

Supervised Learning – Regression (II)



Outline

- Bias-Variance Trade-Off
- Regularisasi pada Regresi Linear
- Ridge and Lasso Modelling
- Model Evaluation
- Gradient Descent

Bias-Variance Trade-Off

Bias dan Varians

Dua konsep pada model machine learning :

- **Bias**
Sejauh mana prediksi model cenderung mengalami deviasi sistematis dari nilai sebenarnya
- **Varians**
Sejauh mana prediksi model bervariasi ketika dilakukan pada dataset berbeda

Bias dan Varians

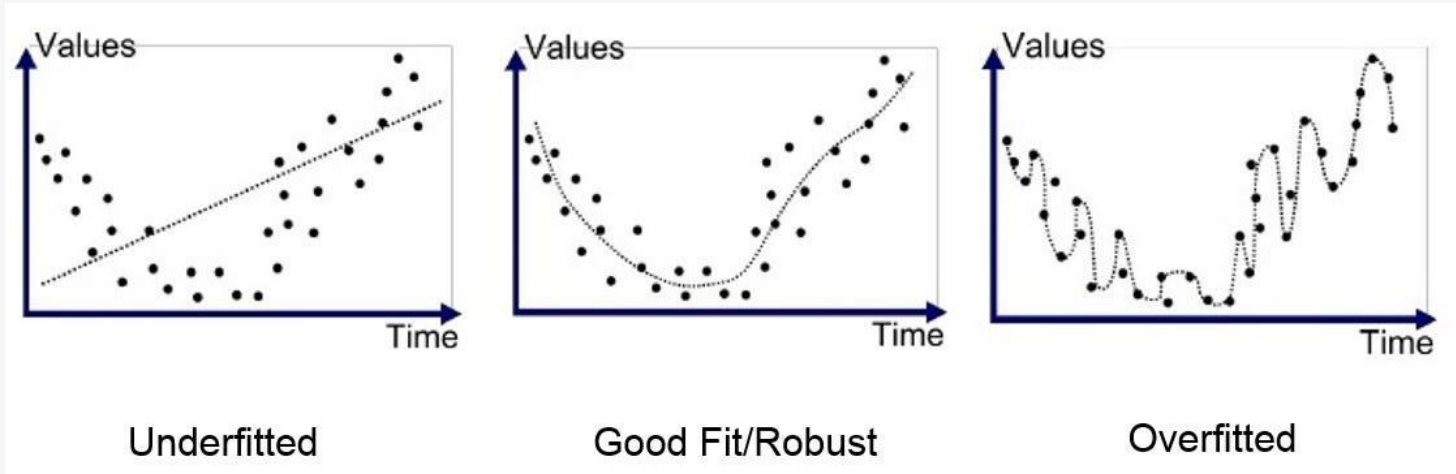
Bias

- Asumsi salah atau model terlalu sederhana
- Bias yang tinggi biasanya terkait dengan **underfitting**
- Bias dapat dikurangi dengan model lebih kompleks atau menambah jumlah fitur pada model

Varians

- Terlalu responsif terhadap data yang digunakan melatih model, tapi tidak bekerja baik pada data baru.
- Varians yang tinggi biasanya terkait dengan **overfitting**
- Varians bisa dikurangi dengan menggunakan teknik regularisasi

