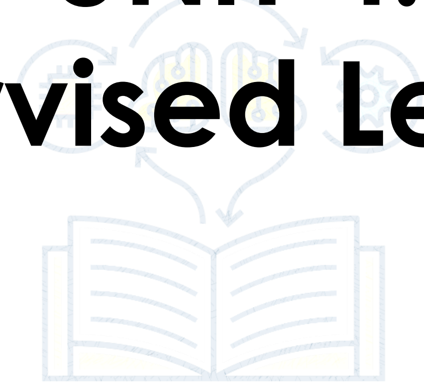
UNIT 4:
Supervised Learning

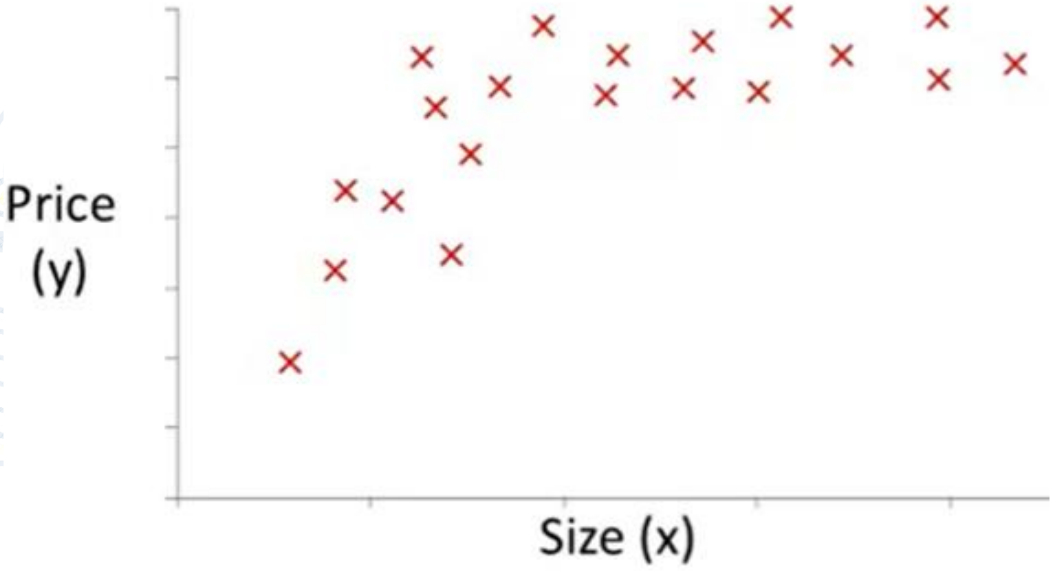


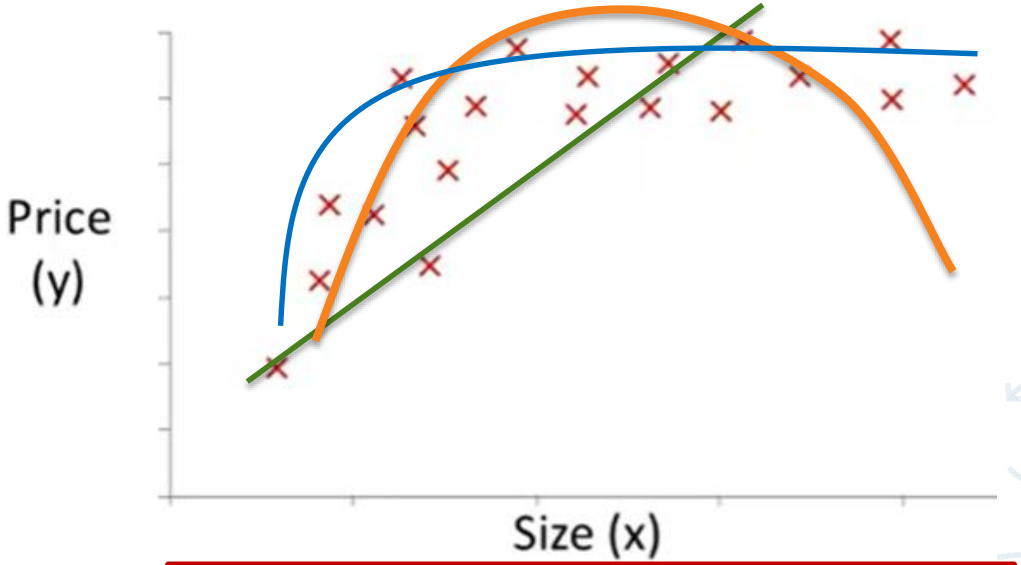
Prepared by Nima Dema

Non-Linear Regression

Polynomial Regression

- Linear regression requires the relation between the dependent variable and the independent variable to be linear.
- Our function need not be linear (a straight line) if that does not fit the data well.





