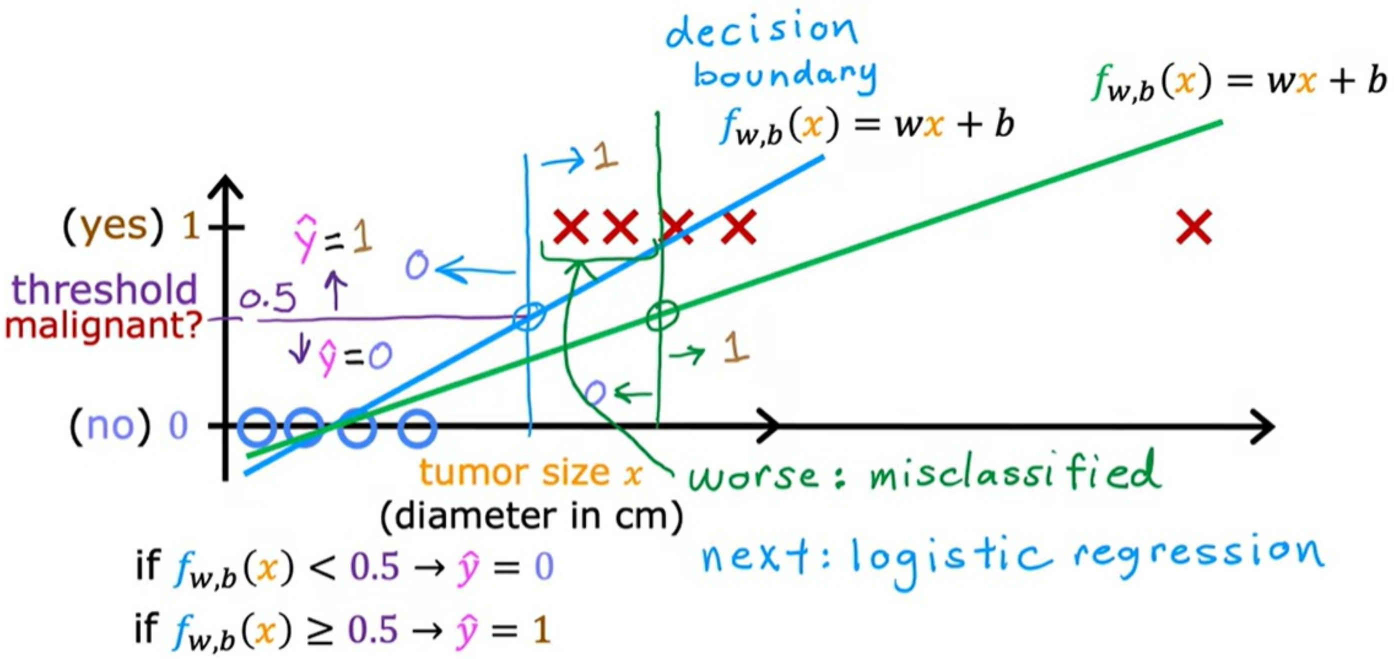
Week 3

Classification with Logistic Regression

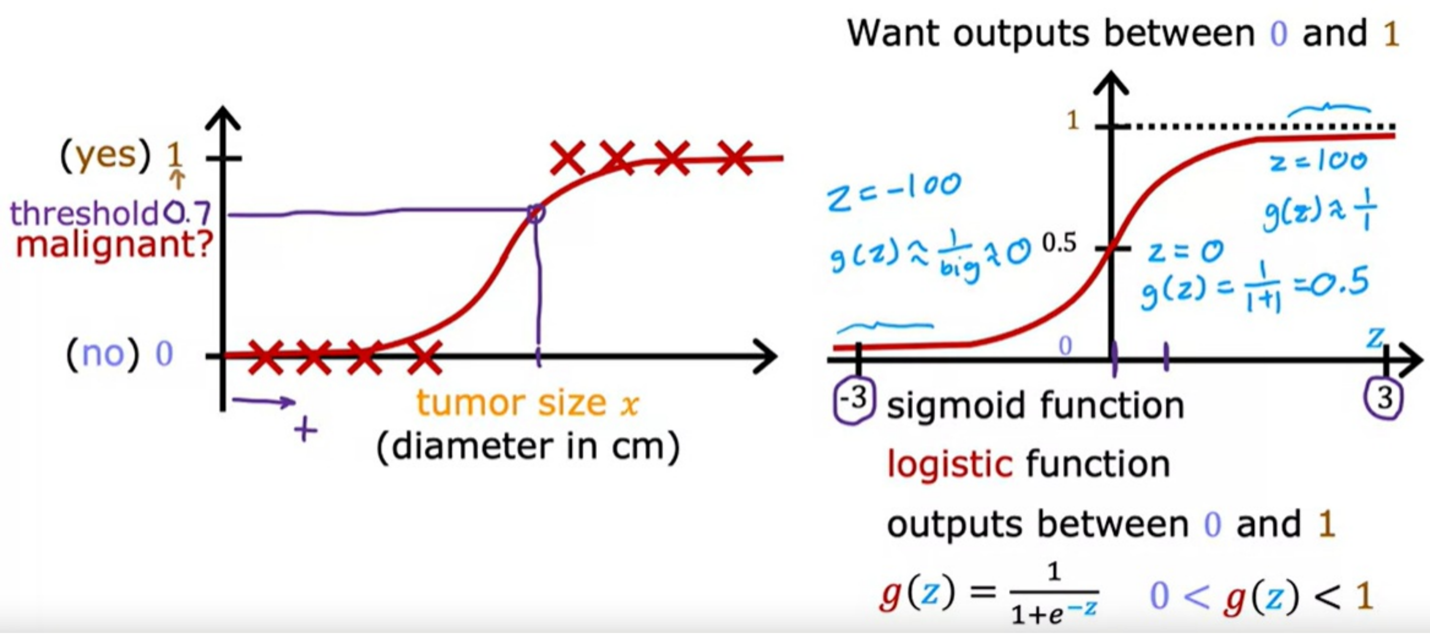
Why not to use Linear Regression on Classification Problems

