Linear regression

November 11, 2016

Agenda

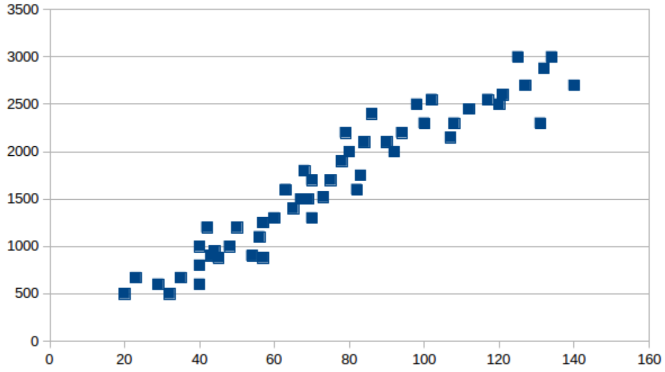
1. Model representation
2. Cost function
3. Gradient descent

Useful resources

1. Coursera. Machine learning (Andrew Ng)
2. HSE course. Week 2. Week 4.

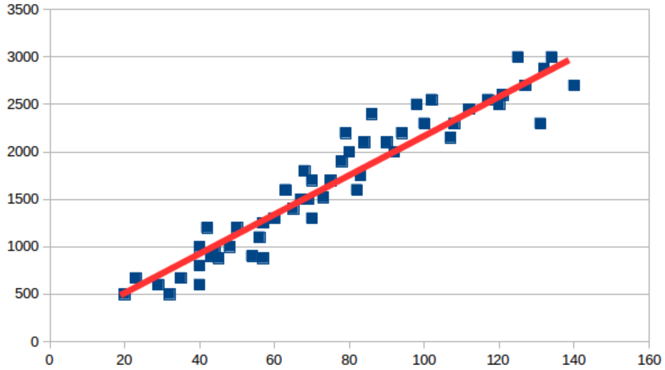
Model representation

House prices



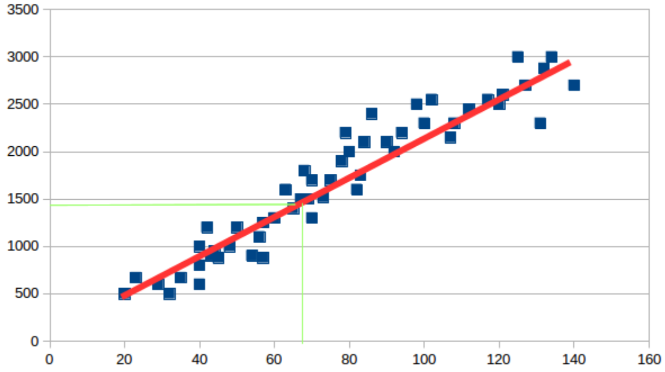
Model representation

House prices



Model representation

Supervised learning. We give right answer for each example of data.
Regression problem - predict real-valued output.



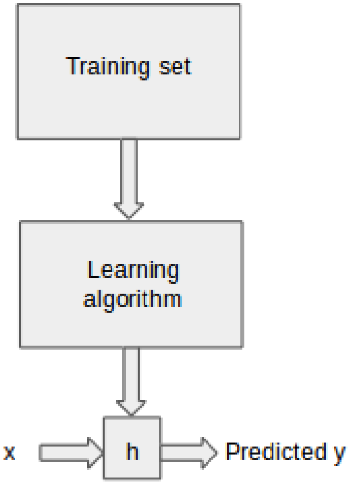
Model representation

Training set of housing prices.

Size (x)	Price (y)
20	500
40	800
65	1300
...	...

