Machine Learning Course

Wellington Ricardo Pinheiro

September 27, 2016

1 Intro

Machine Learning is the science of getting computers to learn, without being explicitly programmed.

There are two approaches to work with ML:

- Supervised Learning: where a set of right answers are given is given to the algorithm
- Unsupervised Learning: where a data set is given and the algorithm finds some structure among the data

1.1 Supervised Learning

When we have a data set with right answers and we want to know the answer for some known property of this data set. For example a **House Price Prediction** or a **Weather Prediction** based on past information.

There can be two approaches to solve a problem using a Supervised Learning algorithm:

- **Regression** problem: predict continuous valued output
- **Classification**: Discrete valued output. Work with discrete f(x) values (x can be continuous)

A Supervised Learning Algorithm can be seen as a function f such as:

$$f(x_1, x_2, ..., x_k) = y$$

Where:

- x_i, para $1 \leq i \leq k$, is a **feature**
- y is the valued output

1.2 Unsupervised Learning Algorithm

When we have a data set without answers and we want to discover some structure among the items. Each structural set found is named **cluster**. The Unsupervised Learning Algorithm is also known as a **clustering algorithm**.

2 Linear Regression

Given a data set with right answers, a linear regression is a way to approximate a result based on a straight line that best fits the data set. It's basically a regression problem.

Remember:

- **Supervised Learning**: given the "right answer" for each example in the data
- **Regression Problem**: Predict real-valued output
- **Classification**: Discrete-valued output

Size(m2)	Price in 1000's
2104	460
1416	232
1534	315
852	178

Table 1: Training set

Table 1 shows a distribution that relates the size of houses in $feet^2$ that maps to their prices in 1000s of dollars).
Based on the training set we can define:

- m number of training examples
- x input variable or feature
- y output variable or target variable
- (x, y) one training example
- $(x^{(i)}, y^{(i)})$ the i_{th} training example

Let's also define h as the **hypothesis**. That is, h is a function that maps from $x's$ to a estimated value. In the example of Table 1 h **is a hypothesis that maps from the size of a house to an estimated price**

2.1 How to represent h

h can be represented as a linear polymon in the form:

$$h_\theta(x) = \theta_0 + \theta_1 * x$$

$h(x)$ is a shorthand for $h_\theta(x)$.

This approach to solve problem is known as **Linear regression with one variable** or **univariate linear regression**.

3 Cost Function

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 * x$, where:

- $\theta_i's$: parameters

H ow to choose $\theta_i's$?

Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y). But how to do that? We have to minimize θ_0, θ_1 so that

$$\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$

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

minimize $J(\theta_0, \theta_1)$
$J(\theta_0, \theta_1)$ is the cost function also called Squared error function

x	**y**
1	2
2	2
3	3

Table 2: Cost function training set

4 Cost Function Intuition

- Hypothesis $h_\theta(x) = \theta_0 + \theta_1 * x$
- Parameters θ_0, θ_1
- Cost function $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$
- Goal minimize $J(\theta_0, \theta_1)$

Let's start with a simplified version of $h_\theta = \theta_1 * x$. Then we're going to minimize $J(\theta_1)$.
Given the following training set
Based on training set in Table 2, we can use a $\theta_1 = 1$, then

$$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{1}{2m}\sum_{i=1}^{m}(\theta_1 * x^{(i)} - y^{(i)})^2 = \frac{1}{2m}(0^2 + 0^2 + 0^2) = 0^2$$

5 Gradient Descent

Have some function $J(\theta_0, \theta_1)$ Want to min $J(\theta_0, \theta_1)$
Outline

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at minimum

5.1 Gradient Descent Algorithm

repeat until convergence (for $j = 0$ and $j = 1$ - simultaneously update θ_0 and θ_1) {

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1)$$

}
where:

- α is the learning rate. The bigger it is the more aggressive the approximation occurs.

Simultaneous update of θ_0 and θ_1 means:

- $temp_0 := \theta_0 - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1)$
- $temp_0 := \theta_1 - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1)$
- $\theta_0 := temp0$
- $\theta_1 := temp1$

5.2 Gradient Descent Intuition

Remember that:

repeat until convergence (for $j = 0$ and $j = 1$ - simultaneously update θ_0 and θ_1) {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

}
Where

- α is the learning rate
- $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ is the derivative term

Again, let's simplify and minimize $J(\theta_1)$, with $\theta_1 \in \mathbb{R}$
If α is too small, gradient descent can be slow. But if α is too large, the gradient descent can overshoot the minimum. It may fail to converge, or even diverge.
Note that as each iteration of the algorithm approaches the local minimum, the derivative term becomes smaller. This way it's not needed to update α value over time.

6 Gradient Descent for Linear Regression

Idea apply gradient descent to the cost function $J(\theta_0, \theta_1)$. Then:

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

Dividing in cases for $j = 1$ and $j = 2$:

- $j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_0(x^{(i)}) - y^{(i)})$
- $j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_0(x^{(i)}) - y^{(i)})x^{(i)}$

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_0(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_0(x^{(i)}) - y^{(i)})x^{(i)}$$

}
About the convergence of $J(\theta_0, \theta_1)$ it's important to say that J is a **convex function**, a bowl shaped function that has a global optimum point.

6.1 Batch Gradient Descent

Each step of gradient descent uses all the training examples this is why we call it **Batch**. There are some cases which not all training examples are needed.

7 Linear Algebra Review

8 Quiz Tips

8.1 Week 1 - Introduction

- In the first question is important to observe that experience E **is the data set**, the information the algorithm uses to learn. The task T **is the action performed by the algorithm**, i.e., the task of predicting the Weather, or a Disease, or a House Price. Each of these predictions is based on features. The performance P **is the output given by the algorithm**.

- For questions two and three remember that when the algorithm output domain is a limited discrete set then the problem can be a classification problem, otherwise it's a regression problem. Be aware when the output domain set in discrete but large, that in these cases the output domain can be seen as a real domain and not discrete.
- In question four it's important to note that when clusters are mentioned or characteristics that are not previously known then it's talking about unsupervised learning.

8.2 Week 1 - Linear Regression with One Variable

- TODO

9 References

- Wiki for Week 1 `https://share.coursera.org/wiki/index.php/ML:Main#Week_1`