- As more data is added, the best fit line can change drastically along with the decision boundary threshold → bad as threshold shouldn't be shifting



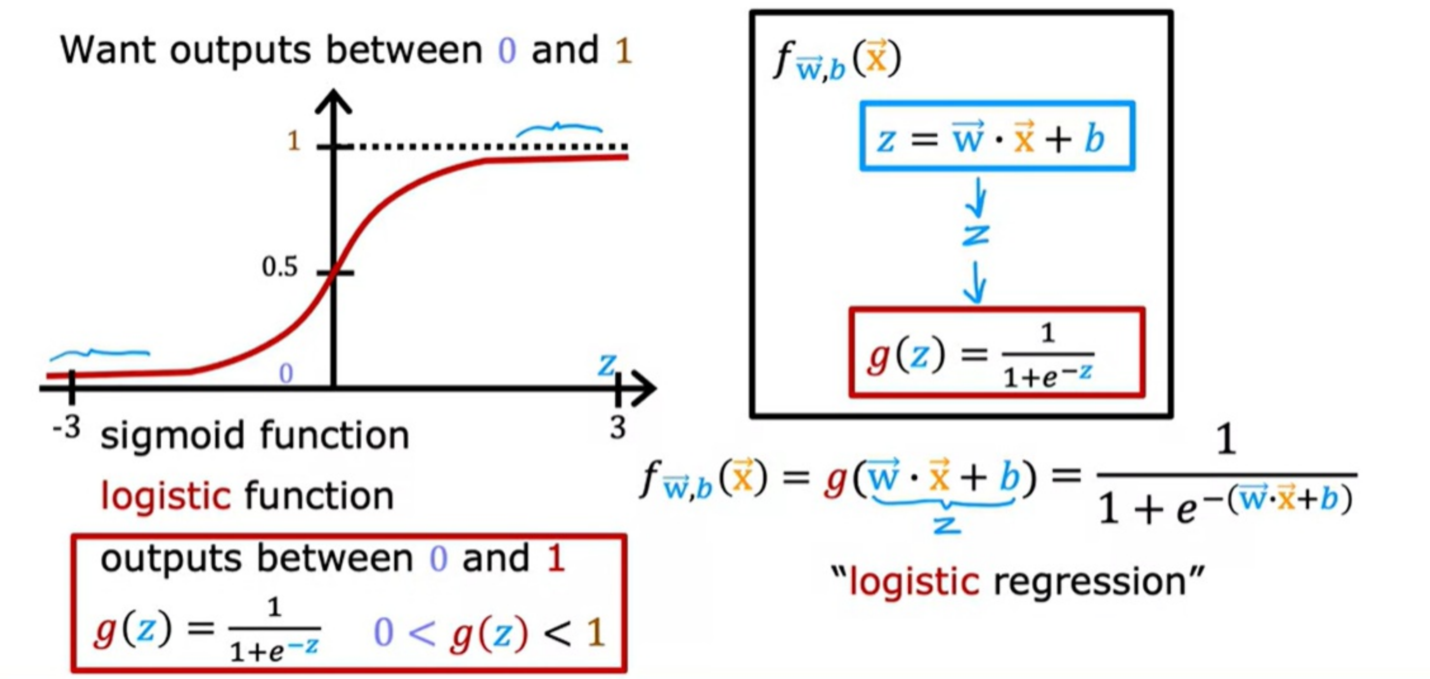
Logistic Regression

- Sigmoid function
- zzz is some constant and eee is euler's number

