Advanced Algorithms

Floriano Zini
Free University of Bozen-Bolzano
Faculty of Computer Science
Academic Year 2013-2014

Lecture 11 – Linear regression

These slides are taken from Andrew Ng, Machine Learning on Coursera - https://class.coursera.org/ml-003/lecture/preview
Machine Learning definition

Arthur Samuel (1959)

- Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed

Tom Mitchell (1998) Well-posed Learning Problem:

- A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E
Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam. What is the task $T$ in this setting?

- Classifying emails as spam or not spam
- Watching you label emails as spam or not spam.
- The number (or fraction) of emails correctly classified as spam/not spam.
- None of the above—this is not a machine learning problem

“A computer program is said to learn from experience $E$ with respect to some task $T$ and some performance measure $P$, if its performance on $T$, as measured by $P$, improves with experience $E$.”

Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam. What is the task $T$ in this setting?

- Classifying emails as spam or not spam
- Watching you label emails as spam or not spam.
- The number (or fraction) of emails correctly classified as spam/not spam.
- None of the above—this is not a machine learning problem

$T$  $E$  $P$

- Classifying emails as spam or not spam
- Watching you label emails as spam or not spam.
- The number (or fraction) of emails correctly classified as spam/not spam.
- None of the above—this is not a machine learning problem
Examples of ML applications (1)

Email spam filtering
• T: to predict which email are spam for a given user
• P: % of emails correctly predicted
• E: a set of emails identified as spam/non spam for the user

Web pages categorization
• T: to categorize Web pages in predefined categories
• P: % of Web pages correctly categorized
• E: a set of Web pages with specified categories

Examples of ML applications (2)

Handwriting recognition
• T: to recognize and classify handwritten words within images
• P: % of words correctly classified
• E: a set of handwritten words with given classifications (i.e., labels)

Robot driving
• T: to drive on public highways using vision sensors
• P: average distance traveled before an error (as judged by human observer)
• E: a sequence of images and steering commands recorded while observing a human driver
Examples of ML applications (3)

Medical diagnosis
- **T**: suggest treatments to doctors
- **P**: % of suggestions accepted by the doctors
- **E**: a database of electronic health records including <observations,treatment> pairs

Movie recommendation
- **T**: to recommend people movies they like
- **P**: overall satisfaction of the users
- **E**: movie ratings given by the users of the system

Machine learning algorithms
- Supervised learning
- Unsupervised learning (e.g., clustering)

In this lecture 1 algorithm for supervised learning
  - Linear regression

These topics are expanded in the courses:
  - Data Mining by Dr. Mouna Kacimi (second semester)
  - Information Search and Retrieval by Prof. Francesco Ricci (second semester)
**Supervised learning**

**Housing price prediction**

![Graph showing housing price prediction](image)

- **Supervised Learning**
  - "right answers" given

- **Regression:** Predict continuous valued output (price)

**Supervised learning**

**Breast cancer (malignant, benign)**

![Graph showing breast cancer classification](image)

- **Classification**
  - Discrete valued output (0 or 1)

In this learning problem we consider just one feature, i.e., the tumor size.
In this learning problem we consider two features, i.e., the tumor size, and the age.

In general, in real cases there are much more features. E.g.,
- Clump Thickness
- Uniformity of Cell Size
- Uniformity of Cell Shape
- ...

You’re running a company, and you want to develop learning algorithms to address each of two problems:
- Problem 1: You have a large inventory of identical items. You want to predict how many of these items will sell over the next 3 months
- Problem 2: You’d like software to examine individual customer accounts, and for each account decide if it has been hacked/compromised

Should you treat these as classification or as regression problems?
- Treat both as classification problems
- Treat problem 1 as a classification problem, problem 2 as a regression problem
- Treat problem 1 as a regression problem, problem 2 as a classification problem
- Treat both as regression problems
You’re running a company, and you want to develop learning algorithms to address each of two problems

- Problem 1: You have a large inventory of identical items. You want to predict how many of these items will sell over the next 3 months
- Problem 2: You’d like software to examine individual customer accounts, and for each account decide if it has been hacked/compromised

Should you treat these as classification or as regression problems?

