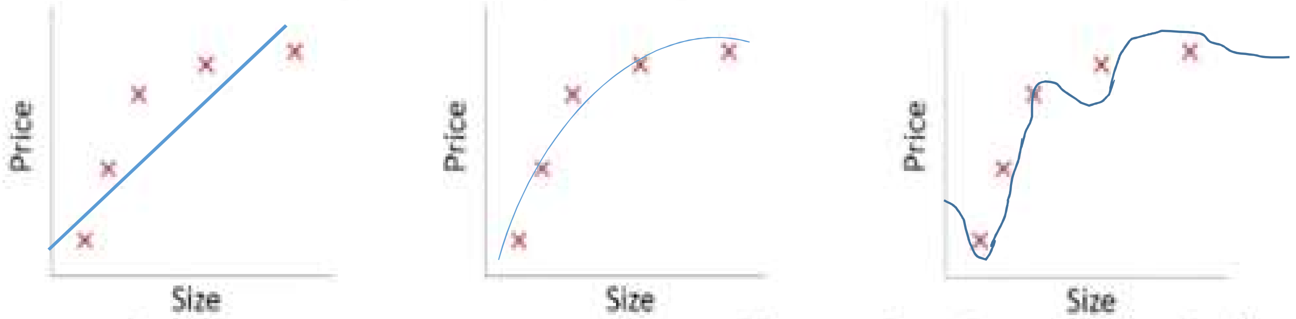
Overfitting

and how to handle

Some slides were adapted/taken from various sources, including Prof. Andrew Ng's Coursera Lectures, Stanford University, Prof. Kilian Q. Weinberger's lectures on Machine Learning, Cornell University, Prof. Sudeshna Sarkar's Lecture on Machine Learning, IIT Kharagpur, Prof. Bing Liu's lecture, University of Illinois at Chicago (UIC), CS231n: Convolutional Neural Networks for Visual Recognition lectures, Stanford University and many more. We thankfully acknowledge them. Students are requested to use this material for their study only and **NOT** to distribute it.

The problem of overfitting

Example: Linear regression (housing prices)



$\theta_0 + \theta_1 x$ $\theta_0 + \theta_1 x + \theta_2 x^2$ $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

