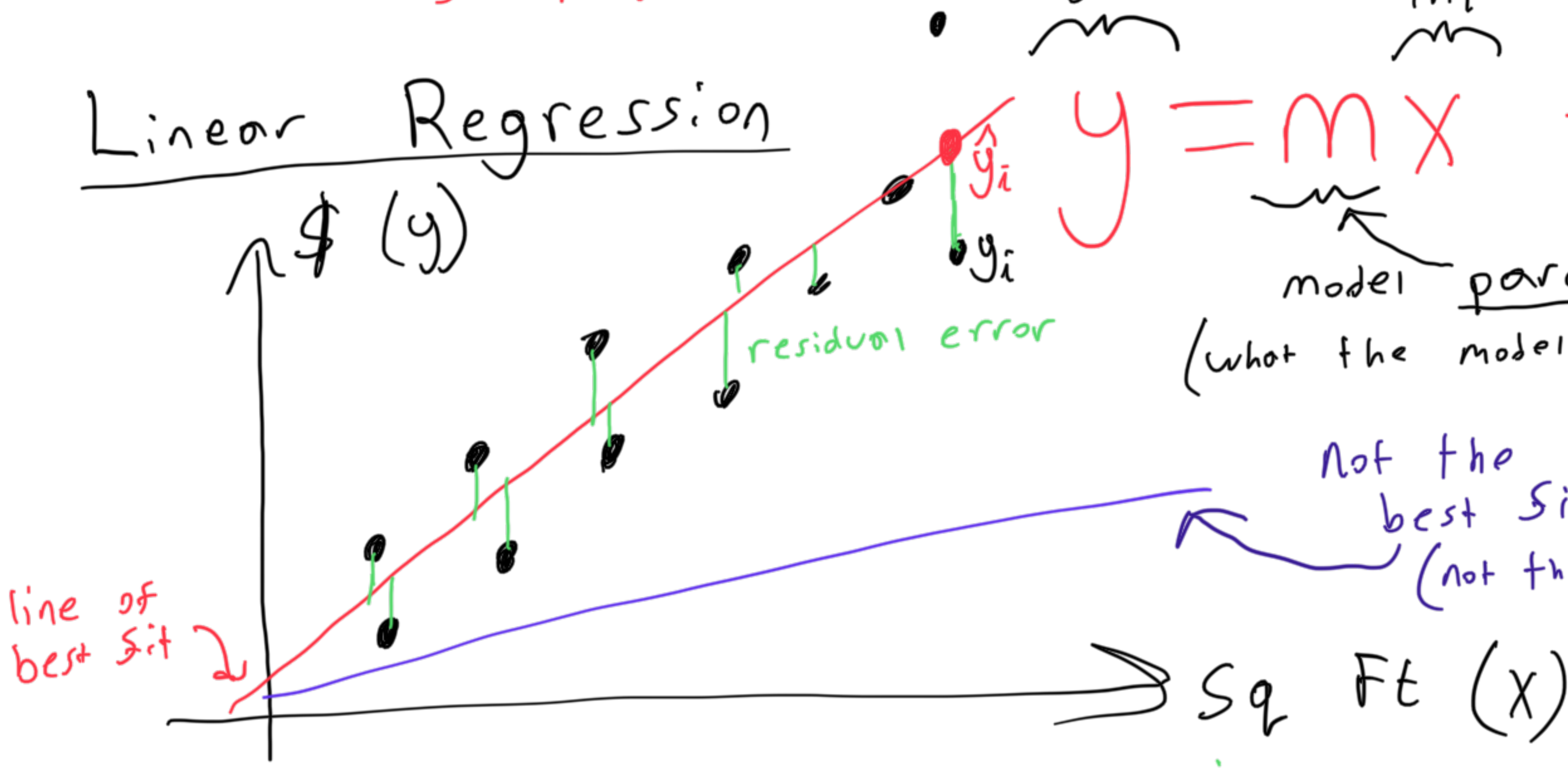
Day 2: Linear Models

Supervised Learning

$$y = f(X)$$

output ("what we predict") — model — input

Linear Regression



$$\overbrace{y}^{\text{predicted output}} = \underbrace{m}\overbrace{x}^{\text{input}} + \underbrace{b}$$

model parameters (what the model learns)

Not the line of best fit (not the optimal model)

