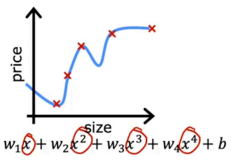
The problem of Overfitting

The problem of overfitting

Overfit vs Underfit

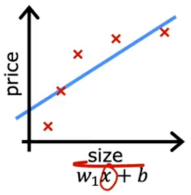
- **Overfitting** occurs when a model learns the training data too well, capturing noise and fluctuations, leading to poor generalization on new data. This is often associated with high variance.



⇒ Fit training data set extremely well.

⇒ **High variance**

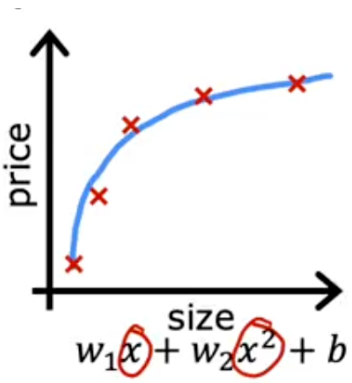
- **Underfitting** happens when a model is too simple to capture the underlying patterns in the data, resulting in poor performance on both training and new data. This is linked to high bias.



⇒ Don't fit training data set well.

⇒ **high bias**

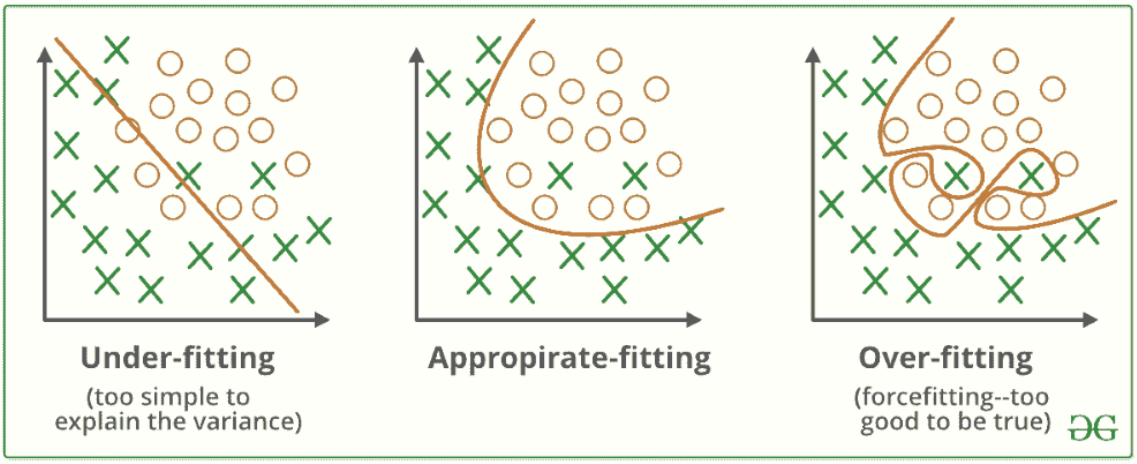
⇒ A fourth-order polynomial model may overfit by fitting the training data perfectly but failing to **generalize** to new examples, indicating high variance.



Just right

⇒ Fit training data set pretty well.

⇒ **generalization**



source: GeekForGeeks

What are Bias and Variance?