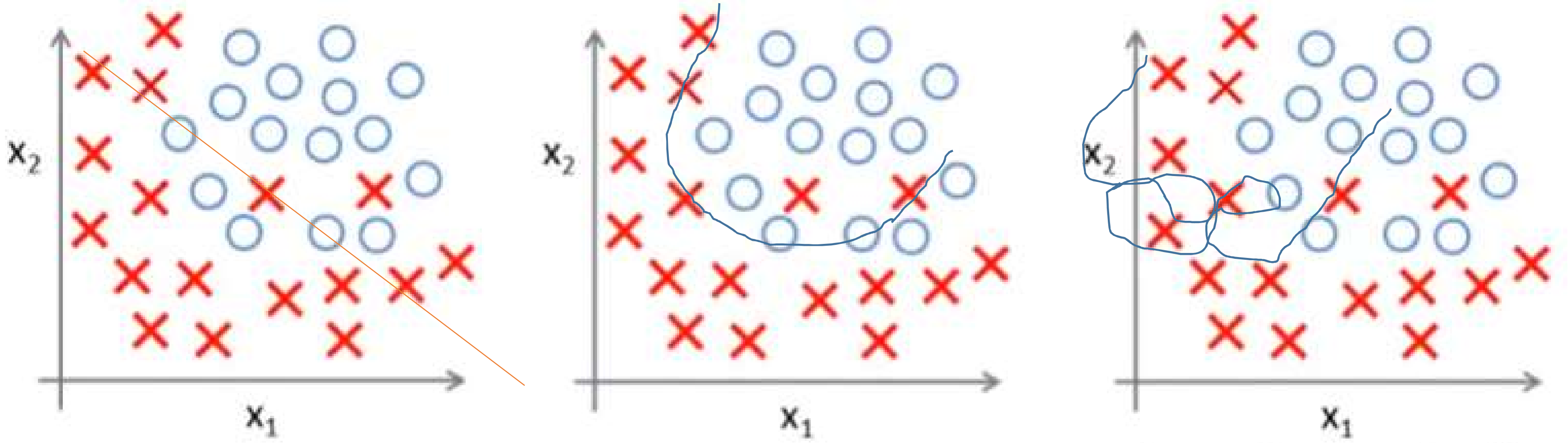
Under fit or High bias

Over fit or High variance

Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

The problem of overfitting

Example: Logistic regression



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2$
$+\theta_3 x_1^2 + \theta_4 x_2^2$
$+\theta_5 x_1 x_2)$

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2$
$+\theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2$
$+\theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

Under fit or High bias

Over fit or High variance

Bias and Variance

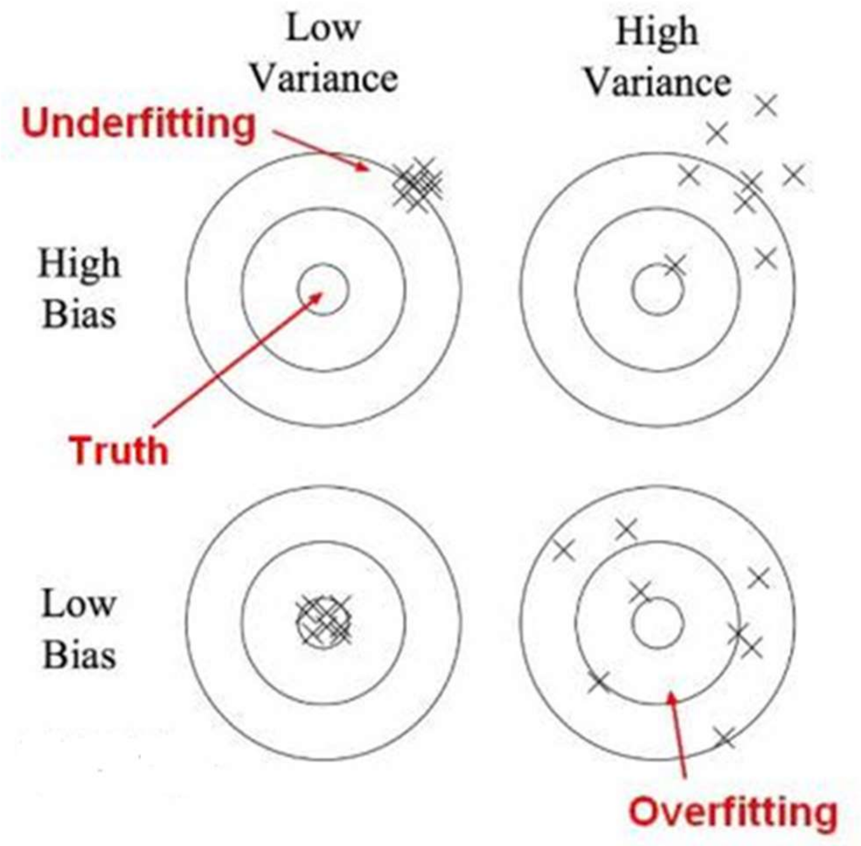
What is bias?

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

What is variance?

Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn’t seen before. As a result, such models perform very well on training data but has high error rates on test data.

Bias and variance using bulls-eye diagram



- Centre of the target is a model that perfectly predicts correct values. As we move away from the bulls-eye our predictions become get worse and worse.

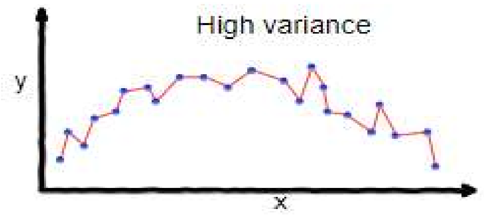
- In supervised learning **underfitting** happens when a model unable to capture the underlying pattern of the data.

