Gradient Descent for Linear Regression

CS 335

Dan Sheldon

Review: Supervised Learning

Observe list of training examples $(x^{(i)}, y^{(i)})$, want to find a function h such that $y^{(i)} \approx h(x^{(i)})$ for all i

Variations:

- Type of x (real number, image, etc.)
- Type of y (real number, 0/1, $\{0, 1, \ldots, k\}$)
- Type of h

Cost function paradigm

Define **parametric** function $h_\theta(x)$ with parameters $\theta_0, \ldots, \theta_n$. E.g.:

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

Define **cost function** $J(\theta_0, \ldots, \theta_n)$ to measure quality (lower is better) of different hypotheses. E.g.:

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

Use a numerical **optimization algorithm** to find $\theta_0, \ldots, \theta_n$ to minimize $J(\theta_0, \ldots, \theta_n)$. E.g., gradient descent.

Gradient descent in higher dimensions

Straightforward generalization to minimize a function $J(\theta_0, \ldots, \theta_n)$ of many variables:

- Intialize θ_j arbitrarily for all j
- Repeat until convergence

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n) \quad \text{for all } j$$

(simultaneuous updates)

Getting started in Python

1. Interactive mode
2. Jupyter
3. Run a script
4. PyCharm