$$y = b + wx$$

$$y = b + w_1 x + w_2 x^2$$

$$y = b + w_1 x + w_2 x^2 + w_3 x^3$$

$$f(x) = y = b + w_1 x_1 + w_2 x_2 + w_3 x_3$$

$$= b + w_1(\text{size}) + w_2(\text{size})^2 + w_3(\text{size})^3$$

$x_1 = size$

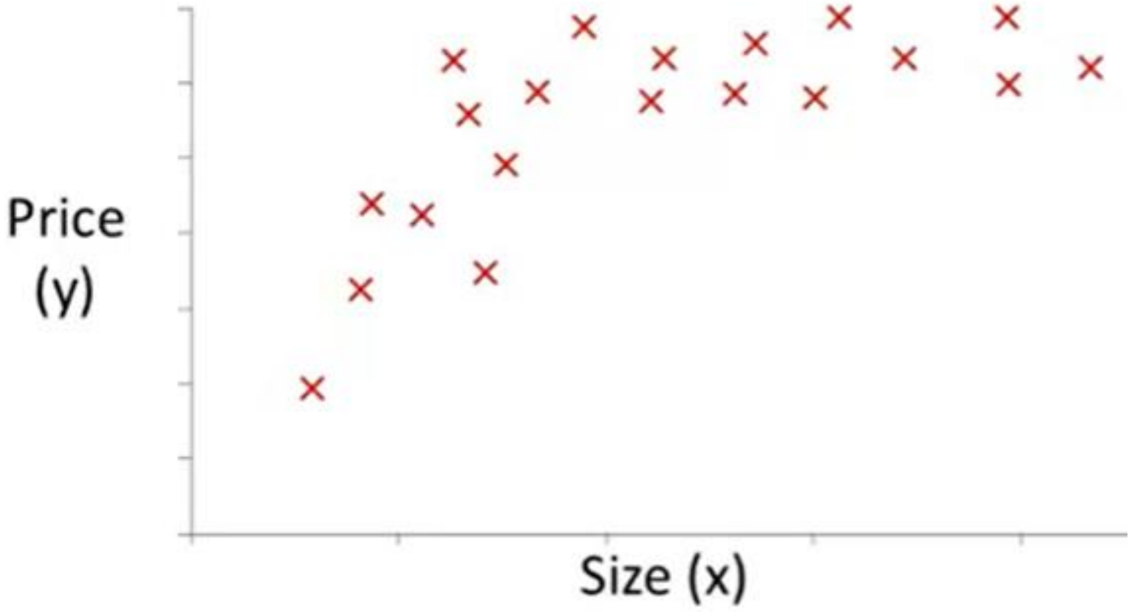
$x_2 = size^2$

$x_3 = size^3$

if x_1 has range 1 - 1000 then range of x_1^2 becomes 1 - 1000000 and that of x_1^3 becomes 1 - 1000000000

Note: One important thing to keep in mind is, if you choose your features this way then feature scaling becomes very important.

Choice of Features



- Simple Linear Regression
- Polynomial function (quadratic Function)
- Polynomial Function (Cubic)

Polynomial Regression in practice

- Implementing polynomial regression with scikit-learn is very similar to linear regression. There is only one extra step: you need to transform the array of inputs to include non-linear terms such as x^2

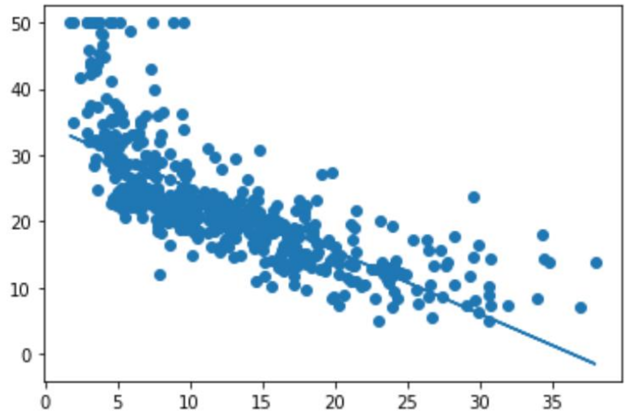
```python
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
```

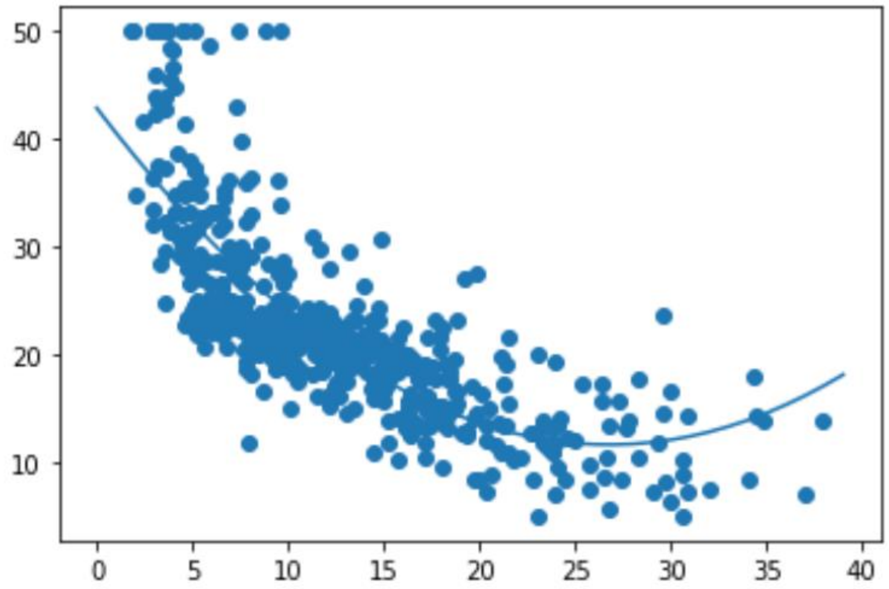
- you need to include x^2 (and perhaps other terms) as additional features when implementing polynomial regression.

- Transform the input array X to contain the additional column(s) with the values of x^2 and many more features.