Underfitting dan Overfitting



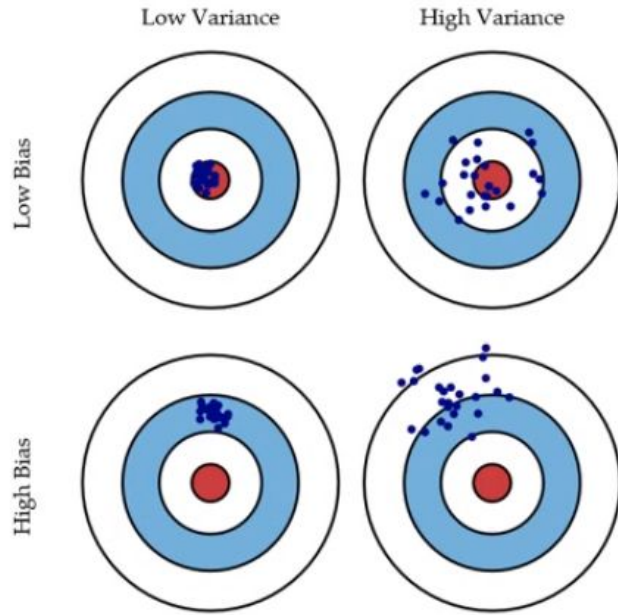
Underfit

- Model yang terlalu sederhana dan tidak menangkap pola
- **Indikasi** : Performa model tidak bagus pada data train dan test

Overfit

- Model yang terlalu kompleks dan menangkap noise, bukan pola yang sebenarnya.
- **Indikasi** : Performa model bagus pada data train, namun tidak bagus pada data test

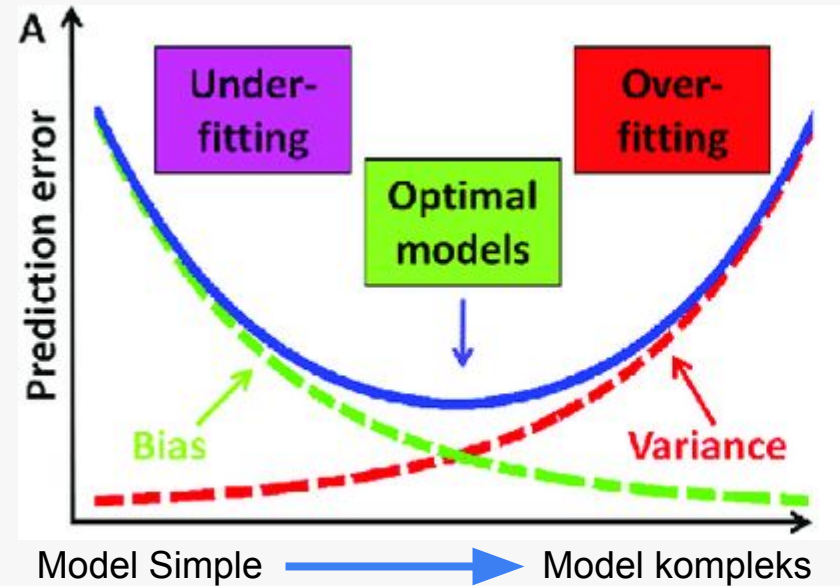
Bias dan Varians



Konsep bias dan varians bisa dilihat pada gambar di samping

Trade-Off

- Ketika mengurangi bias, varians cenderung meningkat dan sebaliknya
- Tujuan utama adalah **mencari keseimbangan antar bias dan varians** untuk mencapai model yang baik dan dapat diandalkan



Trade-Off

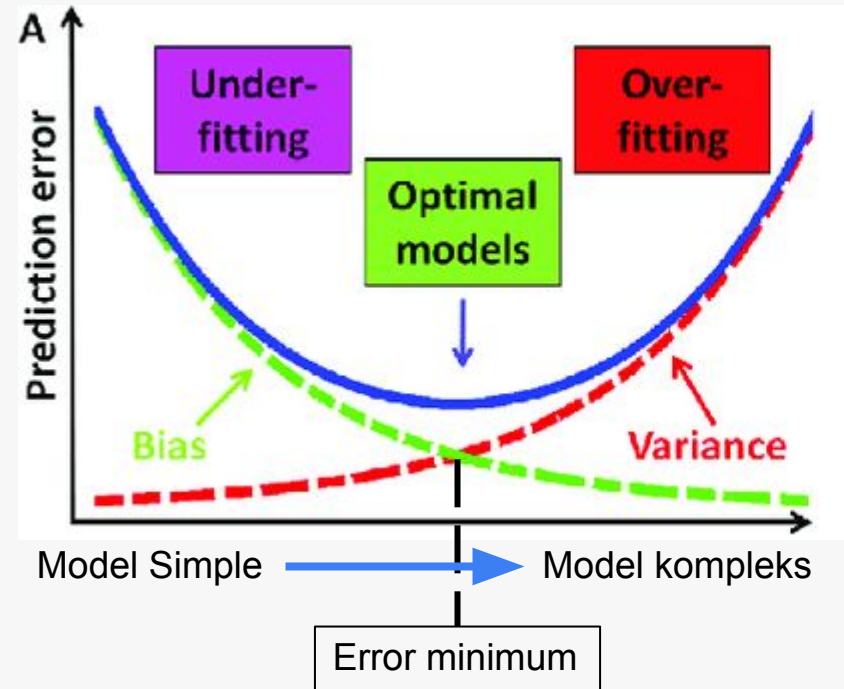
Mengurangi bias dengan meningkatkan variance