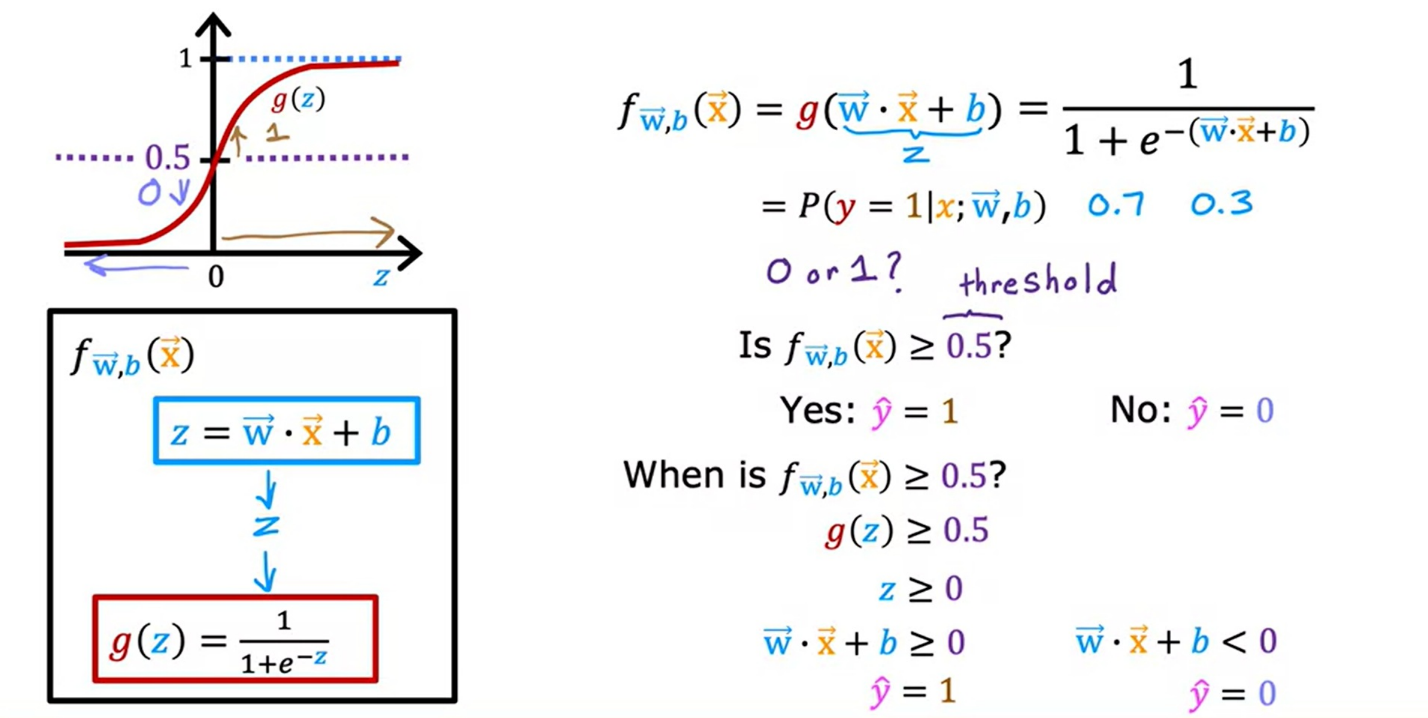


Calculating the logistic function

- Outputting the probability that the function will return 1 (probability of y = 1).
- 0≤f(x)≤10 ≤ f(x) ≤ 1 0 ≤ f(x) ≤ 1



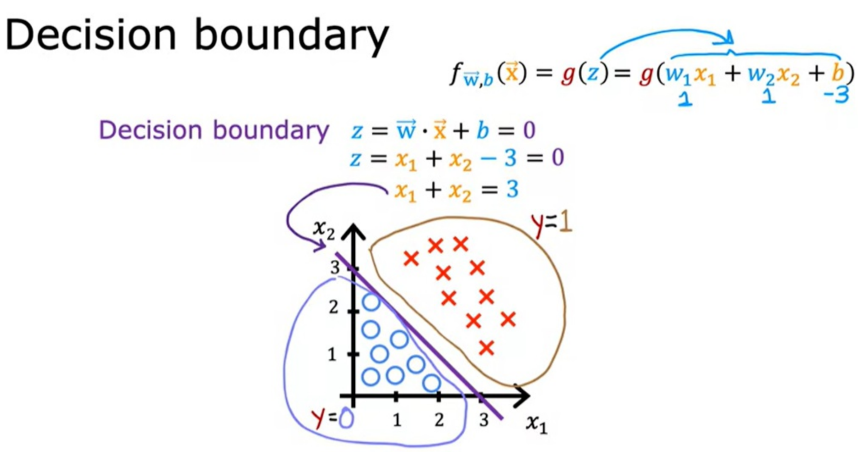
- If z is huge, then g(z) will be closer to 1, vice versa.



