Supervised Learning

Yaya Wihardi, S.Kom., M.Kom.

Email: yayawihardi@upi.edu

Department of Computer Science Education

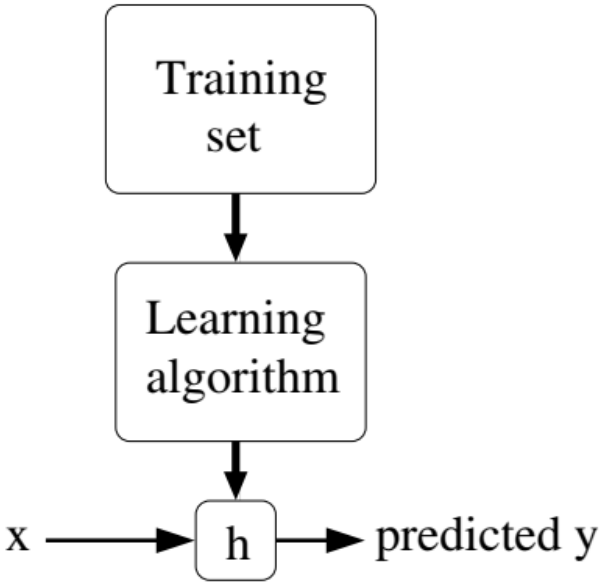
Universitas Pendidikan Indonesia

Outline

- Apa itu Supervised Learning?
- Task dalam Supervised Learning
- Notasi yang sering digunakan
- Regresi Linear
- Least Mean Square
- Gradient Descent

Apa itu Supervised Learning?

- Proses pembelajaran mesin yang mencoba membuat fungsi pemetaan dari input menjadi output berdasarkan data training



Task dalam Supervised Learning

- Regression → memprediksi nilai continue, seperti harga rumah, harga saham, dll

- Classfication → Memprediksi nilai diskrit, seperti kategori, kelas, kelompok, dll

Apakah ini foto Susi, Steak, atau Pizza?

Your Turn

- Anda diberikan history data penjualan item-item produk dalam 3 tahun terakhir, prediksi berapa jumlah penjualan setiap item produk dalam 3 bulan kedepan.
- Anda diberikan data akun dan riwayat aktifitasnya dalam sebuah sistem pemerintahan. Prediksi apakah akun-akun tersebut diretas atau tidak.
- Diberikan data email yang telah dilabeli spam atau bukan spam. Buatlah spam filter.

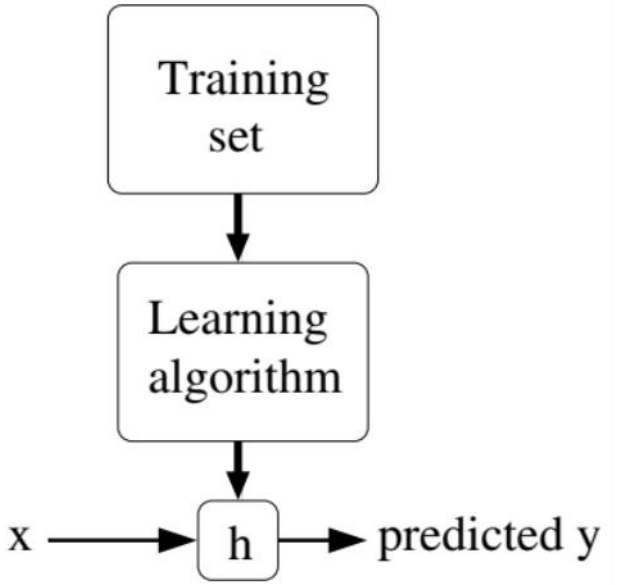
Notasi

Notasi	Keterangan
$x^{(i)}$	Variabel Input/Fitur
$y^{(i)}$	Variabel Output/Target/ Variabel yang akan diprediksi
$(x^{(i)}, y^{(i)})$	Pasangan Training Example/Instance/Data Point
$\{(x^{(i)}, y^{(i)}); i = 1, \dots, n\}$	Dataset/Himpunan Data yg digunakan untuk membangun model
$\dots^{(i)}$	(i) Menyatakan indeks dari data, **bukan pangkat**
R^n	Ruang vector berdimensi n

Regresi: Studi Kasus Real Data

- Diberikan data luas living area (feet^2), jml bedrooms, dan harganya (USD) sbb:

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮



Nilai Continue

- Buat model ML yg dapat memprediksi harga rumah berdasarkan variable input luas living area dan jml bedrooms

Regresi: Studi Kasus Real Data

- Diberikan data luas living area (feet^2), jml bedrooms, dan harganya (USD) sbb:

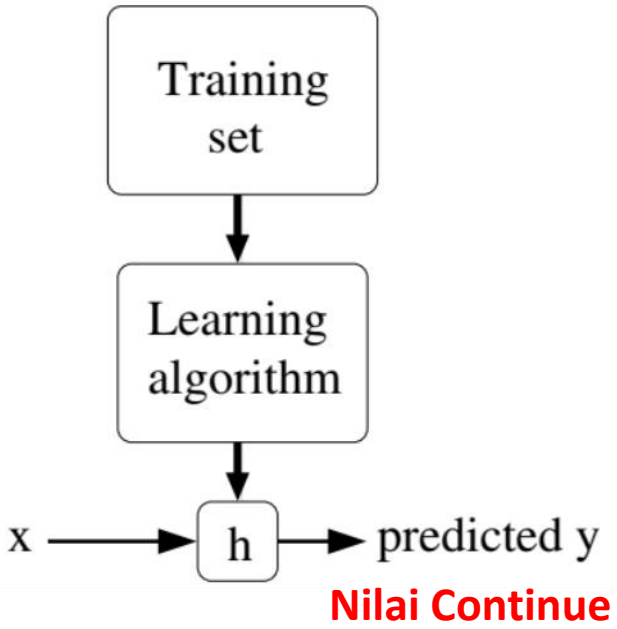
Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

- Utk kasus ini, x merupakan vector berdimensi 2 (R^2)
- $x_1^{(i)}$ menyatakan living area, $x_2^{(i)}$ menyatakan jml bedrooms

Regresi Linear: Studi Kasus Sederhana

- Diberikan data pasangan x dan y sbb:

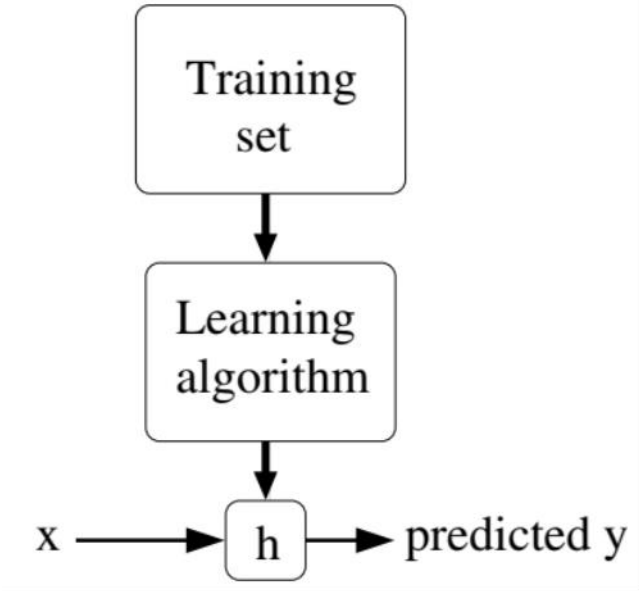
x	y
2	8
5	17
4	13
7	24
8	26
.	.
.	.
.	.



- Buat model ML yg dapat memprediksi nilai y berdasarkan input x

Caranya?

x	y
2	8
5	17
4	13
7	24
8	26
.	.
.	.
.	.



- Cari fungsi hipotesis (h) yang dapat memetakan x ke y
- Ingat prinsip occam's razor: mulailah dari yang sederhana
- Sebagai contoh, bisa digunakan fungsi persaman linear

Model Hipotesis (h)

$$h_\theta(x) = \theta_0 + \theta_1 x$$

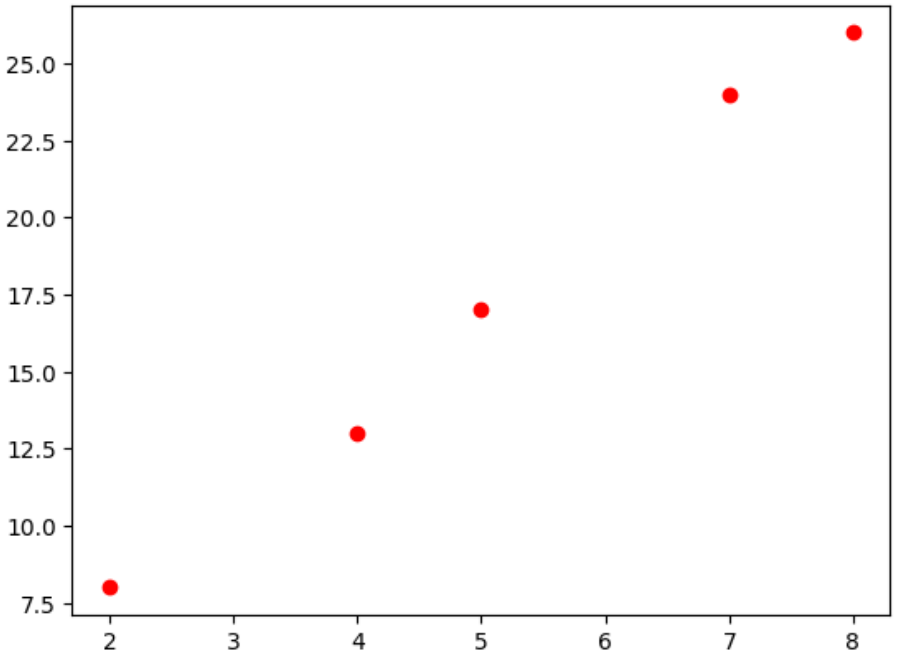
$$h_w(x) = w_0 + w_1 x$$

θ_i : Parameter model

w_i : Parameter model

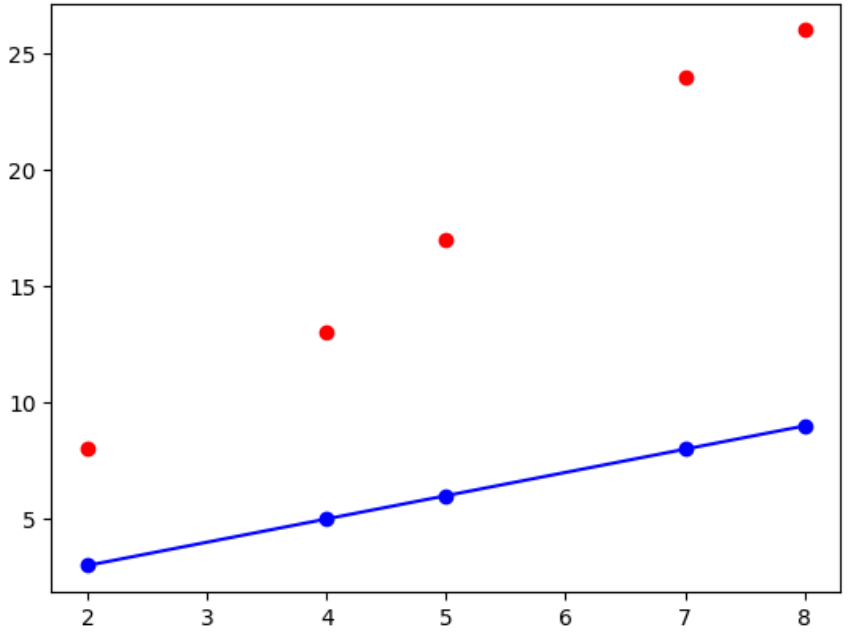
x	y
2	8
5	17
4	13
7	24
8	26
.	.

Bagaimana memilih parameter w_i/θ_i yang tepat?

