Logistic Regression

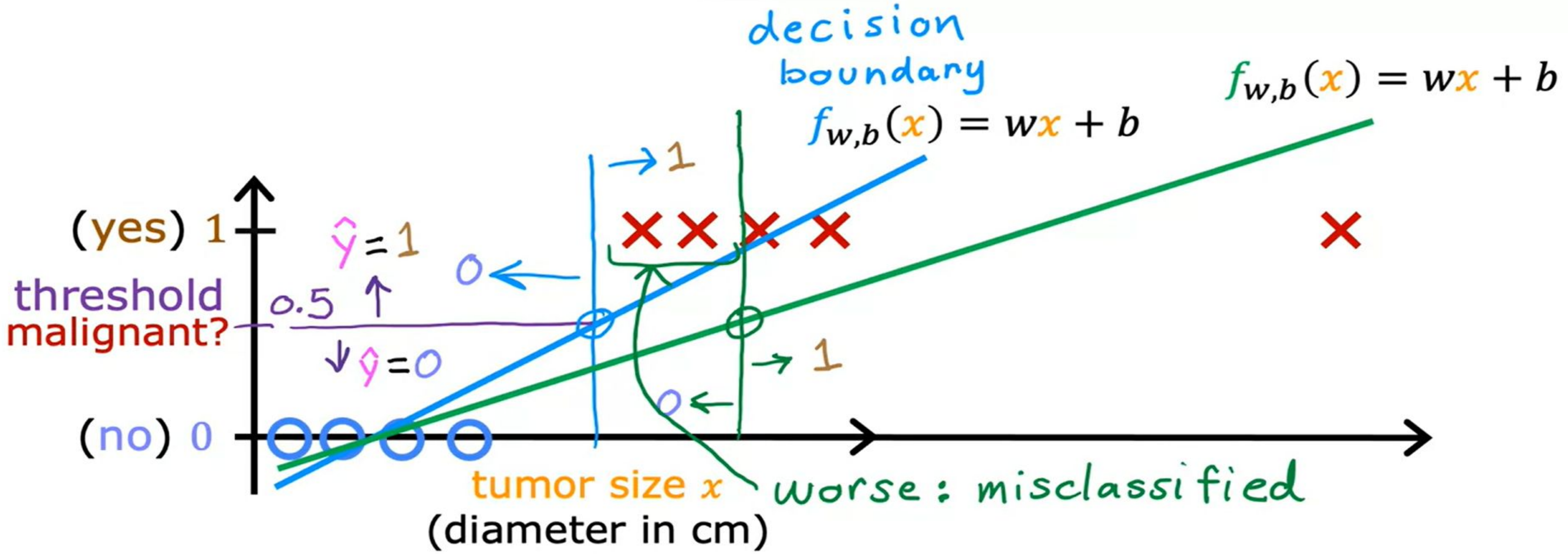
Md Arafat Hussain
Chennupalli Sathwik
Ramesh Chandra Soren
Aryan D Rozario
Avishikta Das
Sanjoy Garai

Objective

- Overview of Logistic Regression
- Sigmoid Function
- Squared error cost
- Gradient descent
- Overfitting Issue
- Regularization

Overview

Problem with linear regression



if $f_{w,b}(x) < 0.5 \rightarrow \hat{y} = 0$

if $f_{w,b}(x) \geq 0.5 \rightarrow \hat{y} = 1$

- Linear regression is used when the outcome (dependent variable) is continuous. It predicts a value based on a linear relationship between the independent variables and the outcome.
- It cannot accurately binarily classify an entity according to categorical outcome variables.
- Logistic regression is specifically designed for binary classification problems, which are prevalent in various fields such as healthcare (disease diagnosis), finance (fraud detection), and marketing (customer segmentation). Linear regression cannot handle binary outcomes directly.