- Treat both as classification problems
- Treat problem 1 as a classification problem, problem 2 as a regression problem
- Treat problem 1 as a regression problem, problem 2 as a classification problem
- Treat both as regression problems

✔

- 0 - not hacked
- 1 - hacked

Linear regression with one variable
Supervised Learning
Given the “right answer” for each example in the data

Regression Problem
Predict real-valued output

Model representation

Housing Prices (Portland, OR)

<table>
<thead>
<tr>
<th>Price (in 1000s of dollars)</th>
<th>Size (feet²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>500</td>
<td>500</td>
</tr>
</tbody>
</table>

Notation:
- \( m \) = Number of training examples
- \( x \)'s = “input” variable / features
- \( y \)'s = “output” variable / “target” variable
- \((x, y)\) = one training example
- \((x^{(i)}, y^{(i)})\) = \(i^{th}\) training example

Training set of housing prices (Portland, OR)

<table>
<thead>
<tr>
<th>Size in feet² (x)</th>
<th>Price ($) in 1000's (y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2104</td>
<td>460</td>
</tr>
<tr>
<td>1416</td>
<td>232</td>
</tr>
<tr>
<td>1534</td>
<td>315</td>
</tr>
<tr>
<td>852</td>
<td>178</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\( m = 47 \)

E.g.,
- \(x^{(1)} = 2104\)
- \(x^{(2)} = 1416\)
- \(y^{(3)} = 315\)
- \((x^{(3)}, y^{(3)}) = (1534, 315)\)
Model representation

Training Set → Learning Algorithm

Size of house \( x \) → \( h \) → Estimated price \( \text{estimated value of } y \)

\( h \) maps from \( x \)'s to \( y \)'s

How do we represent \( h \) ?

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

shorthand: \( h[x] \)

Linear regression with one variable.
Univariate linear regression.

Cost function

<table>
<thead>
<tr>
<th>Training set</th>
<th>Size in feet(^2) (( x ))</th>
<th>Price ($) in 1000's (( y ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>2104</td>
<td>460</td>
<td></td>
</tr>
<tr>
<td>1416</td>
<td>232</td>
<td></td>
</tr>
<tr>
<td>1534</td>
<td>315</td>
<td></td>
</tr>
<tr>
<td>852</td>
<td>178</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>( m = 47 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hypothesis: \( h_\theta(x) = \theta_0 + \theta_1 x \)

\( \theta_i \)'s: Parameters

How to choose \( \theta_i \)'s ?
Cost function

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

\[ h(x) = 1.5 + 0^*x \]
\[ \theta_0 = 1.5 \]
\[ \theta_1 = 0 \]

\[ h(x) = 0.5^*x \]
\[ \theta_0 = 0 \]
\[ \theta_1 = 0.5 \]

\[ h(x) = 1+0.5^*x \]
\[ \theta_0 = 1 \]
\[ \theta_1 = 0.5 \]

**QUIZ**

Consider the plot below of \( h_\theta(x) = \theta_0 + \theta_1 x \). What are \( \theta_0 \) and \( \theta_1 \)?

- \( \theta_0 = 0, \theta_1 = 1 \)
- \( \theta_0 = 0.5, \theta_1 = 1 \)
- \( \theta_0 = 1, \theta_1 = 0.5 \)
- \( \theta_0 = 1, \theta_1 = 1 \)
Cost function

Idea: Choose \( \theta_0, \theta_1 \) so that \( h_\theta(x) \) is close to \( y \) for our training examples \((x, y)\)

Cost function

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

Goal: minimize \( J(\theta_0, \theta_1) \) \( J(\theta_0, \theta_1) \) is also called squared error function

Consider the plot below of \( h_\theta(x) = \theta_0 + \theta_1 x \). What are \( \theta_0 \) and \( \theta_1 \)?

