# Linear Regression

<table>
<thead>
<tr>
<th>Year</th>
<th>Make, Model</th>
<th>Kilometers</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>&quot;Chevrolet Cobalt LS&quot;</td>
<td>70861</td>
<td>$8989</td>
</tr>
<tr>
<td>2007</td>
<td>&quot;Chevrolet Cobalt LS&quot;</td>
<td>71335</td>
<td>$7979</td>
</tr>
<tr>
<td>2007</td>
<td>&quot;Chevrolet Cobalt LS&quot;</td>
<td>55711</td>
<td>$9594</td>
</tr>
<tr>
<td>2007</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>36171</td>
<td>$9994</td>
</tr>
<tr>
<td>2007</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>37606</td>
<td>$10598</td>
</tr>
<tr>
<td>2007</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>8397</td>
<td>$11592</td>
</tr>
<tr>
<td>2007</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>23275</td>
<td>$10392</td>
</tr>
<tr>
<td>2007</td>
<td>&quot;Chevrolet Cobalt SS&quot;</td>
<td>21708</td>
<td>$11697</td>
</tr>
<tr>
<td>2007</td>
<td>&quot;Chevrolet Cobalt SS&quot;</td>
<td>52632</td>
<td>$13993</td>
</tr>
<tr>
<td>2008</td>
<td>&quot;Chevrolet Cobalt LS&quot;</td>
<td>20627</td>
<td>$8989</td>
</tr>
<tr>
<td>2008</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>47503</td>
<td>$9993</td>
</tr>
<tr>
<td>2008</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>46437</td>
<td>$11793</td>
</tr>
<tr>
<td>2008</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>28972</td>
<td>$11994</td>
</tr>
<tr>
<td>2008</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>11539</td>
<td>$11692</td>
</tr>
<tr>
<td>2008</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>21325</td>
<td>$10993</td>
</tr>
<tr>
<td>2008</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>48068</td>
<td>$11991</td>
</tr>
<tr>
<td>2008</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>44414</td>
<td>$9999</td>
</tr>
<tr>
<td>2008</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>53510</td>
<td>$8989</td>
</tr>
<tr>
<td>2008</td>
<td>&quot;Chevrolet Cobalt Sport&quot;</td>
<td>29082</td>
<td>$12894</td>
</tr>
<tr>
<td>2008</td>
<td>&quot;Chevrolet Cobalt Sport&quot;</td>
<td>22278</td>
<td>$12994</td>
</tr>
<tr>
<td>2008</td>
<td>&quot;Chevrolet Cobalt Sport&quot;</td>
<td>20988</td>
<td>$13494</td>
</tr>
<tr>
<td>2008</td>
<td>&quot;Chevrolet Cobalt Sport&quot;</td>
<td>57736</td>
<td>$12794</td>
</tr>
<tr>
<td>2009</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>9830</td>
<td>$12994</td>
</tr>
<tr>
<td>2010</td>
<td>&quot;Chevrolet Cobalt LT&quot;</td>
<td>26590</td>
<td>$12993</td>
</tr>
</tbody>
</table>
Linear Regression

- \( m \) = number of instances
- \( x \) = input variable/"instances"
- \( y \) = output variable/"target variables"

\( x \)=kilometers
\( y \)=price

What is the hypothesis? How do we represent it? Decision tree?

Approximate by a straight line segment

\[
h_\theta(x) = \theta_0 + \theta_1 x
\]
Learning

- How to find the correct values for $\theta_0$, $\theta_1$?
- Pick values that are close to the training data
Learning

- Define a cost function that penalizes hypothesis that are far away from the data points
- Mean squared error

\[ J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \]

- Find the parameters theta that minimize the cost function
- For a fixed \( \theta \), the cost is determined by \( x \) and \( y \)
- In simple cases, we can use the normal equation to find the minimum analytically
  - Approximating a line to a set of data
  - Penrose Moore pseudo inverse method
In more complex cases, calculating the inverse of a large matrix is very expensive.

Try and **search** the values that minimize the cost function using gradient descent.

Until solution has been found:

- Make an initial guess of $\theta_0, \theta_1,\ldots$
- Calculate the gradient of $J(\theta_0, \theta_1,\ldots)$ and take a small step in that direction

How do you know which way to go?
Gradient Descent

- Mathematically: Calculate the partial derivative for $\theta_0, \theta_1, ...$
- Partial derivative is calculating the derivative of the parameter $\theta_j$ assuming all other values are constant.
- Take small step in the direction of the gradient to find the minimum. $\alpha$ is the learning rate.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \text{(for } j = 0 \text{ and } j = 1)$$
Gradient Descent

- Hypothesis
  \[ h_{\theta}(x) = \theta_0 + \theta_1 x \]

- Cost function of the linear regression
  \[ J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_{\theta}(x^{(i)}) - y^{(i)})^2 \]

- Calculate partial derivative analytically
  - Chain rule: \((f \ast g)'(t) = f'(g(t)) \ast g'(t)\)
Gradient Descent

Gradient Descent Update Rule

\[
\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta(x^{(i)}) - y^{(i)} \right) \\
\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}
\]

Update \( \theta_0 \) and \( \theta_1 \) simultaneously

Trick: Create a virtual feature \( x_0 \), which is always 1 and associate with \( \theta_0 \)
Gradient Descent (House Prices)
Gradient Descent (House Prices)

Cost function and trajectory of gradient descent through the cost function
Gradient Descent on Car Price Dataset

Gradient Descent Raw Video of Car Price Dataset
Gradient Descent on Car Price Data Set
What happened?

- Oscillation during gradient descent
  - Reduce learning rate
  - But then it would take much too long to learn

Bug in computation of gradients?

