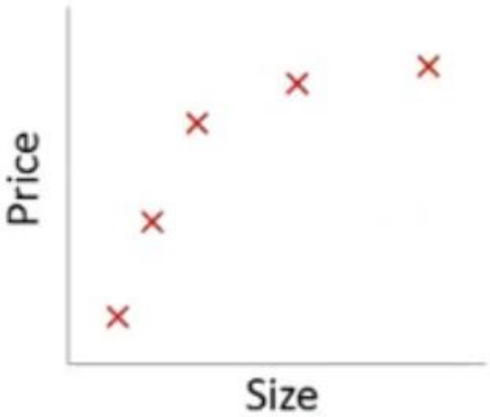
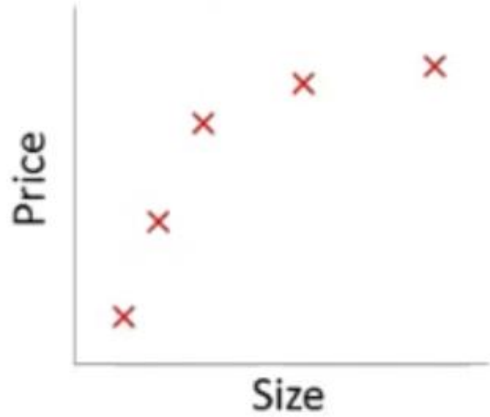
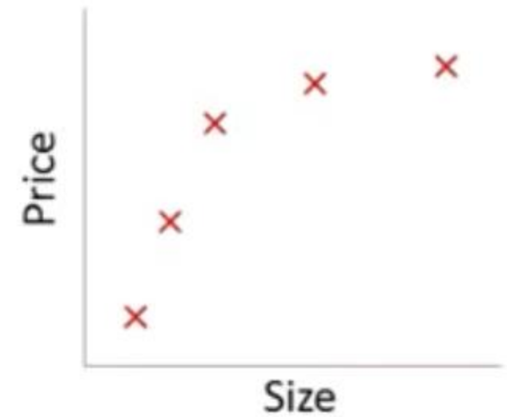
Regularization