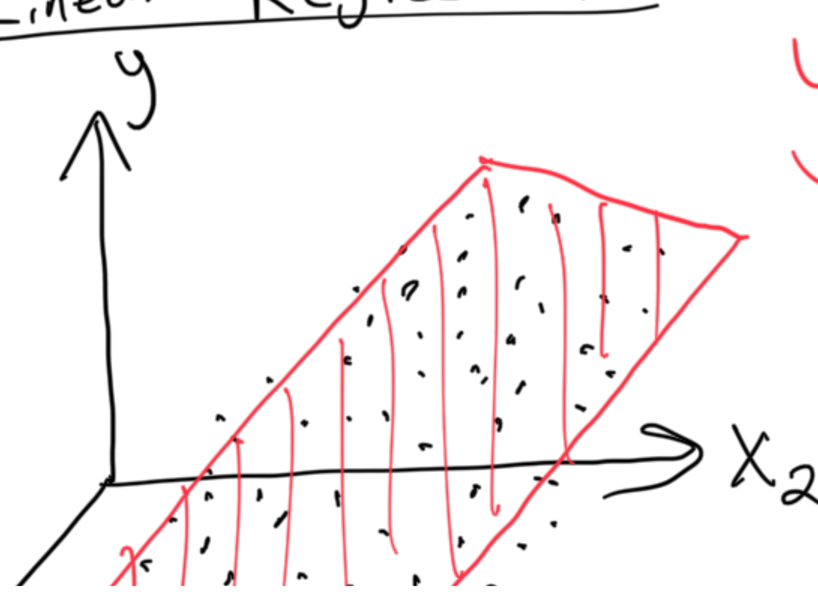
Goal of linear regression:
Learn the m and b with the best fit / minimal error / minimal sum of residuals

y = true values $\hat{y}$ = predicted values

residual for one data point i: $|y_i - \hat{y}_i|$

Minimal sum of residuals: $$\sum_{i=1}^{m} |y_i - \hat{y}_i|$$

2D Linear Regression



$$y = \underbrace{m_1 X_1} + \underbrace{m_2 X_2} + \underbrace{b}$$

Weights / Stuff the model learns

Linear Regression General Form

Same thing

$$y = m_1 x_1 + \dots + m_n x_n + b$$

$$y = \theta_0 x_1 + \dots + \theta_{n-1} x_n + \theta_n$$

$$y = w_1 x_1 + \dots + w_n x_n + b$$

"weights" "bias"

$\theta_{0 \dots n}$ are the model parameters

How do we learn the weights and bias?

Short answer: Optimize a loss function

Loss Function: function which quantifies the error of our model

Linear Regression Loss Function

Mean Absolute Error: (MAE)

$$\frac{1}{n} \sum_{i=1}^{n} \left| y_i - \hat{y}_i \right|$$

In practice, we use:

Mean Square Error: (MSE)

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

Why?

y_i	$\hat{y}_i$	MAE	MSE
3	4	1	1
3	5	2	4
3	6	3	9

Math definition of linear regression:

$$\min_{m,b} \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{average}} \left(y_i - [m x_i + b] \right)^2$$

Non-Linear Regression

Polynomial Regression:

$$f(x) = ax^3 + bx^2 + cx + d$$

a, b, c, and d are the <u>learned parameters</u>

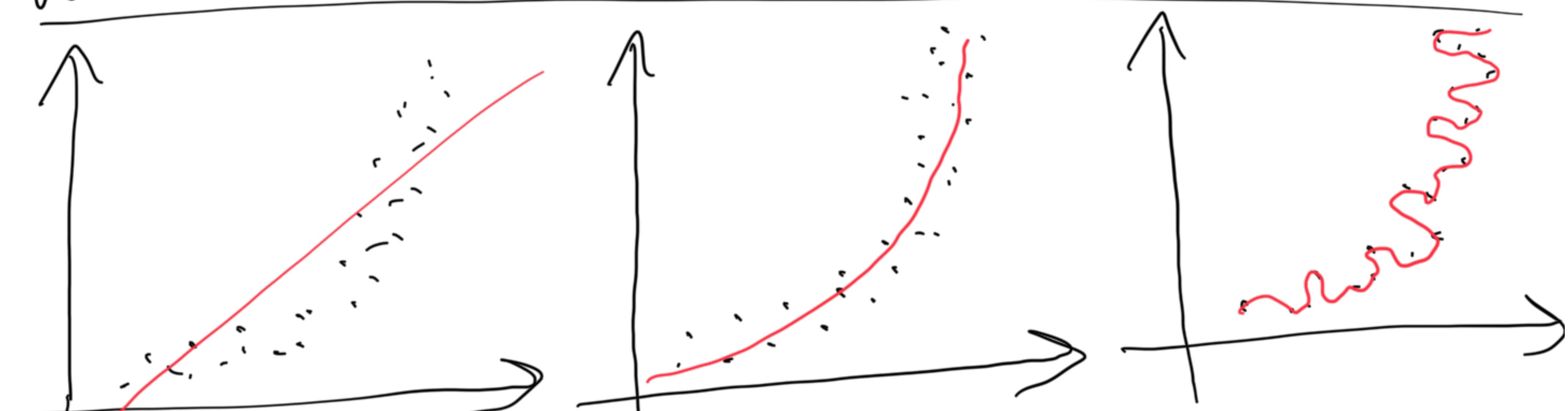
$$f(x) = \theta_0 X_1^2 + \theta_1 X_2^2 + \theta_2 X_1 + \theta_3 X_2 + \theta_4$$

