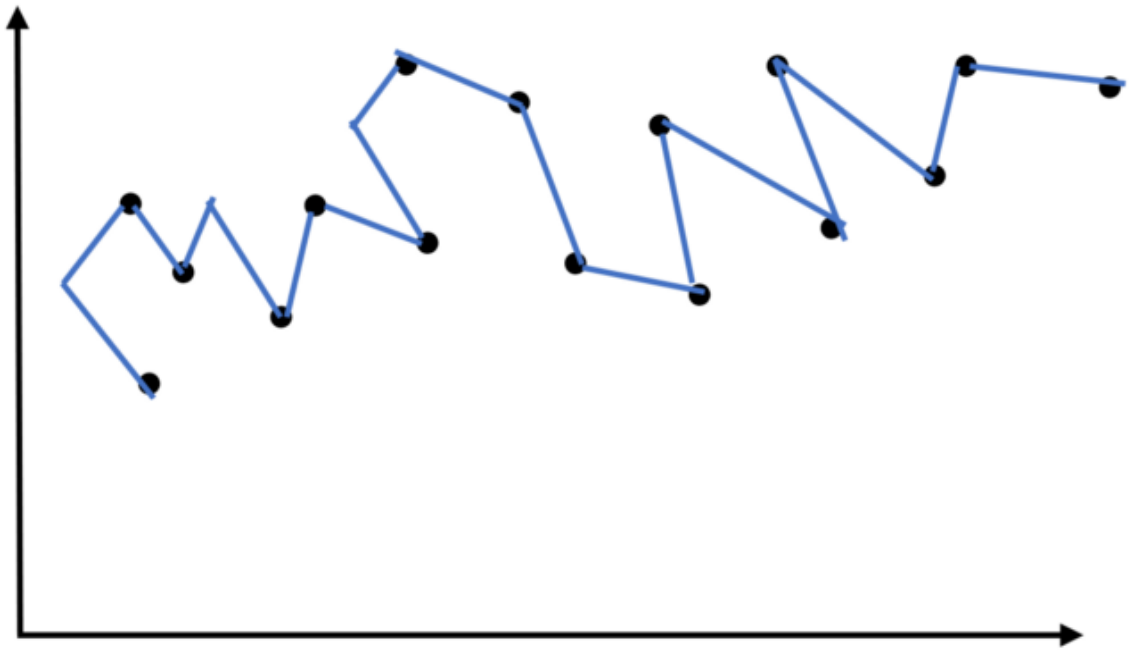
Unit 3 Classification

Problem of Overfitting

Overfitting is a modeling error that occurs when a function or model is too closely fit the training set and getting a drastic difference of fitting in test set. **Overfitting** the model generally takes the form of making an overly complex model to explain Model behavior in the data under study.



Examples of Overfitting

Let's go with examples,

Let's say we need to predict if a student will land a job interview based on his resume. Now assume we train a model from a dataset of 20,000 resumes and their outcomes.

Then we try a model out on the original dataset and it predicts outcomes with 98% Accuracy… Wow! It's Amazing, but not in Reality.

But now comes the bad news. When we run a model out on the new dataset of resumes, we only get 50% of Accuracy.

Our model doesn't get generalized well from our training data to see unseen data. This is known an **Overfitting** and it is a common problem in Data Science.

In fact, **Overfitting** occurs in the real world all the time. We need to handle it to generalize the model.

How to find Overfitting?

The primary challenge in machine learning and in data science is that we can't able to evaluate the model performance until we test it. So the first step to finding the Overfitting is to split the data into the Training and Testing set.

If our model does much better on the training set than on the test set, then we're likely overfitting.

The performance can be measured using the percentage of accuracy observed in both data sets to conclude on the presence of **overfitting**. If the model performs better on the training set than on the test set, it means that the model is likely **overfitting**. For example, it would be a big Alert if our model saw 99% accuracy on the training set but only 50% accuracy on the test set.

How to prevent Overfitting?