Adapted from Andrew Ng

Some data about house prices



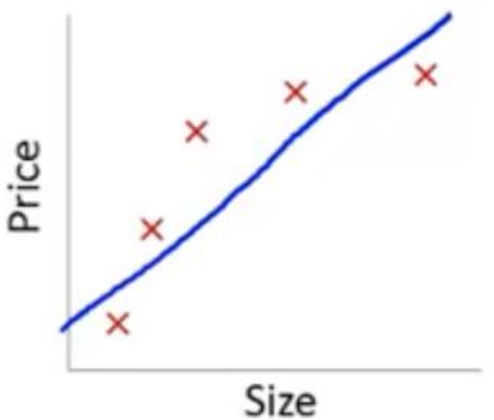


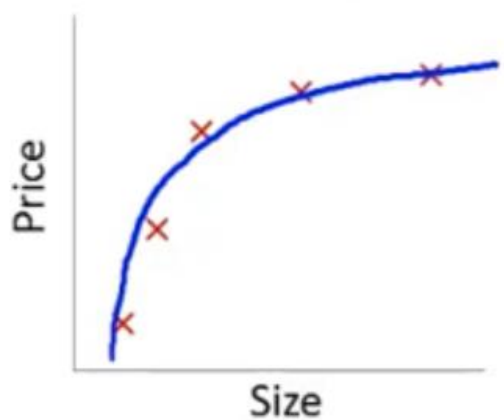


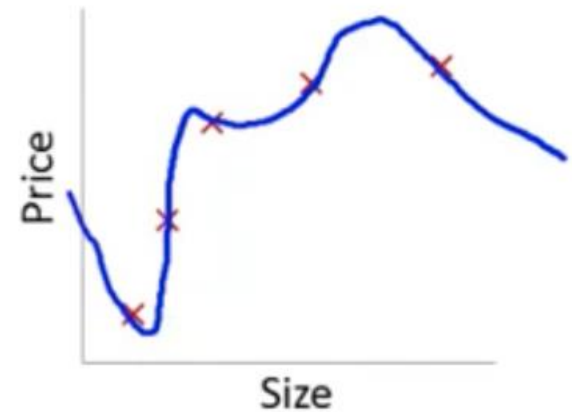
$w_0 + w_1 x$

$w_0 + w_1 x + w_2 x^2$

$w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$







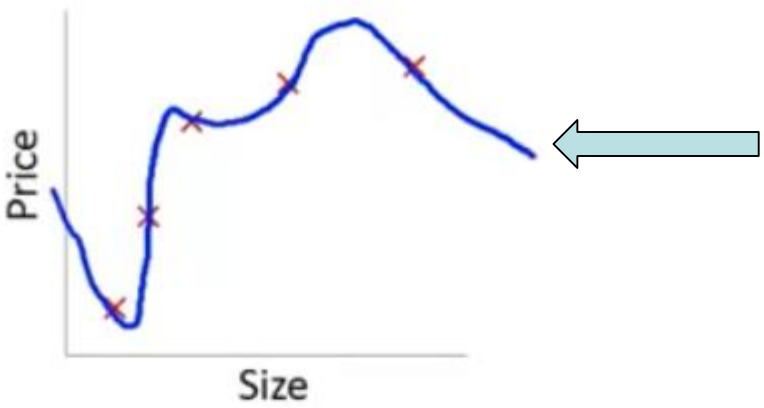
"Underfit" "High bias"

"Just right"

"Overfit" "High variance"

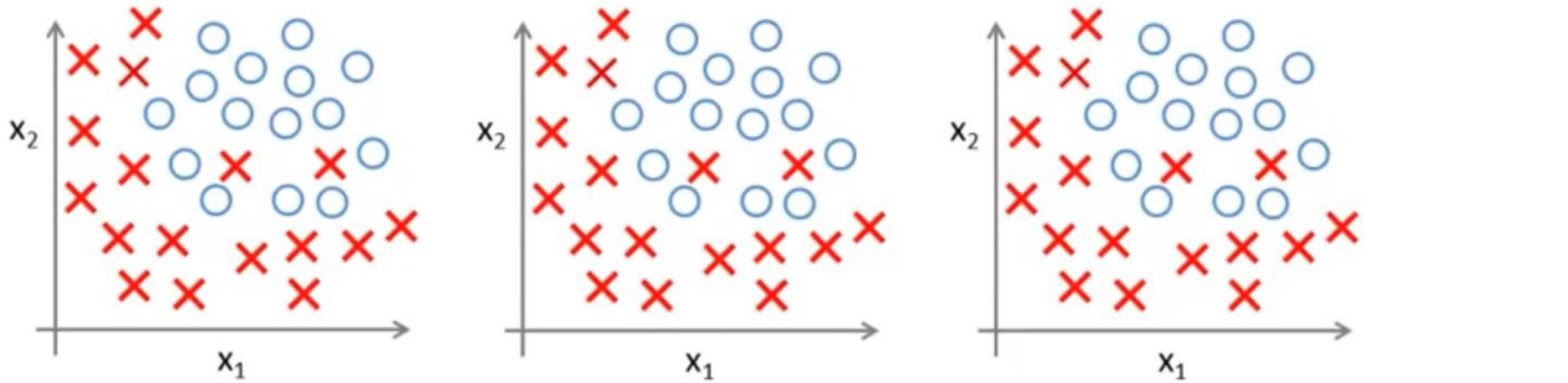
Overfitting

- If we have too many features, the learned hypothesis may fit the training set very well, but fails to generalize to new examples (predict prices on new examples).