Decision Boundary

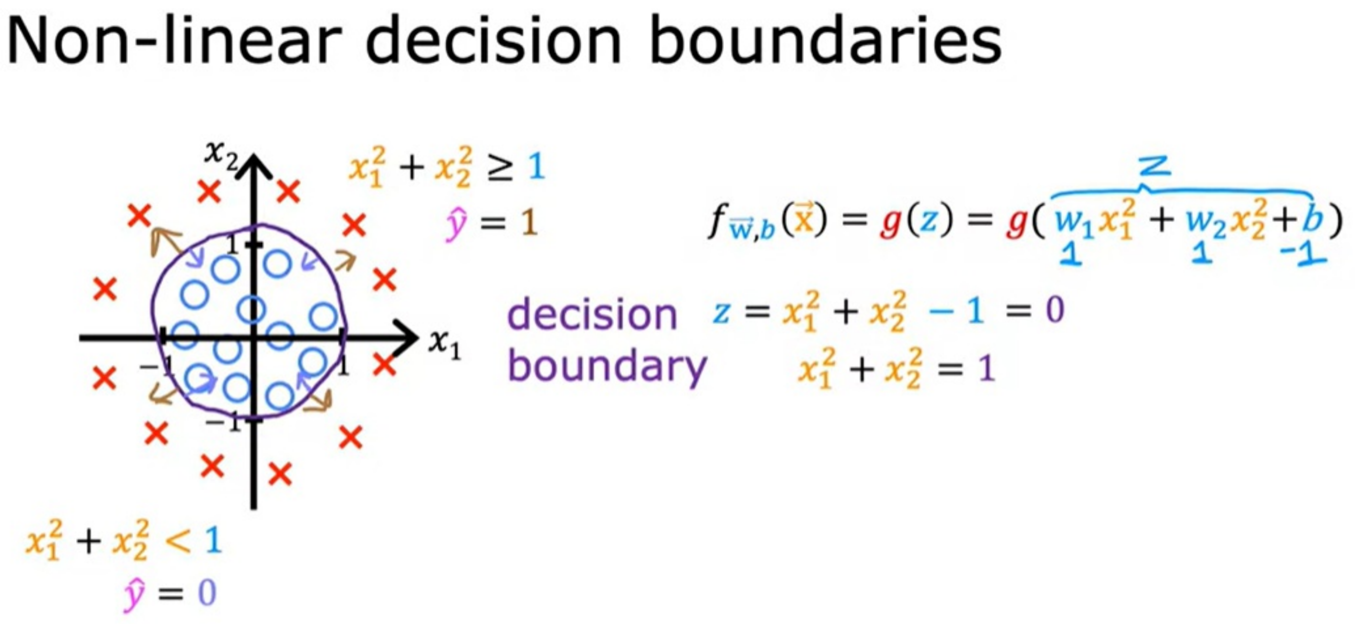
Linear decision boundary

- Calculated by assuming w=1w=1w = 1, then solving for x1x_1x1 and x2x_2x2



Non-linear decision boundary

- polynomial terms form the decision boundary

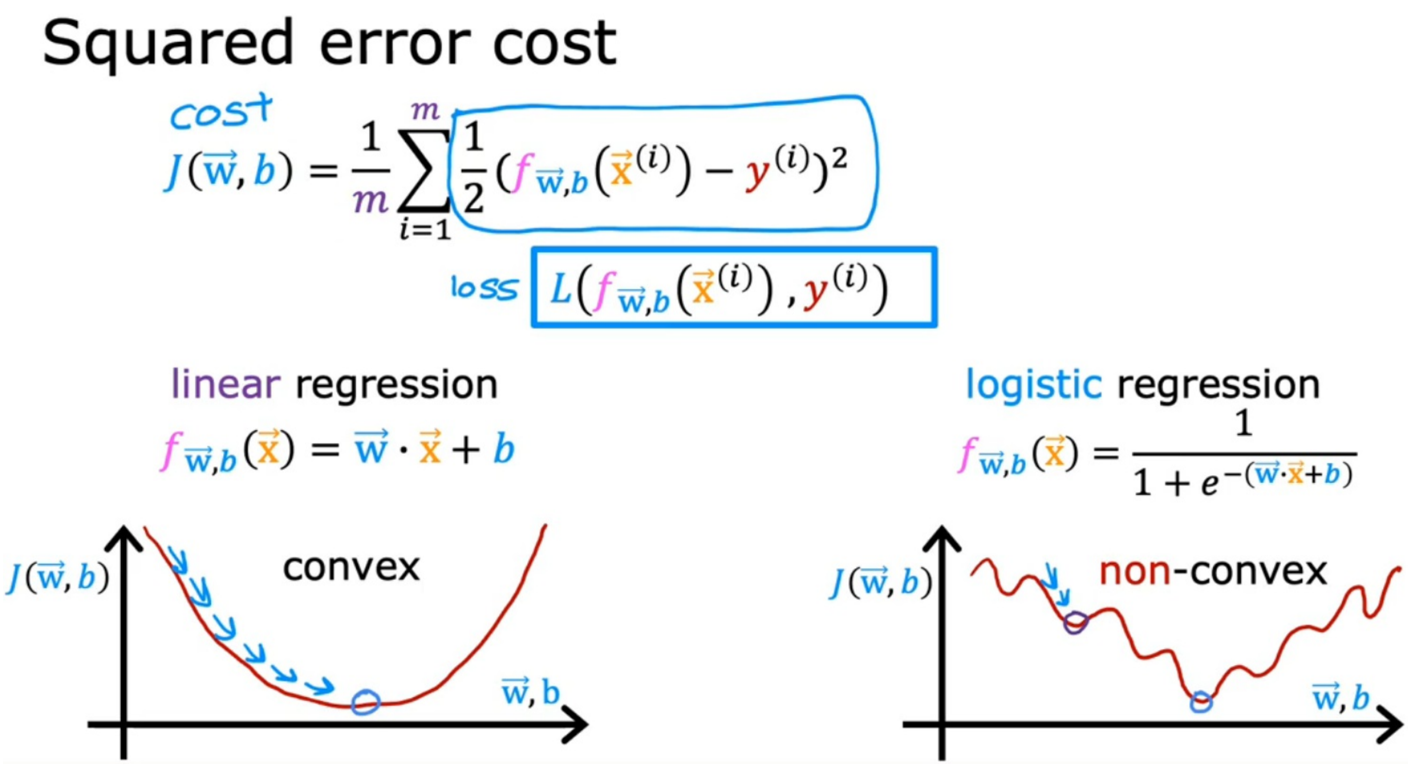


→ low threshold → greater probability that y will be 1

Cost Function for Logistic Regression

Using Cost function in Linear vs Logistic regression

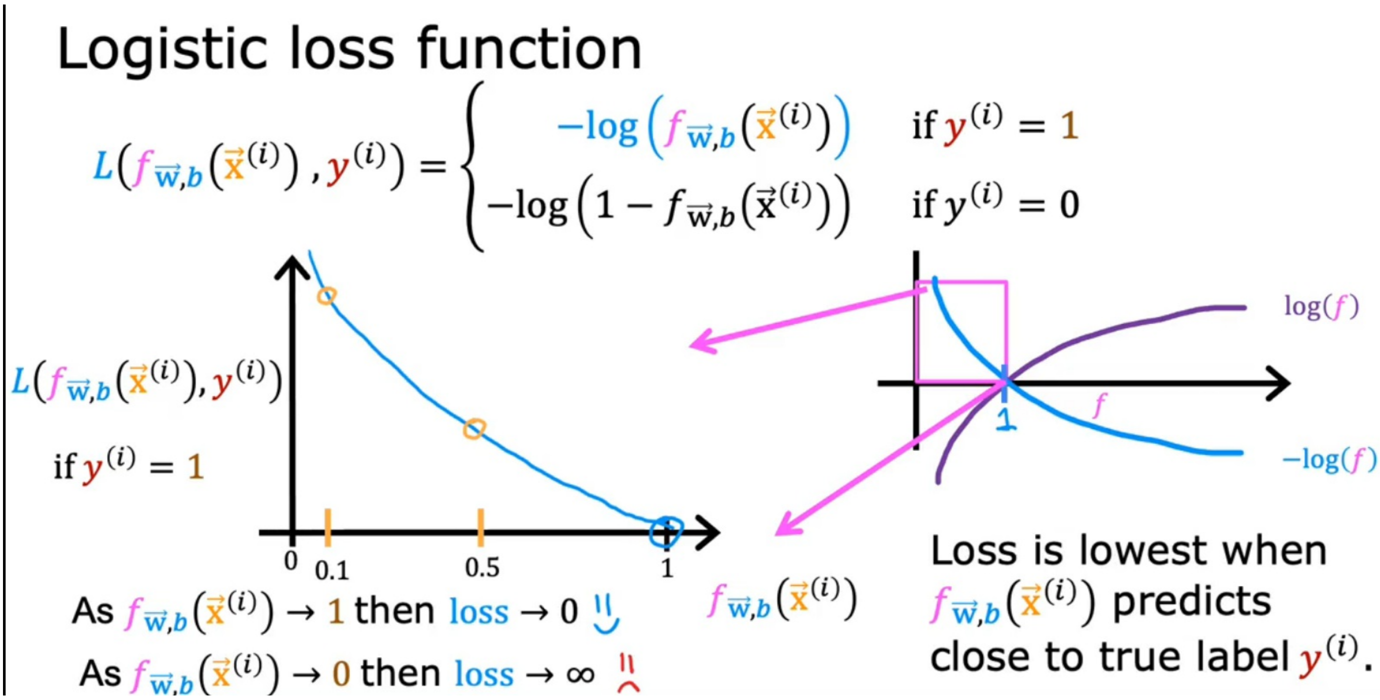