Difference between linear and regression model

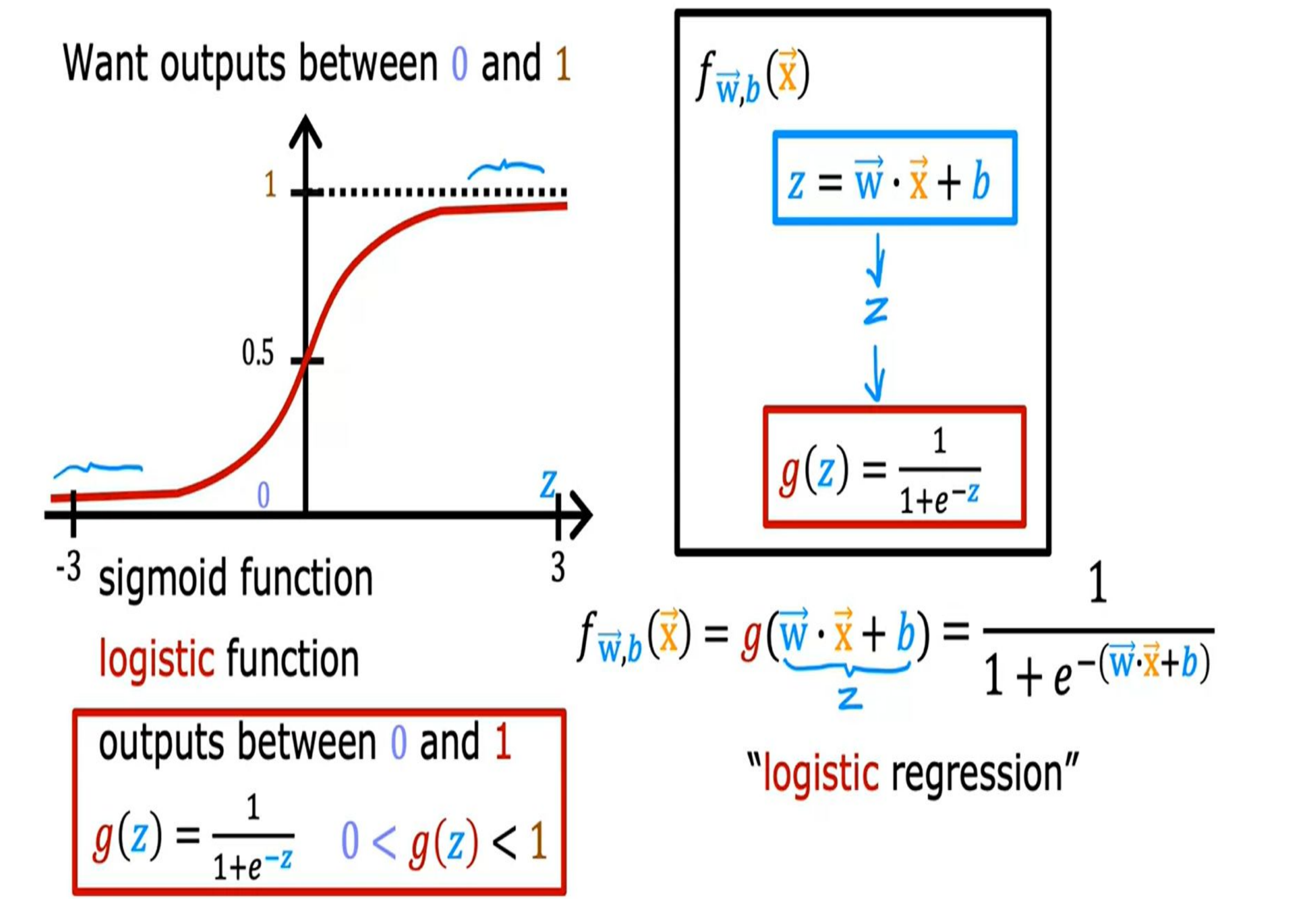
- **Linear Regression**:
  - Predicts continuous numerical values.
  - Models the relationship between dependent and independent variables using a linear equation.
  - Used for regression tasks.
  - Solved by minimizing the least squares error.
- **Logistic Regression**:
  - Predicts categorical outcomes.
  - Models the probability of a binary outcome using a logistic function.
  - Used for classification tasks.
  - Uses the logistic loss function to penalize large errors.

Sigmoid function

Example of Logistic regression

A Linear function cannot be used to classify accurately the given data.