- These models usually have high bias and low variance.

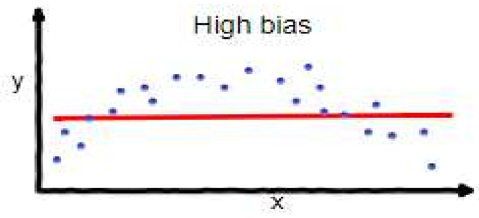
- It happens when we have very less amount of data to build an accurate model or when we try to build a linear model with a nonlinear data.

Bias Variance Tradeoff

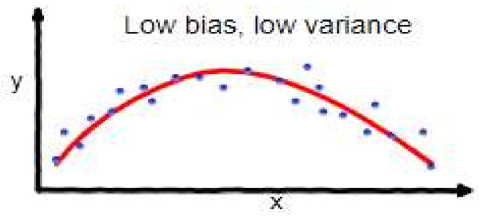
In supervised learning, **overfitting** happens when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over noisy dataset. These models have low bias and high variance. These models are very complex like Decision trees which are prone to overfitting.



overfitting

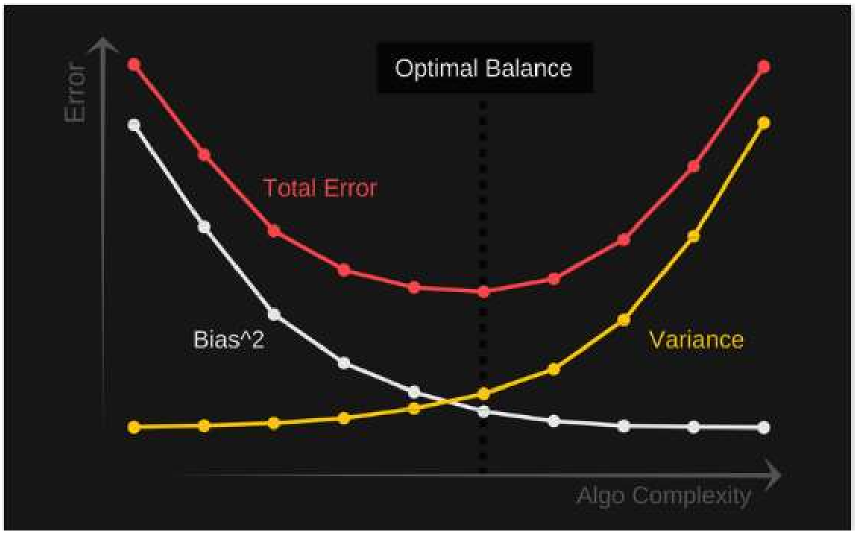


underfitting



Good balance

Bias Variance Tradeoff



Why is Bias Variance Tradeoff?

If our model is too simple and has very few parameters (less complex) then it may have high bias and low variance. On the other hand if our model has large number of parameters (highly complex) then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data.

This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time.

Total Error

To build a good model, we need to find a good balance between bias and variance such that it minimizes the total error.

Total Error = Bias^2 + Variance + Irreducible Error

An optimal balance of bias and variance would never overfit or underfit the model

Overfitting

- A hypothesis function h is said to overfit the training data if there is another hypothesis h’ such that h’ has more error than h on training data but h’ has less error than h on testing data.

- Learning a classifier that classifies a training data perfectly may not lead to the classifier with best generalization performance (why?)
  - There may be noise in training data
  - Training data set is too small

- Simplistically, overfitting occurs when model is too complex whether underfitting occurs when model is too simple.

- Note: Training error is not a good predictor for the testing error.

The problem of overfitting

- Let's consider D, the entire distribution of data, and T, the training set.