Bad prediction

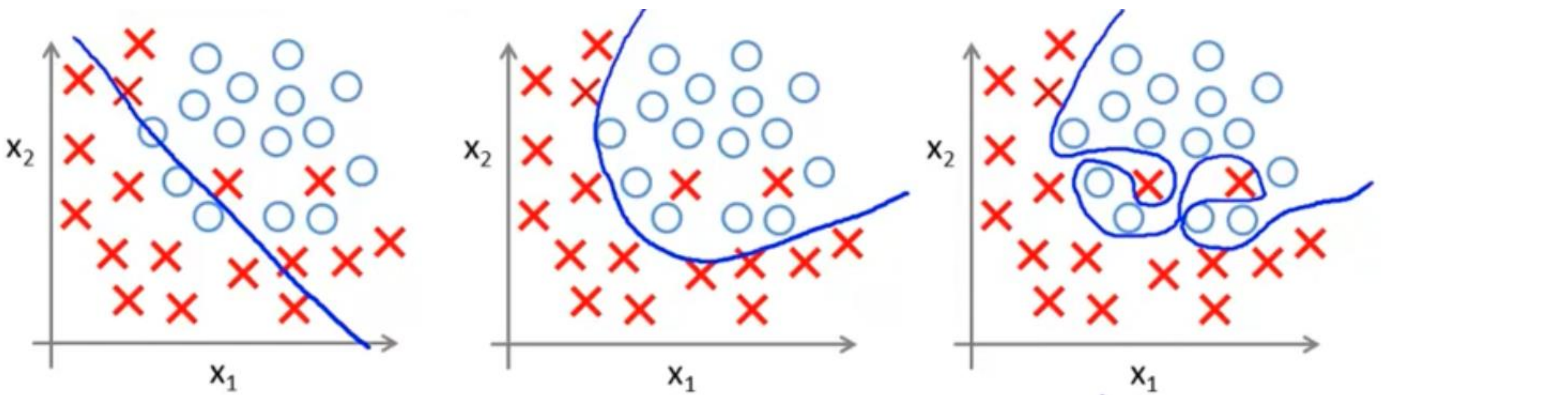
Example: Logistic Regression



$$\sigma(w_0 + w_1 x_1 + w_2 x_2)$$

$$\sigma(w_0 + w_1 x_1 + w_2 x_2 \\ + w_3 x_1^2 + w_4 x_2^2 \\ + w_5 x_1 x_2)$$

$$\sigma(w_0 + w_1 x_1 + w_2 x_2 \\ + w_3 x_1^2 + w_4 x_2^2 \\ + w_5 x_1 x_2 + w_6 x_1^3 x_2 + \cdots)$$

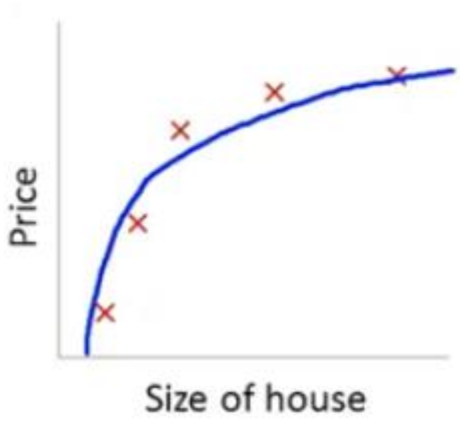


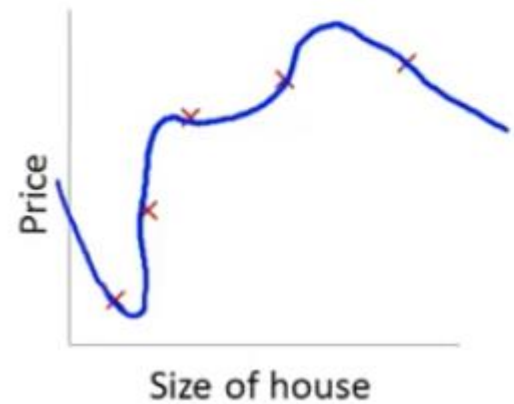
"Underfit" "High bias" "Just right" "Overfit" "High variance"

Address Overfitting

- Reduce number of features
  - Not always easy to do
- Regularization
  - Keep all the features, but reduce magnitude/values of parameters w.
  - Works well when we have a of features, each of which contributes a bit to predicting y.

Regularization - Intuition





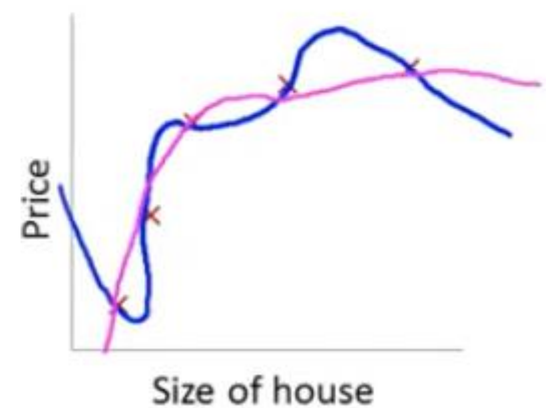
$w_0 + w_1 x + w_2 x^2$

$w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$

Suppose we penalize and make w_3, w_4 small.

$$\min \frac{1}{2n} \sum_{k=1}^{n} \left(y^k - h_{\boldsymbol{w}}(\boldsymbol{x}^k)\right)^2 + 1000 w_3^2 + 1000 w_4^2$$