- Gunakan model **lebih kompleks**

Mengurangi varians dengan meningkatkan bias

- Gunakan model **lebih sederhana** atau regularisasi

Tujuannya mencari keseimbangan antara bias dan varians pada **error minimum**



Regularisasi pada Regresi Linear

Mengapa perlu regularisasi?

- Membut model lebih sederhana.
- Mengurangi varians dengan meningkatkan biasnya.
- Mengurangi kompleksitas model juga bisa mengurangi **overfitting**.

Regularisasi

Ridge

- Linear regression dengan fungsi loss yang dimodifikasi

$$\sum_i (y_i - \hat{y}_i)^2 + \lambda \sum_j \beta_j^2$$

- Jika $\lambda = 0$, menjadi linear regresi biasa
- Semua fitur tidak akan pernah menjadi 0

Lasso

- Linear regression dengan fungsi loss yang dimodifikasi

$$\sum_i (y_i - \hat{y}_i)^2 + \lambda \sum_j |\beta_j|$$

- Jika $\lambda = 0$, menjadi linear regresi biasa
- Beberapa fitur bisa menjadi 0 dan bisa dieliminasi dari model

Bagaimana memilihnya?

Ridge

- Ridge biasanya digunakan untuk mengurangi multikolinearitas
- Interpretasi model lebih sederhana karena semua fitur tetap ada dimana koefisien besar menunjukkan fitur yang lebih berpengaruh
- Lebih stabil terhadap perubahan data karena semua fitur memiliki kontribusi meskipun berkurang

Lasso

- Lasso biasanya digunakan ketika terdapat banyak fitur dan bisa seleksi fitur secara otomatis
- Lasso membuat beberapa fitur yang tidak berpengaruh signifikan menjadi nol
- Hal ini dapat membuat model lebih tidak stabil dalam menanggapi perubahan data karena ada fitur yang menjadi 0
- Efektif mengatasi masalah “curse of dimensionality”

Ridge vs Lasso

Namun, biasanya performa Ridge dan Lasso hampir mirip


Sehingga lebih baik train keduanya dan lihat mana yang menghasilkan performa lebih baik.

Ada gabungan keduanya yaitu Elastic Net Regression

Ridge and Lasso Modelling

Linear Regresi menggunakan Regularisasi

Steps :

- Pisahkan feature dan target
feature dan target
- Split data : train – test
feature_train, feature_test, target_train, target_test
- Multicollinearity Check
Calculate VIF Score dan Analisis Korelasi
- Tentukan model
ridge = Ridge(alpha=0.1) OR lasso = Lasso(alpha=0.1)  **Lambda**
- Train model
ridge.fit(feature_train, target_train) OR lasso.fit(feature_train, target_train)
- Predict test data
ridge.predict(feature_test) OR lasso.predict(feature_test)
- Evaluasi model
Gunakan beberapa metrics untuk regresi (RMSE, MAE, MAPE)

Regularisasi

Bagaimana cara kita menentukan Lambda terbaik?