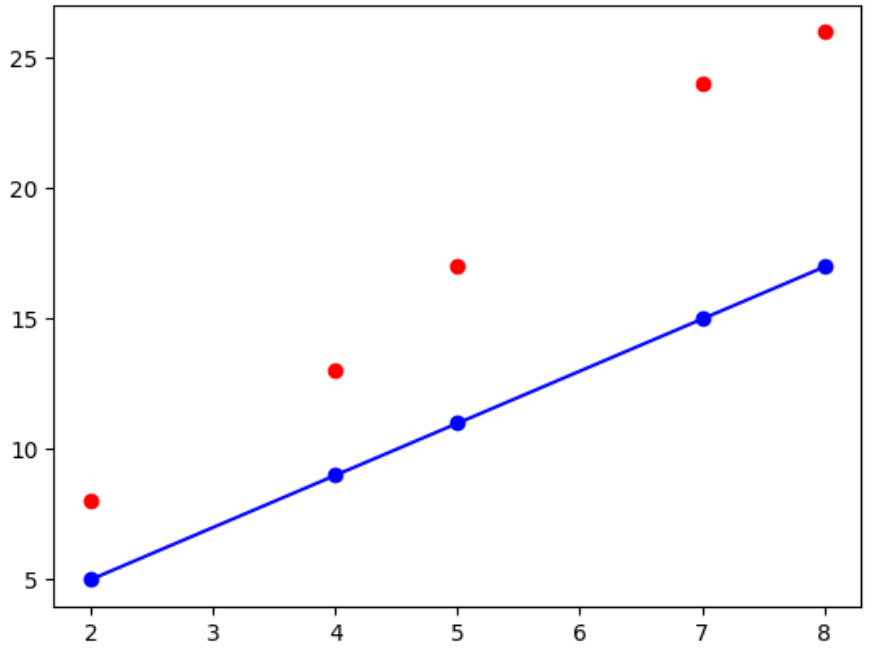


Model Hipotesis

$$h_w(x) = w_0 + w_1 x$$



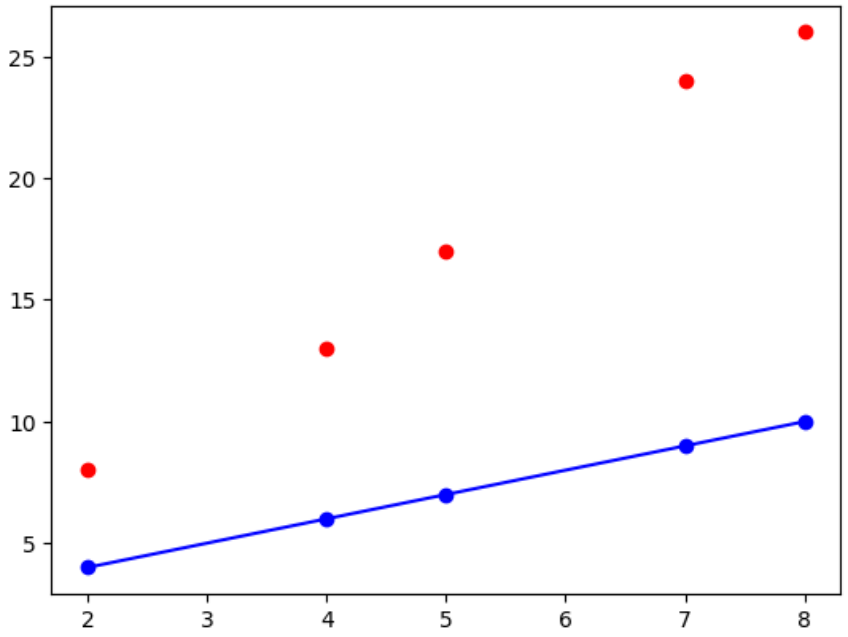
$w_0 = 1$
$w_1 = 1$



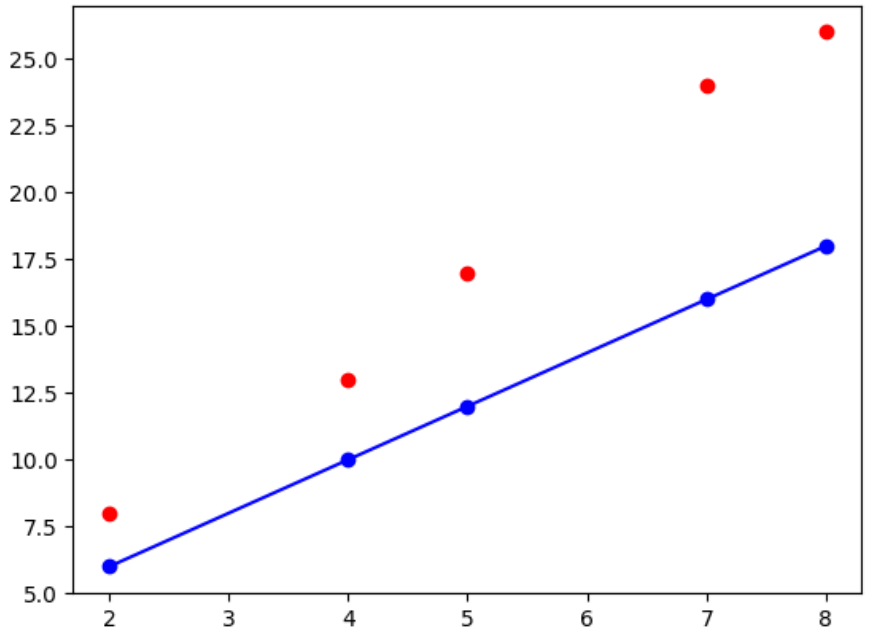
$w_0 = 1$
$w_1 = 2$

Model Hipotesis

$$h_w(x) = w_0 + w_1 x$$



$w_0 = 2$
$w_1 = 1$



$w_0 = 2$
$w_1 = 2$

Memilih Model Hipotesis (h)

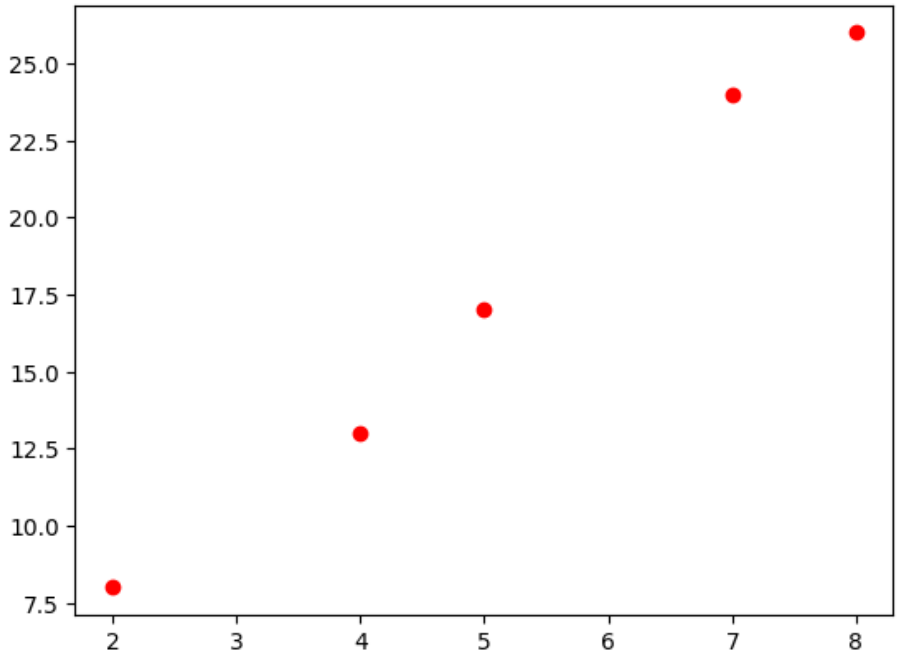
$$h_w(x) = w_0 + w_1 x$$

w_i : Parameter model

x	y
2	8
5	17
4	13
7	24
8	26
.	.

Bagaimana memilih parameter w_i yang tepat?

IDE: Pilihlah w_0, w_1 yang membuat $h_w(x)$ paling mendekati y untuk semua pasangan (x, y)



Cost Function: Least Mean Square (LMS)

Hipotesis:

$$h_w(x) = w_0 + w_1 x$$

Parameter:

$$w_0, w_1$$

Cost Function:

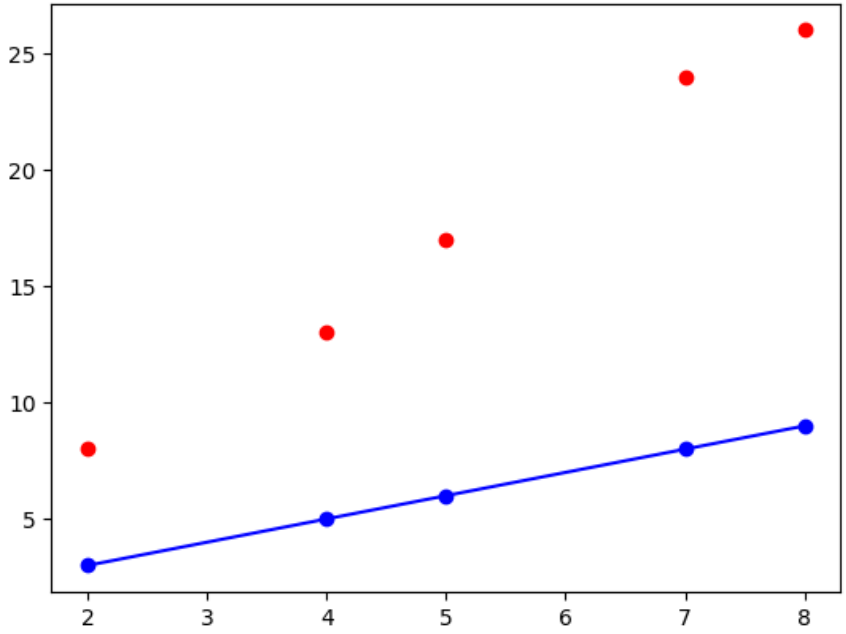
$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2$$

Goal:

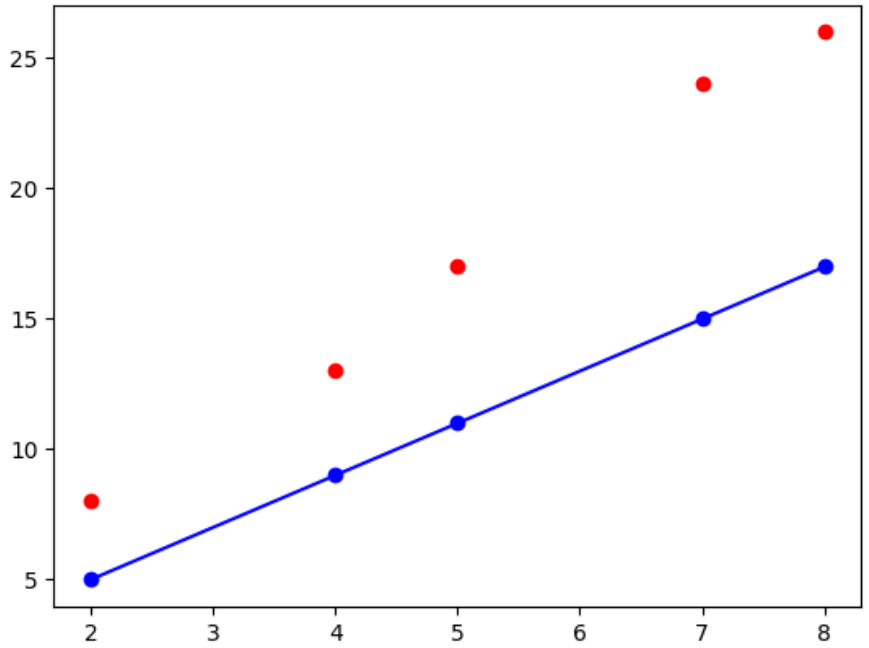
$$\underset{w_0, w_1}{minimize} \; J(w_0, w_1)$$