- Squared error cost function cannot be used for logistic regression as the cost function will be **stuck** at a local minima.
- **Notice that 1/2 is inside the summation**.



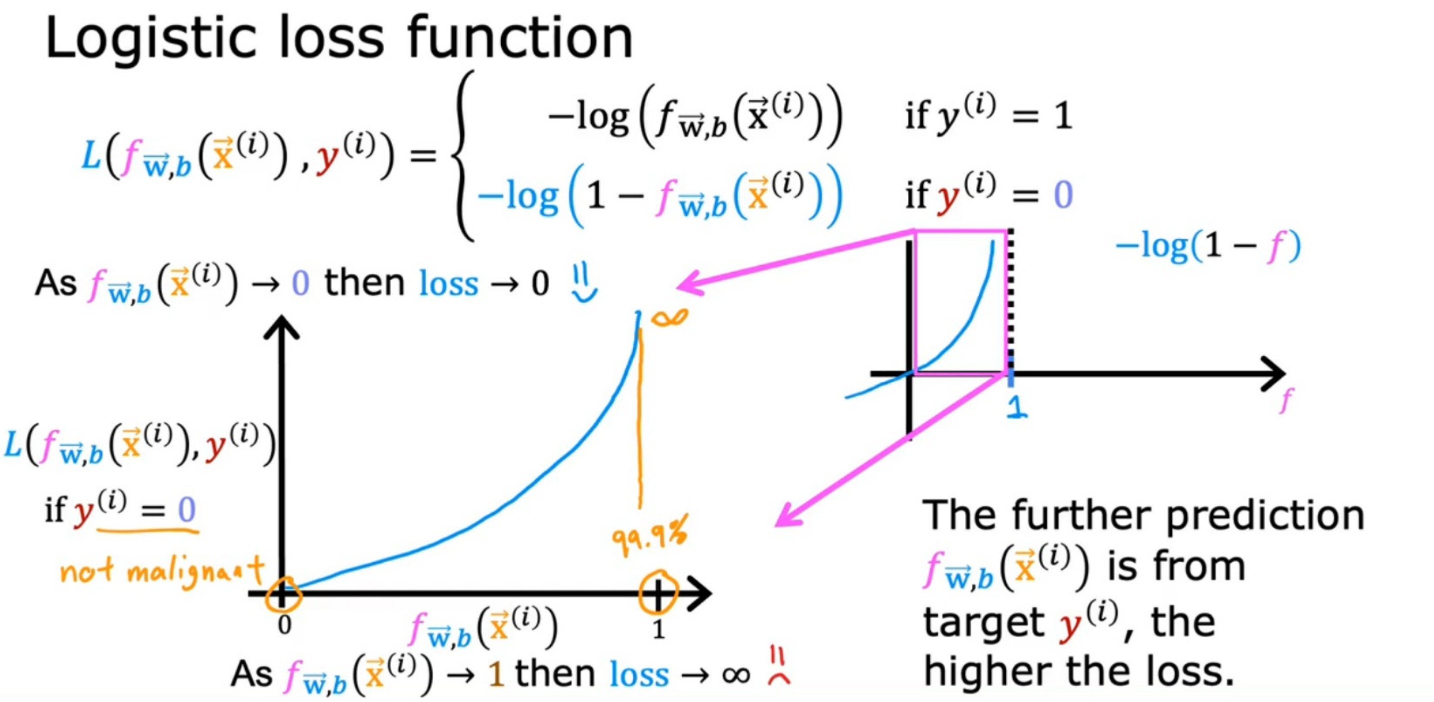
Logistic Loss Function

- Reminder: The loss function measures the acccuracy of one one training set as it measures the entire training set
- Model is penalised if prediction is 0.99 when it is supposed to be 0 → e.g Model predicts tumor to be 0.99 malignant but is actually benign
- y-axis: Loss
- x-axis: value of f (between 0 and 1 only)