- Hypothesis $h \in H$ overfits D if
  $\exists\, h' \neq h \in H$ such that
  (1) $\text{error}_T(h) < \text{error}_T(h')$ [i.e. doing well on training set] but
  (2) $\text{error}_D(h) > \text{error}_D(h')$

- What do we care about most (1) or (2)?
- *Estimate error on full distribution by using test data set.*
  Error on test data: Generalization error (want it low!!)

- *Generalization to unseen examples/data is what we care about.*

The problem of overfitting

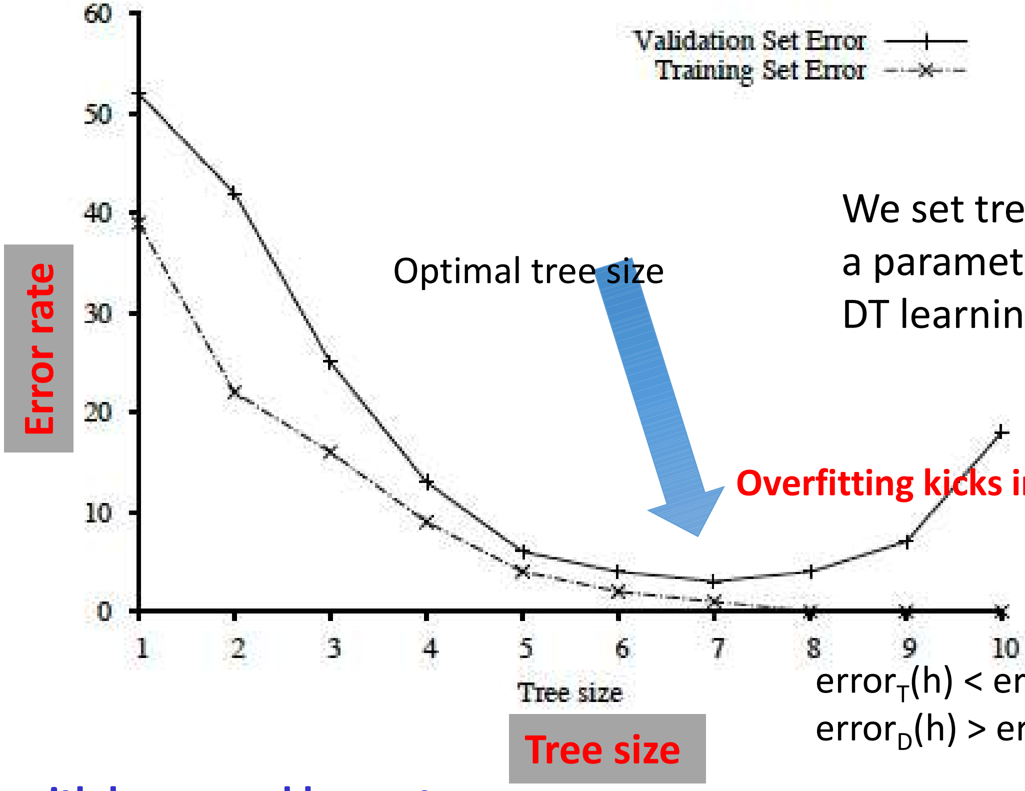
- Data overfitting is the arguably the most common pitfall in machine learning. **Why?**

- Temptation to use as much data as possible to train on. (“Ignore test till end.” Test set too small.) Data “peeking” not noticed.

- *Temptation to fit very complex hypothesis* (e.g. large decision tree). In general, the larger the tree, the better the fit to the training data.

- It’s hard to think of a better fit to the training data as a “worse” result. Often difficult to fit training data well, so it seems that “a good fit to the training data means a good result.”

Key figure in machine learning



Optimal tree size

Overfitting kicks in…

We set tree size as a parameter in our DT learning alg.

$\text{error}_T(h) < \text{error}_T(h')$ but $\text{error}_D(h) > \text{error}_D(h')$

Note: with larger and larger trees,
we just do better and better on the training set!

But note the performance on the validation set degrades!