We need a certain sigmoid function to vary between 0 and 1 for any features of input x to better classify the input according to a threshold value.



w and b are the parameters.

w1,w2,w3…….wn; for x1,x2,x3……xn attributes.

b is the bias term.

Squared Error Cost

Squared Error Cost

- **Squared Error**:
  - Measures the difference between predicted and actual values in regression.
  - Calculated by squaring the difference between predicted and actual values, then summing across all observations.
  - Possesses nice mathematical properties, such as being everywhere differentiable.
  - Induced by an inner product on the underlying space, facilitating geometric interpretation.
- **Choice of Loss Function**:
  - Depends on the nature of the data and the problem being solved. This calculates the loss in each attribute of the given data.

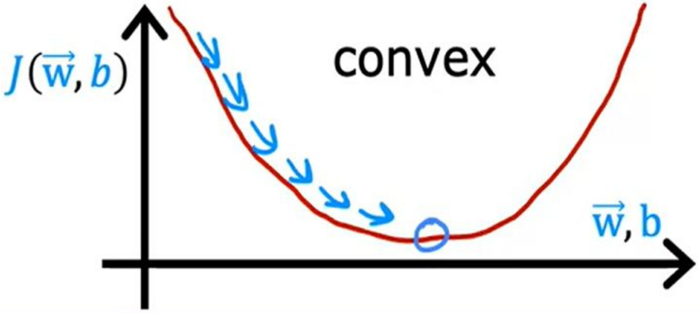
Squared error cost

$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \boxed{\frac{1}{2}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)^2}$$

loss $\boxed{L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right)}$

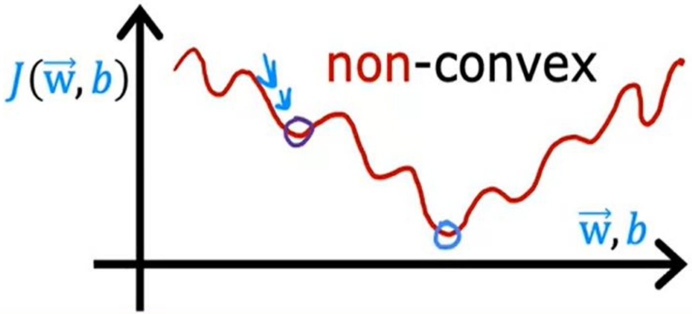
linear regression

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$$



logistic regression

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w}\cdot\vec{x}+b)}}$$



Cost

cost

$$J(\vec{\mathrm{w}}, b) = \frac{1}{m}\sum_{i=1}^{m} L\underbrace{\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right), y^{(i)}\right)}_{\text{loss}}$$

$$= \begin{cases} -\log\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

convex

↳ can reach a global minimum

Simplified cost function

loss

$$L\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right), y^{(i)}\right) = -\,y^{(i)}\log\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)\right)$$

cost

$$J(\vec{\mathrm{w}}, b) = \frac{1}{m}\sum_{i=1}^{m}\left[L\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right), y^{(i)}\right)\right]$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)\right)\right]$$

Gradient Descent

Necessity of Gradient Descent

- **Parameter Reduction**:
  - Gradient descent iteratively adjusts model parameters (coefficients and bias) to minimize the logistic loss function.
  - At each step, gradients of the loss function are computed to determine the direction of parameter updates.
  - Parameters are updated in the opposite direction of the gradients, scaled by the learning rate.
  - This process repeats until convergence, refining parameters to best fit the data and optimize classification accuracy.

Learning Rate(alpha): As in other variants of gradient descent, the learning rate determines the step size of each update. It's a hyperparameter that needs to be carefully chosen to balance convergence speed and stability.