Regularisasi dengan memilih lambda

Steps :

- Pisahkan feature dan target
feature dan target
- Split data : train – validation – test
80% → 80% train, 20% validation
20% test
- Multicollinearity Check
Calculate VIF Score dan Analisis Korelasi
- Train beberapa model ke train data
ridge = Ridge(alpha= λ) OR lasso = Lasso(alpha= λ)
Train dengan beberapa nilai lambda yang berbeda
- Pilih best lambda dari validation set
RMSE paling kecil
- Predict test data
ridge_best.predict(feature_test) OR lasso_best.predict(feature_test)
- Evaluasi model
Gunakan beberapa metrics untuk regresi (RMSE, MAE, MAPE)

Lambda

Regularisasi

Model Training

```
from sklearn.linear_model import Ridge

# define the model
ridge_reg_pointzeroone = Ridge(alpha=0.01, random_state=42)
ridge_reg_pointone = Ridge(alpha=0.1, random_state=42)
ridge_reg_one = Ridge(alpha=1, random_state=42)
ridge_reg_ten = Ridge(alpha=10, random_state=42)

# fit the model (training)
ridge_reg_pointzeroone.fit(X_train, y_train)
ridge_reg_pointone.fit(X_train, y_train)
ridge_reg_one.fit(X_train, y_train)
ridge_reg_ten.fit(X_train, y_train)
```

Regularisasi

Choosing Best Lambda (using validation data)

```
from sklearn.metrics import mean_squared_error
```

```
alphas = [0.01, 0.1, 1., 10]  
ridge_models = [ridge_reg_pointzeroone,  
                 ridge_reg_pointone,  
                 ridge_reg_one,  
                 ridge_reg_ten]
```

```
for model, alpha in zip(ridge_models, alphas):  
    y_predict_validation_ridge = model.predict(X_validation)  
    rmse = np.sqrt(mean_squared_error(y_validation, y_predict_validation_ridge))  
    print(f'RMSE of Ridge Regression model with alpha = {alpha} is {rmse:.4f}')
```

```
RMSE of Ridge Regression model with alpha = 0.01 is 5.1695  
RMSE of Ridge Regression model with alpha = 0.1 is 5.1724  
RMSE of Ridge Regression model with alpha = 1.0 is 5.1693  
RMSE of Ridge Regression model with alpha = 10 is 4.8843
```

**RMSE paling kecil adalah pada saat alpha = 10.
Jadi lambda terbaik adalah 10.**

Model Evaluation

Root Mean Squared Error

Model performance dilakukan pada test data

$$RMSE = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}}$$

Keuntungan :

- Mudah diinterpretasikan karena perhitungan kesalahan dalam satuan yang sama dengan variabel target
- Memberikan bobot lebih besar pada kesalahan yang lebih besar dibandingkan dengan MAE, karena menggunakan kuadrat dari selisih

Kelemahan :

- Lebih sensitif terhadap nilai-nilai outlier dalam data karena menggunakan kuadrat dari selisih, yang dapat memperbesar dampak kesalahan tersebut pada perhitungan keseluruhan
- Tidak menunjukkan arah kesalahan (overestimasi atau underestimasi)

Mean Absolute Error

Model performance dilakukan pada test data

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$

Keuntungan :

- Mudah diinterpretasikan karena perhitungan kesalahan dalam satuan yang sama dengan variabel target
- Lebih tahan terhadap outlier karena menggunakan nilai absolut dari selisih prediksi dan nilai sebenarnya

Kelemahan :

- MAE tidak memberikan bobot lebih pada kesalahan yang lebih besar atau lebih kecil. Dalam beberapa kasus, kesalahan besar mungkin lebih penting daripada kesalahan kecil
- Tidak menunjukkan arah kesalahan (overestimasi atau underestimasi)

Mean Absolute Percentage Error

Model performance dilakukan pada test data

$$\text{MAPE} = \sum_{t=1}^n \left| \frac{y_i - \hat{y}_i}{\hat{y}_i} \right| \times 100\%$$

Keuntungan :

- MAPE memberikan perhitungan kesalahan dalam bentuk persentase, sehingga memudahkan dalam memahami tingkat kesalahan secara relatif.
- MAPE berguna dalam membandingkan kinerja model antara berbagai variabel target yang berbeda skala atau rentang nilainya.