Note: Similar curves can happen when training too long in complex hypothesis space with lots of parameters to set.

Solutions for Overfitting

- K- fold Cross Validation
- Regularization
- Early stopping
- Drop-out
- Pre or post pruning for decision tree
- Minimum description length (MDL) principle

K- fold Cross Validation

Training Set		Testing Set
Training Set	Validation Set	Testing Set

S1	S2	S3	…	S_k

Training Set = S

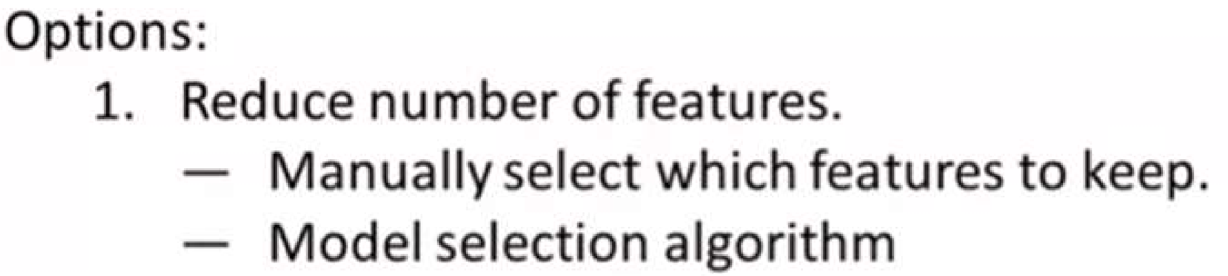
Average test score = $\frac{1}{k}(\sum_{i=1}^{k} S_i)$

Round	Training Set	Validation Set
1	S – S1	S1
2	S – S2	S2
k	S-S_k	S_k

- **Trade-off**:

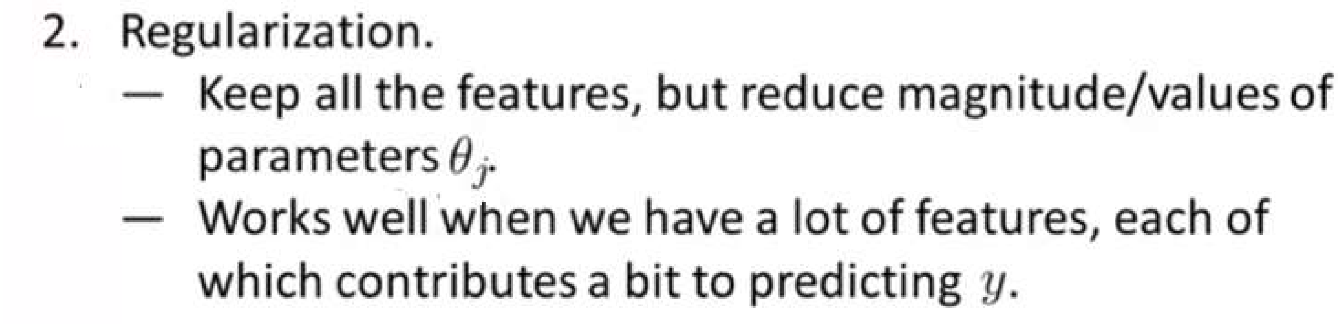
- Complex hypothesis fit the data well → may tend to overfitting
- Simple hypothesis may generalize better → may tend to underfitting
- As the training data samples increase, generalization error decreases.