Gradient descent

(Alpha) is the learning rate.

cost

$$J(\vec{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1 - y^{(i)})\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right)\right]$$

repeat {

j=1...n

$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}J(\vec{w}, b) \qquad \frac{\partial}{\partial w_j}J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})x_j^{(i)}$$

$$b = b - \alpha\frac{\partial}{\partial b}J(\vec{w}, b) \qquad \frac{\partial}{\partial b}J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})$$

} simultaneous updates

Gradient descent for logistic regression

repeat {

$$w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} \right]$$

$$b = b - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right) \right]$$

} simultaneous updates

Same concepts:

- Monitor gradient descent (learning curve)
- Vectorized implementation

Linear regression $f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$

Logistic regression $f_{\vec{w},b}(\vec{x}) = \dfrac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$

Overfitting Issue

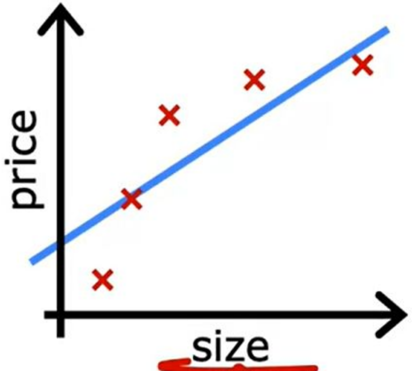
- **Overfitting**:
    - **Definition**: Model memorizes noise in training data rather than learning underlying patterns, leading to poor performance on unseen data.
    - **Causes**: Model complexity, insufficient data, noise in data.
    - **Effects**: Reduced generalization, poor performance on unseen data, unreliable insights.
    - **Prevention and Remedies**: Simplify model, cross-validation, regularization, feature selection, ensemble methods, early stopping.

Overfitting can be addressed by ensuring models are appropriately complex for the data available, validating performance on unseen data, and employing techniques to reduce noise and complexity.

For linear regression

Regression example

Press Esc to exit full screen

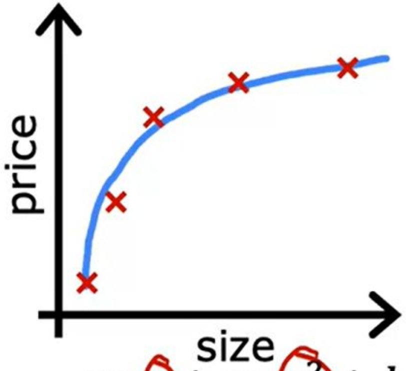


$w_1 x + b$

underfit

- Does not fit the training set well

high bias

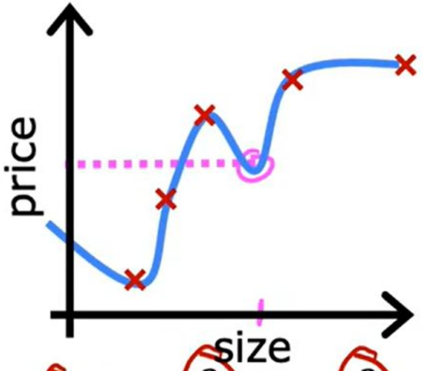


$w_1 x + w_2 x^2 + b$

just right

- Fits training set pretty well

generalization



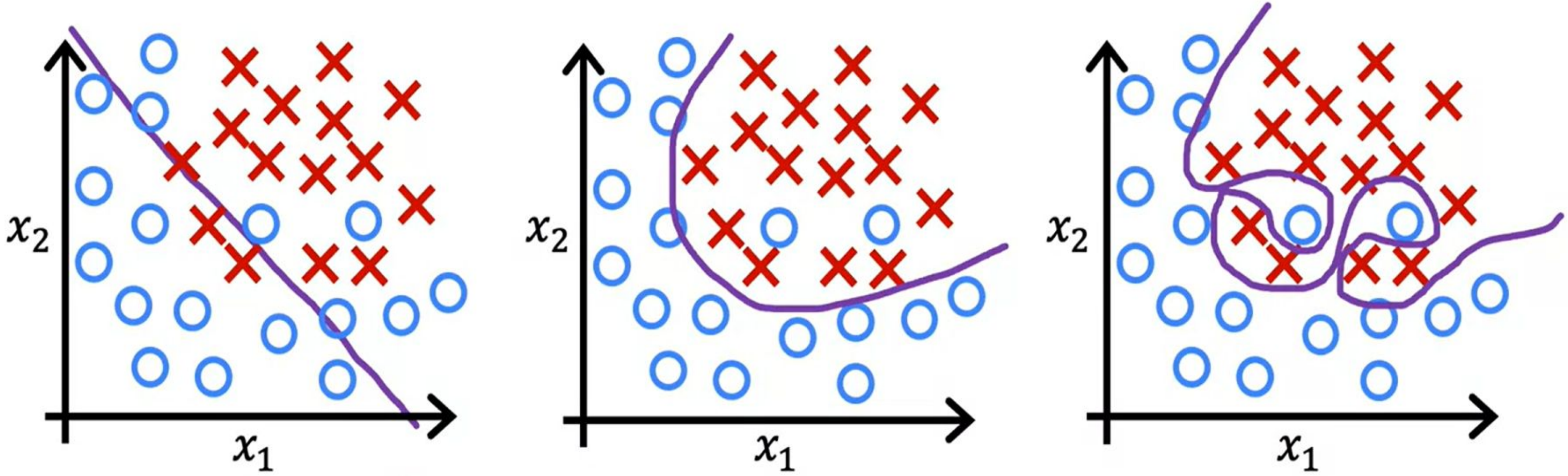
$w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$