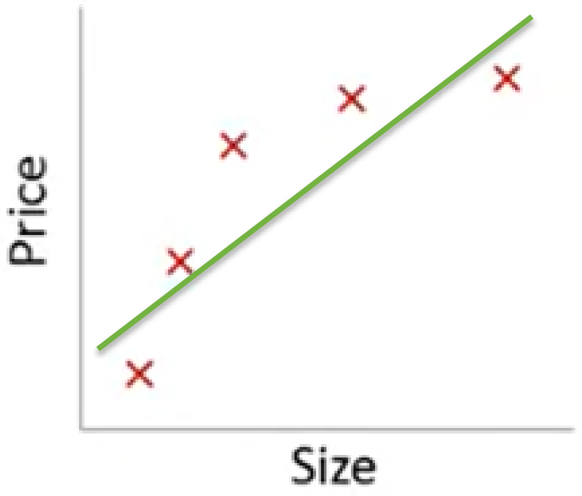
Polynomial Regression in practice

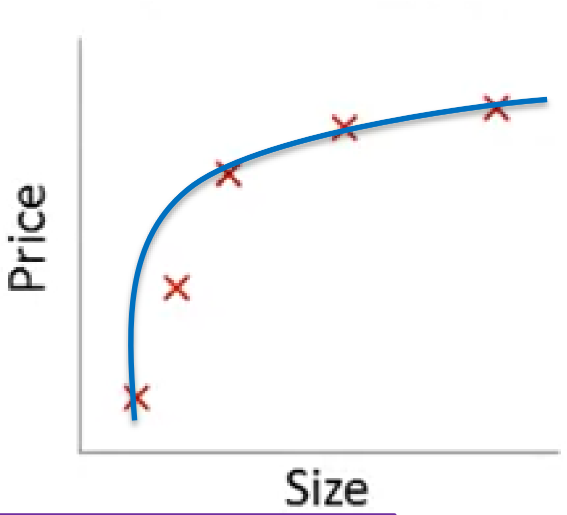
```
poly_features = PolynomialFeatures(degree=2)
X_train_poly = poly_features.fit_transform(X.reshape(-1,1))

model2 = LinearRegression()
model2.fit(X_train_poly,y)
```

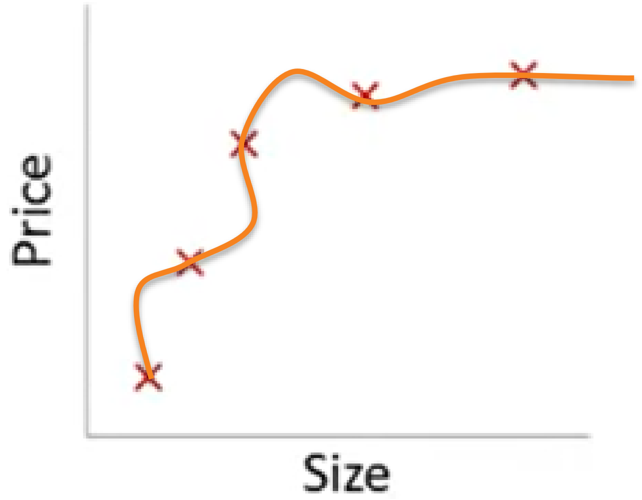




The Problem of Over fitting and Underfitting







$y = b + wx$

$y = b + w_1 x + w_2 x^2$

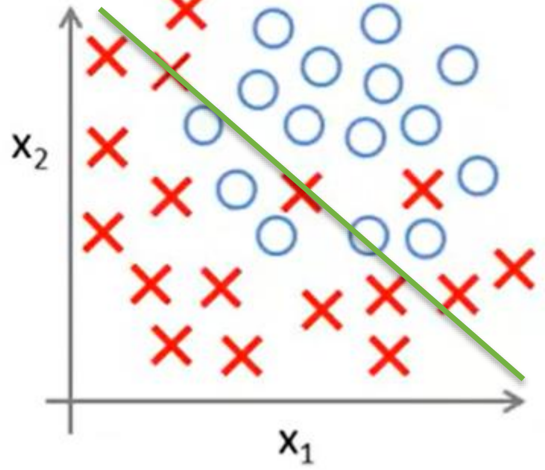
$y = b + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$

" Underfit" and "High bias"

" Just right"

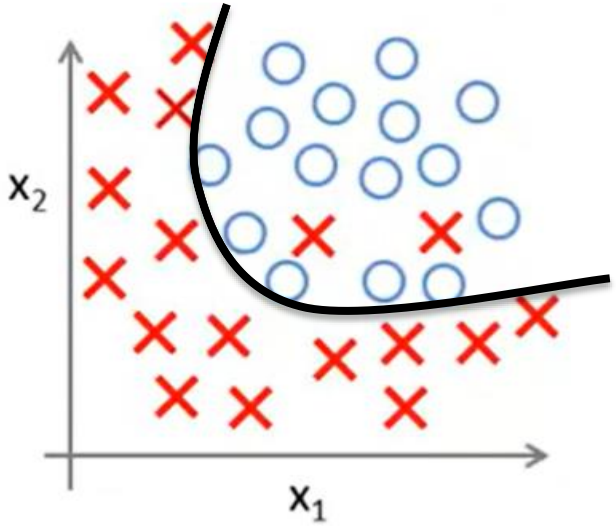
" Overfit" and "High variance"