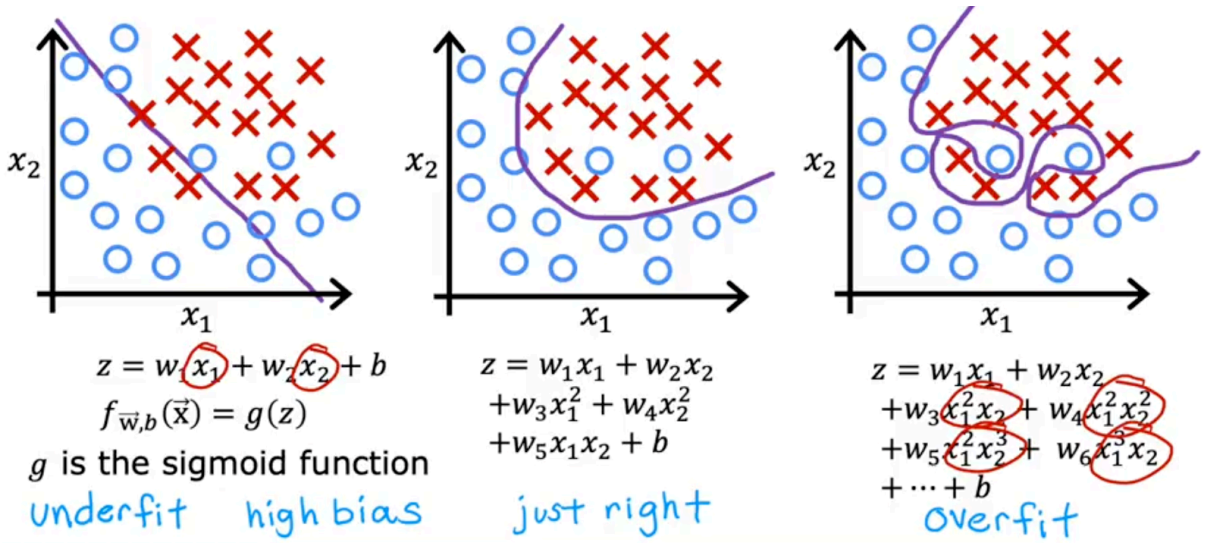
source: GeekForGeeks

- **Bias** refers to the errors which occur when we try to fit a statistical model on real-world data which does not fit perfectly well on some mathematical model. If we use a way too simplistic a model to fit the data then we are more probably face the situation of **High Bias** (underfitting) refers to the case when the model is unable to learn the patterns in the data at hand and perform poorly.
- **Variance** shows the error value that occurs when we try to make predictions by using data that is not previously seen by the model. There is a situation known as **high variance** (overfitting) that occurs when the model learns noise that is present in the data.

Finding the Right Balance

- The goal in machine learning is to find a model that is "just right," meaning it neither underfits nor overfits the data. This balance is crucial for effective predictions on unseen data.

In Classification



💡 **QUES:** Our goal when creating a model is to be able to use the model to predict outcomes correctly for **new examples**. A model which does this is said to **generalize** well.

When a model fits the training data well but does not work well with new examples that are not in the training set, this is an example of:

- A model that generalizes well (neither high variance nor high bias)
- Overfitting (high variance) 👍🏻
- None of the above
- Underfitting (high bias)

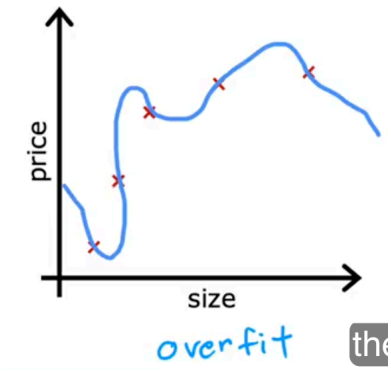
Explain: This is when the model does not generalize well to new examples.

Addressing Overfitting

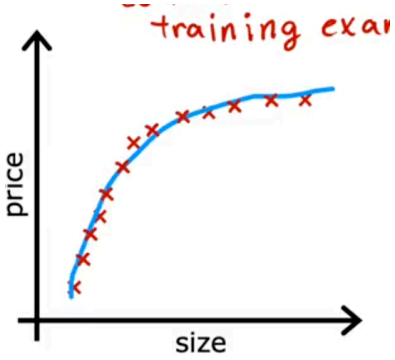
Collect More Data

Increasing the size of the training dataset can help the learning algorithm generalize better and reduce overfitting.

⇒ overfitting happens with small data.