- \( \theta_0 = 0, \theta_1 = 1 \)
- \( \checkmark \) \( \theta_0 = 0.5, \theta_1 = 1 \)
- \( \theta_0 = 1, \theta_1 = 0.5 \)
- \( \theta_0 = 1, \theta_1 = 1 \)
Cost function intuition 1

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

Parameters:
\[ \theta_0, \theta_1 \]

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

Goal: minimize \( J(\theta_0, \theta_1) \)

Simplified
\[ \hat{h}_\theta(x) = \theta_1 x \]

\[ \theta_1 \]

\[ J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{h}_\theta(x^{(i)}) - y^{(i)})^2 \]

minimize \( J(\theta_1) \)
Cost function intuition 1

\( h_\theta(x) \)
(for fixed \( \theta_1 \), this is a function of \( x \))

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

\[ = \frac{1}{2 \times 3} \times [(-0.5-1)^2 + (1-2)^2 + (1.5-3)^2] \approx 0.58 \]

\( J(0.5) = 0.58 \)

\( J(0) = 2.3 \)
Suppose we have a training set with m=3 examples, plotted below. Our hypothesis representation is \( h_\theta(x) = \theta_1 x \), with parameter \( \theta_1 \). The cost function \( J(\theta_1) \) is
\[
J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2.
\]
What is \( J(0) \)?

- 0
- \( \frac{1}{6} \)
- 1
- \( \frac{14}{6} \)
Cost function intuition 2

Hypothesis: \( h_\theta(x) = \theta_0 + \theta_1 x \)

Parameters: \( \theta_0, \theta_1 \)

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

Goal: \( \min_{\theta_0, \theta_1} J(\theta_0, \theta_1) \)
Cost function intuition 2

\( h_\theta(x) \)
(for fixed \( \theta_0, \theta_1 \), this is a function of \( x \))

\( J(\theta_0, \theta_1) \)
(function of the parameters \( \theta_0, \theta_1 \))

Contour plot

\[ h(x) = 800 - 1.5x \]

\[ h(x) = 360 + 0 \cdot x \]
Cost function intuition 2

\[ h_\theta(x) \]
(for fixed \( \theta_0, \theta_1 \), this is a function of \( x \))

\[ J(\theta_0, \theta_1) \]
(function of the parameters \( \theta_0, \theta_1 \))

Gradient descent

Have some function \( J(\theta_0, \theta_1) \)

Want \( \min_{\theta_0, \theta_1} J(\theta_0, \theta_1) \)

More in general:

\( J(\theta_0, \theta_1, \ldots, \theta_n) \)

\( \min_{\theta} J(\theta_0, \theta_1, \ldots, \theta_n) \)

Outline:

- Start with some \( \theta_0, \theta_1 \) (e.g., \( \theta_0 = 0, \theta_1 = 0 \))
- Keep changing \( \theta_0, \theta_1 \) to reduce \( J(\theta_0, \theta_1) \)
  until we hopefully end up at a minimum
Gradient descent

Gradient descent
Gradient descent algorithm

repeat until convergence {
\[ \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \text{(for } j = 0 \text{ and } j = 1) \]
}

Correct: Simultaneous update

\[
\begin{align*}
\text{temp0} & := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\
\text{temp1} & := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\
\theta_0 & := \text{temp0} \\
\theta_1 & := \text{temp1}
\end{align*}
\]

Incorrect:

\[
\begin{align*}
\text{temp0} & := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\
\theta_0 & := \text{temp0} \\
\text{temp1} & := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\
\theta_1 & := \text{temp1}
\end{align*}
\]

Quiz

Suppose \( \theta_0 = 1, \theta_1 = 2 \), and we simultaneously update \( \theta_0 \) and \( \theta_1 \) using the rule:

\[ \theta_j := \theta_j + \sqrt{\theta_0 \theta_1} \] (for \( j = 0 \) and \( j = 1 \))

What are the resulting values of \( \theta_0 \) and \( \theta_1 \)?