overfit

- Fits the training set extremely well

high variance

Classification



$$z = w_1x_1 + w_2x_2 + b$$

$$f_{\vec{w},b}(\vec{x}) = g(z)$$

g is the sigmoid function

$$z = w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2 + b$$

$$z = w_1x_1 + w_2x_2 + w_3x_1^2x_2 + w_4x_1^2x_2^2 + w_5x_1^2x_2^3 + w_6x_1^3x_2 + \cdots + b$$

underfit high bias

just right

overfit

Addressing overfitting

Options

1. Collect more data

2. Select features
   – Feature selection

3. Reduce size of parameters
   – "Regularization"

Regularization

Need for Regularization

- Regularization is a technique to prevent overfitting by adding a penalty term to the model's loss function, discouraging overly complex models. It comes in two main forms:
- Regularization helps in improving model generalization by penalizing complexity, although it may introduce some bias. The choice of regularization strength is crucial and typically requires tuning. Regularization is widely used in various models like linear regression, logistic regression, and neural networks.

Press Esc to exit full screen

Regularization

small values $w_1, w_2, \cdots, w_n, b$

simpler model $w_3 \approx 0$

less likely to overfit $w_4 \approx 0$

size	bedrooms	floors	age	avg income	...	distance to coffee shop	price
x_1	x_2	x_3	x_4	x_5		x_{100}	y

$n = 100$

$w_1, w_1, w_2, \cdots, w_{100}, b$ n features

$$J(\vec{w}, b) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)^2 + \overbrace{\frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2}^{\text{regularization term}} + \overbrace{\frac{\lambda}{2m} b^2}\right]$$

can include or exclude b

"lambda"

regularization parameter

$\lambda > 0$

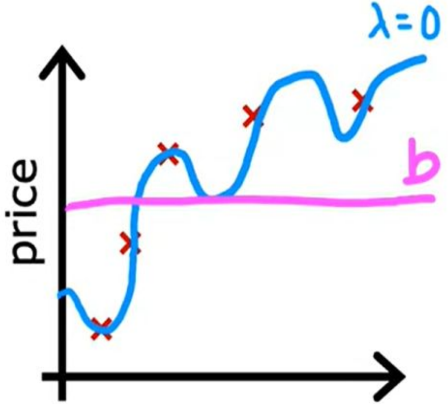
Regularization

$$\min_{\vec{w},b} J(\vec{w},b) = \min_{\vec{w},b}\left[\overbrace{\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2}^{\text{mean squared error}} + \overbrace{\frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2}^{\text{regularization term}}\right]$$

fit data

Keep w_j small

λ balances both goals



choose $\lambda = 10^{10}$

$$f_{\vec{w},b}(\vec{x}) = \underbrace{\cancel{w_1 x}}_{\approx 0} + \underbrace{\cancel{w_2 x^2}}_{\approx 0} + \underbrace{\cancel{w_3 x^3}}_{\approx 0} + \underbrace{\cancel{w_4 x^4}}_{\approx 0} + b$$

$$f(x) = b$$

Choose λ

Thank You