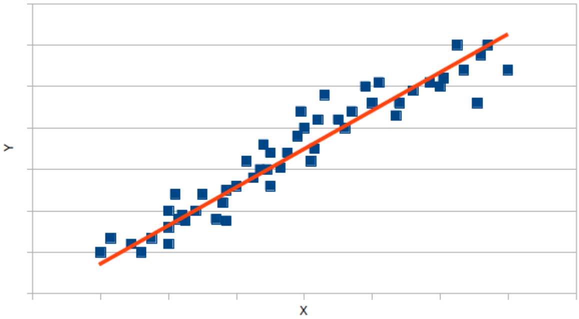
Notation:
m = number of training example
x = input variable features
y = output variable target variable

Model representation



Model representation

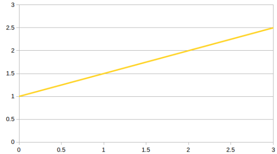
$h_\theta(x) = \Theta_0 + \Theta_1 x$ Linear regression with one variable.



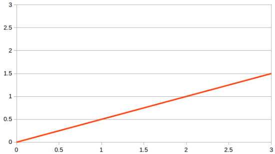
Model representation

How to choose Θ_i

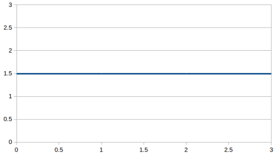
$$h_\theta(x) = \Theta_0 + \Theta_1 x$$



(a) $\Theta_0 = 1 \ \Theta_1 = 0.5$



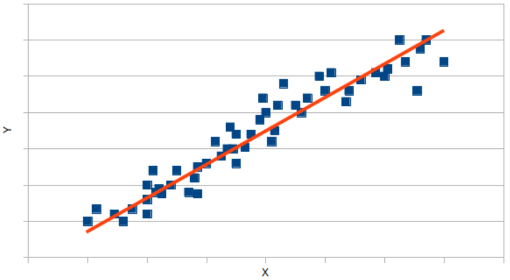
(b) $\Theta_0 = 0 \ \Theta_1 = 0.5$



(c) $\Theta_0 = 1.5 \ \Theta_1 = 0$

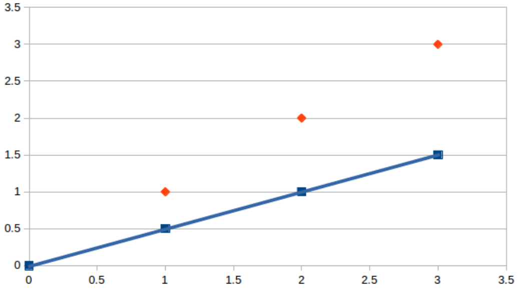
Cost function

$h_\theta(x) = \Theta_0 + \Theta_1 x$
We need to choose Θ_0, Θ_1 that $h_0(x)$ is close to y for our training examples (x,y).
So we should minimize $J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_0^i(x) - y^i)^2$
$J(\Theta_0, \Theta_1)$ - **Loss function**, squared error function.



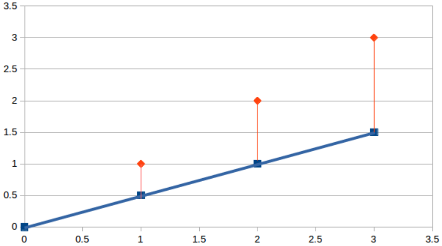
Cost function

$$h_\theta(x) = \Theta_0 + \Theta_1 x \text{ - hypothesis function.}$$
$$\Theta_0 = 0, \;\; \Theta_1 = 0.5$$