- \( \theta_0 = 1, \theta_1 = 2 \)
- \( \theta_0 = 1 + \sqrt{2}, \theta_1 = 2 + \sqrt{2} \)
- \( \theta_0 = 2 + \sqrt{2}, \theta_1 = 1 + \sqrt{2} \)
- \( \theta_0 = 1 + \sqrt{2}, \theta_1 = 2 + \sqrt{(1 + \sqrt{2}) \cdot 2} \)
Gradient descent intuition

Gradient descent algorithm

repeat until convergence {
\[ \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \]
(simultaneously update \(j = 0\) and \(j = 1\))
}

Learning rate \(\alpha\)
Derivative \(\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)\)

Simplification
\[ \min_{\theta_0, \theta_1} J(\theta_0, \theta_1) \in \mathbb{R} \]
Gradient descent intuition

\[ J(\theta_1), \theta_1 \in \mathbb{R} \]

\[ \theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1) \]

\[ \theta_1 := \theta_1 - \alpha \ast (\text{positive number}) \]

\[ \frac{d}{d\theta_1} J(\theta_1) \leq 0 \]

\[ \theta_1 := \theta_1 - \alpha \ast (\text{negative number}) \]

If \( \alpha \) is too small, gradient descent can be slow.

If \( \alpha \) is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.
What if $\theta_1$ is already at the local minimum?

$\theta_1$ at local optima

Current value of $\theta_1$

Gradient descent intuition

Gradient descent can converge to a local minimum, even with the learning rate $\alpha$ fixed.

$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$

- As we approach a local minimum, gradient descent will automatically take smaller steps.
- So, no need to decrease $\alpha$ over time.
Gradient descent for linear regression

Gradient descent algorithm

repeat until convergence {
\[ \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \]
(for \( j = 1 \) and \( j = 0 \))
}

Linear Regression Model

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

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

We apply gradient descent to minimize \( J(\theta_0, \theta_1) \):
\[ \min_{\theta_0, \theta_1} J(\theta_0, \theta_1) \quad \theta_0, \theta_1 \in \mathbb{R} \]

Gradient descent for linear regression

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

\[ = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2 \]

\[ j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \]

\[ j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)} \]
Gradient descent for linear regression

Gradient descent algorithm

repeat until convergence 

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

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

\[ \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \]

update \( \theta_0 \) and \( \theta_1 \) simultaneously

In general, the local optimum of the cost function \( J \) found by the gradient descent algorithm depends on the initialization of the parameters.
Gradient descent for linear regression

The cost function $J$ for linear regression is always a **convex** (or “bowl-shaped”) function

Only 1 global optimum

Gradient descent for linear regression

$h_\theta(x)$ (for fixed $\theta_0, \theta_1$, this is a function of $x$) $J(\theta_0, \theta_1)$ (function of the parameters $\theta_0, \theta_1$)
Gradient descent for linear regression

\[ h_\theta(x) \]
(for fixed \( \theta_0, \theta_1 \), this is a function of \( x \))

\[ J(\theta_0, \theta_1) \]
(function of the parameters \( \theta_0, \theta_1 \))
Gradient descent for linear regression

\[ h_\theta(x) \]
(for fixed \( \theta_0, \theta_1 \), this is a function of \( x \))

\[ J(\theta_0, \theta_1) \]
(function of the parameters \( \theta_0, \theta_1 \))
Gradient descent for linear regression

\[ h_\theta(x) \]
(for fixed \( \theta_0, \theta_1 \), this is a function of \( x \))

\[ J(\theta_0, \theta_1) \]
(function of the parameters \( \theta_0, \theta_1 \))
Gradient descent for linear regression

\[ h_\theta(x) \quad (\text{for fixed } \theta_0, \theta_1, \text{ this is a function of } x) \]

\[ J(\theta_0, \theta_1) \quad (\text{function of the parameters } \theta_0, \theta_1) \]
Gradient descent for linear regression

“Batch” Gradient Descent

Each step of gradient descent uses all training examples in calculating the derivatives

\[
\begin{align*}
\theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \\
\theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}
\end{align*}
\]

There are variants:

- **Stochastic**: uses 1 example in each iteration
  - Faster algorithm, but may not reach the global minimum, even tough it gets close to it
- **Mini-batch**: uses b examples in each iteration, \(1 < b < m\)

---

**QUIZ**

Which of the following are true statements? Select all that apply.

- [ ] To make gradient descent converge, we must slowly decrease \(\alpha\) over time.