Regularization



Options:

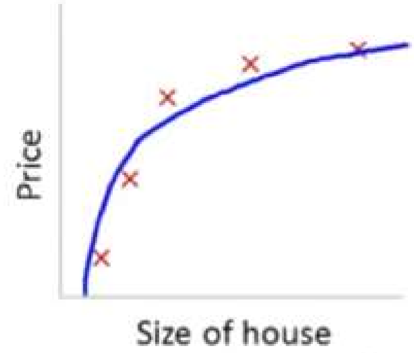
1. Reduce number of features.
   - Manually select which features to keep.
   - Model selection algorithm



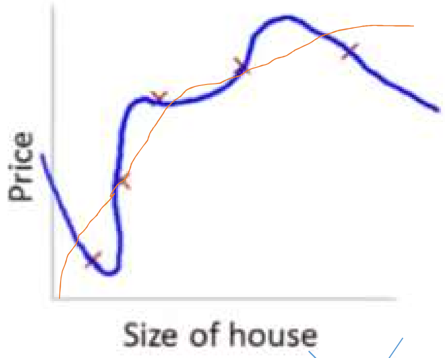
2. Regularization.
   - Keep all the features, but reduce magnitude/values of parameters θ_j.
   - Works well when we have a lot of features, each of which contributes a bit to predicting y.

Regularization

Intuition



$$\theta_0 + \theta_1 x + \theta_2 x^2$$



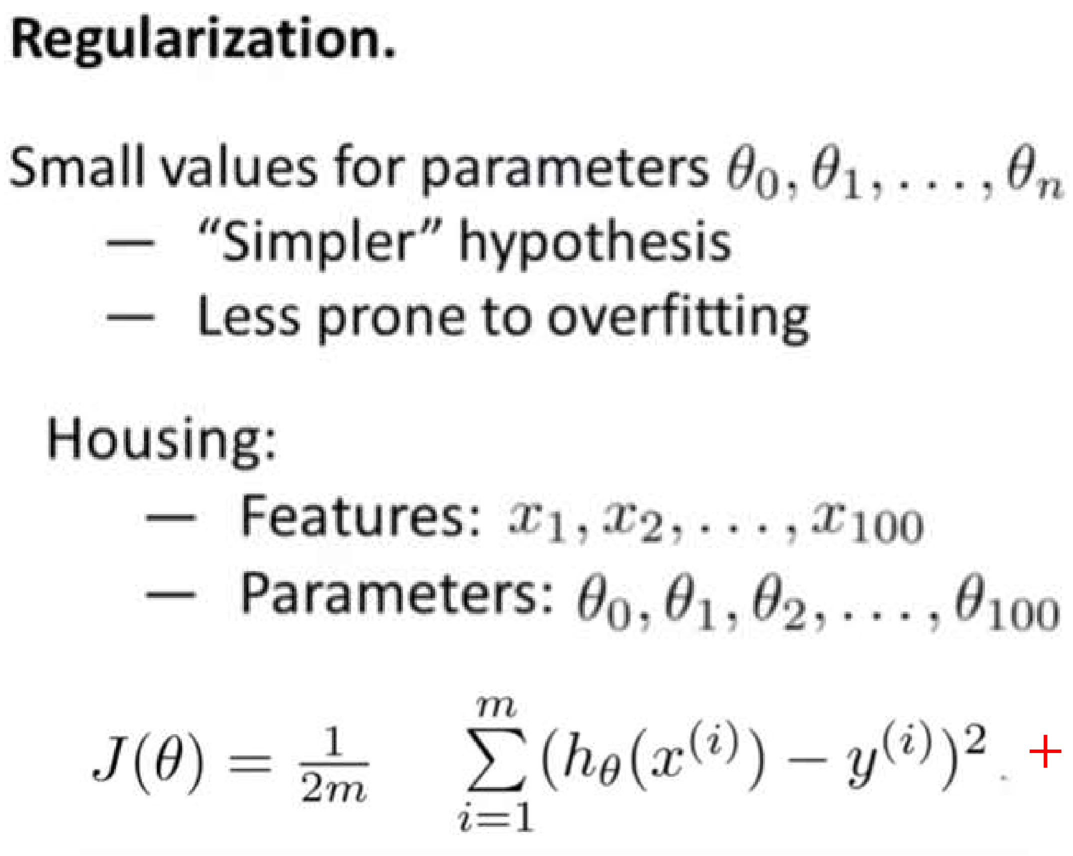
$$\theta_0 + \theta_1 x + \theta_2 x^2 + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4}$$

Suppose we penalize and make θ_3, θ_4 really small.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

$$\theta_3 \cong 0 \qquad \theta_4 \cong 0$$

Regularization



Regularization.

