Machine Learning by Stanford University

Vocabulary

Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed [1].

Supervised Learning: In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output. Supervised learning problems are categorized into "regression" and "classification" problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output.

Unsupervised Learning: Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables. We can derive this structure by clustering the data based on relationships among the variables in the data.

Linear regression
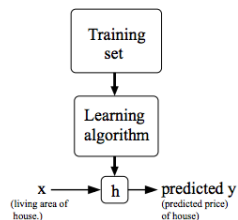
- n: number of features
- m: number of training set
- α: Learning Rate

Hypothesis: $h_\theta(x) = \theta^T x$ with $x_0 = 1$
Parameters: $\theta_0, \theta_1, ..., \theta_n$
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: for fast convergence, Get every feature into approximately a $-1 \leq x_i \leq 1$ range

Normal Equation

$\theta = (X^T X)^{-1} X^T y$
There is no need to do feature scaling with the normal equation.

Comparaison

Gradient Descent	Normal Equation
Need to choose α	No need to choose α
Needs many iterations	No need to iterate
$O(kn^2)$	$O(n^3)$ for $X^T X$
Works well when n is large	Slow if n is very large

Classification

Sigmoid function
$h_\theta(x) = p(y = 1/x; \theta) = \frac{1}{1 + \exp{-\theta^T x}}$
Decision Boundary
$h_\theta(x) \geq 0.5 \Rightarrow y = 1 \quad h_\theta(x) < 0.5 \Rightarrow y = 0$
Logistic regression cost fonction
$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$

$$\begin{cases} \text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) & \text{if y = 1} \\ \text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) & \text{if y = 0} \end{cases}$$

$$\begin{cases} \text{Cost}(h_\theta(x), y) = 0 & \text{if } h_\theta(x) = y \\ \text{Cost}(h_\theta(x), y) \to \infty & \text{if } y = 0 \text{ and } h_\theta(x) \to 1 \\ \text{Cost}(h_\theta(x), y) \to \infty & \text{if } y = 1 \text{ and } h_\theta(x) \to 0 \end{cases}$$

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

Gradient Descent Repeat
$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} y^{(i)} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$
simultaneously update for every j={0,...,n}
Advanced Optimization Matlab

Listing 1: Function

```
1 function [jVal, gradient] = costFunction(
      theta)
2   jVal = [...code to compute J(theta)
      ...];
3   gradient = [...code to compute
      derivative of J(theta)...];
4 end
```

Listing 2: Fminunc

```
1 options = optimset('GradObj', 'on', '
      MaxIter', 100);
2 initialTheta = zeros(2,1);
3   [optTheta, functionVal, exitFlag] =
      fminunc(@costFunction,
      initialTheta, options);
```

Multiclass Classification

One-vs-all
Train a logistic regression classifier $h_\theta(x)$ for each class i to predict the probability that $y = i$.
$h^i(\theta) = P(y = i/x; \theta)$
To make a prediction on a new x, pick the class i that maximizes $h_\theta(x)$.

Overfitting

There are two main options to address the issue of overfitting:

- Reduce the number of features.

- Regularization: Keep all the features, but reduce the magnitude of parameters θ_j

Regularization Cost Function
$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)} - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2$
Gradient Descent

Repeat { $\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$

$\theta_j := \theta_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right]$

$j \in \{1, 2...n\}\}$

Normal Equation $\theta = (X^T X + \lambda \cdot L)^{-1} X^T y$

$$\text{where } L = \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

If $m < n$, then $X^T X$ is non-invertible. However, when we add the term λL, then $X^T X + \lambda L$ becomes invertible.

References

[1] Machine Learning andrew ng. `https://www.coursera.org/learn/machine-learning`.