- [ ] Gradient descent is guaranteed to find the global minimum for any function \(J(\theta_0, \theta_1)\).

- [ ] Gradient descent can converge even if \(\alpha\) is kept fixed. (But \(\alpha\) cannot be too large, or else it may fail to converge.)

- [ ] For the specific choice of cost function \(J(\theta_0, \theta_1)\) used in linear regression, there are no local optima (other than the global optimum).
Linear regression with multiple variables

QUIZ

Which of the following are true statements? Select all that apply.

- To make gradient descent converge, we must slowly decrease $\alpha$ over time.
- Gradient descent is guaranteed to find the global minimum for any function $J(\theta_0, \theta_1)$.
- Gradient descent can converge even if $\alpha$ is kept fixed. (But $\alpha$ cannot be too large, or else it may fail to converge.)
- For the specific choice of cost function $J(\theta_0, \theta_1)$ used in linear regression, there are no local optima (other than the global optimum).
## Multiple features

**Single feature (variable)**

<table>
<thead>
<tr>
<th>Size (feet$^2$)</th>
<th>Price ($1000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2104</td>
<td>460</td>
</tr>
<tr>
<td>1416</td>
<td>232</td>
</tr>
<tr>
<td>1534</td>
<td>315</td>
</tr>
<tr>
<td>852</td>
<td>178</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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

**Multiple features (variables)**

<table>
<thead>
<tr>
<th>Size (feet$^2$)</th>
<th>Number of bedrooms</th>
<th>Number of floors</th>
<th>Age of home (years)</th>
<th>Price ($1000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2104</td>
<td>5</td>
<td>1</td>
<td>45</td>
<td>460</td>
</tr>
<tr>
<td>1416</td>
<td>3</td>
<td>2</td>
<td>40</td>
<td>232</td>
</tr>
<tr>
<td>1534</td>
<td>3</td>
<td>2</td>
<td>30</td>
<td>315</td>
</tr>
<tr>
<td>852</td>
<td>2</td>
<td>1</td>
<td>36</td>
<td>178</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Notation:**

- \( n = \) number of features
- \( x(0) = \) input (features) of \( i^{th} \) training example
- \( x(0)_j = \) value of feature \( j \) in \( i^{th} \) training example

\[
x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}
\]

\[
x^{(2)}_3 = 2
\]
Multiple features

Hypothesis:

Previously: \( h_\theta(x) = \theta_0 + \theta_1 x \)

Now: \( h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 \)

E.g.: \( h_\theta(x) = 80 + 0.1 x_1 + 0.01 x_2 + 3 x_3 - 2 x_4 \)

| size | # bedrooms | # floors | age |

Multiple features

For convenience of notation, define \( x_0 = 1 \) \( x_0 = 1 \)

\[
x = \begin{bmatrix}
  x_0 \\
  x_1 \\
  x_2 \\
  \vdots \\
  x_n
\end{bmatrix} \quad \theta = \begin{bmatrix}
  \theta_0 \\
  \theta_1 \\
  \theta_2 \\
  \vdots \\
  \theta_n
\end{bmatrix}
\]

\( h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \theta^T x \)

Multivariate linear regression
Gradient descent for multiple variables

Hypothesis: \( h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \)

Parameters: \( \theta_0, \theta_1, \ldots, \theta_n \rightarrow \theta \in \mathbb{R}^{n+1} \) \( n+1 \) 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 \( j = 0, \ldots, n \))

**Gradient Descent**

Previously \( n=1 \):
Repeat \{ \\
\( \theta_0 := \theta_0 - \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \) \\
\( \theta_1 := \theta_1 - \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \) \\
\} (simultaneously update \( \theta_0, \theta_1 \))

New algorithm \( n \geq 1 \):
Repeat \{ \\
\( \theta_j := \theta_j - \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \) \\
\} (simultaneously update \( \theta_j \) for \( j = 0, \ldots, n \))

E.g., \( n=2 \):
\begin{align*}
\theta_0 &:= \theta_0 - \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \\
\theta_1 &:= \theta_1 - \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \\
\theta_2 &:= \theta_2 - \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}
\end{align*}