The Problem of Over fitting and Underfitting

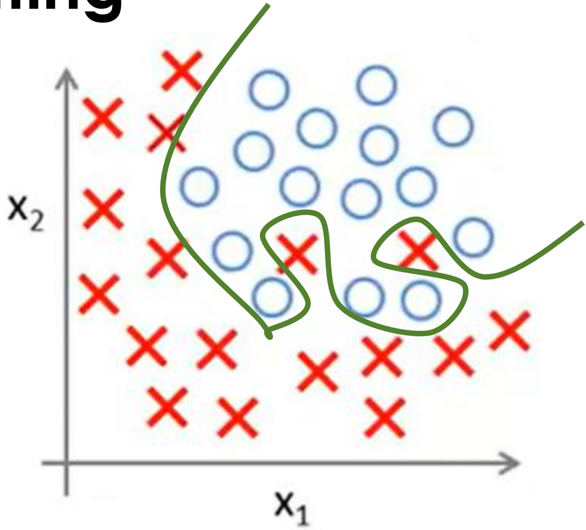


$f(x) = g(b + w_1x_1 + w_2x_2)$

" Underfit" and "High bias"



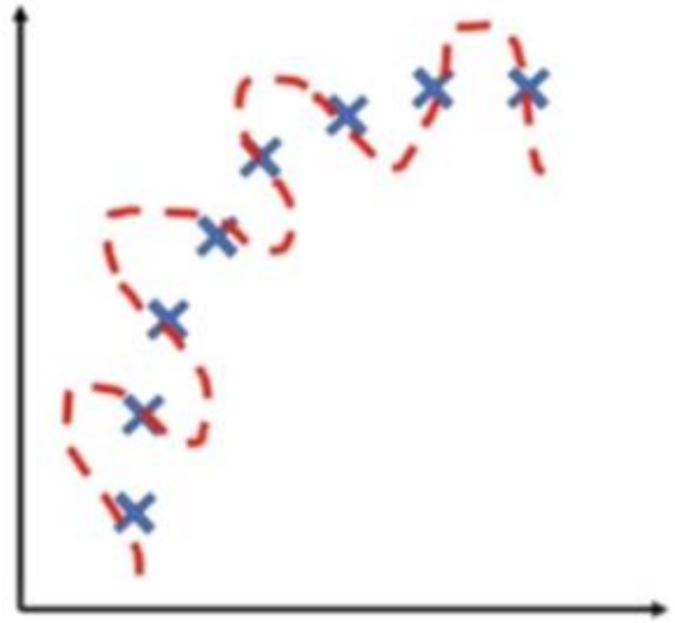
$g(b + w_1x_1 + w_2x_2 + w_3\,x_1^2 + w_4\,x_2^2 + w_5\,x_1x_2)$



$g(b + w_1x_1 + w_2x_2 + w_3\,x_1^2 \ldots + w_6\,x_2^4 \ldots)$

" Overfit" and "High variance"

The Problem of Over fitting

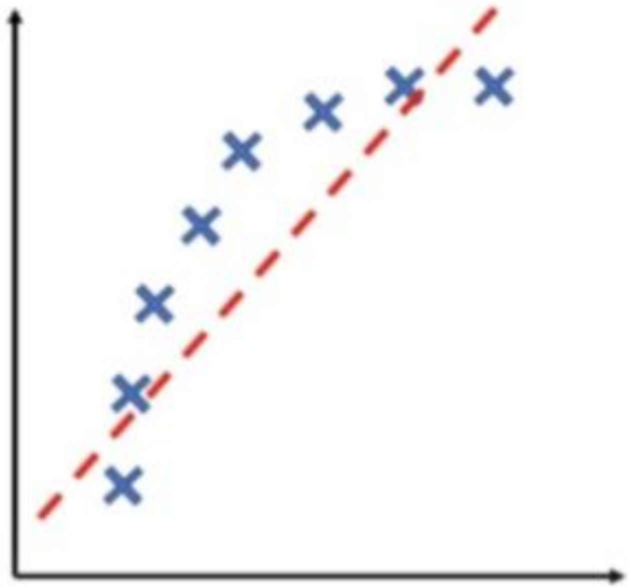


- Variance defines the algorithm's sensitivity to specific sets of data. A model with a high variance pays a lot of attention to training data and does not generalize

- **Overfitting:** IF we have too many features, the learned model may fit the training set very well (Cost Function = 0), but fails to generalize to new example (predict price on new examples).

- Generally, when overfitting occurs, the training accuracy will be very high but the test accuracy will be relatively low.

The Problem of Under fitting



- A Bias occurs when an algorithm has limited flexibility to learn from data. Such models pay very little attention to the training data and oversimplify the model.

- **Underfitting**: If we have very less features, the learned model may not fit the training set very well (Cost Function is very high). It fails to capture relation between input and output data, and also fails to generalize.

- Generally when underfitting occurs, both training and test accuracy will be very low.