Cost function

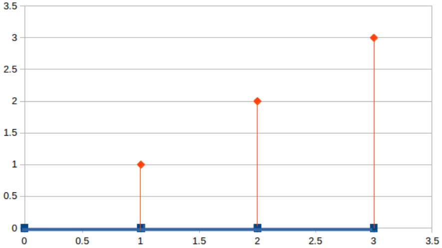
$$h_\theta(x) = \Theta_0 + \Theta_1 x \text{ - hypothesis function.}$$
$$\Theta_0 = 0, \;\; \Theta_1 = 0.5$$



$$J(0, 0.5) = \frac{1}{2*3}((0.5-1)^2 + (1-2)^2 + (1.5-3)^2) = 0.58$$

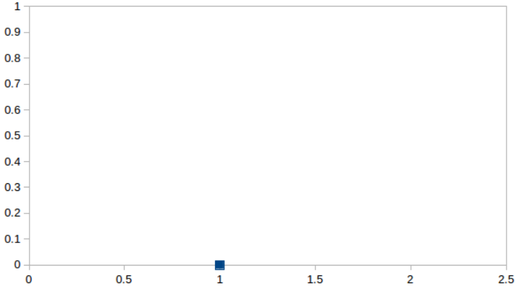
Cost function

$$h_\theta(x) = \Theta_0 + \Theta_1 x \text{ - hypothesis function.}$$
$$\Theta_0 = 0, \ \ \Theta_1 = 0$$