$\theta_{0\ldots4}$ are the <u>learned parameters</u>

X_1, X_2 are the <u>model inputs</u>

$f(x) = y$ is the <u>model output</u>

What if we have too many Variables?



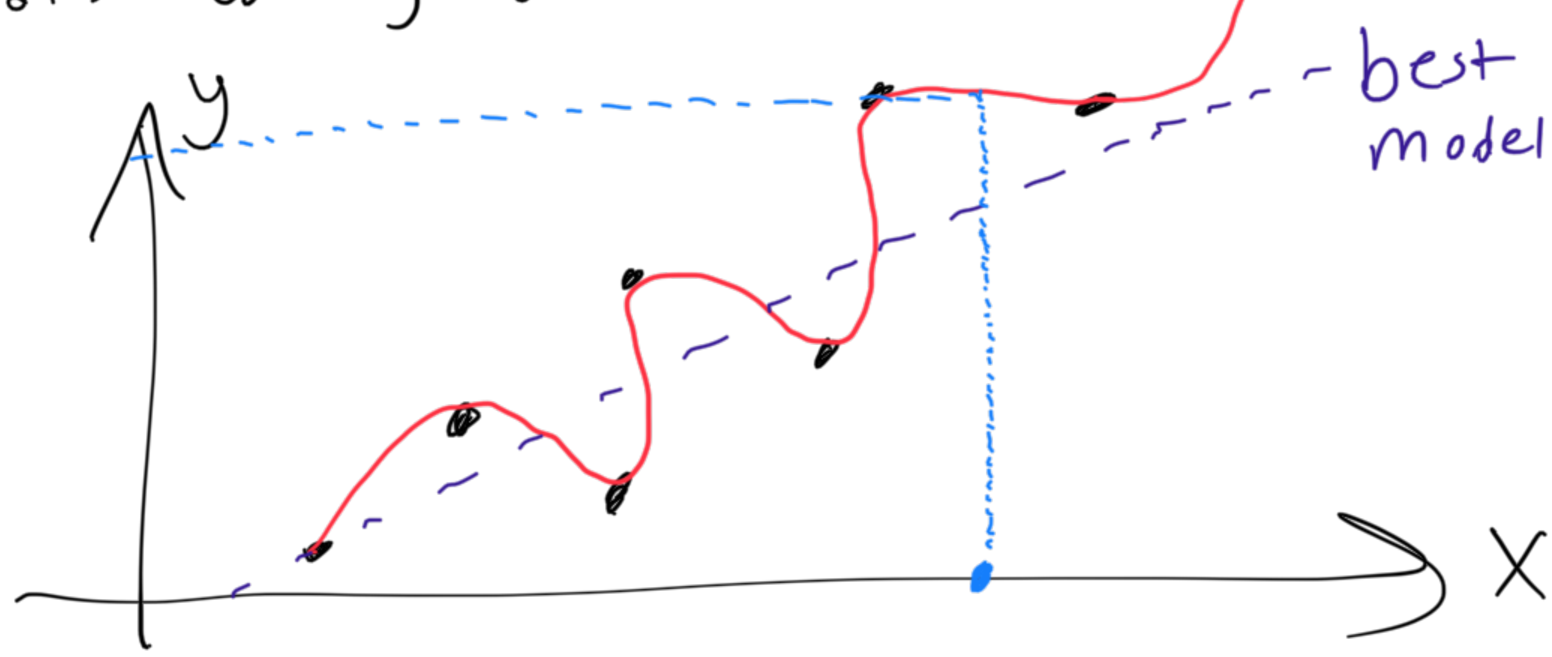
underfitting "high bias"

just right

overfitting "high variance" "<u>memorizing</u> the

data"
learns the noise
in the data

What's wrong with overfitting?