Model Hipotesis

$$h_w(x) = w_0 + w_1 x$$



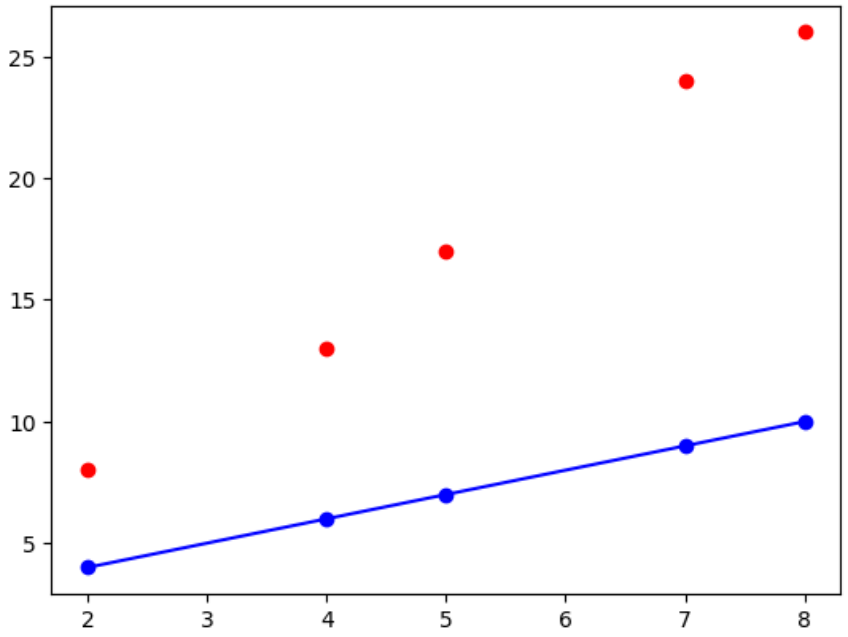
$w_0 = 1$ J =75.5
$w_1 = 1$



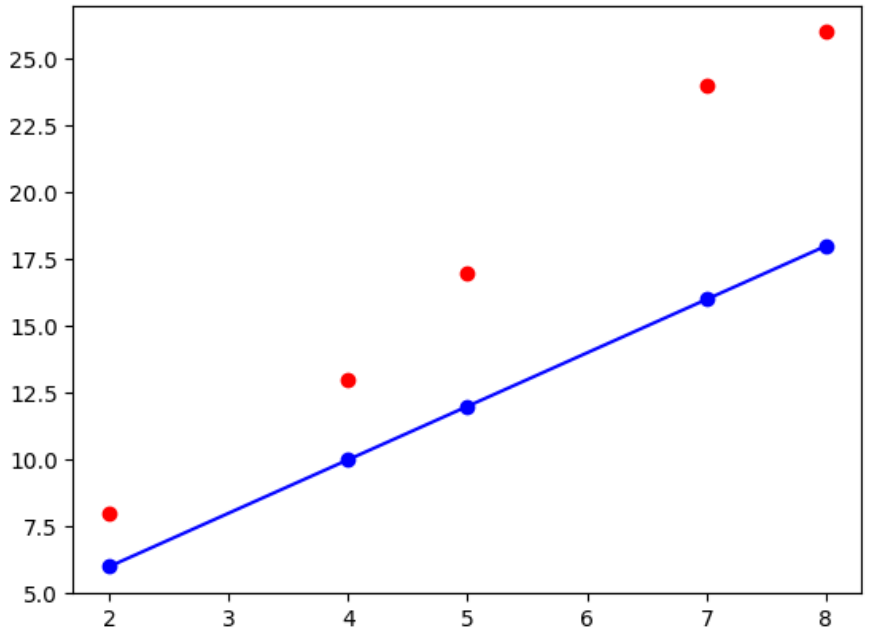
$w_0 = 1$ J =22.3
$w_1 = 2$

Model Hipotesis

$$h_w(x) = w_0 + w_1 x$$



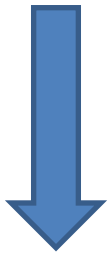
$w_0 = 2$ $J = 64.6$
$w_1 = 1$



$w_0 = 2$ $J = 16.6$
$w_1 = 2$

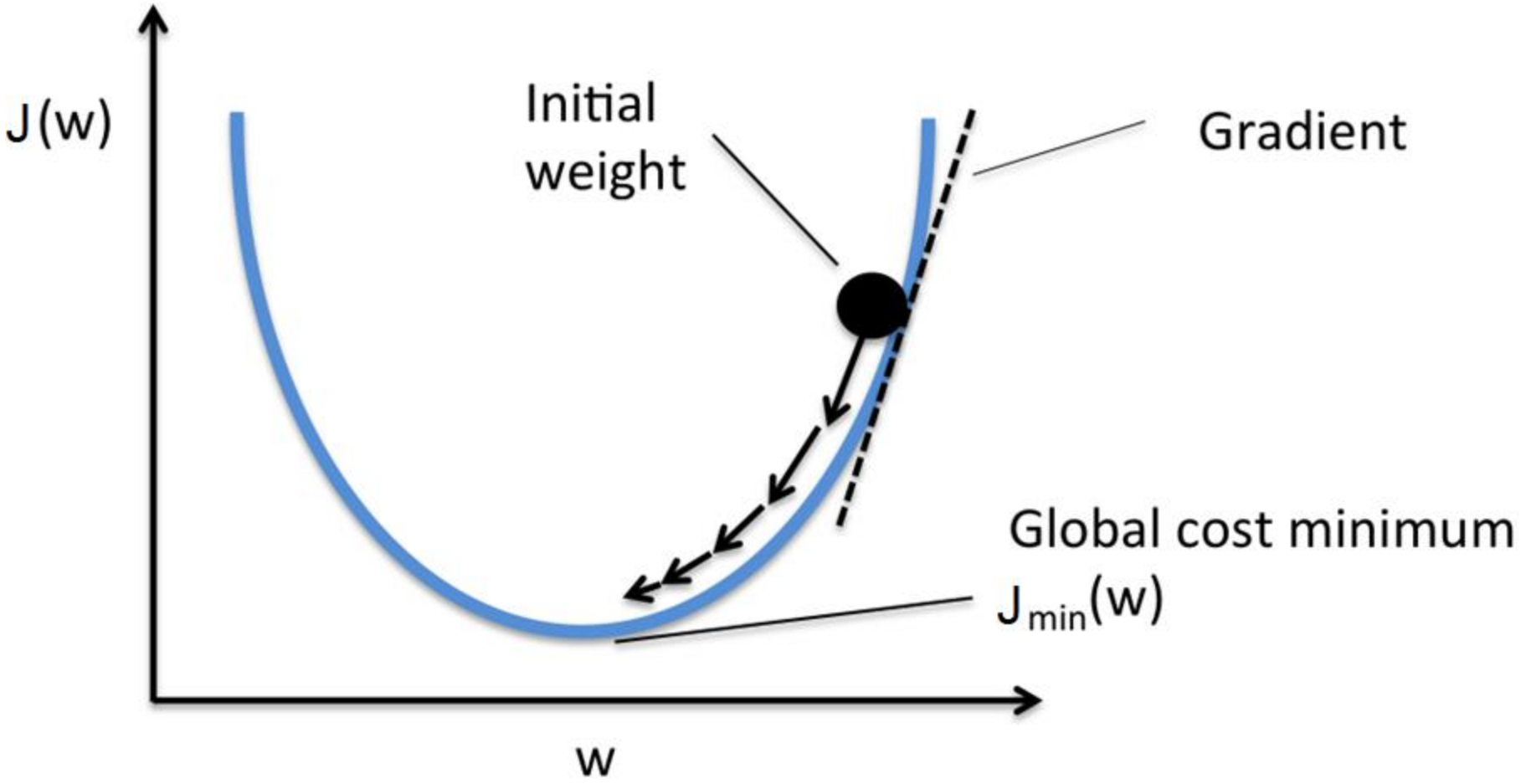
Tujuan Training

Meminimalkan Cost Function

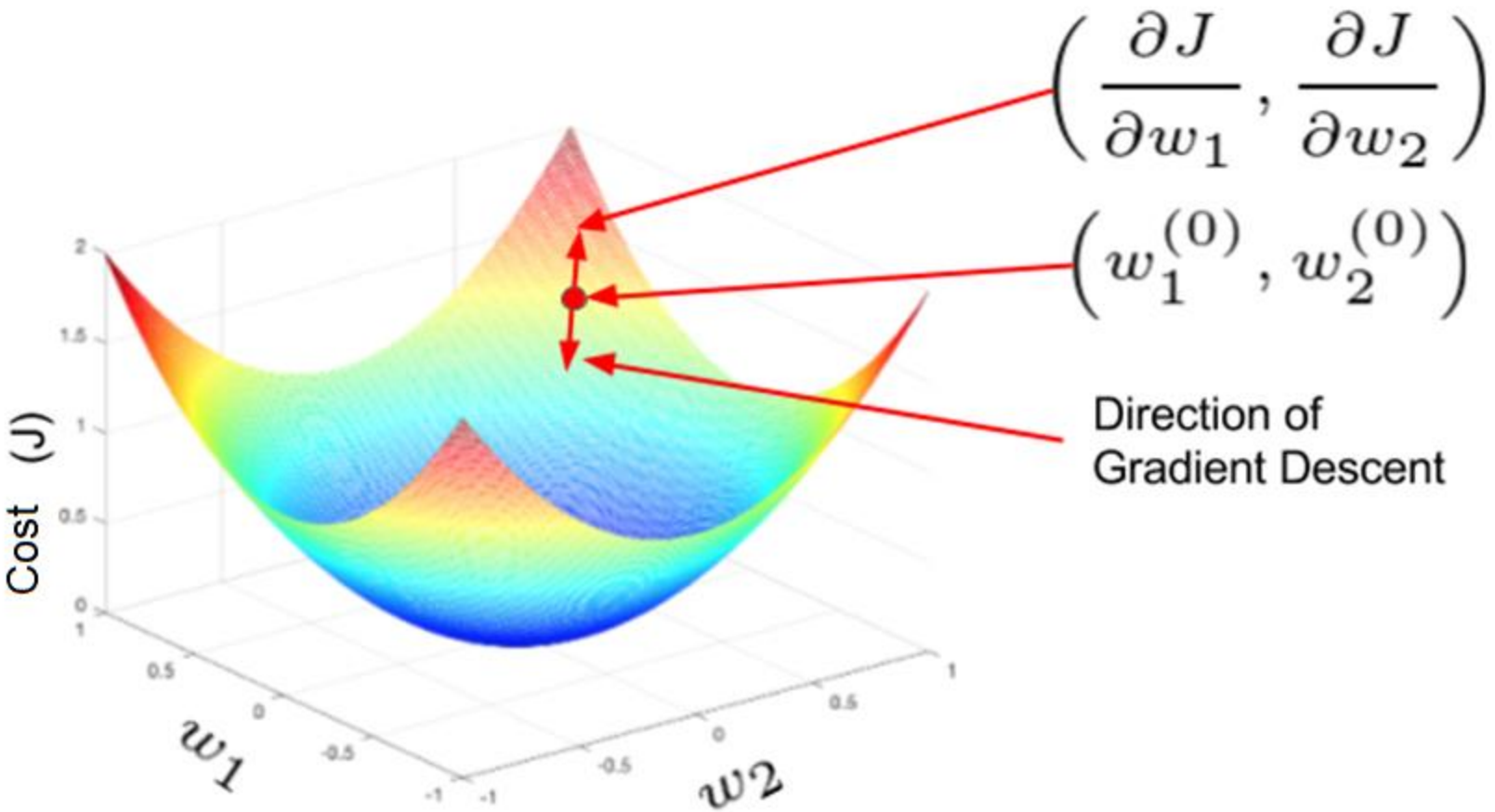


Dg Merubah Nilai Parameter Model

Gradient Descent



Gradient Descent



Proses Training

Langkah:

1. Inisialisasi w_0 dan w_1 dengan nilai random, misal:
   $w_0 = 0.1$ dan $w_1 = 0.03$
2. Ubah w_0 dan w_1 untuk menurunkan loss/error $J(w_0, w_1)$
3. Ulangi terus Langkah 2 sampai dengan didapat nilai loss yang paling minimum

Hipotesis: $h_w(x) = w_0 + w_1 x$

Parameter: w_0, w_1

Cost Function:

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2$$

Goal: $\underset{w_1, w_1}{minimize}\ J(w_0, w_1)$

Gradient Descent: Mengubah w_0 dan w_1

$$repeat\ until\ convergen\ \{$$

...

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w)$$

...

$$\}$$

Update Rule

Untuk j $= \{0, \dots, n\}$

$$temp_0 = w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$

$$temp_1 = w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$

$$w_0 = temp_0$$

$$w_1 = temp_1$$

Keterangan:

α : Learning Rate

$\frac{\partial}{\partial w_0} J(w_0, w_1)$: Partial Derivative

Gradient Descent: Mengubah w_0 dan w_1

$$repeat\ until\ convergen\ \{$$

$$\dots$$

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w)$$

Untuk j $= \{0, \dots, n\}$

$$\dots$$

$$\}$$

$$temp_0 = w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$

$$w_0 = temp_0$$

$$temp_1 = w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$

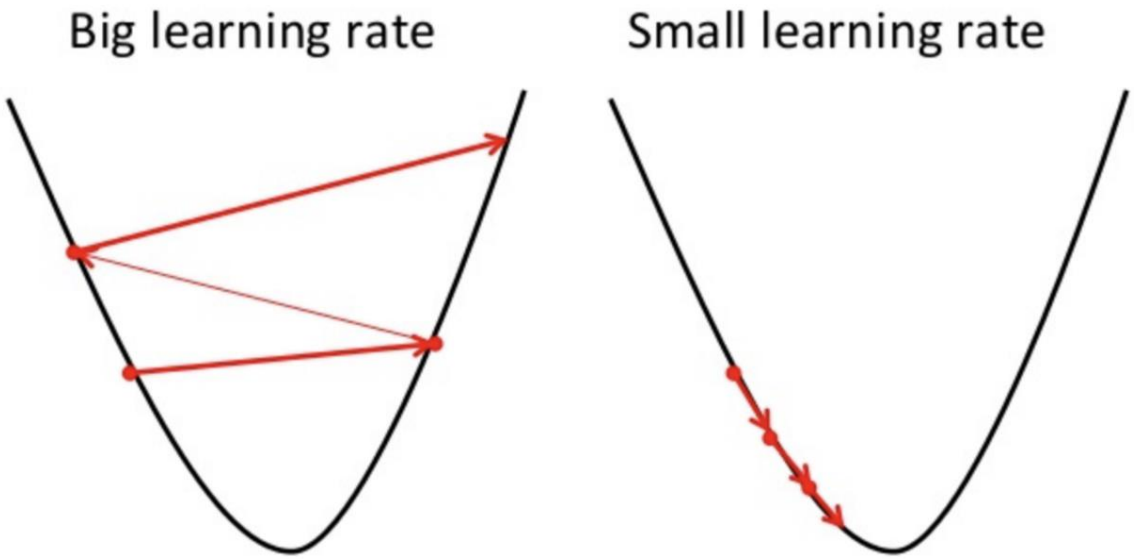
$$w_1 = temp_1$$

Keterangan:

α : Learning Rate

$\frac{\partial}{\partial w_0} J(w_0, w_1)$: Partial Derivative

Pengaruh Learning Rate



Learning rate menentukan **seberapa besar**(cepat) **perubahan pada bobot**, dkl. **step-size** dari proses update.