When actual data is positive (y=1):



When actual data is negative (y=0):



Simplified Loss Function

- Cases of yi=0y^i = 0yi = 0 and yi=1y^i =1yi = 1

Simplified loss function

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

$$\boxed{L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -\underbrace{\cancel{y^{(i)}\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)}}_{0} - \underbrace{\left(1 - y^{(i)}\right)}_{(1-0)}\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)}$$

if $y^{(i)} = 1$:

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

if $y^{(i)} = 0$:

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = \underbrace{-(1-0)\log\left(1 - f(\vec{x})\right)}$$

- **Reminder: cost function measures the average loss across datasets**
- Cost function rewritten using the simplified loss function formula:

Simplified cost function

loss

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

cost

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

convex
(single global minimum)

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

maximum likelihood
(don't worry about it!)

Gradient Descent for Logistic Regression

- Same concept as Linear Regression
- Simultaneously updates w and b to minimize the cost
- Feature scaling is applied the same way

Gradient descent for logistic regression

repeat { looks like linear regression!

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

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

} simultaneous updates

Same concepts:
- Monitor gradient descent (learning curve)
- Vectorized implementation
- Feature scaling

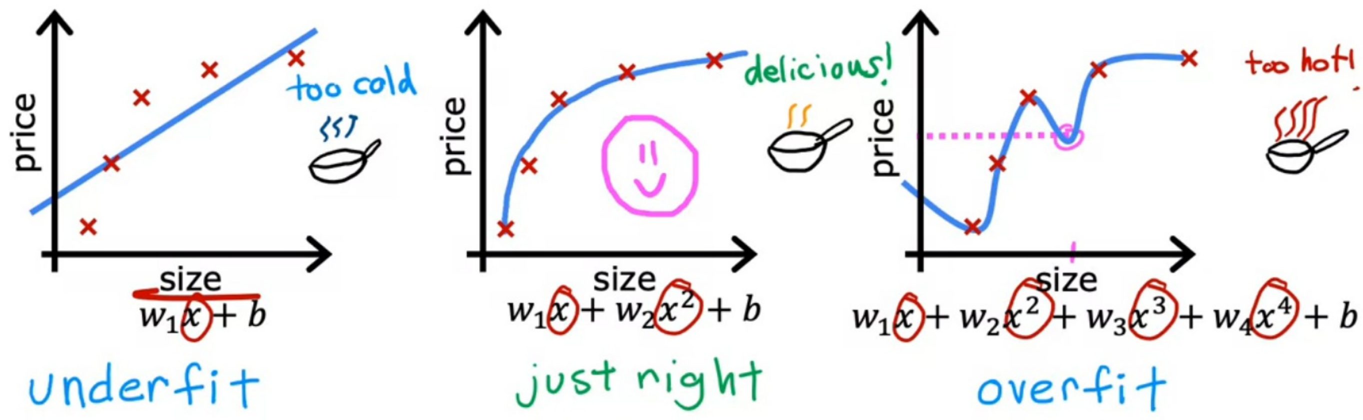
Linear regression $f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$

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

Overfitting

- Underfitting → Too few features → Low variance, high bias
- Overfitting → Too many features, model tries too hard to fit all the data and result in a weird curve that leads to inaccurate predictions → Model does not generalize well to new examples