Bias vs Variance in Machine Learning:
https://www.youtube.com/watch?v=BO1qMFMAgUQ

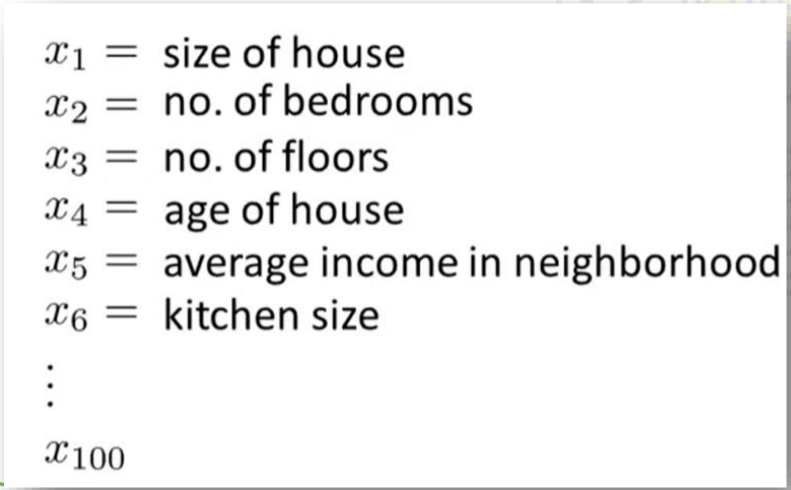
Addressing Under fitting

- Increase More features

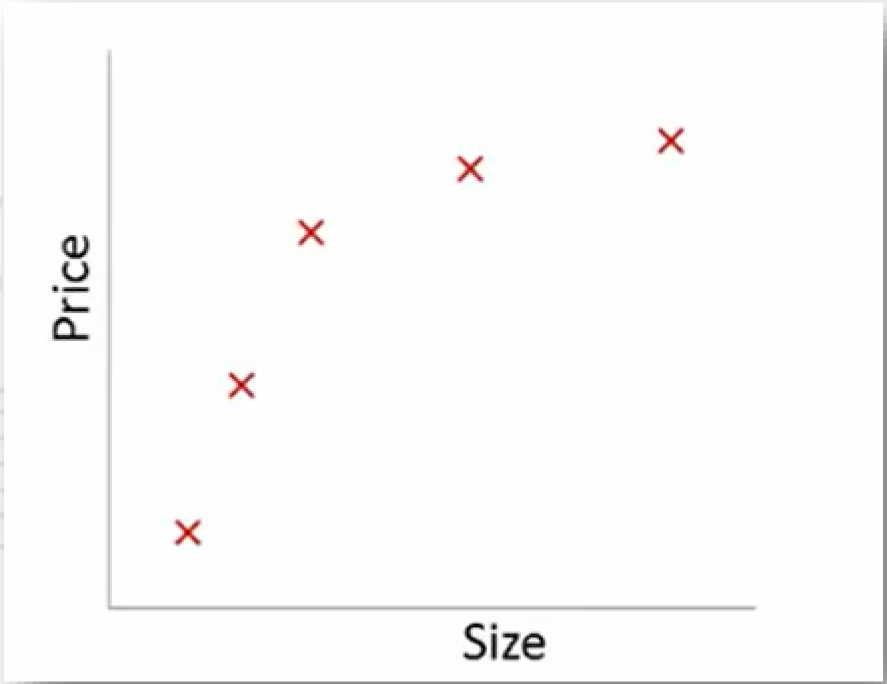
- Increase model complexity (Polynomial Features)

Addressing Overfitting

- Multiple number of features
- Not just the choice of degree of polynomial, but it's harder to plot the data



$x_1 =$ size of house
$x_2 =$ no. of bedrooms
$x_3 =$ no. of floors
$x_4 =$ age of house
$x_5 =$ average income in neighborhood
$x_6 =$ kitchen size
$\vdots$
x_{100}



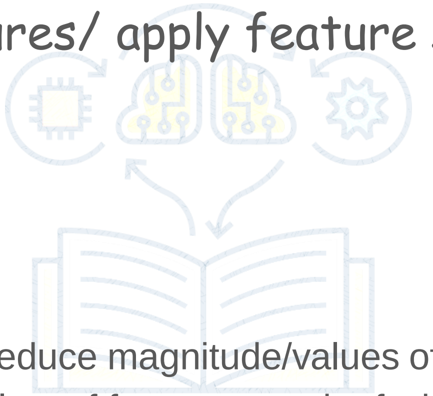
Addressing Overfitting

Options:

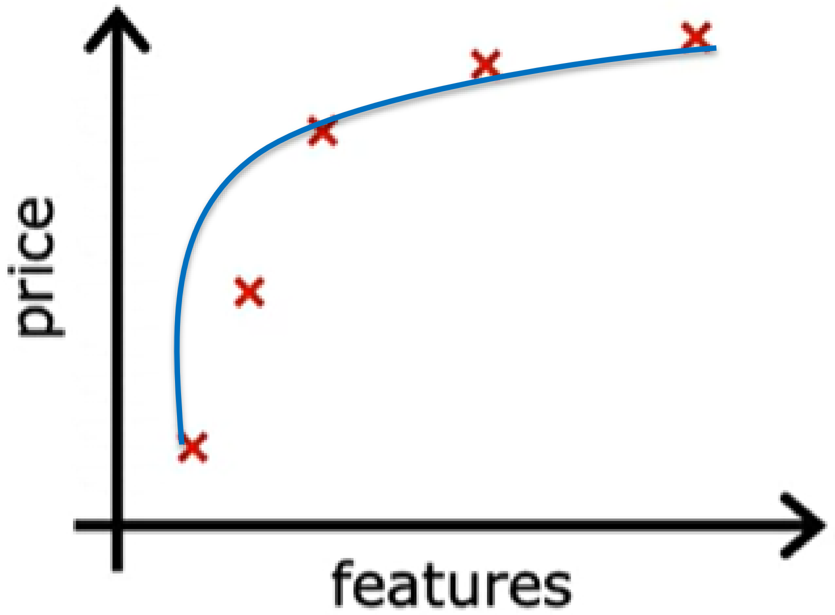
1. Reduce number of features/ apply feature selection