Studi Kasus Sederhana

Hipotesis:

$$h_w(x) = w_0 + w_1 x$$

Cost Function:

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2$$

x	y
2	8
5	17
4	13
7	24
8	26
.	.
.	.
.	.

Update Rule:

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1)$$

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2$$

Ket:
i : Index data
j : Index parameter

Partial Derivatif LMS

$$\frac{\partial J(w)}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} \frac{\partial}{\partial w_j} (h_w(x^{(i)}) - y^{(i)})^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} \frac{\partial}{\partial w_j} \left(h_w(x^{(i)}) - y^{(i)}\right) \left(h_w(x^{(i)}) - y^{(i)}\right)$$

$$= \frac{1}{2m} \sum_{i=1}^{m} 2\left(h_w(x^{(i)}) - y^{(i)}\right) \frac{\partial}{\partial w_j} \left(h_w(x^{(i)}) - y^{(i)}\right)$$

$$= \frac{1}{2m} \sum_{i=1}^{m} 2\left(h_w(x^{(i)}) - y^{(i)}\right) (x_j^{(i)}) \qquad = \frac{1}{m} \sum_{i=1}^{m} \left(h_w(x^{(i)}) - y^{(i)}\right) (x_j^{(i)})$$

$jika\ f(x) = u(x)\ v(x)$

$maka\ f'(x) = u'(x)\ v(x) + u(x)\ v'(x)$

$$h_w(x) = w_0 + w_1 x$$

Studi Kasus Sederhana

Hipotesis:

$$h_w(x) = w_0 + w_1 x$$

Cost Function:

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2$$

x	y
2	8
5	17
4	13
7	24
8	26
.	.

Update Rule:

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1) = w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_w(x^{(i)}) - y^{(i)}\right) (x_j^{(i)})$$

$$w_0 = w_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_w(x^{(i)}) - y^{(i)}\right) (x_0^{(i)})$$

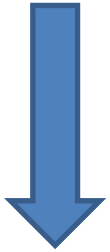
$$w_1 = w_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_w(x^{(i)}) - y^{(i)}\right) (x_1^{(i)})$$

Ket:
i : Index data
j : Index parameter

Gradient Descent

Tujuan Training

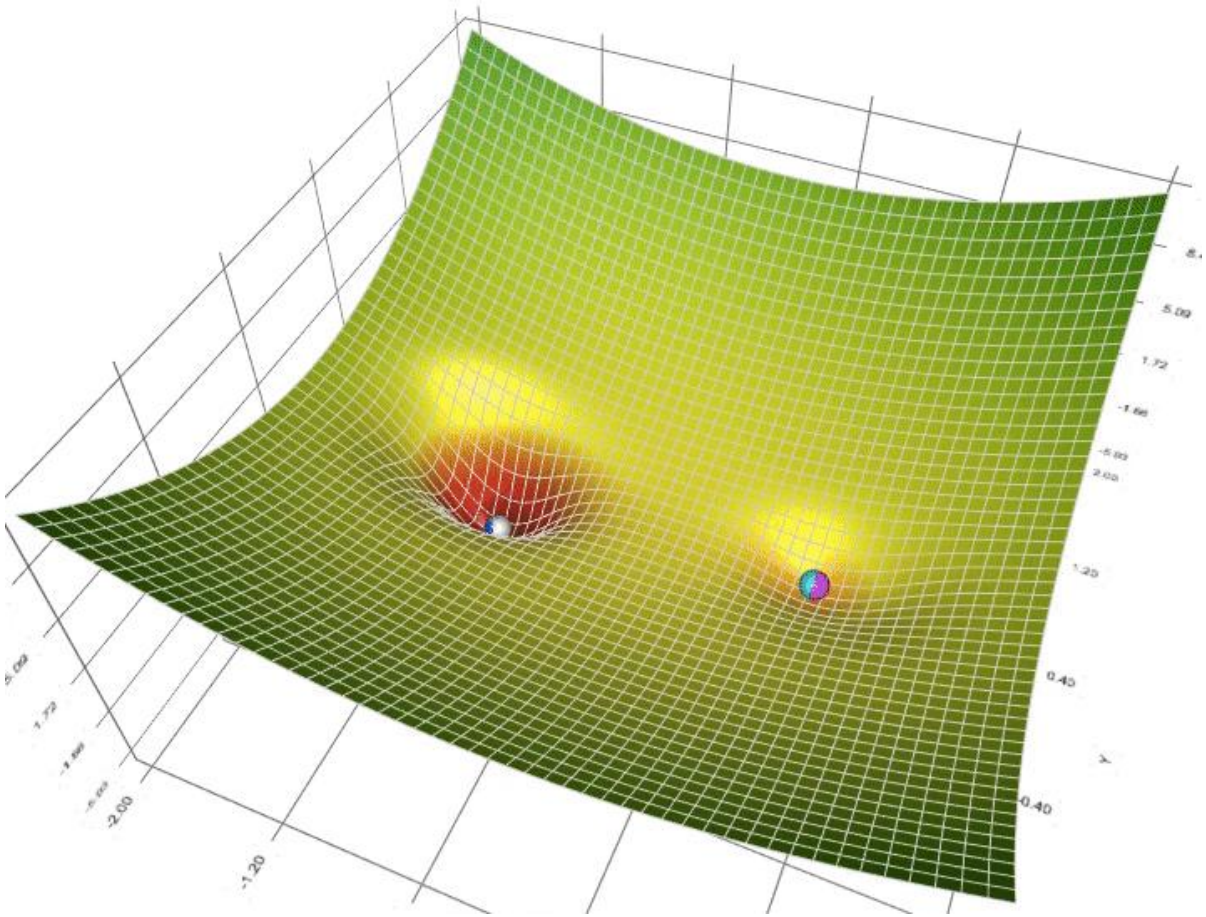
Meminimalkan Nilai Error

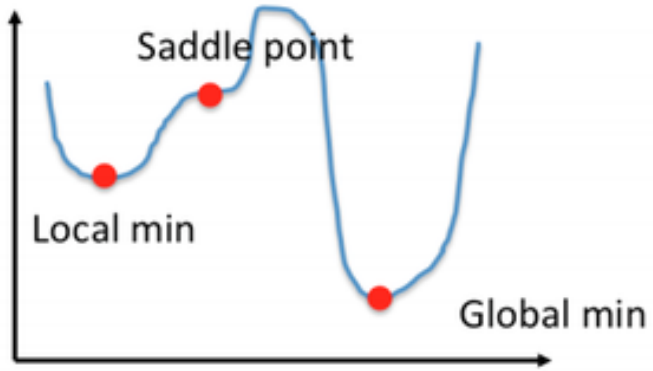


Dg Merubah Nilai Bobot

Requirement Algoritma Learning

- Menemukan global minima dengan cepat dan reliabel
- Tidak terjebak di local minima, saddle point, atau bahkan pada plateu



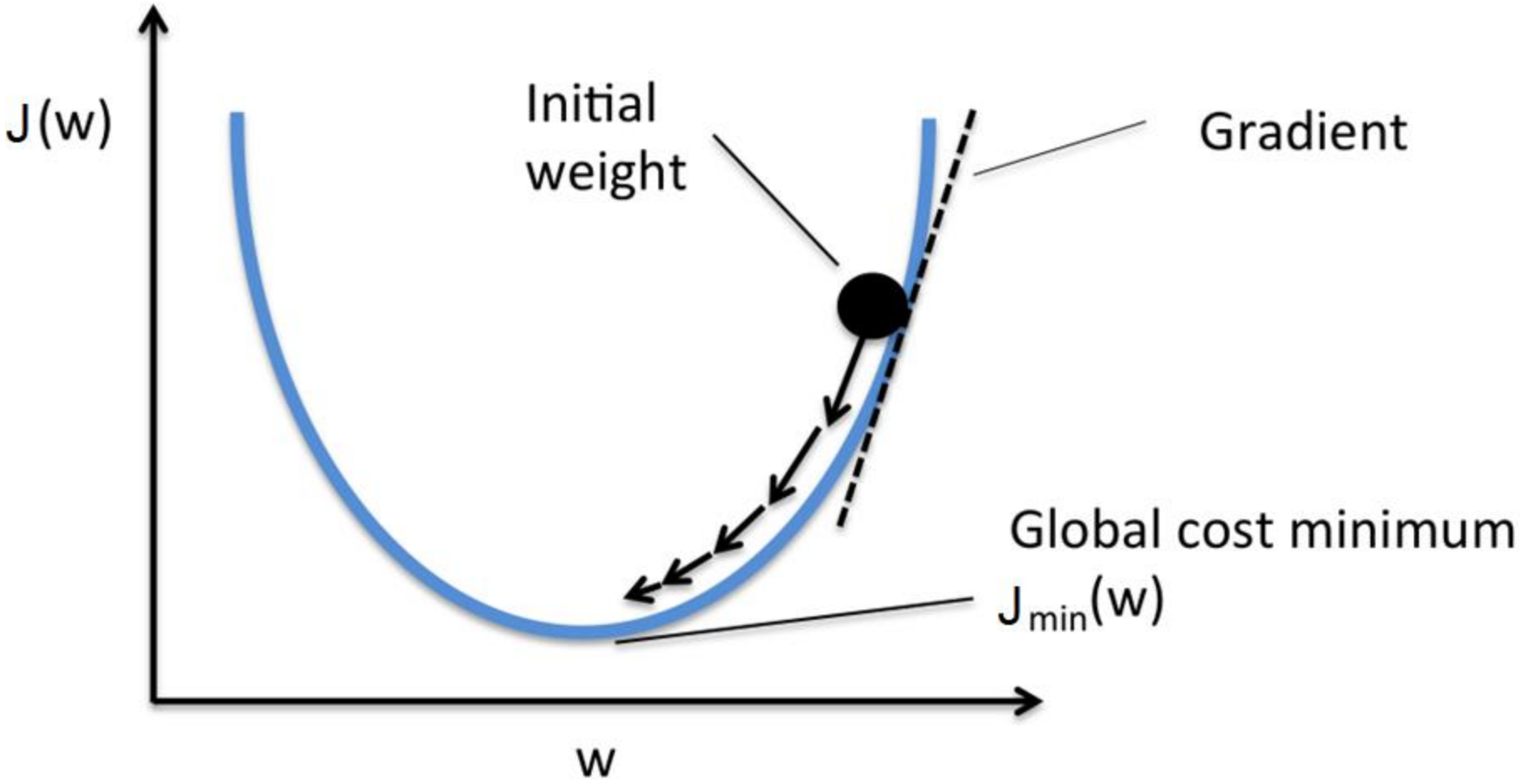


Gradient Descent

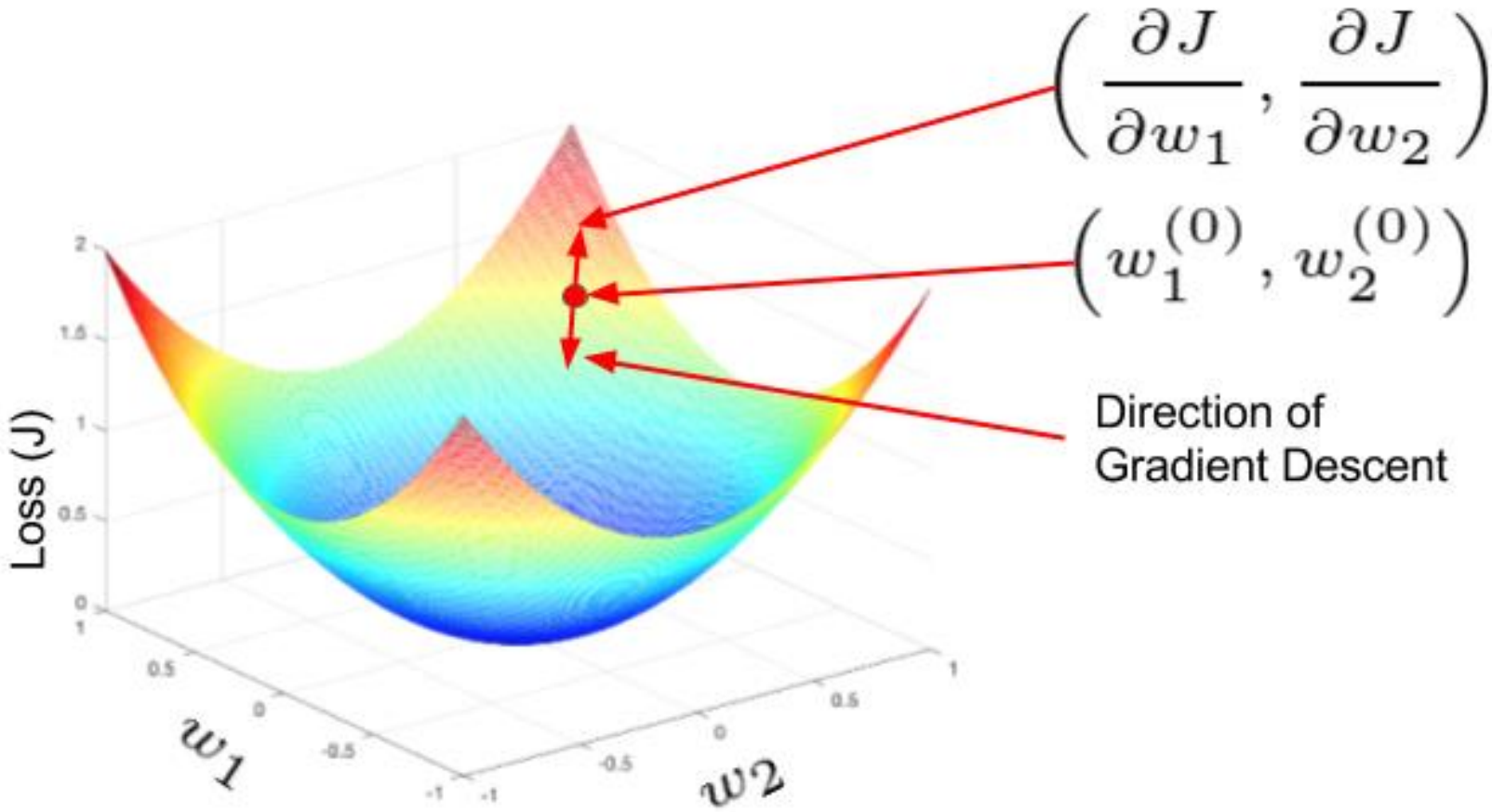
- Salah satu algoritma optimasi yg dapat digunakan utk meminimalkan fungsi error berdasarkan bobot pada JST
- **Learning rate**: factor pengali yang menentukan seberapa besar nilai update untuk suatu bobot (i.e.: 0.1)
- Satu **epoch**: Satu Langkah lengkap yang memproses seluruh data training
- Gradien menunjukan tingkat kemiringan suatu fungsi pada suatu titik tertentu (x,y), dapat dicari dengan menggunakan turunan pertama fungsi tersebut terhadap x.
- Contoh fungsi error sederhana (linear):
  Error = Y(Predicted) – T(Target/Actual)

