Machine Learning Course

Wellington Ricardo Pinheiro

October 2, 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

How 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). To do this we define a cost function $J(\theta_0, \theta_1)$, also known as **Squared Error Function**, as:

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

Minimizing $J(\theta_0, \theta_1)$ will give the θ_0 and θ_1 that makes $h_\theta(x)$ close to training examples.

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 $J(\theta_0, \theta_1)$

x	y
1	2
2	2
3	3

Table 2: Cost function training set

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

Size(feet^2)	**Number of bedrooms**	**Number of floors**	**Age of home (years)**	**Price (\$1000ŝ)**
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

Table 3: Multiple Features (variables)

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 {

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

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

}
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 Liner Regression with Multiple Features

In previous chapters the linear regression dealt with only one feature. In this chapters is presented how this approach also works with multiple features. Table 3 shows an example of multiple features that influences the price of a house.
Let's define:

- n: number of features
- $x^{(i)}$: input (features) of i^{th} training example
- $x_j^{(i)}$: value of feature j in i^{th} training example

The hypothesis using multiple features is defined as:

$$h_\theta = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Example: $h_\theta = 80 + 0.1x_1 + 0.01x_2 + 3x_3 - 2x_4$
For convenience we define $x_0 = 1$, so it's possible to rewrite h_θ as:

$$h_\theta = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

X and θ can be represented as matrices:

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \cdots \\ \theta_n \end{bmatrix}$$

Based on the the matrices X and θ $h_\theta(x)$ can be written as:

$$h_\theta(x) = \theta^T x$$

7.1 Gradient Descent for Multiple Variables

For multiple variables the cost function J is defined as: $J(\theta_0, \theta_1, \cdots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$
Instead of writing $J(\theta_0, \theta_1, \cdots, \theta_n)$ will be only used $J(\theta)$.
Gradient Descent
repeat ({

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

} (simultaneously update for every $j = 0, 1, \cdots, n$)
Applying the partial derivative

repeat ({

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

} (simultaneously update for every $j = 0, 1, \cdots, n$)

7.2 Feature Scaling

Make sure features are on similar scale.
Ex: x_1 = size $(0 - 2000 feet^2)$ and x_2 = number of bedrooms $(1 - 5)$.
If scales are very different it'll take a long time to gradient descent finds the local optimum. The approach here is to scale the features.

Get every feature into approximately a $-1 < x_i < 1$ range by dividing each feature by its maximum value. That is:

$$x_1 = \frac{size(feet^2)}{2000} \qquad x_2 = \frac{numberofbedrooms}{5}$$

7.3 Mean Normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (do not apply to $x_0 = 1$). Also divide $x_i - \mu_i$ by the max value minus the min value for the feature. The formula is:

$$\frac{x_i - \mu_i}{S_i}$$

where S_i is the max $-$ min value for feature i.

7.4 The Learning Rate

Convergence test: when $J(\theta)$ decreases less than 10^{-3} it could be considered that θ is an answer.
Summary about learning rate:

- if α is too slow there will be a slow convergence
- if α is too large $J(\theta)$ may not decrease on every iteration; may not converge

7.5 Features and Polynomial Regression

House price prediction

$$h_\theta(x) = \theta_0 + \theta_1 * frontage + \theta_2 * depth$$

in the previous expression $frontage$ is the like x_1 and $depth$ is the x_2. In fact, it's possible to create and use other features in the linear regression. For example, a new feature $area = frontage * depth$ can be used and the hypothesis will be $h_\theta(x) = \theta_0 + \theta_1 x$. Sometimes using new feature can lead to better models.

7.5.1 Polynomial Regression

Depending on how your data is distributed, other model than linear can fit better. For example, can be used a quadratic distribution with a hypothesis $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$ or a cubic $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$.
For example, when dealing with a feature $size$ related to the size of a house, the hypothesis can be defined as $h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2 + \theta_3(size)^3$, then defining $x_1 = size$, $x_2 = (size)^2$ and $x_3 = (size)^3$. Doing this it's possible to use the already known liner regression mechanics for $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$. **A important note** is that in this case feature scaling really matters here.
Other models than can also be used, for example, $h_\theta(x) = \theta_0 + \theta_1 size + \theta_2 \sqrt{size}$.

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`