Kelemahan :

- Masalah Ketika terdapat nilai actual yang bernilai 0 yang akan menjadi pembagi sehingga menghasilkan perhitungan yang tidak terdefinisi.
- Tidak menunjukkan arah kesalahan (overestimasi atau underestimasi)

Gradient Descent

Gradient Descent

Algoritma optimasi yang digunakan untuk mencari nilai minimum (atau maksimum) dari suatu fungsi untuk mengoptimalkan loss function dalam model.

Proses optimalisasi yang terjadi secara iteratif menggunakan gradient yang dihitung dari hasil backpropagation untuk memperbarui weight dan bias dimulai dengan inisialisasi acak pada parameter model.

Gradient Descent

Kelebihan:

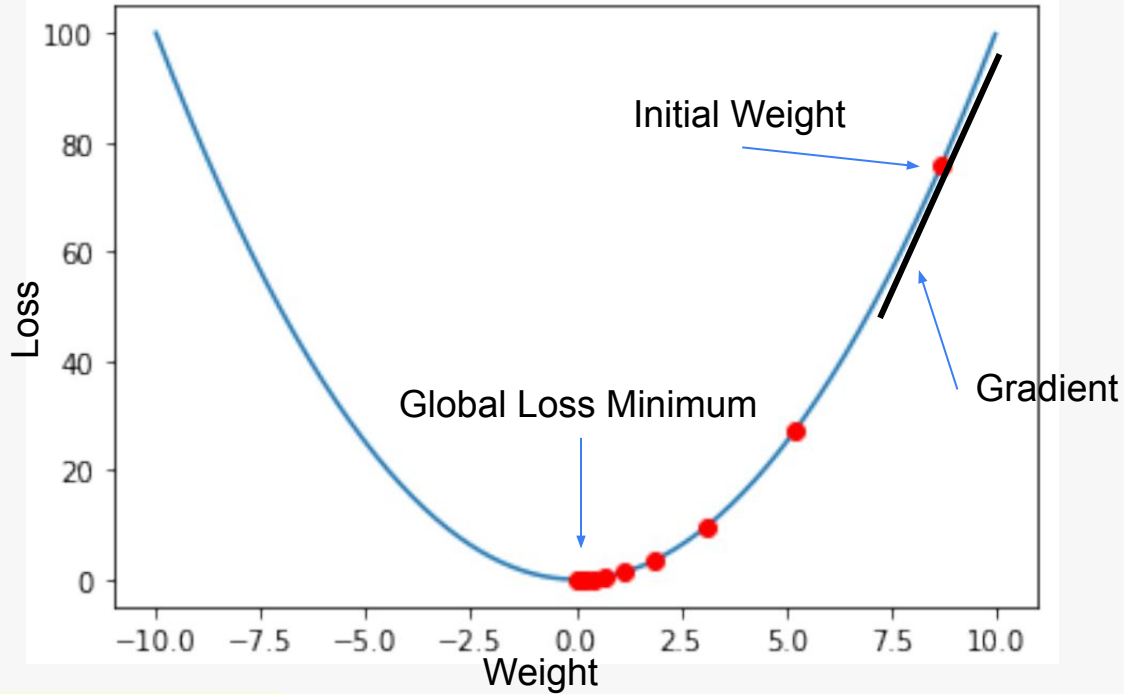
- Dapat mencapai minimum global dan menghasilkan model optimal secara global
- Efisien dengan hanya menggunakan sejumlah kecil sampel data pada setiap iterasi
- Dapat bekerja dengan baik pada dataset besar, karena komputasi dihitung parallel untuk mempercepat proses optimisasi.
- Dapat diterapkan pada berbagai jenis model machine learning

Gradient Descent

Kekurangan:

- Dapat memiliki kecepatan konvergensi yang lambat terutama saat mendekati minimum local
- Sangat sensitif terhadap pemilihan learning rate. Diusahakan tidak terlalu besar dan juga tidak terlalu kecil.
- Dalam beberapa kasus, dapat terjebak pada minimum local dan gagal mencapai minimum global yang lebih optimal
- Jika skala variable pada dataset berbeda, akan kesulitan menemukan minimum yang optimal karena gradient yang dihasilkan tidak seimbang

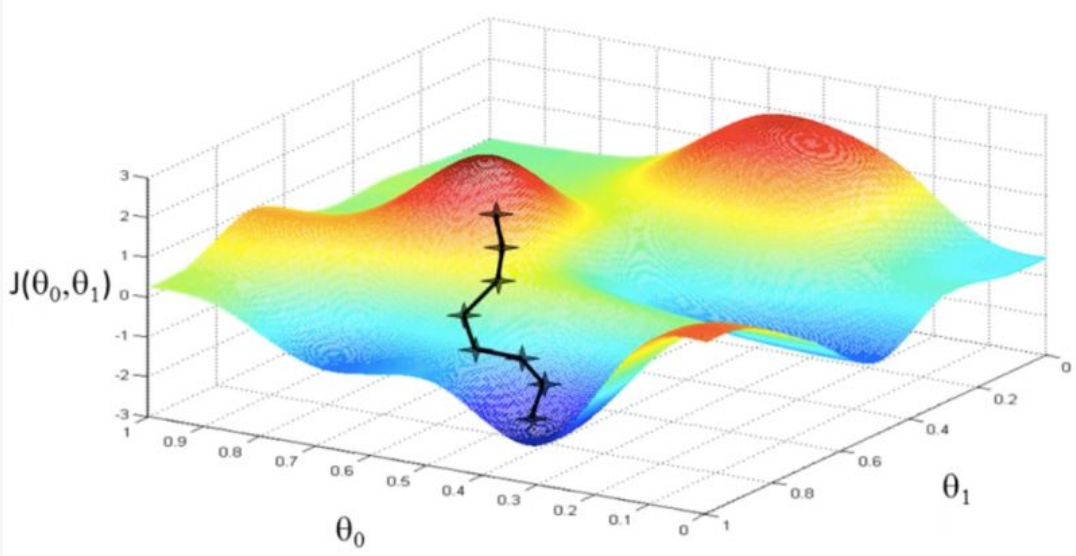
Gradient Descent



Steps:

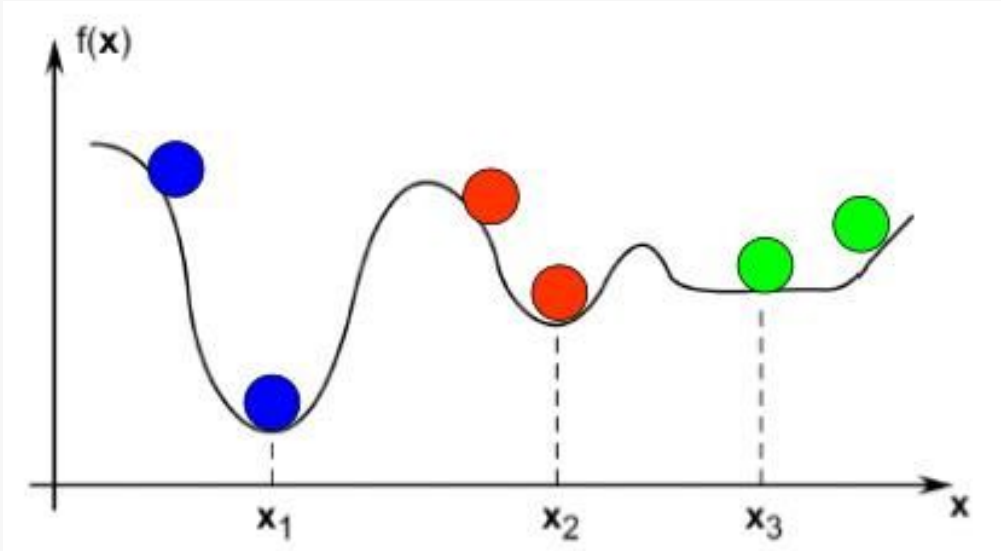
1. Inisialisasi weight
2. Weight akan terus berubah hingga mencapai global loss minimum