1. Training with more data
2. Data Augmentation
3. Cross-Validation
4. Feature Selection
5. Regularization

Regularization :

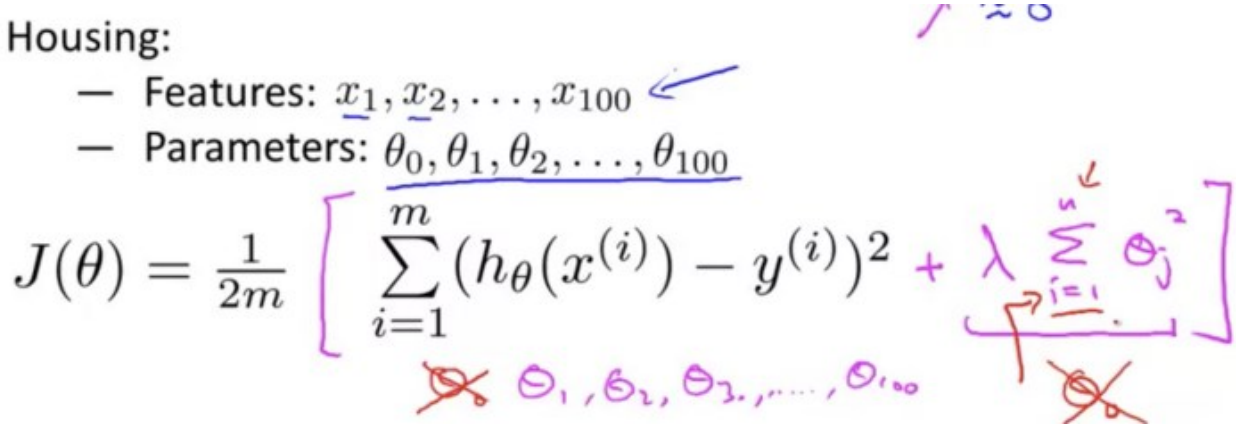
- keep all features but reduce the magnitude/value of parameters (theta j) to make the value smaller

*Works well when we have a lot of features, each of which contributes a bit to predicting y.

How does regularization work?

Regularization (makes values smaller)

- make the “simpler” hypothesis
- less prone to overfitting



Make theta 3 and theta 4 close to 0

Modify the cost function by **adding an extra regularization term** in the end to **shrink every single parameter** (e.g. close to 0)

lambda (regularization parameter) controls the tradeoff between two goals:

former formula — 1st goal: fit the training data well

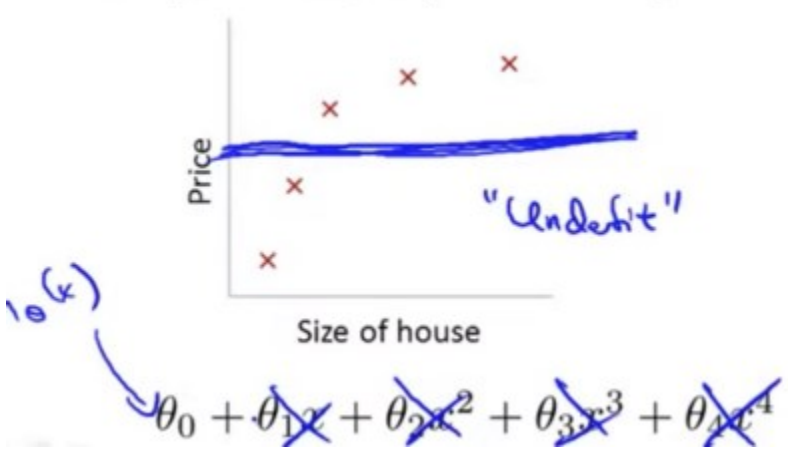
extra lambda (purple) — 2nd goal: keep the parameters small to avoid overfitting

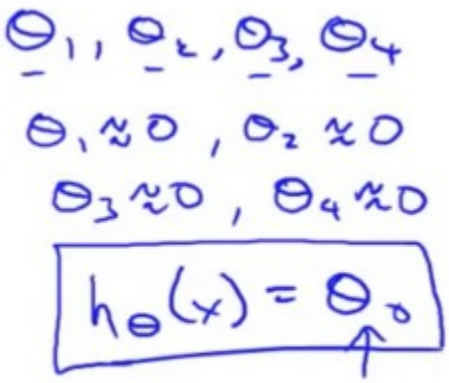
In regularized linear regression

In regularized linear regression, we choose θ to minimize

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?





If all parameters (theta) are close to 0, the result will be close to 0. -> it will generate a **flat straight line** that **fails to fit the features well** → **underfit**

- To sum up, if **lambda** is chosen to be too **large,** it may **smooth out the function too much** and cause **underfitting**.