We will end up with small w_3, w_4



Regularization

- Small values for parameters $w_1, w_2, \dots, w_m$
  - Simpler hypothesis
  - Less prone to overfitting
- Housing:
  - Features: $x_1, x_2, \dots, x_{100}$
  - Parameters: $w_1, w_2, \dots, w_{100}$

$$E(\boldsymbol{w}) = \frac{1}{2n}\left[\sum_{k=1}^{n}\left(y^k - h_{\boldsymbol{w}}(\boldsymbol{x}^k)\right)^2 + \lambda\sum_{i=1}^{m} w_i^2\right]$$

By convention, we don't include w_0 in the second sum. However, we can include it, without causing much of a difference.

Regularization

$$E(\boldsymbol{w}) = \frac{1}{2n}\left[\sum_{k=1}^{n}\left(y^k - h_{\boldsymbol{w}}(\boldsymbol{x}^k)\right)^2 + \lambda\sum_{i=1}^{m} w_i^2\right]$$

- **λ** is the **regularization parameter** controls trade off between two goals
  - 1) Want to fit the training set well
  - 2) Want to keep parameters small
- If **λ** is large we penalize ALL the parameters so all the parameters end up being close to 0
  - If this happens, it's like we got rid of all the terms in the hypothesis
    - This results in **underfitting**
- So, **λ** should be chosen carefully - not too big...
  - We look at some automatic ways to select **λ** later in the course

If **λ** is large, say 10^{10}

$$w_0 + \cancel{w_1}x + \cancel{w_2}x^2 + \cancel{w_3}x^3 + \cancel{w_4}x^4$$

Linear Regression – Gradient Descent

Repeat {

$$w_0 = w_0 - \alpha \frac{1}{n} \sum_{k=1}^{n} \left(h_{\boldsymbol{w}}\left(\boldsymbol{x}^k\right) - y^k \right) x_0^k$$

$$w_j = w_j - \alpha \left(\frac{1}{n} \sum_{k=1}^{n} \left(h_{\boldsymbol{w}}\left(\boldsymbol{x}^k\right) - y^k \right) x_j^k + \frac{\lambda}{n} w_j \right)$$

$$j = 1,2, \dots, m$$

}

$$w_j = w_j \underbrace{\left(1 - \alpha \frac{\lambda}{n}\right)} - \alpha \underbrace{\frac{1}{n} \sum_{k=1}^{n} \left(h_{\boldsymbol{w}}\left(\boldsymbol{x}^k\right) - y^k \right) x_j^k}$$

We are slightly shrinking w_j, then performing an update as before.

Less than 1, e.g. 0.99

Same as before (without regularization)

Linear Regression – Canonical Eq.

We had:

$$\boldsymbol{w} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$$

Now, with regularization it becomes:

$$\boldsymbol{w} = \left(\boldsymbol{X}'\boldsymbol{X} + \lambda \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \right)^{-1} \boldsymbol{X}'\boldsymbol{y}$$

- **X** is the $n \times (m+1)$ data matrix
  - one row of m+1 elements for each data instance
    - without the y attribute
- **y** is the n-vector of class values
- **X'X** is $(m+1) \times (m+1)$ matrix
  - Good if number m of attributes is not too big.
- **w** is m-vector, i.e. $(m+1) \times 1$

Regularization – Logistic Regression

- Before:

$$E(\mathbf{w}) = \frac{1}{n}\sum_{k=1}^{k} \ln\left(1 + e^{-y^k \mathbf{w}^T \mathbf{x}^k}\right)$$

- Now:

$$E(\mathbf{w}) = \frac{1}{n}\sum_{k=1}^{k} \ln\left(1 + e^{-y^k \mathbf{w}^T \mathbf{x}^k}\right) + \frac{\lambda}{2n}\sum_{j=1}^{m} w_j^2$$

Gradient Descent Algorithm

Initialize **w**=**0**

For t=0,1,2,…do

Compute the gradient $$\nabla_E(\mathbf{w}) = -\frac{1}{n}\sum_{k=1}^{n}\frac{y^k\mathbf{x}^k}{1+e^{y^k\mathbf{w}^T\mathbf{x}^k}} + \frac{\lambda}{n}w_j$$

Update the weights $$\mathbf{w} \leftarrow \mathbf{w} - \kappa\nabla_E(\mathbf{w})$$

Iterate with the next step until **w** doesn't change too much (or for a fixed number of iterations)

Return final **w**.