Machine Learning Course
Notebook

Mariana Jó

June 3, 2019

What is Machine Learning?

- Definitions of ML
  "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."
- Kinds of ML algorithms

Supervised learning

Some definitions:

- Supervised Learning: We give the dataset the "right answers"
- Regression: predict continous valued output
- Classification: discrete valued output
- Support Vector Machines: an algorithm that can deal with an infinite number of features

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. With unsupervised learning there is no feedback based on the prediction results."

- Clustering: groups things that are somehow similar and/or related
- Non-clustering: finds structure. Ex.: cocktail party algorithm.

Model representation

Notation:

- m = Number of training examples
- x = "input" variable / features
- y = "output" variable / "target" variable

- $(x, y) =$ one training example
- $\theta =$parameter

About hypothesis h:

- Hypothesis h: takes the input x and outputs the estimated value y, i.e., h maps from x's to y's.
- How do we represent h?
  We choose an initial choice
  $$h_\theta(x) = \theta_0 + \theta_1 x, \tag{1}$$
  which is a linear function.
- Linear Regression with one variable $=$ Univariate Linar Regression

Cost function:

- We try to minimize $\theta_0\theta_1$
- So we try to minimize
  $$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i \right)^2 \tag{2}$$
  where $h_\theta(x)$ is given by 1 and $J(\theta_0, \theta_1)$ is called the **cost function**. This function is also called **Squared error function** or **Mean Squared Error (MSE)**.
- The mean is halved $(\frac{1}{2})$ as a convenience for the computation of the gradient descent, as the derivative term of the square function will cancel out the $\frac{1}{2}$ term.

Parameter Learning

Gradient descent

- Given $J(\theta_0, \theta_1)$, we want to minimize it.
- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$
- Gradient descent algorithm:
  Repeat until convergence:
  $$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1), \quad (\text{for } j = 0 \text{ and } j = 1) \tag{3}$$
  where α is the **learning rate** and $\alpha \geqslant 0$.
- It is importante to make the simultaneous update of θ_0 and θ_1.
- "Batch" Gradient descent means that each step of gradient descent uses all the training examples, i.e., all the m examples.

For Linear Regression:

- $J(\theta_0, \theta_1)$ will always be a Convex Function, i.e., a "bowl-shaped function"
- This implies that it does not have any local optima, only the global one.

Prediction and Inference

- Prediction: you use variables to predict **values**
- Inference: you use variables to understand behaviors, find structures, etc

Prediction accuracy vs. Model interpretability

Usually, when you want to *predict*, you don't care very much about **interpretability**, because what matters is the accuracy and not how variables are correlated, and then you can have more **flexibility**. On the other hand, when you want to make an *inference*, you also want to **interpret** the algorithm's outcomes, therefore you will end up using less **flexible** methods.

> *"In general, as the flexibility of a method increases, its interpretability decreases."* - ISLR-Gareth [1].

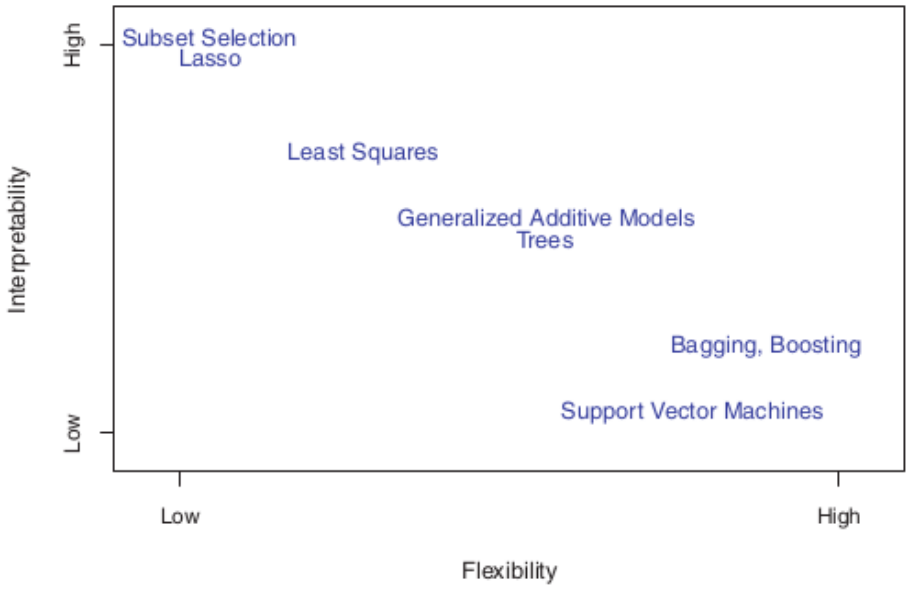

Figure 1: *How the learning algorithms are classified in terms of flexibility vs. interpretability, from ISLR-Gareth [1].*

Multivariate Linear Regression

Let

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \quad \text{and} \quad \theta = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \tag{4}$$

then the Multivariate Linear Regression is defined by

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \theta^T \mathbf{x} \tag{5}$$

Gradient descent

In the case $n \geq 1$ case, you should repeat:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \tag{6}$$

There are some tricks to help the Gradient Descent converge more quickly:

- Feature scaling

  get every feature into approximately a $-1 \leq x \leq 1$ range

  mean normalization: $\frac{x_i - \mu_i}{s_i}$ where μ_i is the mean of the x_i and s_i is the range.

- Debugging: your $J(\theta)$ should descrease at every iteration. If it's increasing, or if it shows an "weird" behavior (like descreasing and increasing over and over) maybe you should use a smaller α. But if $alpha$ is too small, gradient descent can be slow to converge.

Normal equation

It is a method to solve for θ "analytically". In this case, you don't need to use feature scaling.

$$\theta = (X^T X)^{-1} X^T y \tag{7}$$

Gradient descent

- Need choose α
- Needs many itrations
- Works well even when n is large

Normal equation

- No need to chosse α
- Don't need to iterate
- Need to compute $(X^T X)^{-1}$
- Slow if n is very large (usually for $n \geq 10000$)

References

[1] Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, *Introduction to Statistical Learning with Applications in R*, Springer, 2017.