Regression example



- Does not fit the training set well

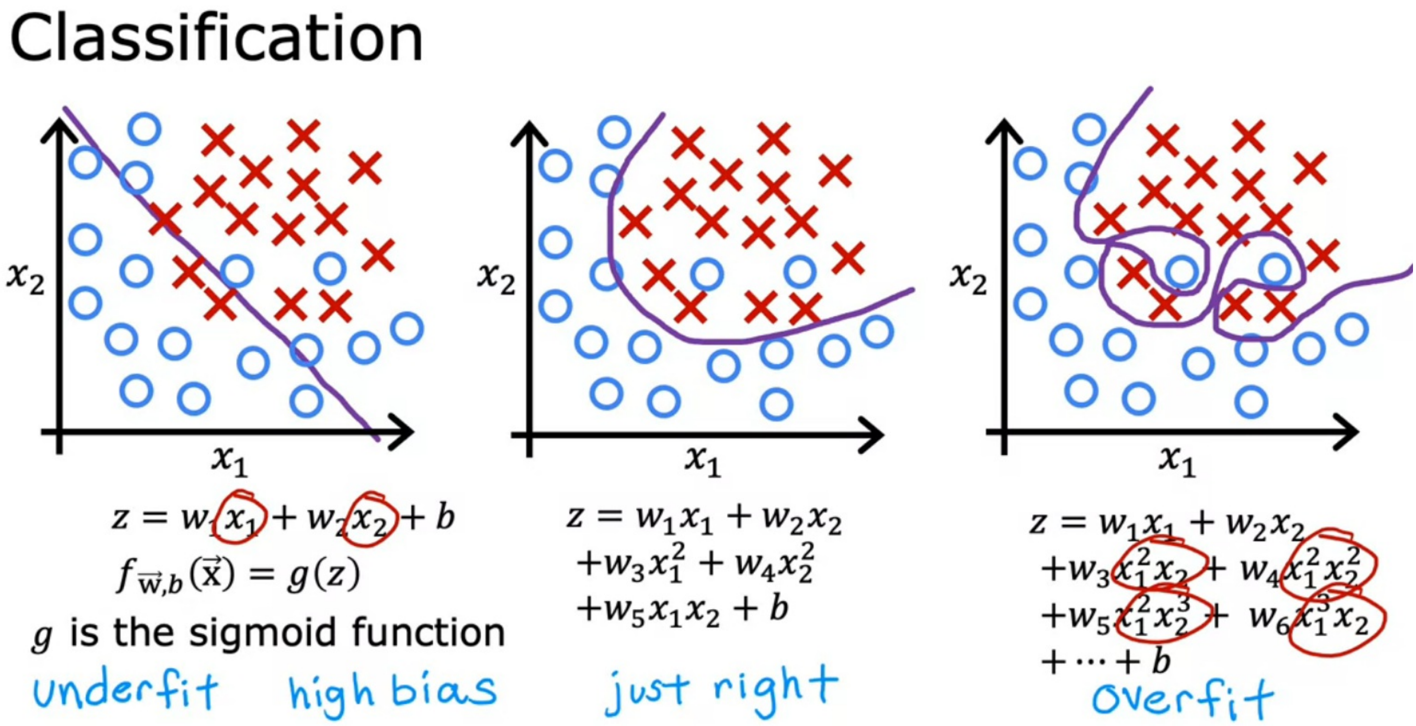
high bias

- Fits training set pretty well

generalization

- Fits the training set extremely well

high variance

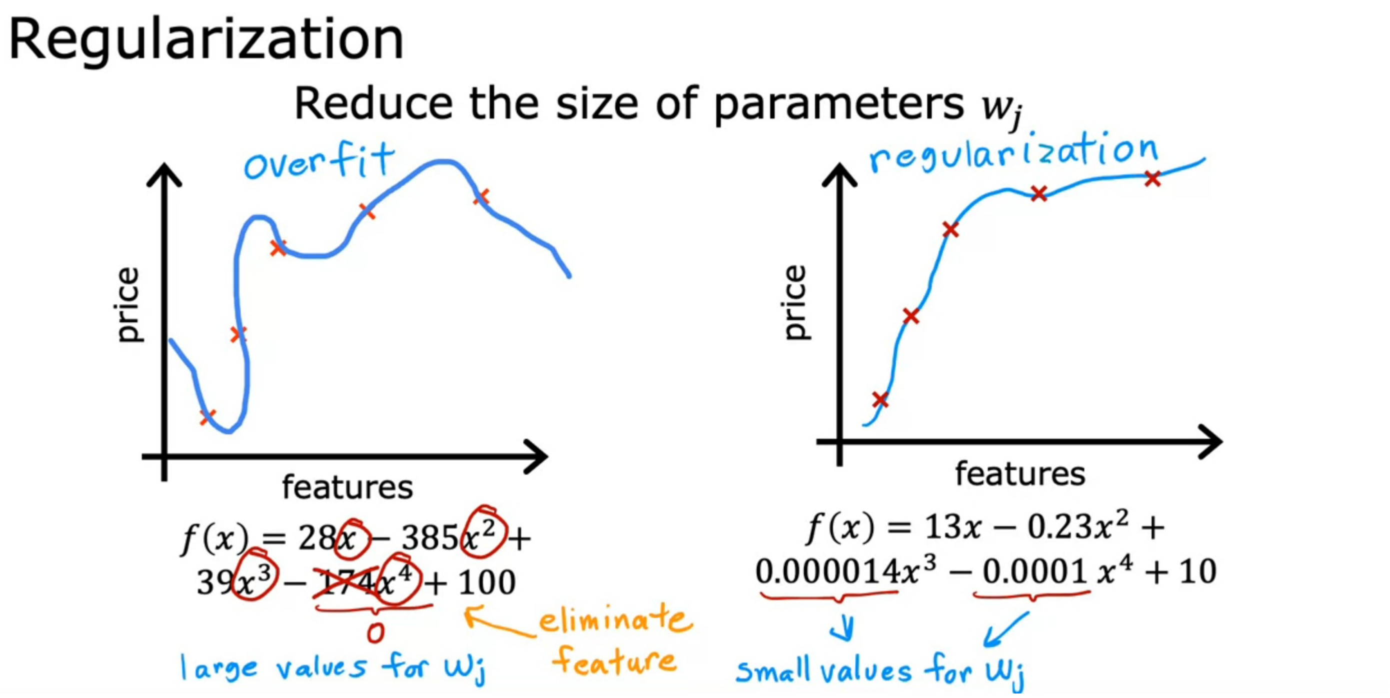


Addressing overfitting

- Collect more training data
- Select features to include/exclude (Maybe exclude some features) → Pick the most appropriate features to use especially when you don't have much data. However, useful features could potentially be lost.