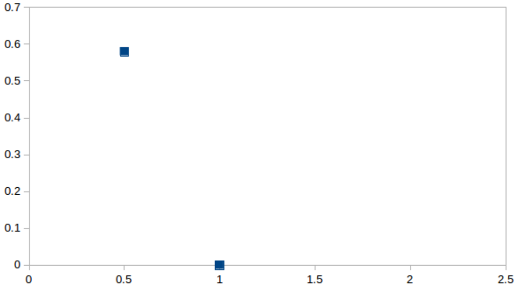


$$J(0,0) = \frac{1}{2*3}((0-1)^2 + (0-2)^2 + (0-3)^2) = 2.33$$

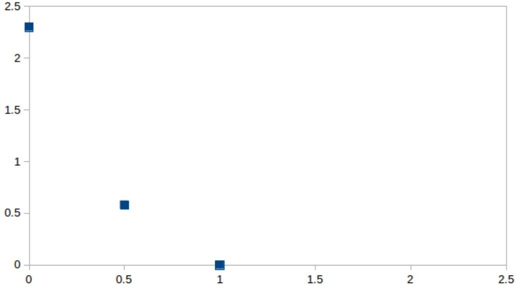
Cost function



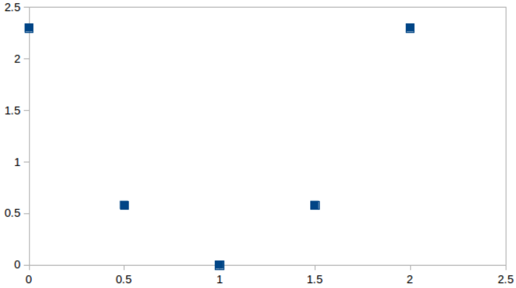
Cost function



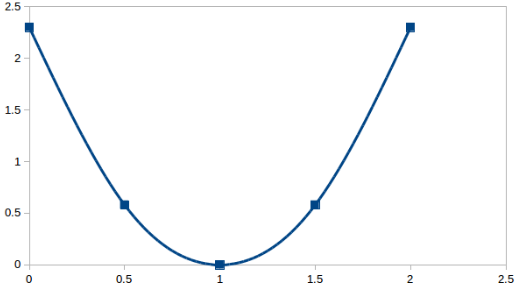
Cost function



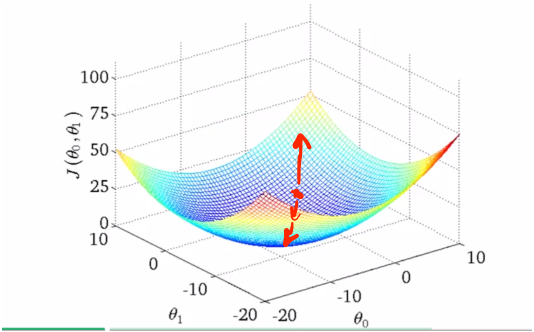
Cost function



Cost function



Cost function

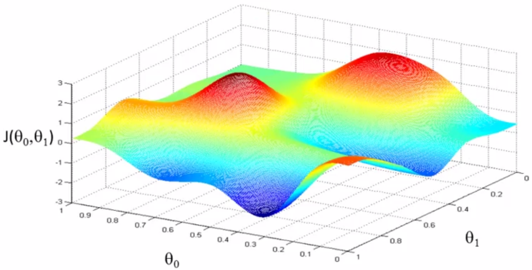


Gradient descent

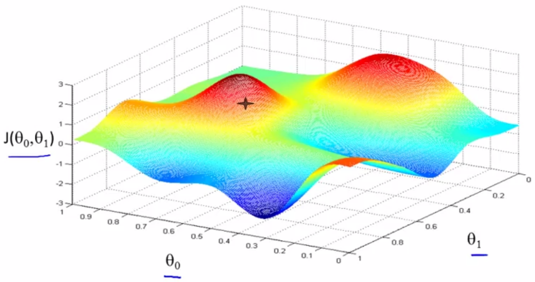
Have some function $J(\Theta_0, \Theta_1)$
Want to find *min* $J(\Theta_0, \Theta_1)$

1. Start from initialΘ_0, Θ_1
2. Keep changing Θ_0, Θ_1 to reduce $J(\Theta_0, \Theta_1)$ until we find minimum

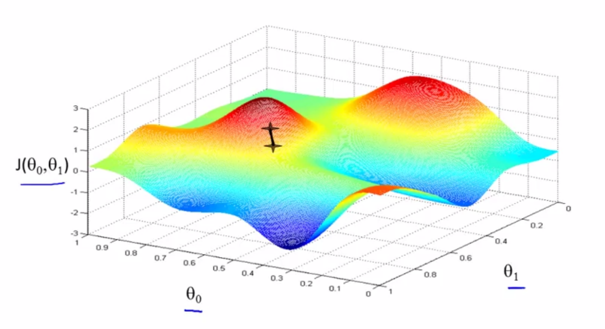
Gradient descent



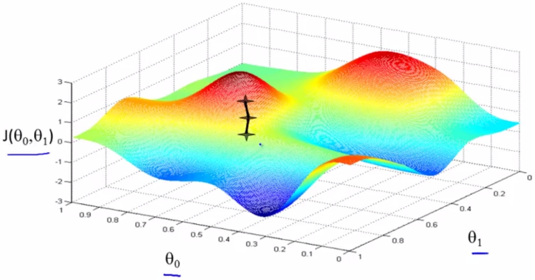
Gradient descent



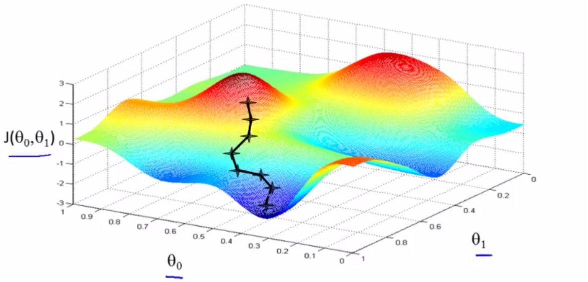
Gradient descent



Gradient descent



Gradient descent



Gradient descent

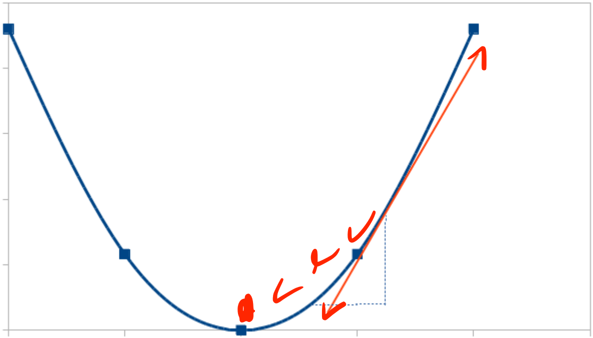
Gradient descent algorithm.