Small values for parameters $\theta_0, \theta_1, \dots, \theta_n$

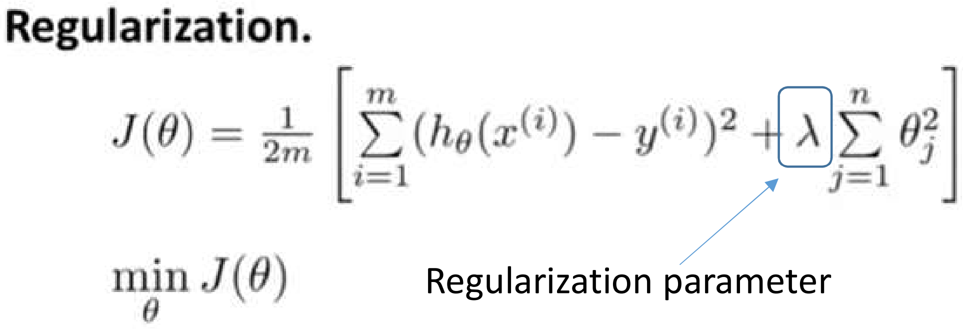
- "Simpler" hypothesis
- Less prone to overfitting

Housing:

- Features: $x_1, x_2, \dots, x_{100}$
- Parameters: $\theta_0, \theta_1, \theta_2, \dots, \theta_{100}$

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

Regularization

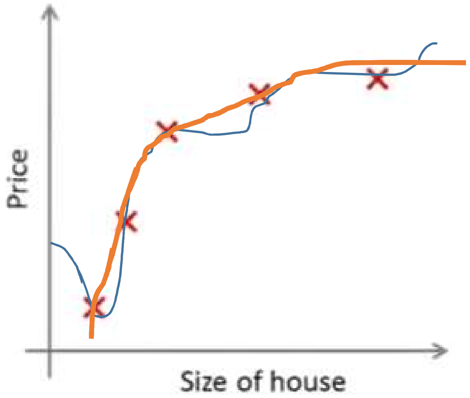


Regularization.

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

$$\min_\theta J(\theta)$$

Regularization parameter



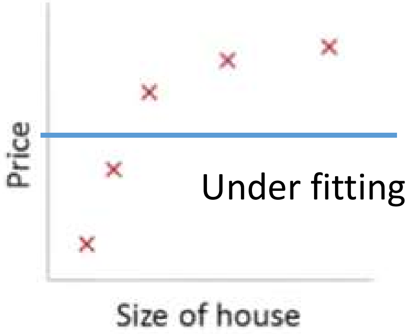
- Fitting the data points well
- Keeping the no. of parameters (Θs) small

Regularization

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 $\lambda = 10^{10}$)?



$\Theta_1 \cong 0;\ \Theta_2 \cong 0; \Theta_3 \cong 0;\ \Theta_4 \cong 0;$

$h_\Theta(x) = 0$

$$h_\Theta(x) = \theta_0 + \cancel{\theta_1 x} + \cancel{\theta_2 x^2} + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4}$$

