lalu berjalan lambat menyusuri dasar menuju bintang.

Gradient descent dari nol dengan NumPy

```python
import numpy as np

def grad_mse(X, y, beta):
    n = X.shape[0]
    return (2.0 / n) * X.T @ (X @ beta - y)  # gradien MSE

def gradient_descent(X, y, eta=0.05, n_iter=2000,
                     tol=1e-10):
    beta = np.zeros(X.shape[1])          # titik awal
    for t in range(n_iter):
        g = grad_mse(X, y, beta)
        if np.linalg.norm(g) < tol:      # kriteria berhenti
            break
        beta = beta - eta * g            # satu langkah turun
    return beta

x = np.array([1, 2, 3, 4, 5], dtype=float)
y = np.array([2, 4, 5, 4, 5], dtype=float)
X = np.column_stack([np.ones_like(x), x])  # kolom 1: intersep
print(gradient_descent(X, y))              # [2.2 0.6]
print(np.linalg.solve(X.T @ X, X.T @ y))   # [2.2 0.6]
```

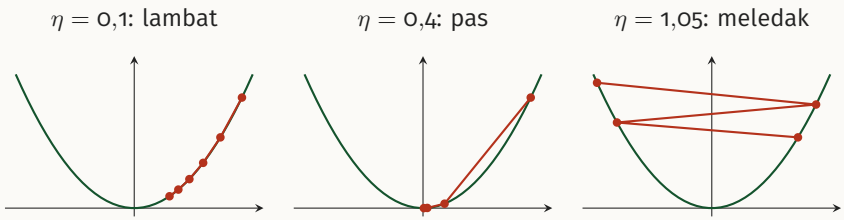
bab11-optimasi-gradient-descent/gd_numpy.py

- Satu iterasi: satu perkalian matriks dan satu pengurangan.
- Kolom konstanta 1 membuat intersep ikut dilatih seperti *weight* biasa.
- Hasilnya dibandingkan dengan persamaan normal untuk menguji implementasi.

Gradient checking

Bandingkan *gradient* buatan sendiri dengan selisih hingga $\big(L(\theta+\epsilon) - L(\theta-\epsilon)\big)/2\epsilon$, $\epsilon \approx 10^{-6}$. Di $\beta = (0{,}4;\ 1{,}32)$ keduanya memberi $(0{,}72;\ 5{,}04)$. Teknik ini dipakai lagi untuk *backpropagation* di Pertemuan 10.

Learning rate: terlalu kecil, pas, terlalu besar



Pada $L(\theta) = \theta^2$, *gradient* 2θ memberi $\theta_{t+1} = (1 - 2\eta)\,\theta_t$. Dengan $\eta = 0{,}4$ faktornya 0,2 dan minimum tercapai dalam beberapa langkah. Dengan $\eta > 1$ nilai mutlak faktornya lebih dari 1: setiap langkah mendarat lebih tinggi di lereng seberang, dan *loss* akhirnya menjadi `inf` atau `nan`.

Batas stabil untuk *loss* kuadratik: $0 < \eta < 2/\lambda_{\max}(H)$

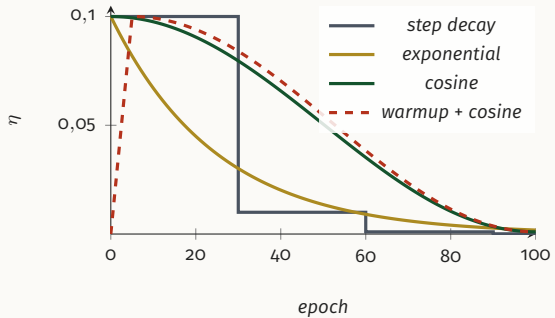
Pada data jam belajar $\lambda_{\max} \approx 23{,}66$, jadi $\eta < 0{,}0845$: $\eta = 0{,}08$ memberi MSE 0,483 setelah 100 iterasi, $\eta = 0{,}09$ memberi sekitar 6×10^{11}. Dengan $\eta = 0{,}08$, di arah landai jarak ke minimum menyusut dengan faktor $1 - \eta\lambda_{\min} = 0{,}973$ per iterasi, sehingga perlu sekitar 84 iterasi untuk menyusutkannya sepuluh kali ($\eta = 0{,}05$: faktor 0,983, sekitar 135 iterasi).