⇒ more data ⇒ fit well



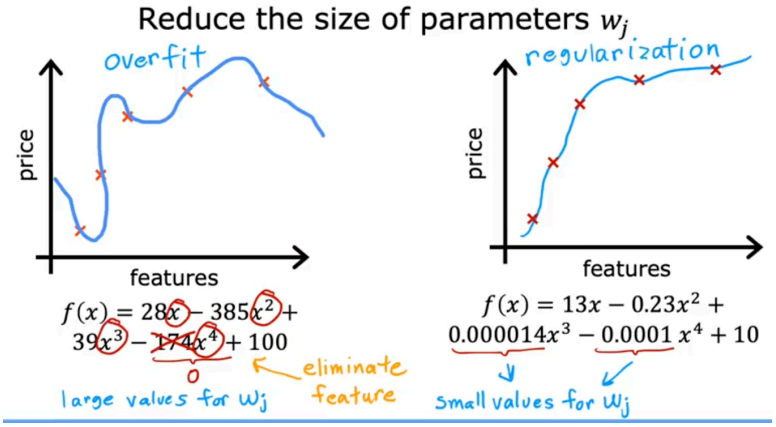
Feature Selection

(in course 2): Reducing the number of features used in the model can also mitigate overfitting. Selecting only the most relevant features can improve model performance.

Reduce size of parameter

Regularization: This technique helps to control overfitting by shrinking the values of the model parameters, allowing all features to be retained while minimizing their impact.

Parameter Management: Regularization encourages smaller parameter values, which can lead to better fitting of the training data without completely discarding any features.



💡 **QUES:** Applying regularization, increasing the number of training examples, or selecting a subset of the most relevant features are methods for...

- Addressing overfitting (high variance) 👍
- Addressing underfitting (high bias)

Explain: These methods can help the model generalize better to new examples that are not in the training set.

Cost function with regularization

What is Regularization?

- Regularization helps prevent **overfitting** by keeping the model's parameters (like $w_1, w_2, \ldots, w_n$) small.
- It does this by *modifying* the **cost function** to include a **penalty** for large weights, encouraging simpler, more generalizable models.

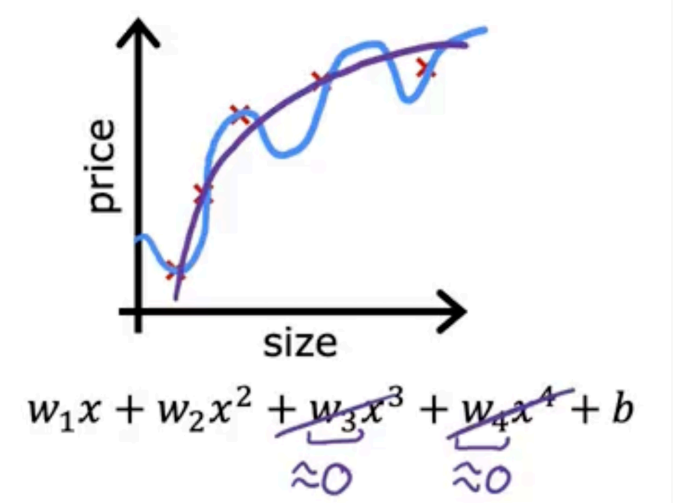
💡 Because our target is always minimize the cost function ⇒ Therefore, if we add more
$1000w_3^2 + 1000w_4^2$ then to minimize cost function the values of w_3, w_4 must be as small as possible

make w_3, w_4 really small (≈ 0)

$$\min_{\vec{w},b} \frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + 1000\,w_3^2 + 1000\,w_4^2$$

$w_3 \approx 0.001$, $w_4 \approx 0.002$

⇒ $w_3, w_4 \approx 0$

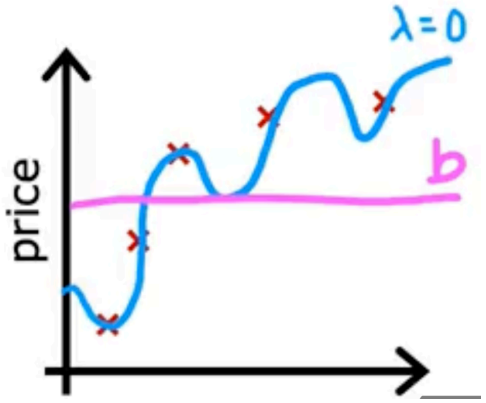


Updated Cost Function

- The new cost function includes both the **mean squared error** $[\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{i}\right) - y^{i}\right)^2]$ and a **regularization term** $(\frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2)$ (e.g., $1000 \times w_3^2$).
- This penalizes large weights, pushing the model toward smoother, less complex functions that better generalize.