Repeat until convergence
$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta_0, \Theta_1)$

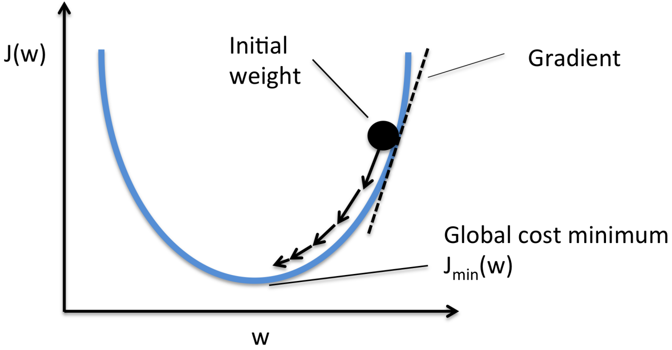
Simultaneous update:
$temp0 = temp0 - \alpha \frac{\partial}{\partial \Theta_0} J(\Theta_0, \Theta_1)$
$temp1 = temp1 - \alpha \frac{\partial}{\partial \Theta_1} J(\Theta_0, \Theta_1)$
$\Theta_0 = temp0$
$\Theta_1 = temp1$

Gradient descent

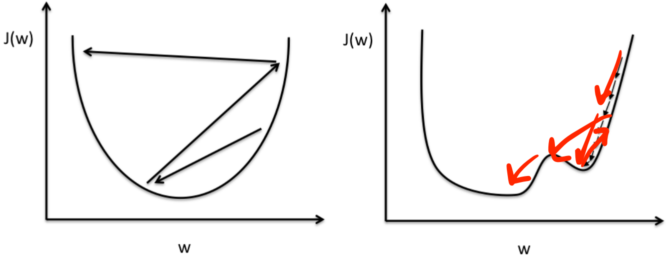
$$\Theta_1 = \Theta_1 - \alpha \frac{\partial}{\partial \Theta_1} J(\Theta_0, \Theta_1)$$



Gradient descent



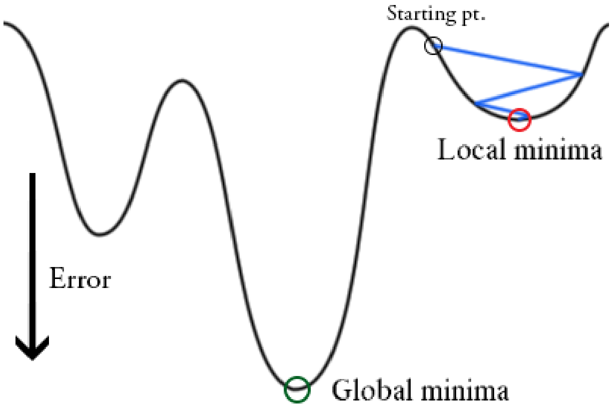
Gradient descent



Large learning rate: Overshooting.

Small learning rate: Many iterations until convergence and trapping in local minima.

Gradient descent



Gradient descent

$$\alpha \frac{\partial}{\partial \Theta_1} \frac{1}{2m} \sum_m \left(\Theta_0 + \Theta_1 x^i - y^i \right)^2 =$$

$$1 \qquad x_1 \qquad 1$$

Gradient descent for linear regression.

Repeat until convergence

$$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta_0, \Theta_1) = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} \frac{1}{2m} \sum_m^{i=1} (\Theta_0 + \Theta_1 x^i - y^i)^2$$

$$\Theta_0 = \Theta_0 - \alpha \frac{\partial}{\partial \Theta_j} \frac{1}{m} \sum_m^{i=1} (h_\Theta(x^i) - y^i)$$

$$\Theta_1 = \Theta_1 - \alpha \frac{\partial}{\partial \Theta_j} \frac{1}{m} \sum_m^{i=1} (h_\Theta(x^i) - y^i) x^i$$

$$= \frac{2}{2m} (\Theta \ldots) x^i$$

Multiple features

Training set of housing prices.