Minimasi Fungsi Error dg Merubah Bobot

Merubah nilai initial w → Menuju titik J minimum → Dipandu oleh gradient



Minimasi: w lebih dari satu var/fitur

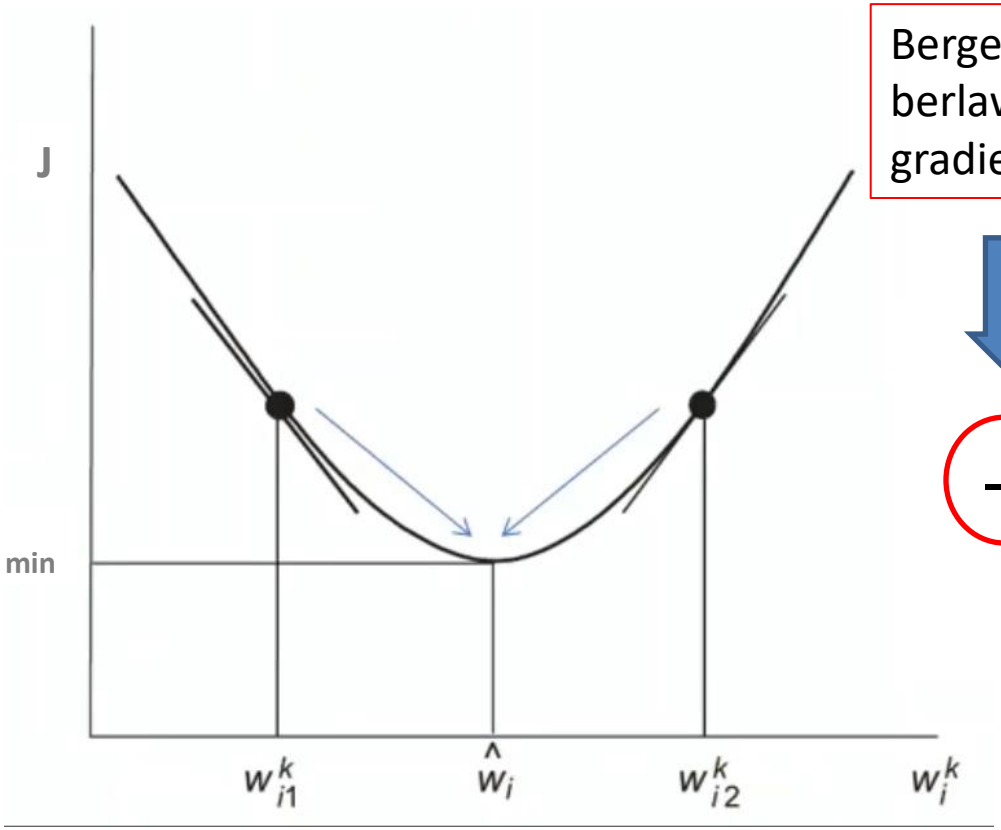


Backprop dan Updating Rule

Meminimalkan Nilai Error → **Dg Merubah Nilai Bobot**

Gradien Negatif:
(pada posisi w^k_{i1})
Jika w ditambah, maka Loss function menurun.
-(-m) = + → w harus dinaikan/ditambah (bergerak ke kanan)

Gradien Positif:
(pada posisi w^k_{i2})
Jika w ditambah, maka Loss function naik.
-(+m) = - → w harus diturunkan/dikurangi (bergerak ke kiri)

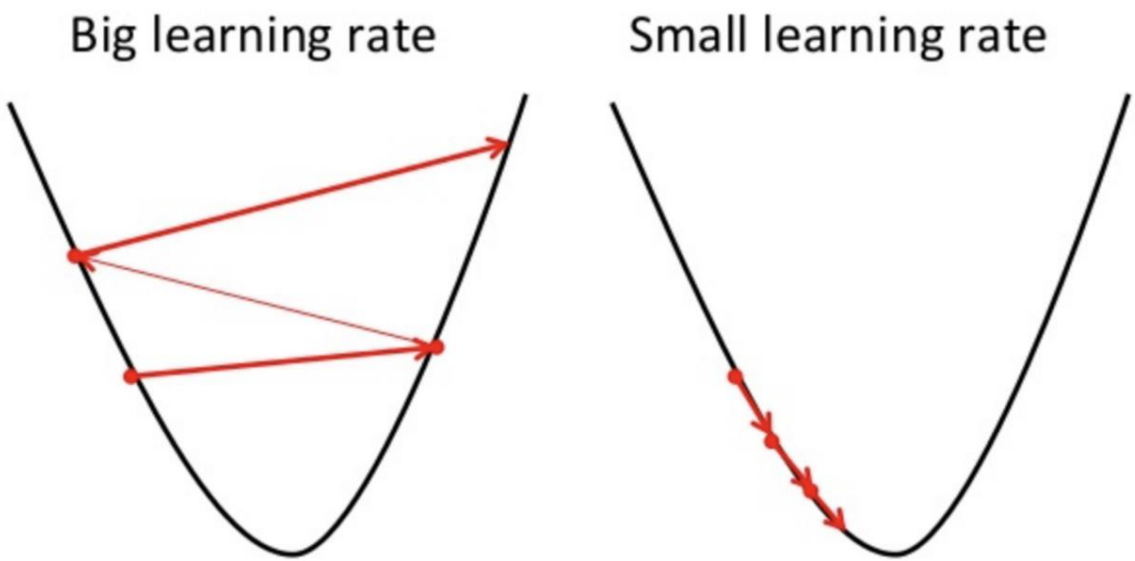


Bergerak ke arah yg berlawanan dengan gradien

$$-\frac{\partial J}{\partial w_i^k}$$

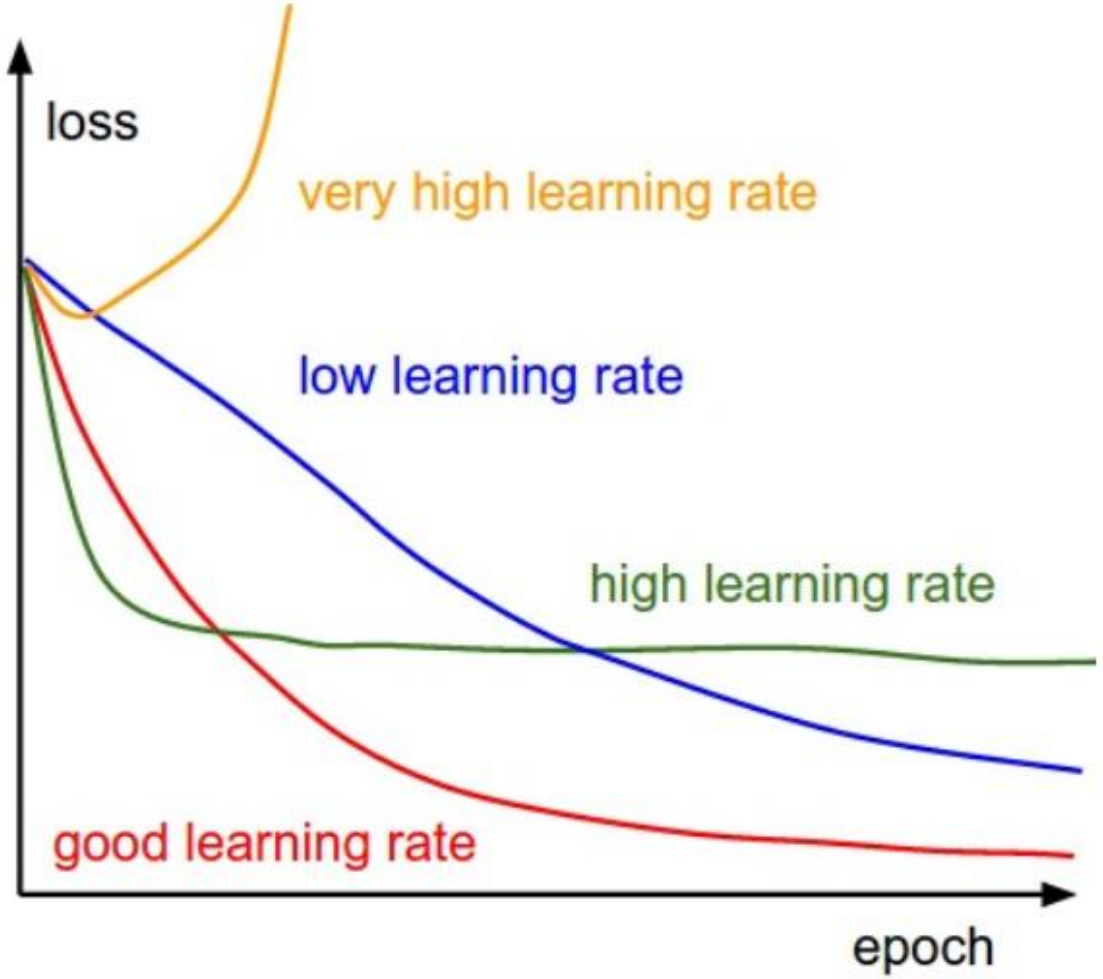
Weight Update Rule

bobot baru		bobot lama	negasi dari gradien	learning rate	gradient
w	$=$	w	$-$	α	$\frac{\partial J}{\partial w}$



Learning rate menentukan **seberapa besar**(cepat) **perubahan pada bobot**, dkl. **step-size** dari proses update.

Pengaruh Learning Rate



Learning rate yg banyak digunakan: *0.001, 0.003, 0.01, 0.03, 0.1, 0.3*

(Vanilla) Gradient Descent: Algoritma

1. Inisialisasi bobot secara random
2. Hitung output (o) dari model
3. Hitung gradient: $\frac{\partial J}{\partial w}$
4. Update bobot menggunakan Weight Update Rule
5. Cek apakah kondisi berhenti terpenuhi? Jika ya, selesai. Jika tidak, ulangi Langkah 2.

Kondisi berhenti:

- Slope ditemukan (missal: error <= 0.001)
- Maksimum epoch tercapai
- Konvergen → Error tidak berubah (flat)
- Dll.