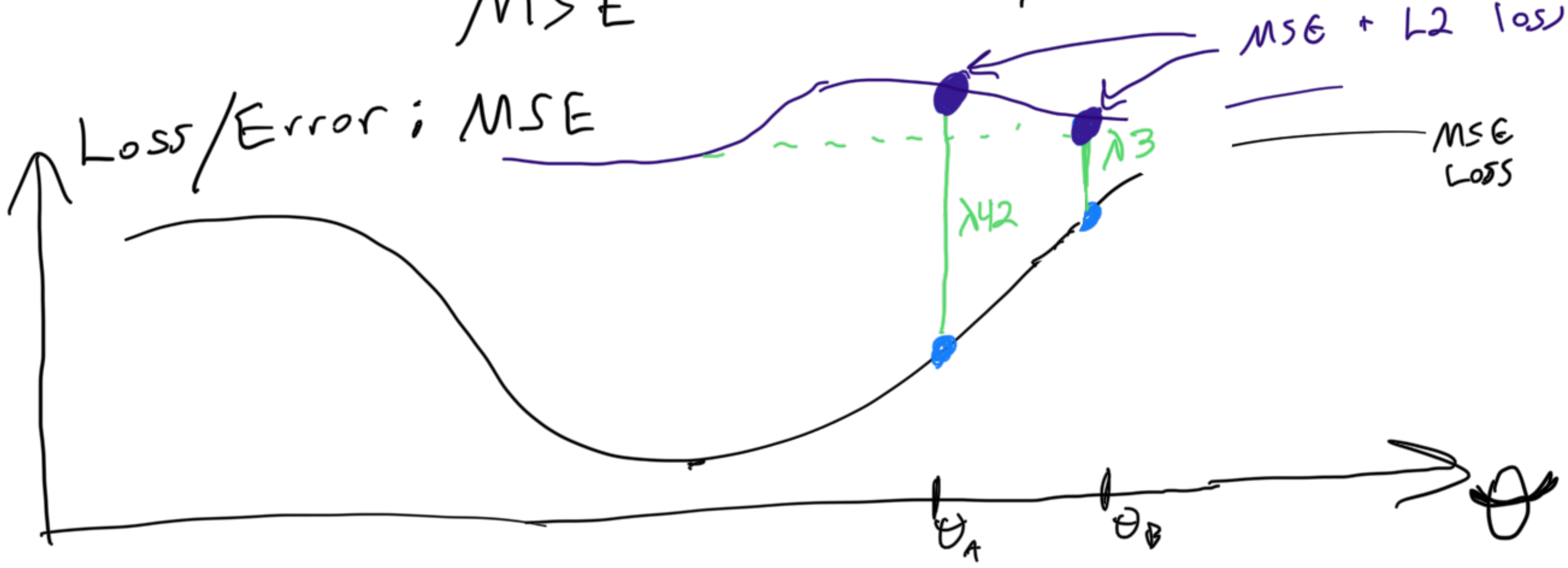
One Solution to overfitting: Regularization

L1 Regularization: add $\lambda \sum_{j=1}^{p} |\theta_j|$ to loss function

L2 Regularization: add $\lambda \sum_{j=1}^{p} \theta_j^2$ to loss function

p = model parameters

Loss Function with L2 Regularization: $$\underbrace{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}_{\text{MSE}} + \underbrace{\lambda \sum_{j=1}^{p}\theta_j^2}_{\text{penalize model weights}}$$



With MSE loss, θ_A gives lower loss
With MSE + L2 loss, θ_B gives lower loss

$f(x) = ax^4 + bx^3 + cx^2 + dx + e$

$f(x) = \ldots$

$$\theta_A \left\{ = 3x^4 + 4x^3 + 2x^2 + 2x - 3 \right.$$

↳ add $\lambda \cdot [3^2 + 4^2 + 2^2 + 2^2 + 3^2]$ to loss function

$$= 0x^4 + 0x^3 + 1x^2 + x + 1 \quad \Big\} \theta_B$$

↳ add $\lambda \cdot [1^2 + 1^2 + 1^2]$ to loss function

penalizing loss less

λ = <u>regularization strength</u>

λ is set by the ML engineer

(<u>hyperparameter</u>)

hyperparameter: a value set by the ML engineer before learning begins