Feature scaling dan penjadwalan *learning rate*

Feature scaling

Condition number besar biasanya muncul karena *feature* berbeda skala (gaji dalam jutaan rupiah, umur dalam tahun). Pada data jam belajar, memusatkan x (kurangi $\bar{x} = 3$) mengubah Hessian menjadi $\mathrm{diag}(2,\ 4)$: *condition number* turun dari 70 ke 2, kontur hampir bundar, dan η boleh jauh lebih besar. Parameter *scaler* dihitung dari *training data* saja.

Learning rate range test: naikkan η eksponensial dari 10^{-6} ke 1 sambil mencatat *loss*, lalu pilih sedikit di bawah titik ketika *loss* melonjak.



Langkah besar di awal, kecil di akhir. *Cosine annealing*: $\eta_t = \eta_{\min} + \frac{1}{2}(\eta_0 - \eta_{\min})\big(1 + \cos(\pi t/T)\big)$. *Warmup* menaikkan η dari hampir nol pada iterasi awal, lazim pada *transformer*.

Batch, SGD, dan *mini-batch*

Gradient adalah **rata-rata**, dan rata-rata dapat ditaksir dari sampel. Akarnya metode aproksimasi stokastik Robbins–Monro (1951).

- *Batch* GD: *gradient* dari seluruh n contoh.
- ***Stochastic gradient descent*** (SGD murni): dari **satu** contoh acak.
- ***Mini-batch***: dari B contoh acak, biasanya 32 sampai beberapa ribu; dalam praktik "SGD" biasanya berarti versi ini.

Satu ***epoch*** $= \lceil n/B \rceil$ iterasi. *Gradient mini-batch* tak bias dengan *variance* $\propto 1/B$: menaikkan B dari 32 ke 128 hanya memangkas simpangan bakunya setengah, dengan *cost* empat kali.

Dua langkah SGD murni, $(0, 0)$**,** $\eta = 0{,}05$

Gradient satu contoh: $(2e,\ 2ex)$.

Langkah 1, contoh $(1, 2)$: $e = -2$, $\mathbf{g} = (-4;\ -4)$, $\beta \leftarrow (0{,}2;\ 0{,}2)$.

Langkah 2, contoh acak $(3, 5)$: $e = 0{,}2 + 0{,}6 - 5 = -4{,}2$, $\mathbf{g} = (-8{,}4;\ -25{,}2)$, $\beta \leftarrow (0{,}62;\ 1{,}46)$.

GD penuh setelah satu langkah berada di $(0{,}4;\ 1{,}32)$ dan sudah membaca lima baris. Dua langkah SGD membaca dua baris. Arahnya tidak persis sama, tetapi rata-ratanya menuju tempat yang sama.

Mini-batch SGD dan perilaku tiga varian

```python
def eta_cosine(t, T, eta_maks=0.1, eta_min=0.001):
    return eta_min + 0.5 * (eta_maks - eta_min) * (
        1 + np.cos(np.pi * t / T))

def minibatch_sgd(X, y, ukuran_batch=32, n_epoch=10,
                  seed=0):
    rng = np.random.default_rng(seed)
    n, d = X.shape
    beta = np.zeros(d)
    T = n_epoch * int(np.ceil(n / ukuran_batch))
    t = 0
    for epoch in range(n_epoch):
        urutan = rng.permutation(n)  # acak tiap epoch
        for mulai in range(0, n, ukuran_batch):
            idx = urutan[mulai:mulai + ukuran_batch]
            Xb, yb = X[idx], y[idx]
            g = (2.0 / len(idx)) * Xb.T @ (
                Xb @ beta - yb)
            beta -= eta_cosine(t, T) * g
            t += 1
    return beta
```

	Batch	**SGD murni**	***Mini-batch***
Contoh/langkah	n	1	B
Kebisingan	nol	tinggi	sedang
Kurva *loss*	*smooth*	bergerigi	agak bergerigi
Jadwal η	tidak wajib	wajib	dianjurkan
Cocok untuk	data kecil	*online*	*neural network*

- Data **diacak ulang** setiap *epoch*; *gradient* dibagi `len(idx)` karena *batch* terakhir bisa lebih pendek.
- Pada data sintetis 10.000 baris, sepuluh *epoch* menyamai bentuk tertutup sampai dua angka di belakang koma.
- Dengan η tetap SGD berkeliaran di sekitar minimum; jadwal $\eta_t = \eta_0/(1 + kt)$ memenuhi syarat Robbins–Monro $\sum \eta_t = \infty$, $\sum \eta_t^2 < \infty$.