Gradient Descent



Terus turun hingga mencapai minimum global

Gradient Descent



Masalah yang mungkin muncul adalah bisa terjebak pada minimum lokal

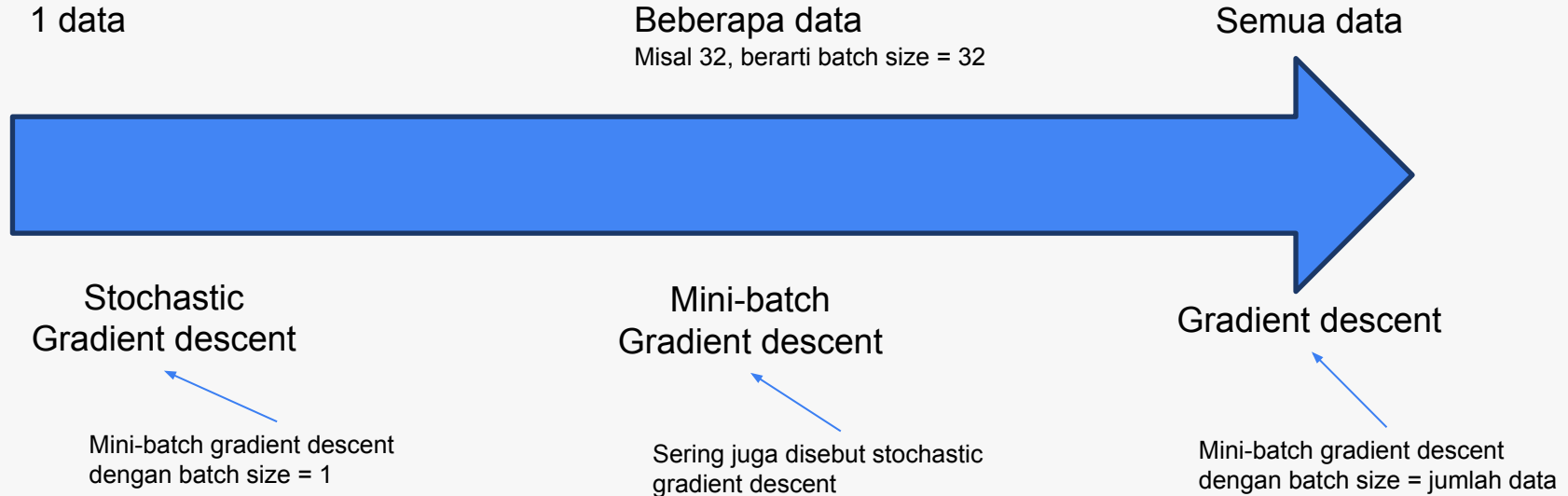
Learning Rate

- Parameter yang mengontrol seberapa besar langkah yang diambil pada setiap iterasi
- Nilai learning rate yang tepat bisa mempercepat konvergensi algoritma dan memastikan bahwa solusinya tepat
- Jika terlalu besar, algoritma mungkin melompati-lompati minimum
- Jika terlalu kecil, algoritma mungkin butuh waktu lama atau bisa terjebak minimum lokal

Batch Size

- Jumlah sampel data yang diproses dalam satu iterasi
- Batch size yang tepat bisa mempercepat training, stabilitas konvergensi, dan kebutuhan memori
- Ukuran batch besar dapat mempercepat pelatihan, namun bisa menyebabkan kekurangan memori
- Ukuran batch kecil dapat membantu model mencapai konvergensi lebih stabil dan akurat, namun meningkatkan waktu komputasi

Tipe Gradient Descent



Terima Kasih

Thanks!