Size (x)	Price (y)
20	500
40	800
65	1300
...	...

Notation:
m = number of training example
x = input variable features
y = output variable target variable
$h_\Theta(x) = \Theta_0 + \Theta_1 x$

Multiple features

Training set of housing prices.

Size (x) x_1	Bedrooms x_2	Floors x_3	Age x_4	Price (y) y
20	5	1	45	500
40	3	2	20	800
65	3	3	14	1300
...	...	...	...	...

Notation:
n = number of features
x^i = input (features) of i^{th} training example.
x^i_j = value of feature j in i^{th} training example
$h_\Theta(x) = \Theta_0 + \Theta_1 x + \Theta_2 x + \Theta_3 x + \Theta_4 x$

Multiple features

$h_\Theta(x) = \Theta_0 x_0 + \Theta_1 x + \Theta_2 x + \Theta_3 x + \Theta_4 x$ - hypothesis function.

Let's define $x_0 = 1$, so in matrix notation:
$h_\Theta(x) = \underline{\Theta^T x}$

Multiple features

Gradient descent

Hypothesis: $h_\Theta(x) = \Theta_0 + \Theta_1 x + \Theta_2 x + \Theta_3 x + ... + \Theta_n x$
Parameters: $\Theta-$ n-dimensional vector
Cost function: $J(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\Theta(x^i) - y^i)^2$
Gradient descent: Repeat $\{\Theta_j = \Theta_j - \alpha\frac{\partial}{\partial\Theta_j}J(\Theta)\}$ (simultaneously update for every $j = 0, ..., n$

Feature scaling

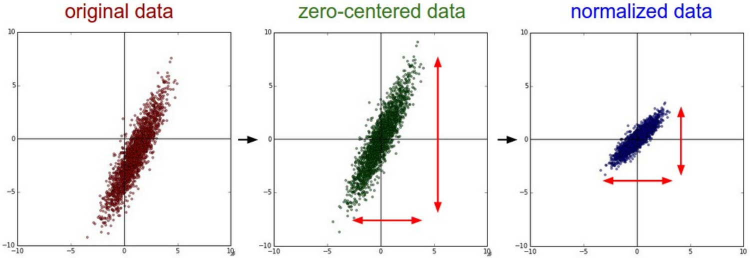
Make sure that your features are on a similar scale.

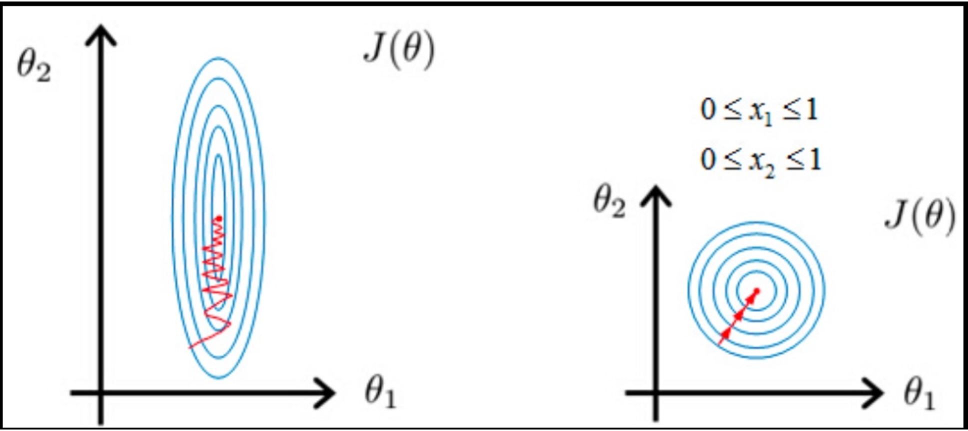
E.g. x_1 – *size* $(0 - 150m^2) \to x_1 = \frac{size}{150}$