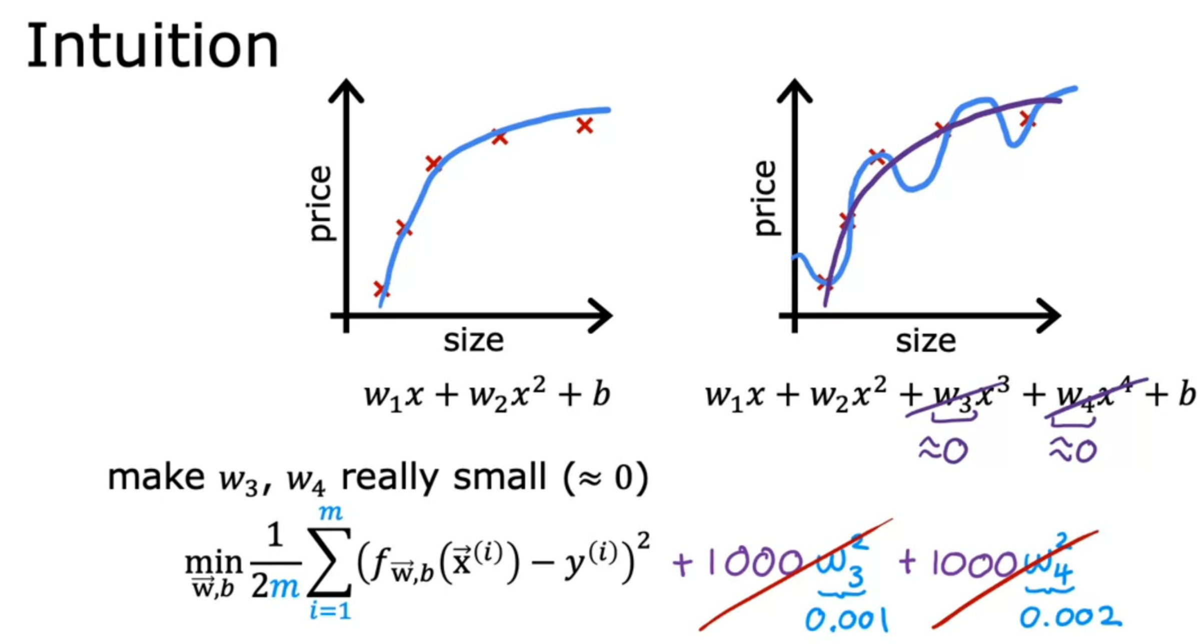
Regularization

- Reduce size of parameters
- By convention, regularize w only and not b

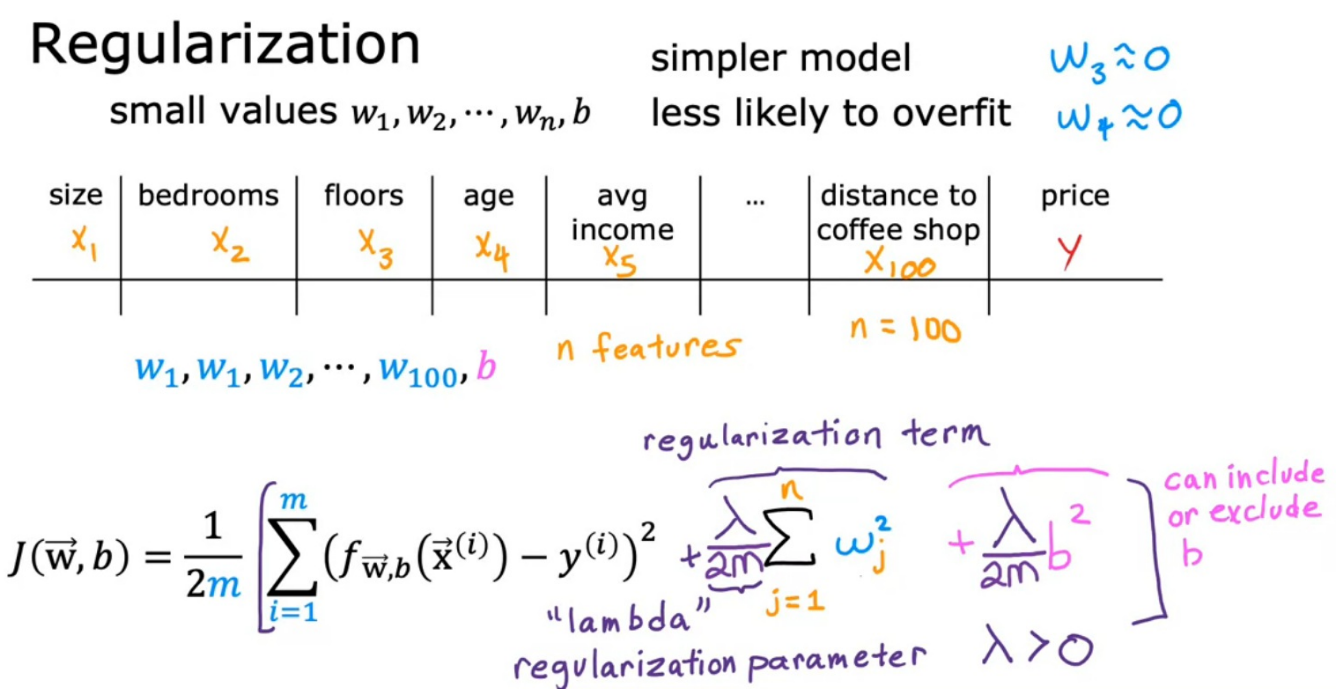


Cost function with regularization

- Idea: Penalize the model for having a huge w value by multiplying a large number and adding it outside of the cost function → Prevent overfitting



- Reminder: By convention, b is not regularized
- Lambda = regularization parameter
- Choose lambda appropriately
- Penalize on every w parameter



Regularization for Gradient Descent

- For both linear and logistic regression

Implementing gradient descent

repeat {

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

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

} simultaneous update $j = 1 \ldots n$

$$w_j = \underbrace{1 w_j - \alpha \frac{\lambda}{m} w_j}_{w_j \left(\underbrace{1 - \alpha \frac{\lambda}{m}}_{\text{shrink } w_j} \right)} - \underbrace{\alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}\left(\vec{x}^{(i)}\right) - y^{(i)} \right) x_j^{(i)}}_{\text{usual update}}$$

$$\alpha \frac{\lambda}{m}$$

$$0.01 \frac{1}{50} = 0.0002$$

$$w_j (\underbrace{1 - 0.0002}_{0.9998})$$