Regularized Linear Regression

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

$$\min_\theta J(\theta)$$

Gradient descent

Repeat {

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha\left[\overbrace{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\Theta_j}^{\frac{\partial}{\partial\Theta_j}J(\Theta)}\right]$$

$$(j = \cancel{0}, 1, 2, 3, \ldots, n)$$

}

Regularized Linear Regression

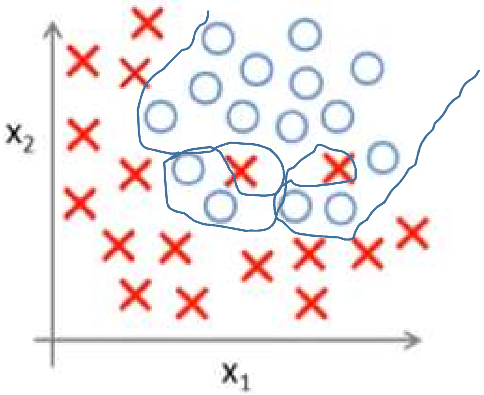
$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \Theta_j \right]$$

$$(j = 0, 1, 2, 3, \ldots, n)$$

$$\theta_j := \theta_j \overbrace{(1 - \alpha \frac{\lambda}{m})}^{\text{Shrinkage}} - \overbrace{\alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}^{\text{Parameter updation}}$$

$$(1 - \alpha \frac{\lambda}{m}) < 1$$

Regularized Logistic Regression



$$
\begin{aligned}
h_\theta(x) = g(&\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 \\
&+\theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 \\
&+\theta_5 x_1^2 x_2^3 + \ldots)
\end{aligned}
$$

Cost function:

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

$$+ \frac{\lambda}{2m} \sum_{i=1}^{n} \theta_j^2$$

Regularized Logistic Regression

$$\theta_j := \theta_j - \alpha \left[\underbrace{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\Theta_j}_{\frac{\partial}{\partial\Theta_j}J(\Theta)} \right]$$

$$(j = 0, 1, 2, 3, \ldots, n)$$

For Logistic Regression:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Early Stopping

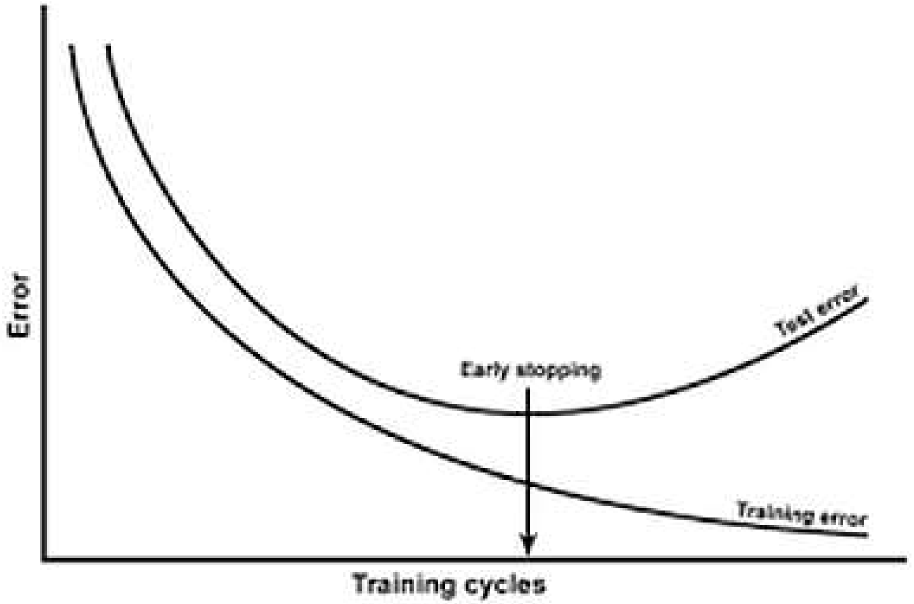
- Early stopping is a form of regularization while training a model with an iterative method, such as gradient descent.

- This method update the model so as to make it better fit the training data with each iteration.

- Up to a point, this improves the model's performance on data on the test set.

- Past that point however, improving the model's fit to the training data leads to increased generalization error.

- Early stopping rules provide guidance as to how many iterations can be run before the model begins to overfit.



In the Fig., we can see, after some iterations, test error has started to increase while the training error is still decreasing. Hence the model is overfitting. So to combat this, we stop the model at the point when this starts to happen.

Thanks