Logistic regression singkat: dari garis ke peluang

Untuk kelas biner, *output* linier diperas ke $(0, 1)$ oleh **sigmoid**:

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \qquad \hat{p} = \sigma(\mathbf{w}^\top \mathbf{x} + b)$$

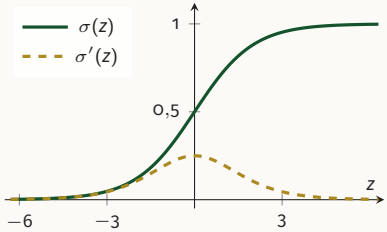
z adalah *logit*, yaitu log-*odds*; batas keputusan $z = 0$ berupa *hyperplane*. MSE pada sigmoid tidak konveks dan nyaris berhenti belajar saat model yakin tetapi salah. *Maximum likelihood* Bernoulli memberi ***binary cross-entropy***:

$$L = -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i) \right]$$

Dengan $\sigma'(z) = \sigma(z)(1 - \sigma(z))$, *derivative*-nya $\partial \ell / \partial z = \hat{p} - y$:

$$\boxed{\nabla_{\mathbf{w}} L = \tfrac{1}{n} X^\top (\hat{\mathbf{p}} - \mathbf{y}), \quad \tfrac{\partial L}{\partial b} = \tfrac{1}{n} \textstyle\sum_i (\hat{p}_i - y_i)}$$

Bentuknya sama dengan *gradient* MSE: *error* kali *feature*.



Derivative sigmoid paling besar 0,25 di $z = 0$ dan mendekati nol untuk $|z|$ besar, fakta yang penting lagi saat membahas *vanishing gradient* pada jaringan dalam. *Logistic regression* adalah satu neuron: *weighted sum*, bias, lalu fungsi tak linier.

Contoh terhitung: satu langkah *logistic regression*

Jam belajar $x = (1, 2, 3, 4, 5)$, label lulus $y = (0, 0, 1, 1, 1)$, mulai dari $w = b = 0$, $\eta = 0{,}5$.

Maju. Semua $z = 0$, jadi $\hat{p} = 0{,}5$ dan *log loss* $= -\log 0{,}5 = 0{,}693$.

Error. $\hat{p} - y = (0{,}5;\ 0{,}5;\ -0{,}5;\ -0{,}5;\ -0{,}5)$

Gradient.

$$\frac{\partial L}{\partial b} = \frac{0{,}5+0{,}5-1{,}5}{5} = -0{,}1$$
$$\frac{\partial L}{\partial w} = \frac{0{,}5+1{,}0-1{,}5-2{,}0-2{,}5}{5} = -0{,}9$$

Pembaruan. $b \leftarrow 0 - 0{,}5(-0{,}1) = 0{,}05$, $w \leftarrow 0 - 0{,}5(-0{,}9) = 0{,}45$.

Periksa. $\hat{p} = \sigma(0{,}45x + 0{,}05)$:

x	1	2	3	4	5
$\hat{p}$	0,62	0,72	0,80	0,86	0,91

Log loss turun ke 0,543.

Peluang lulus kini naik seiring jam belajar. Batas keputusannya ($z = 0$ di $x \approx -0{,}11$) masih jauh dari tempat yang benar, antara 2 dan 3; iterasi berikutnya menggesernya.

Data yang terpisah sempurna

Kelas di sini terpisah sempurna oleh $x = 2{,}5$, sehingga *log loss* terus turun bila w membesar dan tanpa *regularization* $w \to \infty$. Karena itu `LogisticRegression` di scikit-learn memakai L2 secara bawaan. Versi NumPy pada data *breast cancer* (buku Kode `logreg_numpy.py`) mencapai *test accuracy* 0,951, sama dengan scikit-learn.

Praktikum Pertemuan 9

Bagian A: evaluasi *regression*

1. Jalankan kode data diabetes; pastikan angka RMSE, MAE, dan R^2 sama dengan slide.
2. Tambahkan `Ridge` dengan α dipilih lewat `GridSearchCV` di dalam *nested* CV.
3. Bandingkan *linear regression* dan *ridge* dengan *paired t-test* atas MSE per lipatan (`KFold` yang sama), lalu gambarkan plot *residual* model terbaik.

Bagian B: *gradient descent* dari nol

4. Pakai `fetch_california_housing`; bagi latih/uji dan standardisasi dari *training data*.
5. Tulis `grad_mse` dan `gradient_descent`; periksa *gradient* dengan selisih hingga dan bandingkan β dengan `np.linalg.lstsq`.
6. Gambar MSE terhadap iterasi untuk $\eta \in \{0{,}001;\ 0{,}01;\ 0{,}1;\ \text{batas stabil}\}$; cocokkan dengan $2/\lambda_{\max}$.
7. Ulangi tanpa standardisasi, lalu jalankan *mini-batch* SGD dengan $B \in \{1, 8, 32, 256, n\}$; tabelkan waktu dan MSE uji.