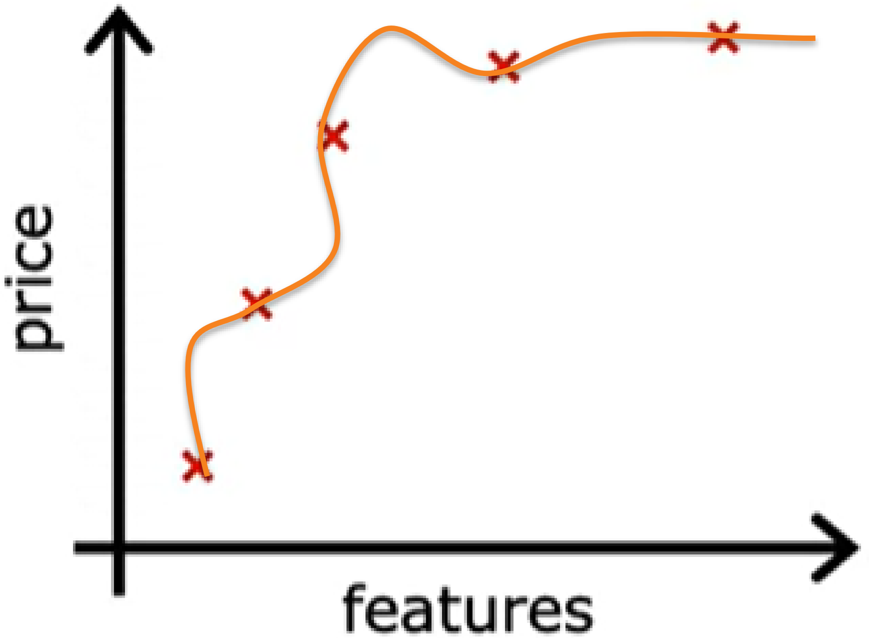
1. Get more training data.

1. Regularization.
   - Keep all the features, but reduce magnitude/values of parameters(w)
   - Works well when we have lots of features, each of which contributes a bit to predicting y.



Regularization

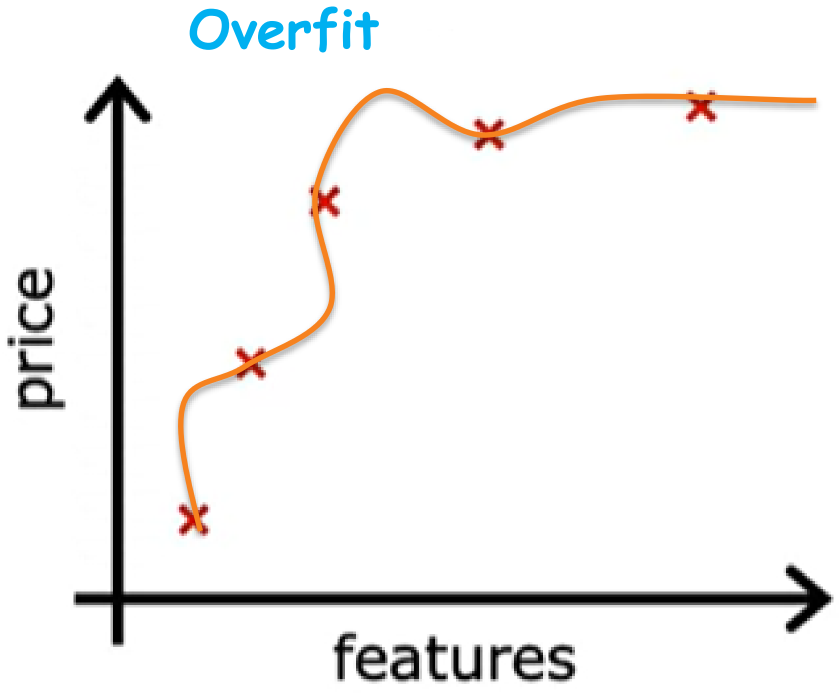




$$y = b + w_1 x + w_2 x^2$$

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

Regularization



- Often parameters w1,w2… are large values.

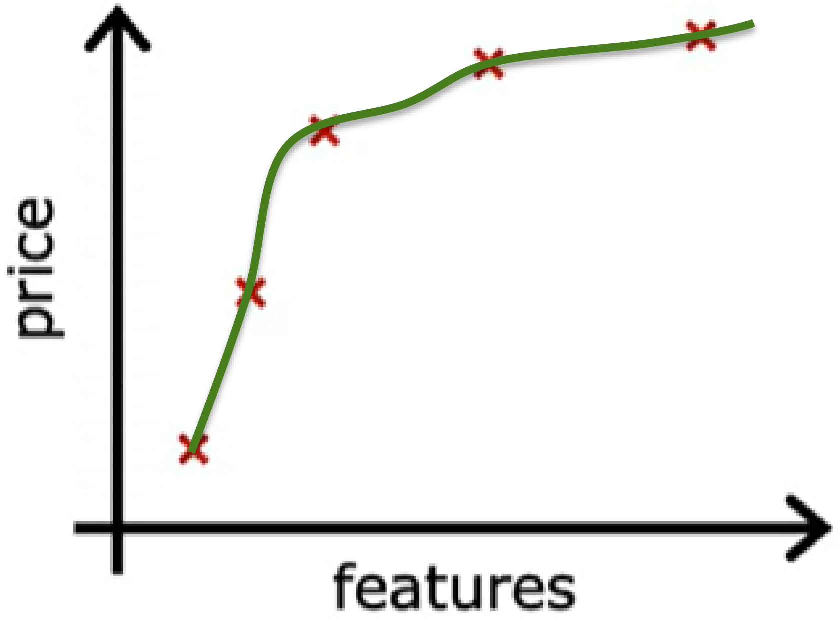
$$\underbrace{f(x) = 28x - 385x^2 + 39x^3 - 174x^4}_{\text{large values for } w_j} + 100$$

- Making parameters w1,w2… = 0 is as same as eliminating features.