Aturan Pembelajaran

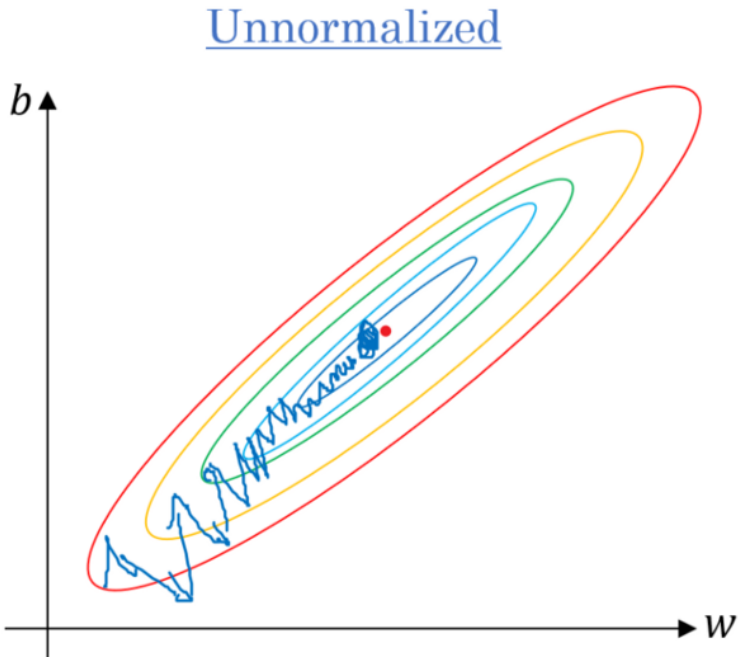
$$\text{Gradient} \quad \nabla J[\vec{w}] \equiv \frac{\partial J}{\partial w} \equiv \left[\frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}, \ldots, \frac{\partial J}{\partial w_n}\right]$$

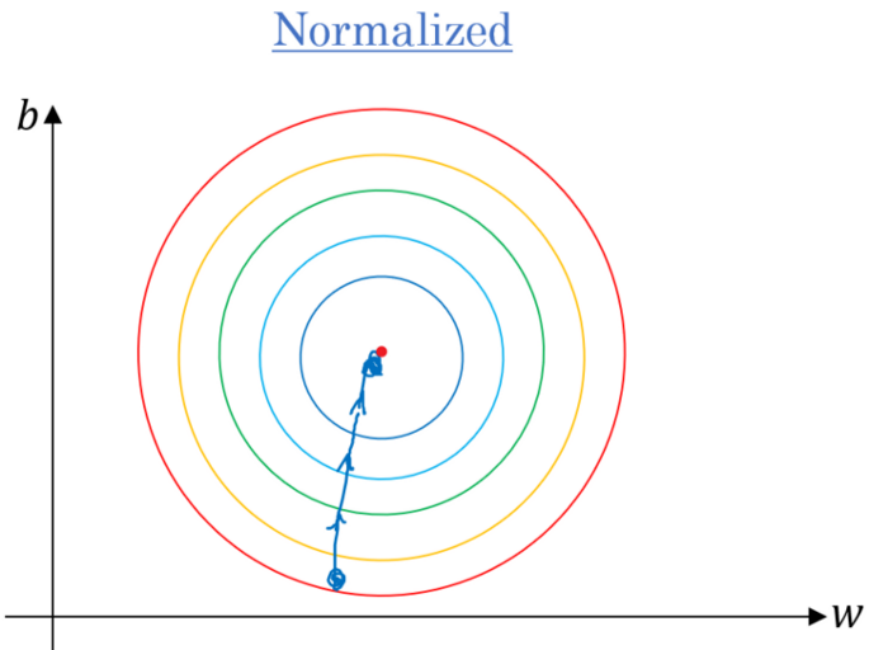
$$\text{Training rule:} \ \Delta w = -\alpha \nabla J[\vec{w}]$$

$$maka \qquad \Delta w_i = -\alpha \frac{\partial J}{\partial w_i}$$

Normalisasi Data

- Data yang digunakan harus dinormalisasi
- Pada GD ini cocok digunakan scaling data menjadi range antara 0 s.d. 1





Strategi Update Bobot

- Variasi: Berbeda pada jumlah data yg digunakan untuk menghitung gradient pada setiap epoch-nya.
- Trade-off: Akurasi vs Waktu Eksekusi
- Variasi Strategi:
  - Batch Gradient Descent
  - Mini-batch Gradient Descent
  - Stochastic Gradient Descent

Batch Gradient Descent

- Untuk melakukan update bobot, harus menggunakan semua data dalam proses perhitungan gradient-nya

```
for i in range(num_epochs):
    grad = compute_gradient(data, bobot)
    bobot = bobot – learning_rate * grad
```

- Dalam setiap epoch, hanya sekali melakukan update bobot
- **Kelebihan**:
  - Stabil, convergence
  - Grafik fungsi loss lebih smooth
- **Kekurangan**:
  - Butuh memory yang besar
  - Waktu yang dibutuhkan lama, apalagi datanya banyak

Mini-Batch Gradient Descent

- Untuk melakukan update bobot, hanya menggunakan sebagian data (disebut **batch**) dalam proses perhitungan gradient-nya

```
for i in range(num_epochs):
    np.random.shuffle(data)
    for batch in radom_minibatches(data, batch_size=32):
        grad = compute_gradient(batch, bobot)
        bobot = bobot – learning_rate * grad
```

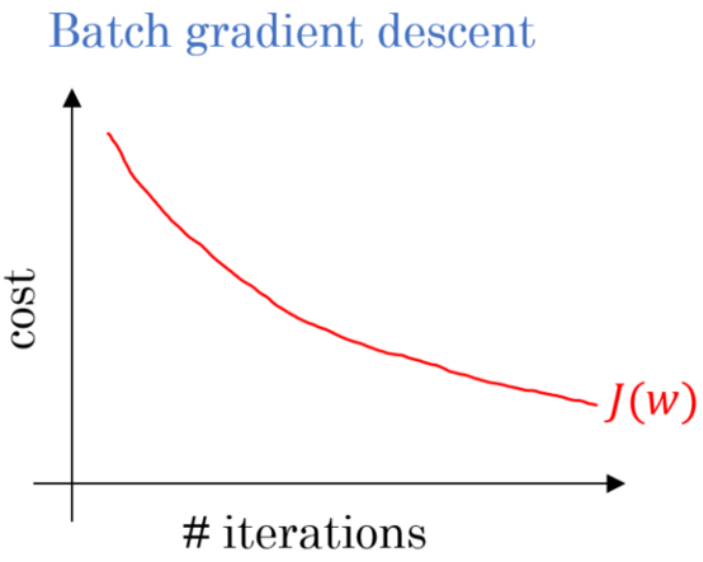
- Setiap epoch, dilakukan (**jml / jml-per-batch**) kali update bobot
- Misal jml data = 10.000, jml per batch = 32, maka dalam setiap epoch dilakukan 10.000/32 = 313 kali update
- Jika jml data dibagi jml data per batch tidak bulat, maka sisa data pada partisi terakhir dianggap satu batch
- Jika jml data per batch sama dengan jml data, maka sama dg batch-GD

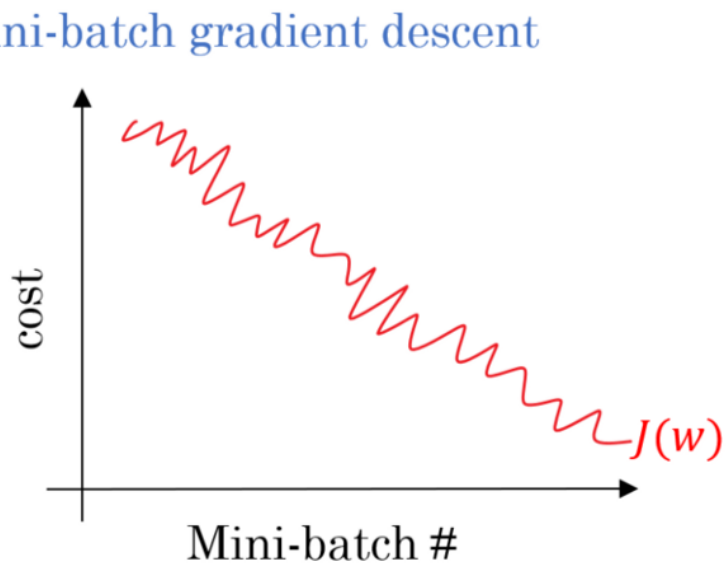
Mini-Batch Gradient Descent

- **Kelebihan**:
  - Lebih cepat waktu belajarnya dibanding dg batch GD
  - Ada tambahan noise, sehingga meningkatkan generalization
  - Lebih ramah memory, karena dalam satu proses hanya mengolah sejumlah kecil data

- **Kekurangan**:
  - Sulit convergence, karena ada tambahan noise
  - Grafik Loss leih fluktuatif

Batch vs Mini-Batch GD





Berapa Batch Size yang Tepat?

Jml data per batch biasanya merupakan hasil perpangkatan dengan basis 2: 32, 64, 128, 256, 512, dll, karena GPU performanya lebih optimal pada nilai-nilai tsb.

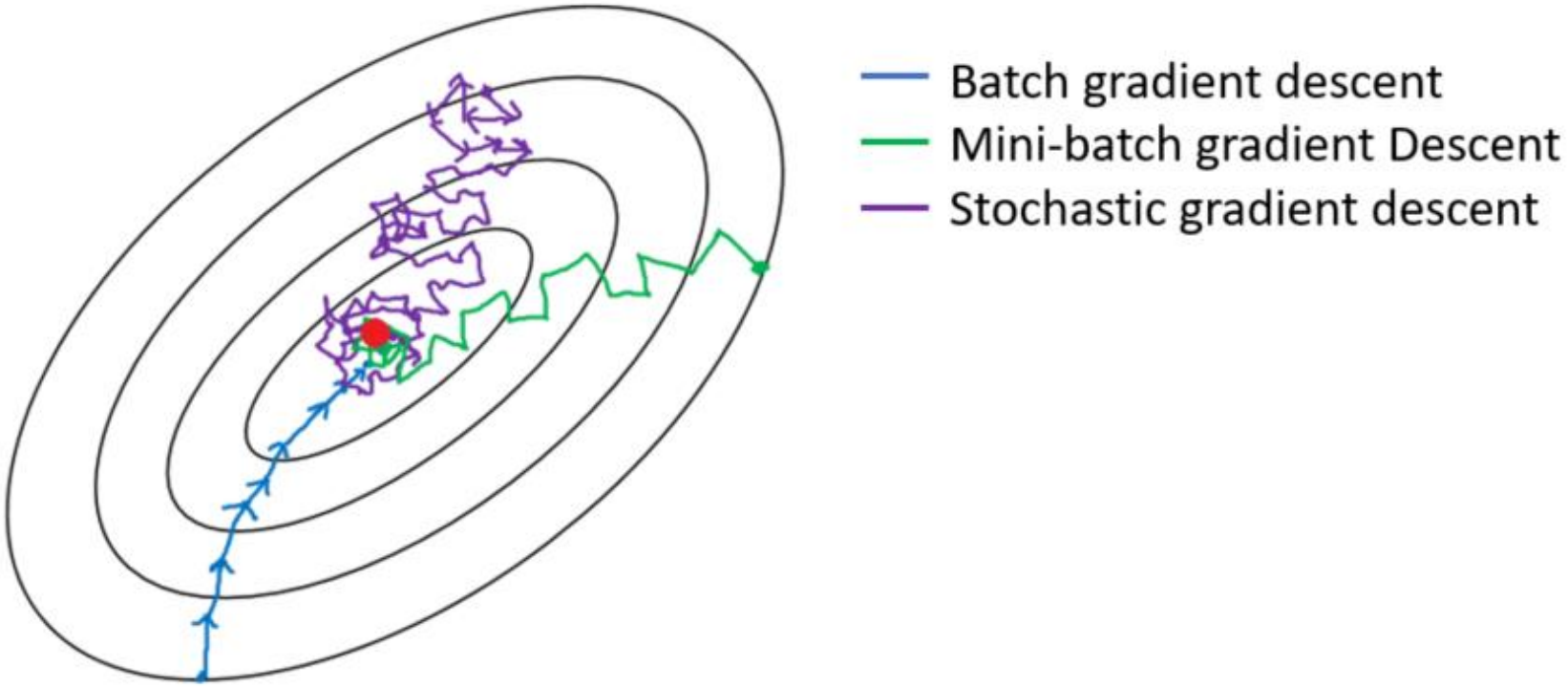
Stochastic Gradient Descent

- Update bobot dilakukan untuk setiap data, artinya gradient dihitung untuk setiap data sample

```
for i in range(num_epochs):
    np.random.shuffle(data)
    for example in data:
        grad = compute_gradient(example, bobot)
        bobot = bobot – learning_rate * grad
```