Yang dikumpulkan

Notebook yang dapat dijalankan ulang dari awal, beserta satu halaman pembahasan: model *regression* mana yang Anda pilih dan buktinya, mengapa η tertentu meledak, dan kompromi ukuran *batch* yang Anda lihat.

Kriteria penilaian

Protokol evaluasi tanpa kebocoran	25%
Uji statistik dan pembacaannya	15%
Implementasi GD dan uji *gradient*	30%
Analisis η dan ukuran *batch*	20%
Kerapian *notebook*	10%

Ringkasan

Topik	Pokok yang perlu diingat
Linear regression	Minimalkan RSS (setara *maximum likelihood* dengan *error* Gaussian); $\hat{\beta} = (X^\top X)^{-1} X^\top \mathbf{y}$ adalah proyeksi ortogonal. Contoh: $\hat{y} = 2{,}2 + 0{,}6x$, RSS 2,40, $R^2 = 0{,}60$, $t = 2{,}12 < 3{,}182$. Hitung dengan `lstsq`.
Error sebagai variabel acak	Selang $\mathrm{err} \pm z_N \sqrt{\mathrm{err}(1-\mathrm{err})/n}$; 12 salah dari 40 berarti 16%–44%; lebar turun sebanding $1/\sqrt{n}$.
Metric dan protokol	Pada data timpang *accuracy* menyesatkan; pilih *metric* menurut *cost*. *Test data* dikunci, *preprocessing* di dalam `Pipeline`, varian CV menurut struktur data.
Membandingkan model	*Paired t-test* pada lipatan yang sama ($t \approx 8{,}6$ nyata, $t \approx 0{,}99$ tidak); lipatan saling tumpang tindih, jadi baca p dekat 0,05 dengan hati-hati.
Evaluasi *regression*	RMSE, MAE, dan R^2_{test} (bisa negatif) terhadap *baseline* rata-rata; `KFold` untuk target kontinu.
Gradient descent	$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla L$ dengan $\nabla L = \frac{2}{n} X^\top (X\beta - \mathbf{y})$; MSE $17{,}2 \to 1{,}6464 \to \cdots \to 0{,}48$. Stabil bila $\eta < 2/\lambda_{\max}$; *condition number* besar membuatnya merangkak.
SGD dan *logistic regression*	*Mini-batch* menaksir *gradient* secara tak bias dengan *variance* $\propto 1/B$. *Logistic regression*: sigmoid, *binary cross-entropy*, *gradient* $\frac{1}{n} X^\top (\hat{\mathbf{p}} - \mathbf{y})$.

Sumber: Mitchell Bab 5; Hastie dkk. Bab 3 dan 7; James dkk. Bab 3 dan 5; Bottou, Curtis, dan Nocedal (2018).

Kuis singkat: periksa pemahaman Anda

1. Model salah pada 27 dari 150 *test example*. Hitung *sample error* dan selang 90%. 0,18; [0,129 ; 0,231]
2. $TP = 40$, $FN = 10$, $FP = 60$, $TN = 890$. Hitung *accuracy*, *precision*, *recall*, dan F_1. 0,93; 0,40; 0,80; 0,533
3. Pada contoh lima titik, ganti y_3 dari 5 menjadi 3. Hitung ulang $\hat{\beta}_1$, $\hat{\beta}_0$, RSS, dan R^2. 0,6; 1,8; 1,6; 0,69
4. Mengapa R^2 pada *test data* bisa negatif, padahal R^2 latih tidak pernah negatif?
5. Ulangi langkah pertama GD pada data jam belajar dengan $\eta = 0{,}02$. Di mana β sesudahnya? (0,16; 0,528)
6. Mengapa $\eta = 0{,}09$ meledak pada data jam belajar, sedangkan $\eta = 0{,}08$ konvergen?
7. Mengapa *logistic regression* dilatih dengan *binary cross-entropy* dan tidak dengan MSE?

Pertemuan depan

Satu neuron hanya menarik garis lurus, sehingga gagal pada XOR. Pertemuan 10 menumpuk neuron menjadi *perceptron* berlapis (MLP) dan menurunkan *backpropagation*. Bacaan: buku ajar Bab 12.

Terima kasih

Pertanyaan?