Choosing λ (Lambda)

- The **regularization parameter** λ controls the strength of the penalty.
  - A small λ (like 0) may cause **overfitting**.
  - A very large λ may result in **underfitting**.
- Choosing the right λ is key to balancing **fit** and **simplicity** for optimal model performance.



Example:

Choosing $\lambda = 10^{10}$, then:

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

💡 **QUES:** For a model that includes the regularization parameter λ (lambda), increasing λ lambda will tend to...

- Increases the size of the parameters $w_1, w_2, \ldots w_n$
- Increase the size of parameter b.
- Decrease the size of parameters $w_1, w_2, \ldots w_n$. 👍👍
- Decrease the size of the parameter b.

Explain: Increasing the regularization parameter λ reduces overfitting by reducing the size of the parameters. For some parameters that are near zero, this reduces the effect of the associated features.

Regularized Linear Regression

With regularization added, the cost function changes—so the gradient descent updates must also be adjusted to include the regularization term. (the derivative)

D

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

Only a minor adjustment to w_j is needed in the gradient descent update.

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

Note: The parameter b remains **unchanged** by regularization because it isn't included as a weighted term in the regularization component of the cost function

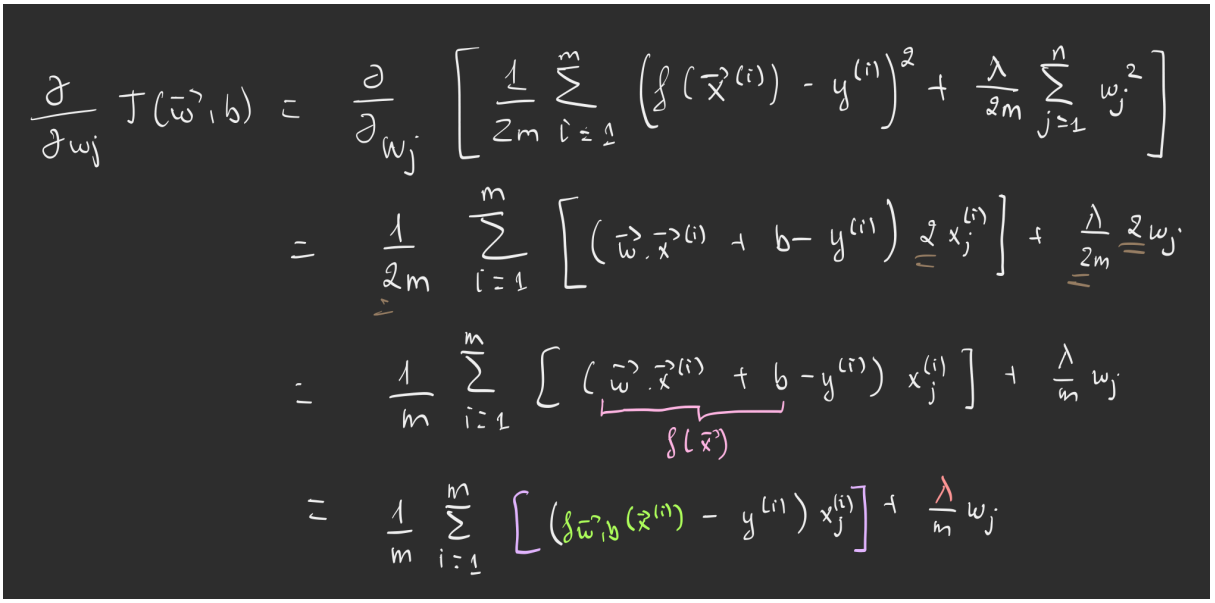
Gradient descent of regularized linear regression

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

$\text{simultaneously update } j = 1, ..., n$

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

How we get the derivative term



Regularized Logistic Regression

Applying regularization to logistic regression follows closely to linear regression. Considering the follow cost function equation for logistic regression, the only modification needed is to add the regularization term to the end of the equation,

Moving onto the gradient descent algorithm, consider the following equations,

When solving for the derivatives, they result in the nearly the exact same equations as those found for linear regression. The only difference is the model function (f_{\vec{w}, b}\left(\vec{x}^i\right)).

Combining the equations above, the complete gradient descent is formed,

Gradient descent of regularized logistic regression