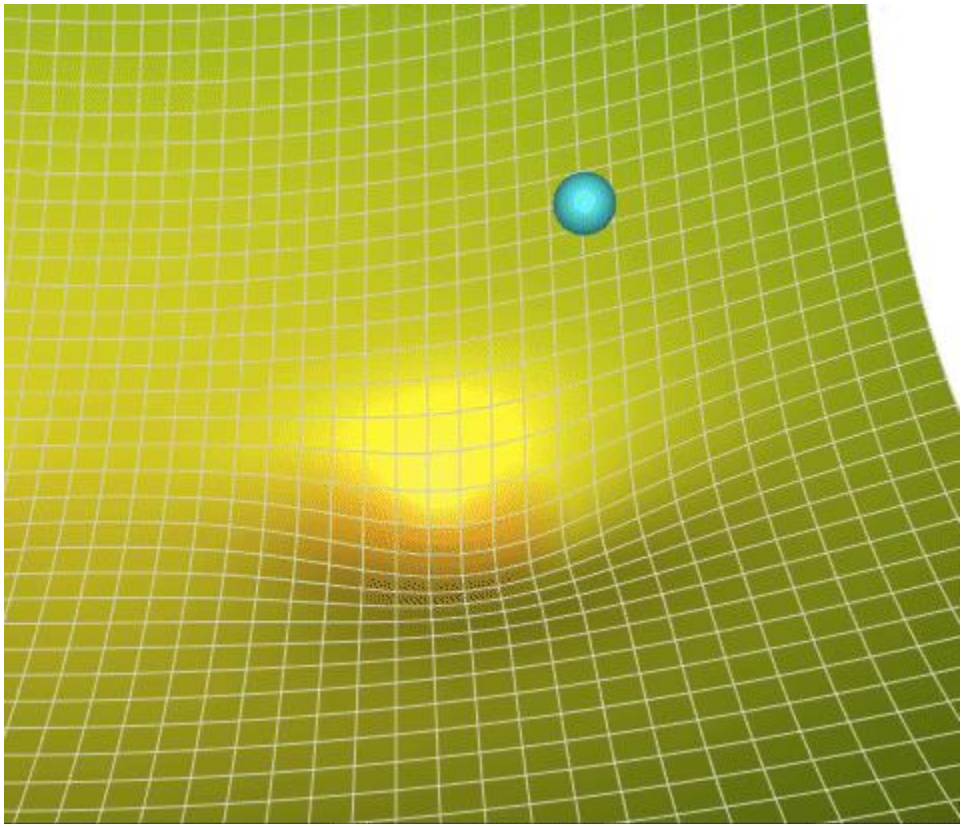
- Setiap epoch, dilakukan update bobot sebanyak jumlah data
- Hemat memory, komputasi cepat, efektif utk jml data yang besar
- Namun, noisenya sangat tinggi

Perbandingan Batch, Mini-Batch, dan Stochastic



Gradient Descent Improvements

Vanilla Gradient Descent lambat dan sering terjebak pada local minima



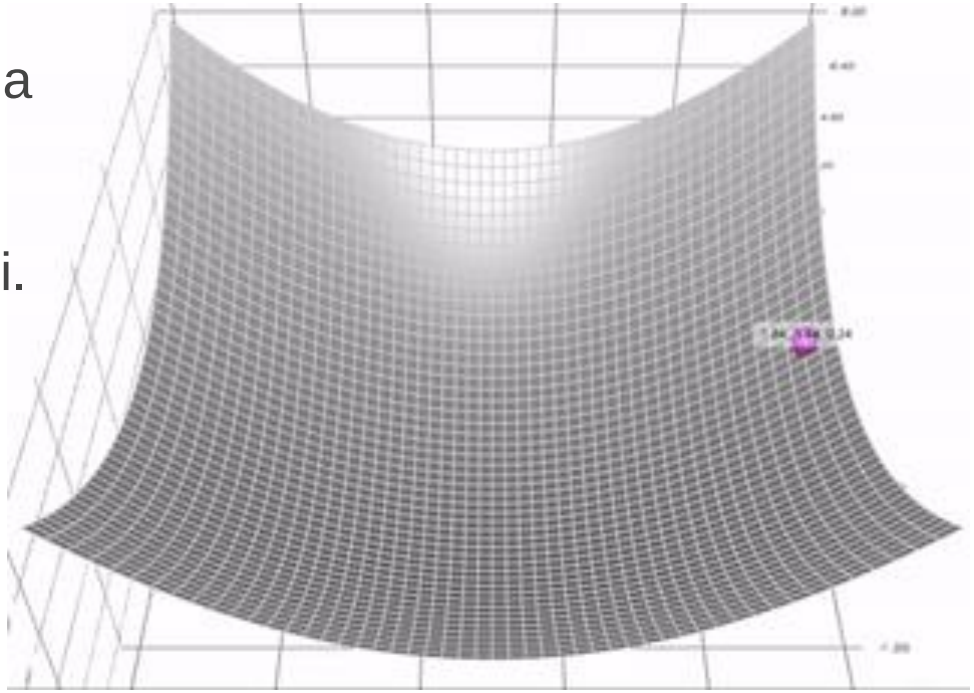
Gradient Descent Improvements

- Beberapa upaya peningkatan GD:
  - Momentum Based Gradient (Momentum)
  - Adaptive Gradient (AdaGrad)
  - Root Mean Square Propagation (RMSProp)
  - Adaptive Moment Estimation (Adam)

Momentum Based Gradient (Momentum)

- Menerapkan konsep momentum dalam ilmu Fisika
- Momentum adalah besaran yang dihasilkan dari jumlah perkalian skala massa benda dan vektor kecepatan gerak benda

  p = m v
- Bayangkan sebuah bola digelindingkan pada sebuah mangkuk dengan zero gesekan,
  dengan adanya momentum, bola tersebut alih-alih berhenti pada cekungan, malah akan terus bergerak bolak-balik tanpa henti.
- Konsep ini dapat diterapkan pada vanilla GD



Momentum Based Gradient (Momentum)

- Vanilla GD:

  delta = - (learning_rate * gradient)

  bobot = bobot + delta

- Momentum

  delta = - (learning_rate * gradient)+ **previous_delta * decay_rate**

  bobot = bobot + delta

- Faktorisasi:

  sum_of_grad = gradient + previous_sum_of_grad * decay_rate

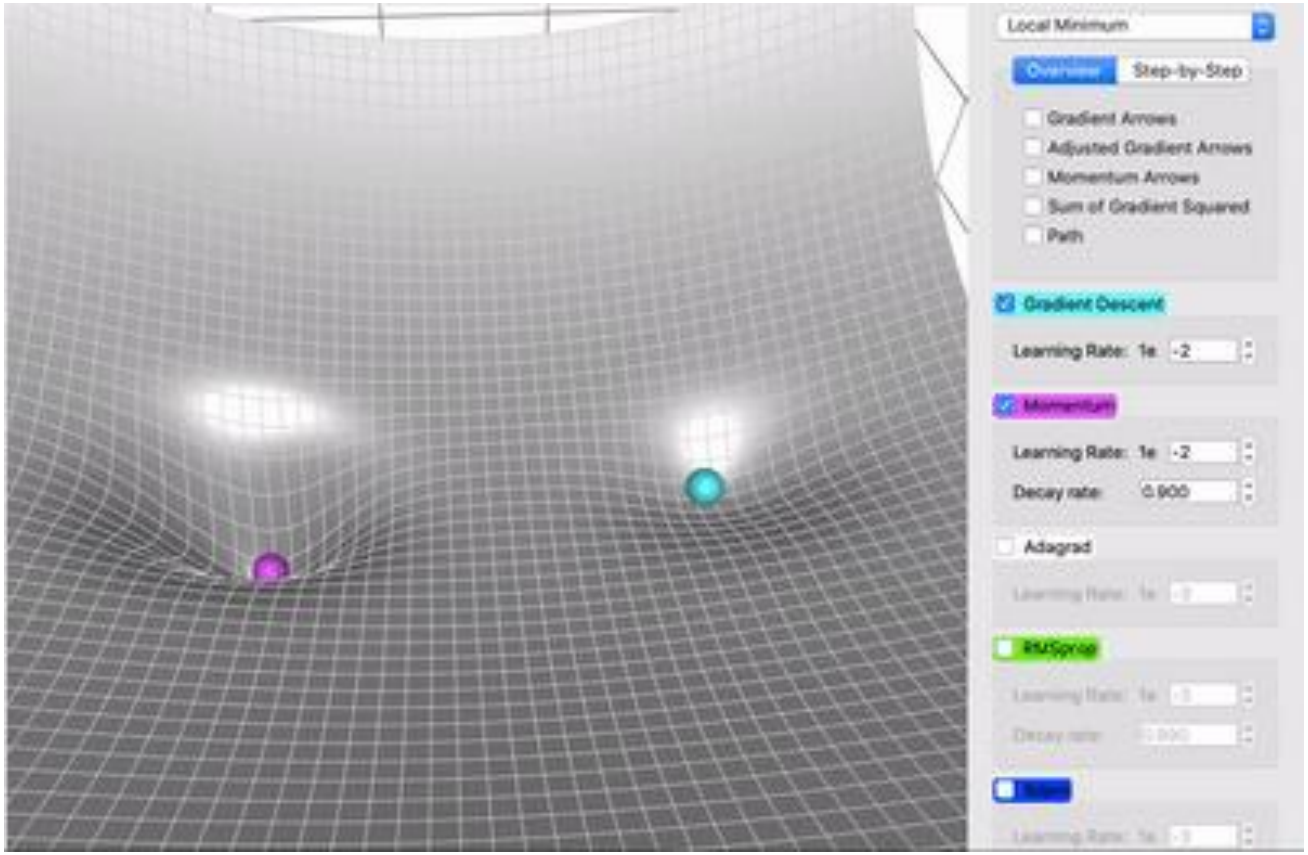
  delta = -learning_rate * sum_of_grad

  bobot = bobot + delta

- decay_rate = 0 → Vanilla Gradient Descent
- decay_rate = 1 → Tak akan pernah berhenti
- decay_rate ideal → 0.8 s.d. 0.9 → perlahan akan berhenti

Momentum Based Gradient (Momentum)

- Momentum membuat pergerakan menjadi lebih cepat
- Mampu menghindari/keluar dari jebakan local minima
- Momentum (magenta) vs. Gradient Descent (cyan):



Adaptive Gradient Descent (AdaGrad)

- Ada kalanya fitur yang digunakan bersifat sparse
- Rataan dari gradient fitur yang sparse cenderung kecil, berakibat pada lamanya proses training, karena pergerakannya lambat.
- AdaGrad mengatasi ini dengan ide: Semakin besar update pada bobot, makan akan semakin kecil update selanjutnya, hal ini memberi kesempatan pada fitur yang sparse untuk melakukan penyesuaian.
- Pada implementasinya, AdaGrad menggunakan akumulasi kuadrat gradient untuk memandu arah pergerakan ini.
- Hal ini membuat AdaGrad (dan variasinya) mampu keluar dari jebakan saddle point.
- AdaGrad:

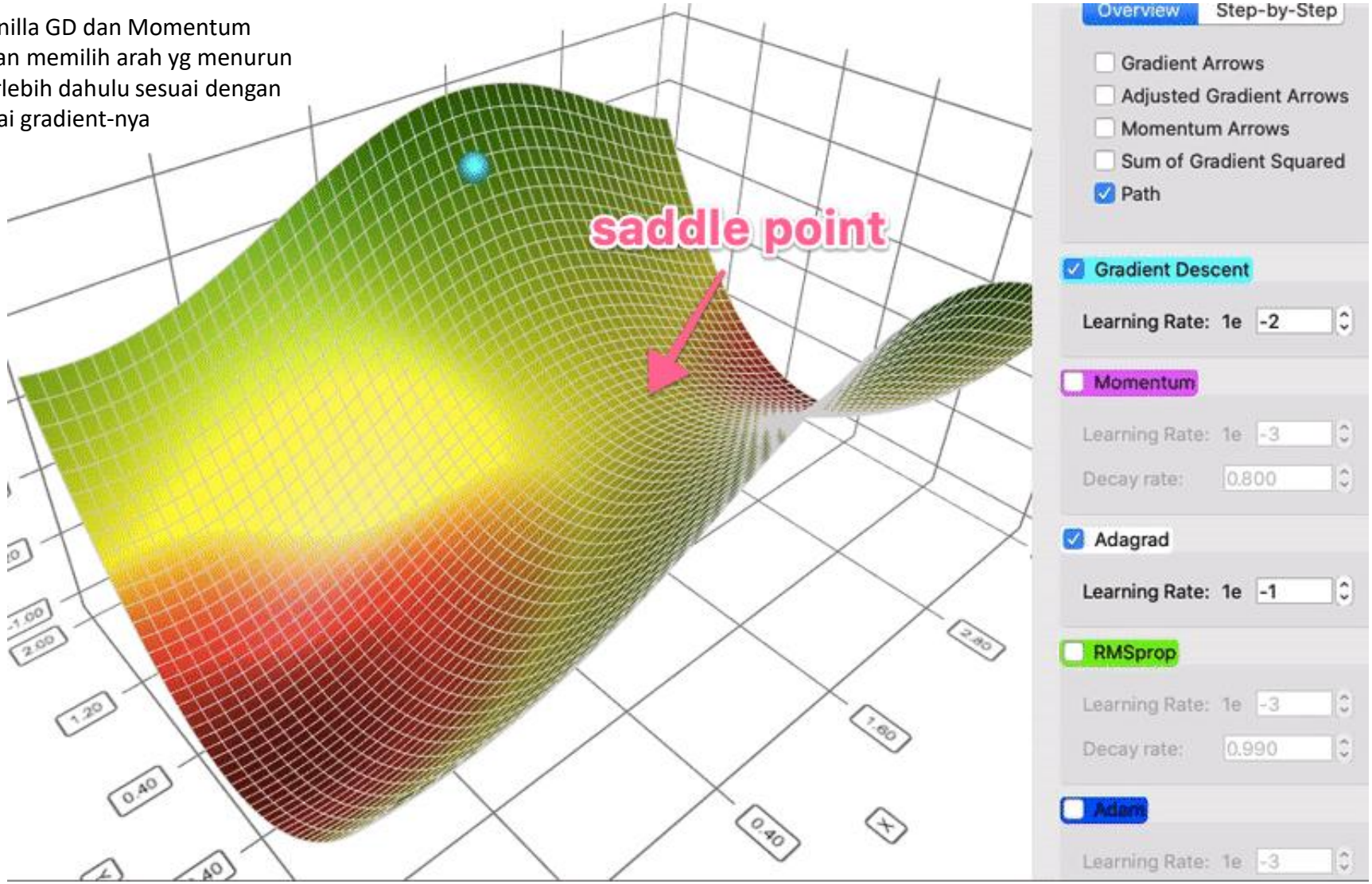
sum_of_grad_squared = previous_sum_of_grad_squared + **gradient^2**

delta = -learning_rate * gradient / sqrt(sum_of_grad_squared)

bobot = bobot + delta

AdaGrad (white) vs. gradient descent (cyan)