x_2 – *number of bedrooms* $(1 - 5) \to x_2 = \frac{number}{5}$

Mean normalization

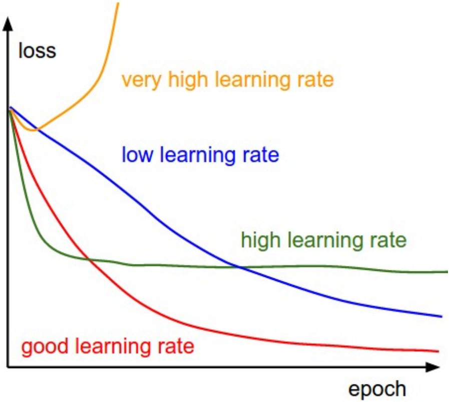
Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean




θ_2
$J(\theta)$
θ_1
$0 \le x_1 \le 1$
$0 \le x_2 \le 1$
θ_2
$J(\theta)$
θ_1

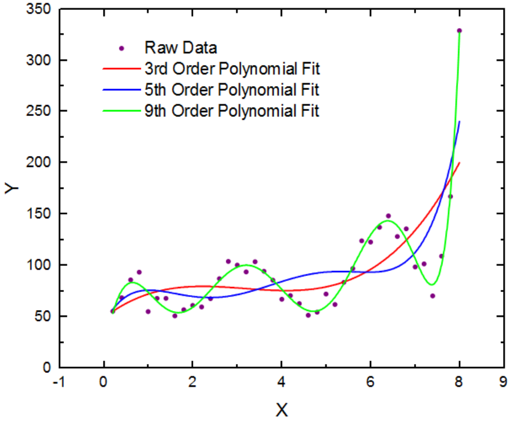
Gradient descent

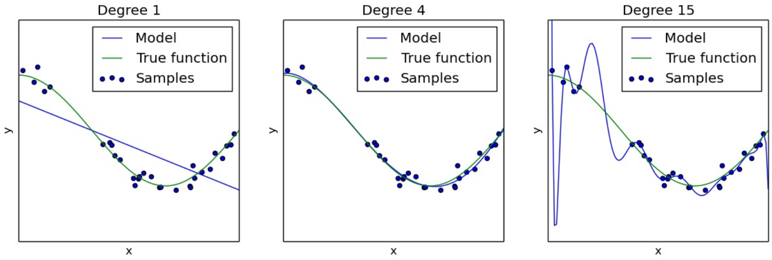
Making sure gradient descent is working correctly.



Polynomial regression

$$h_\Theta(x) = \Theta_0 + \Theta_1 x + \Theta_2 x + \Theta_3 x = \Theta_0 + \Theta_1(size) + \Theta_2(size)^2 + \Theta_3(size)^3$$

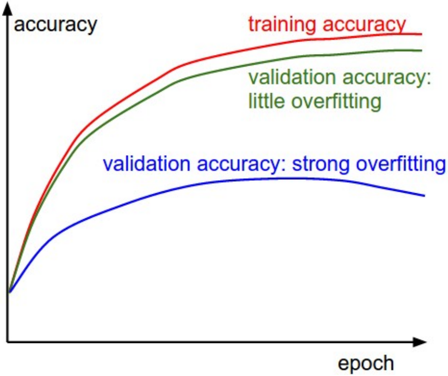




Degree 1
Model
True function
Samples
y
x
Degree 4
Model
True function
Samples
y
x
Degree 15
Model
True function
Samples
y
x

Overfitting

Overfitting refers to a model that models the training data too well. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance on the model on new data.



Homework

Logistic regression

1. Coursera. Andrew Ng course. Week 3.
2. Scikit-learn documentation