- Regularization is way to more gently reduce the impact of some features without actually removing features.

Regularization

Overfit



- Regularization encourages to shink values of parameters w1,w2,w3… without making them actually 0

$$\boxed{\underbrace{f(x) = 13x - 0.23x^2 + 0.000014x^3 - 0.0001\,x^4}_{\text{Small values for } w_j} + 10}$$

- It turns out that even if we fit higher order polynomial, it allows to shrink parameters values very near to 0 and ends up getting better curve that fits data well.

- Regularization helps to keep all features but it just prevent features from having overly large effects.

Regularization

Suppose we penalize and make w_3 and w_4 really small

Cost Function Minimize

$$\frac{1}{n}\sum_{i=1}^{n}\underbrace{\left(f\left(x^i\right)-y^i\right)^2 + 1000w_3^2 + 1000w_4^2}$$

- We want to eliminate the influence of w_3 and w_4. Without actually getting rid of these features or changing the form of our model's function, we can instead modify our **cost function.**
- Now, in order for the cost function to get close to zero, we will have to reduce the values of w_3 and w_4 to near zero.
- This will in turn greatly reduce the values of $w_3x^3$and w_4x^4 in our model's function.
- As a result, we see that the new model's function looks like a quadratic function but fits the data better due to the extra small terms of $w_3x^3$and w_4x^4

Regularization

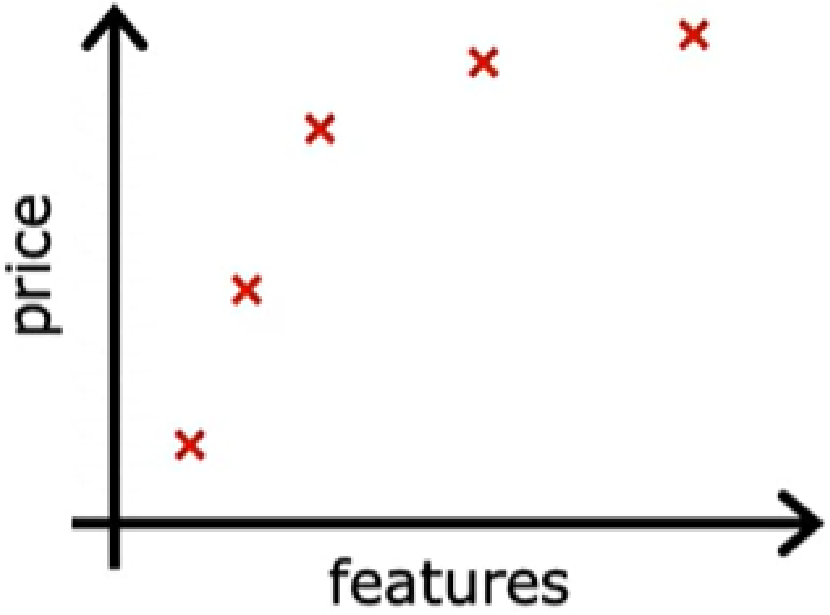
- Small values for parameters $w_{1,}\ w_{2,}\ w_{3,}\ w_4$ ……., w_r
  - Simpler hypothesis
  - Less prone to overfitting
- Housing price prediction problem,
  - Features: $x_{1,}\ x_{2,}\ x_{3,}\ x_4$ ……., x_{100}
  - Parameters: : $w_{1,}\ w_{2,}\ w_{3,}\ w_4$ ……., w_{100}

Regularization

Cost Function = $\frac{1}{n}\sum_{i=1}^{n}(f(x^i) - y^i)^2 + \lambda \sum_{j=1}^{r} w_j^2$

- First term is the cost function and second term is the regularization term where λ is the regularization parameter.

- λ controls the trade off between two different goals. First term aims to fit training data well, Second term aim to keep parameters small.

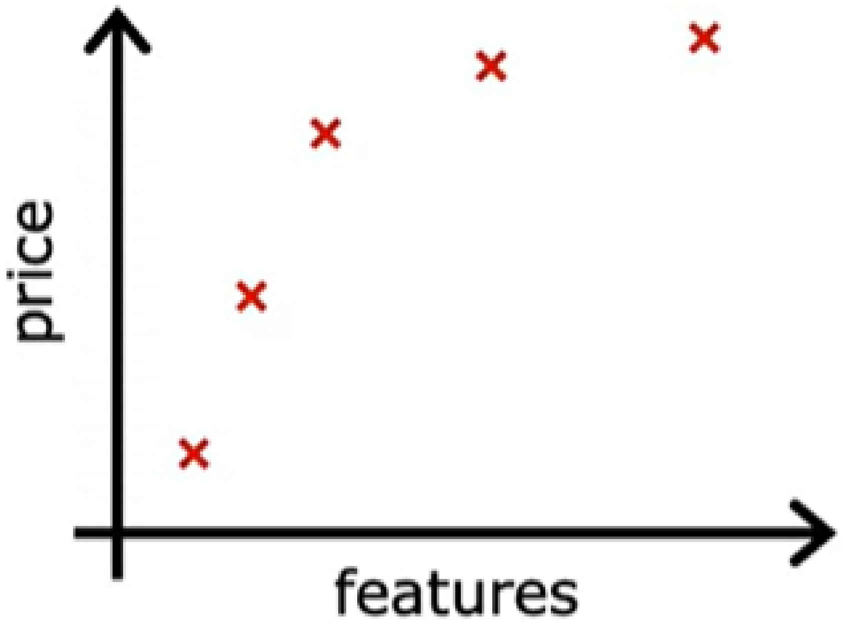


Regularization

In regularized linear regression, we choose parameters **w** to minimize our cost Function.