Hypothesis $\theta$ has one entry for every feature

Check gradient numerically

```python
e = 1.0e^{-4}  # A small constant
for i in num_features
    p(...) = 0; p(i) = e
    l1 = J(\theta - p)
    l2 = J(\theta + p)
    grad(i) = (l2-l1)/(2*e)
```

$g_{calc} - g_{numeric} < 1.0e^{-18}$ // So very close match in the beginning

$g_{calc} - g_{numeric} > 1.0e^{100}$ // At the end, $g_{calc} = NaN$
Gradient Descent Car Price Dataset

- What happened?
- Oscillation during gradient descent
  - Reduce learning rate
  - But then it would take much too long to learn
- Look at the features
  - 1
  - 10,000
- The gradient in the second attribute (kilometers) will have much more influence then the first attribute (offset=1)
Feature Scaling

- A common problem in Machine Learning is that features may have vastly different ranges.
- If features have values that differ by orders of magnitude, then those features will dominate the descent.
- Feature scaling:
  - Try reduce feature values into a normal range (e.g., -1 .. +1).
- Fixed Scaling:
  - Multiply each feature by some constant.
  - E.g., kilometers * 1/10,000.
- Mean Normalization:
  - Calculate mean \( \mu \).
  - Calculate standard deviation \( \sigma \) of the feature.
  - Calculate new feature \( x'_i \):
    \[
    x'_i = \frac{(x_i - \mu)}{\sigma}
    \]
Gradient Descent with Feature Scaling
Gradient Descent: Convergence

- In linear systems, the cost function is shaped like a bowl
- Only one (global) minimum
- Gradient descent can not be trapped in a local minimum
  - if the learning rate is small enough
  - otherwise it may oscillate as shown in the car price example
Gradient Descent

- Cost function $J$ is a function of $\theta$. Calculate slope around current estimate. Take one small step in the direction of the slope.
Gradient Descent

- May lead to a different end point
Gradient Descent: Learning Rate

- The learning rate is between 0 to 1
- If the learning rate is too small
  - Slow convergence
  - Slow learning
- If the learning rate is too high
  - Faster convergence
  - Oscillation
  - May even lead to divergence
- Use exponential scale to try and find a good learning rate
  - 0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, ...
  - Once performance improves check in that neighbourhood
- As the hypothesis approaches the minimum, the partial derivatives are going to flatten out
  - Convergence is going to slow down
  - No need to decrease learning rate as time progresses
Multivariate Gradient Descent

- We can use the same approach with instances with multiple features
- So we can have more complex instances
  - Kilometers, Age, Maximum speed, Gas mileage, ...
- The hypothesis will be given by

\[ h(\theta) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots \]

- where
  - \(\theta_0\) = offset, \(x_0 = 1\) for all instances
  - \(\theta_1\) = influence of kilometers
  - \(\theta_2\) = influence of age
  - ...
Polynomial Regression

- Sometimes our data does not match a straight line
Polynomial Regression

- Once you notice that linear regression stops, you can look at the instances that failed
- Create a new virtual feature to help the system
  - $x_2 = \sqrt{x_1}$
- Now we still have a linear regression problem, since the relationship between the hypothesis and the instances is linear
- Can you standard multivariate linear regression algorithm
Logistic Regression

- Logistic regression is an extension of linear regression to discrete classification problems (e.g., heart attack risk)
- Assume that we have two classes
  - $y=0$ (Healthy)
  - $y=1$ (Not healthy)
- First attempt
  - Threshold classifier
  - If $h_{\theta}(x) > 0.5$, then 1, 0 else
Logistic Regression

- Problems if data is not spread uniformly
- Classification is 0 or 1 only
- $h_\theta(x)$ can be from - infinity to +infinity
- Scale $h_\theta(x)$ into 0 to 1
Logistic Regression: Sigmoid Function

- Logistic/Sigmoid function
- Decision boundary
  - $h_\theta(x) = 0.5$?
  - $\Rightarrow \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 = 0$?

$$g(z) = \frac{1}{1 + e^{-z}}$$
Logistic Regression

- Result now reflects probability that $y=1$
  - $h_\theta(x) = 0.7$ :- $x$ has a 70% chance of being at risk

- Cost function
  - To apply gradient descent we must define the cost function
  - Prefer a convex cost function with no local minimas
  - Standard cost function from linear regression leads to many oscillations. Many local minimas
Logistic Regression: Cost Function

● If $y = 1$
  ○ values close to 1 should have small penalty
  ○ values close to 0 should have high penalty
  ○ function should have a simple derivative
● $\text{Cost}(h_\theta(x)) = -\log(h_\theta(x))$
● If $y = 0$
  ○ $\text{Cost}(h_\theta(x)) = -\log(1 - h_\theta(x))$
Logistic Regression

- Define new cost function that takes both cases
- \( J(\theta) \)

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

- if \( y=1 \), then right part is 0
- if \( y=0 \), then left part is 0
- Minimize the cost function using gradient descent
- Make a prediction by calculating

\[
h_{\theta}(x) = \frac{1}{1 + e^{-(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots)}}
\]
Regression: Update Rule

- The update rule for regression is given by

\[ \theta_j = \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \]

- Same update rule as linear regression
- Same update rule as polynomial regression
Regression: Multiple classes

• One vs All classification is used to apply logistic regression to classification problems with multiple classes \((c_0, c_1, c_2)\)
  ○ Task 1: \(y = 1\) if \(c_0 = 1\), 0 else
  ○ Task 2: \(y = 1\) if \(c_1 = 1\), 0 else
  ○ Task 3: \(y = 1\) if \(c_2 = 1\), 0 else

• Return the classification which maximizes \(h_\theta(x^i)\)