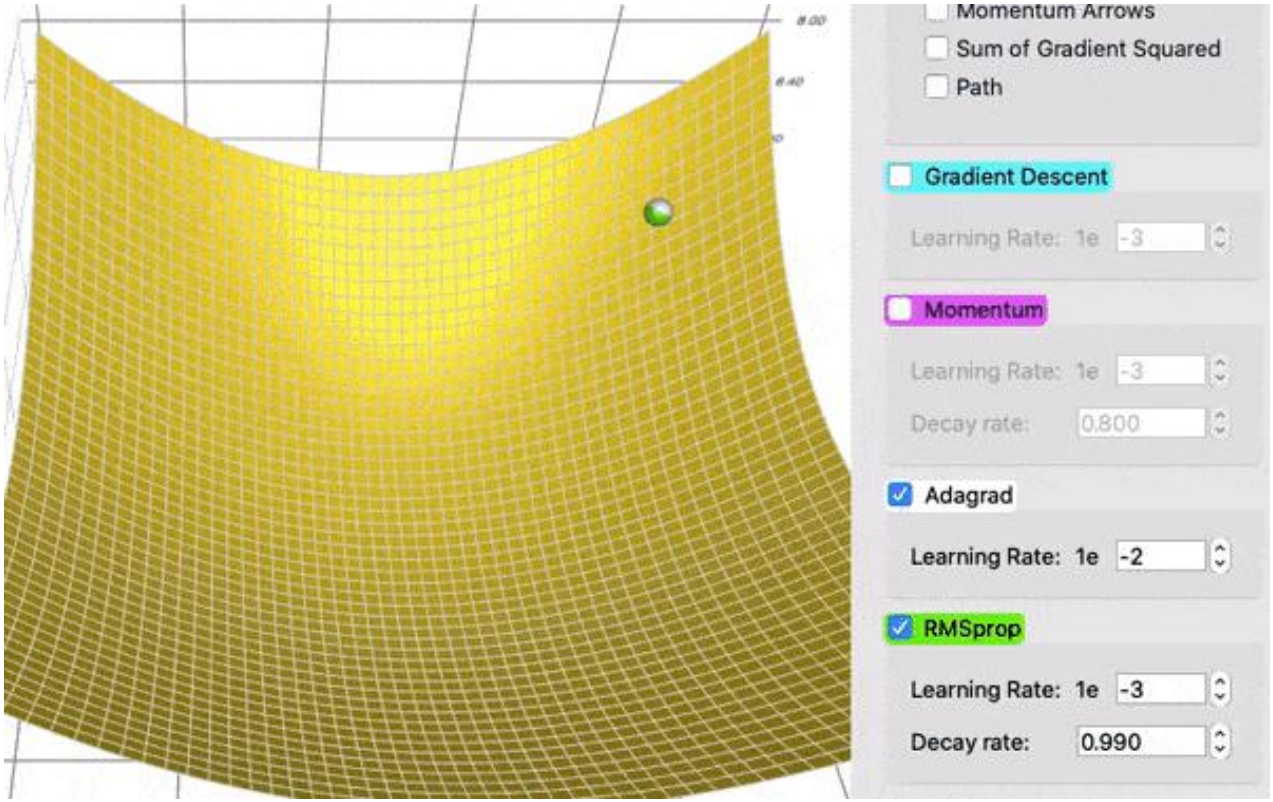
Vanilla GD dan Momentum akan memilih arah yg menurun terlebih dahulu sesuai dengan nilai gradient-nya



Root Mean Square Propagation (RMSProp)

- Kekurangan dari AdaGrad adalah lambat, karena sum_of_gradient _squared hanya akan bertambah terus menerus tanpa penurunan.
- RMSProp mengatasinya dengan menambahkan decay_rate

RMSProp (green)
vs AdaGrad (white)



- RMSProp:

sum_of_grad_squared = previous_sum_of_grad_squared * decay_rate + gradient^2 * (1-decay_rate)

delta = -learning_rate * gradient/sqrt(sum_of_grad_squared)

bobot = bobot + delta

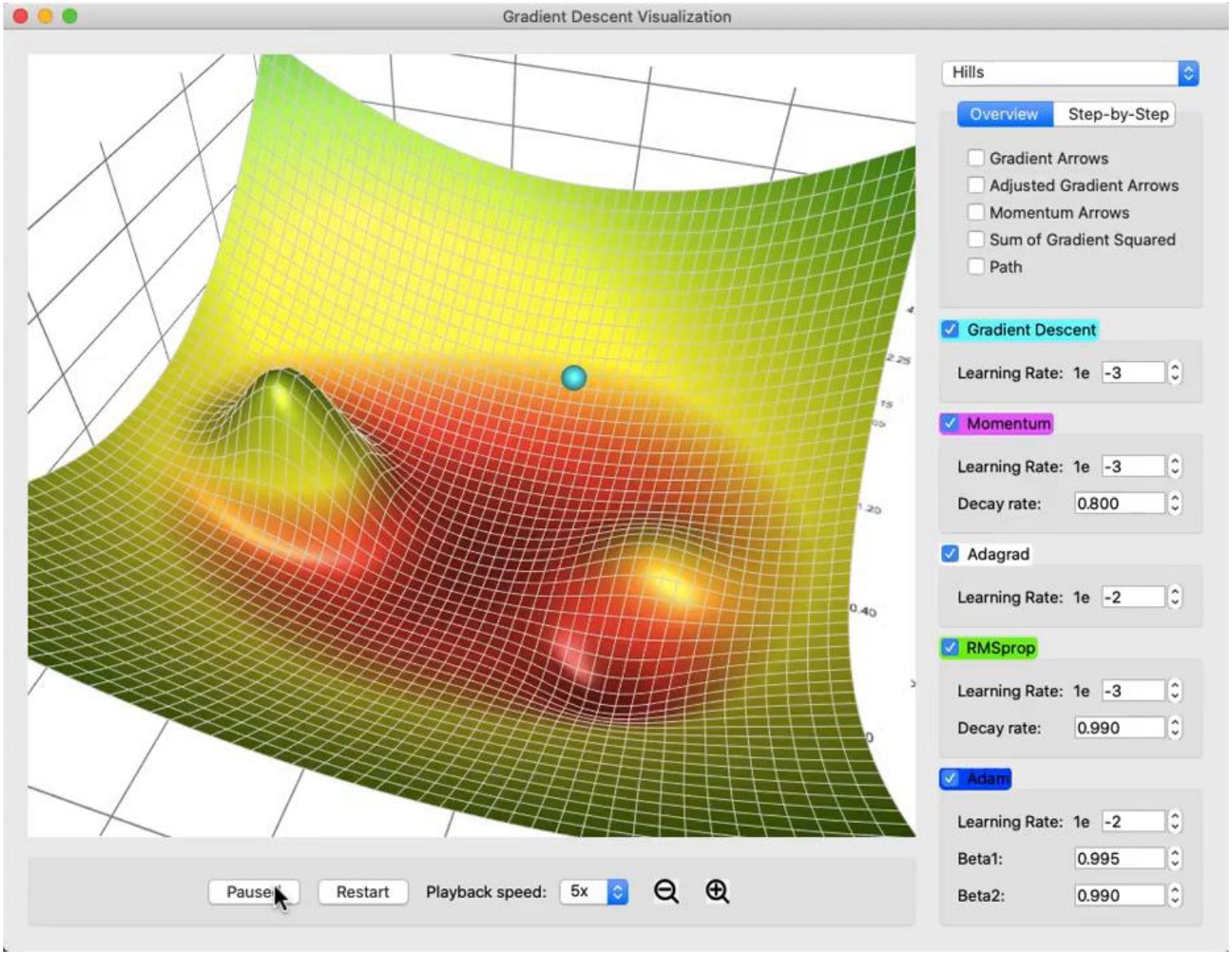
Adaptive Moment Estimation (Adam)

- Mengabungkan keunggulan RMSProp dan Momentum
- Adam:

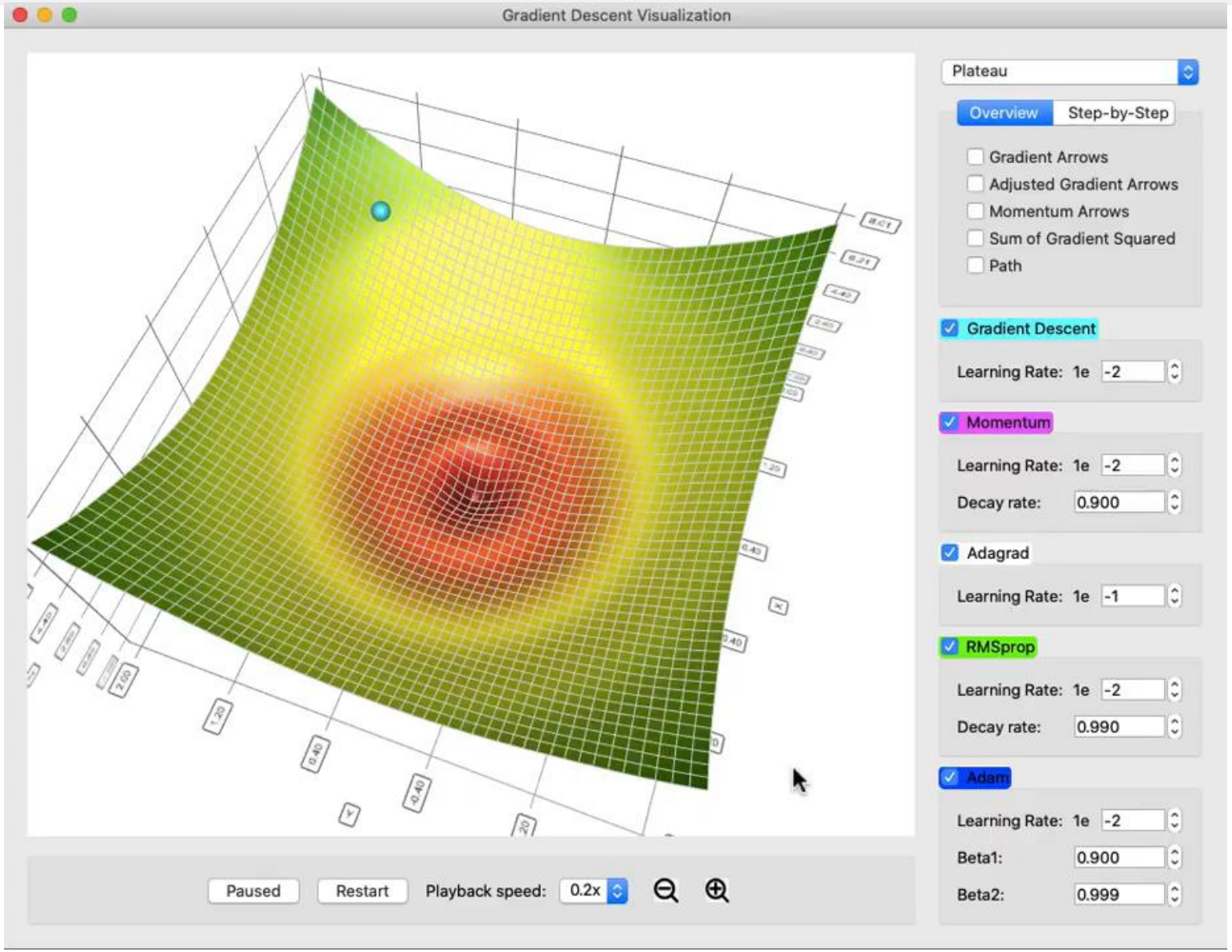
sum_of_grad = previous_sum_of_grad * beta1 + gradient * (1-beta1)
[Momentum]
sum_of_grad_squared = previous_sum_of_grad_squared * beta2 + gradient^2 * (1-beta2)
[RMSProp]
delta = -learning_rate * sum_of_gradient / sqrt(sum_of_grad_squared)
bobot = bobot + delta

- Beta1 merupakan decay_rate untuk moment pertama (biasanya: 0.9)
- Beta2 merupakan decay_rate untuk moment kedua (biasanya: 0.999)
- Adam mewarisi kecepatan dari momentum dan RMSProp sebagai pemandu arah gerakannya

Viz: GD on Hills



Viz: GD on Plateu



Thank You