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

$b + w_1x + w_2x^2 + w_3x^3 + w_4x^4$



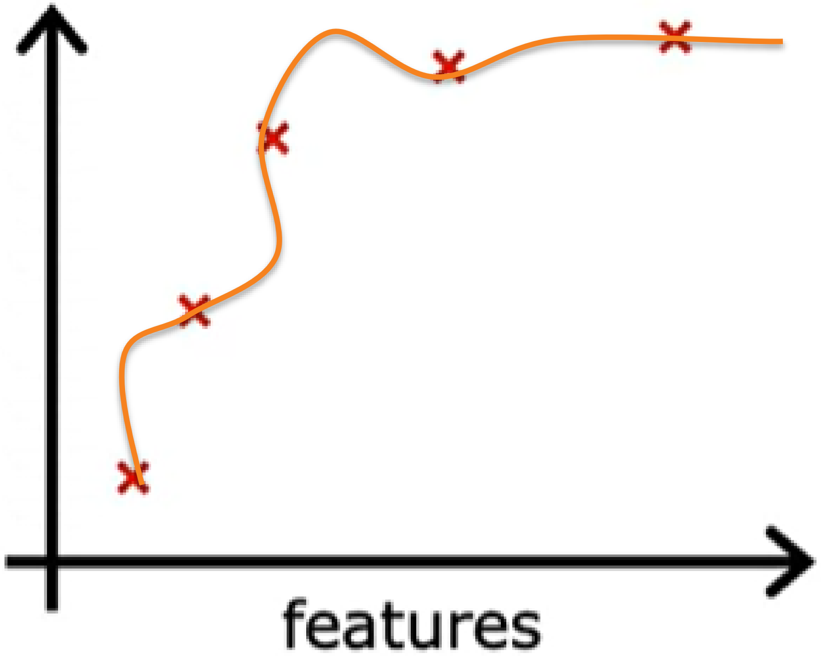
The effort is needed to choose the best regularization parameter (λ)

Regularization

When you choose λ = 0

$$\text{Cost Function} = \frac{1}{n}\sum_{i=1}^{n}\left(f\left(x^{i}\right)-y^{i}\right)^{2} + \underbrace{\lambda \sum_{j=1}^{r} w_j^{2}}_{\text{Regularization term becomes 0}}$$

- No effect of regularization term. Get same overfitted function



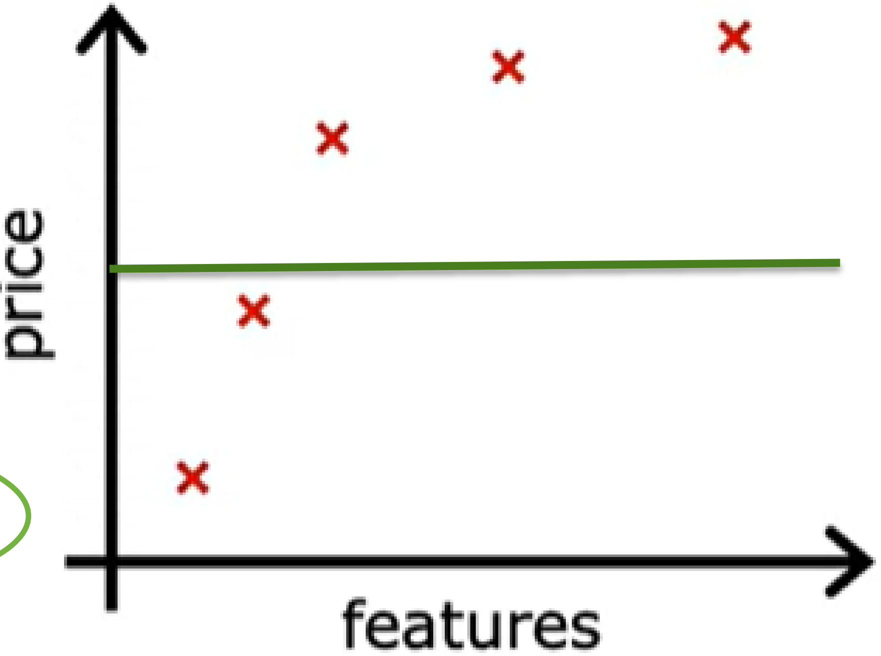
Regularization

When you choose λ = very large value (10^{10})

$$\text{Cost Function} = \frac{1}{n}\sum_{i=1}^{n}\left(f\left(x^i\right)-y^i\right)^2 + \underbrace{\lambda \sum_{j=1}^{r} w_j^2}$$

Makes parameters $w_j = 0$

- It eliminates impact of all the features in the given datasets. Only intercept term remains. Get oversimplified model like constant straight line



THANK YOU ☺