Machine Learning 101

Supervised Learning

"right answers" given

task: estimate more "right answers"

- Regression Problem: continuous value answer
- Classification Problem: discrete valued output

Unsupervised Learning

no "right answer" → clusters

can find structure in data

Linear Regression

Notation

- Training set ⇝ dataset for given problem
- m ⇝ number of examples in training set
- x's ⇝ "input" variables
- y's ⇝ "output"/"target" variables
- $(x^{(i)}, y^{(i)})$ ⇝ ith training example
- hypothesis ⇝ function given by learning algorithm
  - ↳ maps from x's to y's

Cost Function

given: $h_\theta(x) = \theta_0 + \theta_1 x$ ← input; ← hypothesis; ← parameters

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

} minimize this: get θ values that "cost less" :: fit data best

Gradient Descent

* Can reach different local min depending on starting parameters
  - ↳ works best in cases in which there is only one min point (elliptic paraboloid)

Algorithm:

while !convergence { ← learning rate

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

} * Simultaneous update: in literal algorithm, next state must be calculated and stored into temp values all at once, then updated all at once or well

↝ Learning Rate (α)
- too small: gradient descent is too slow
- too big: "big steps", may lead to overshooting and failure to converge

↝ Gradient Descent for Linear Regression

model $\begin{cases} h_\theta(x) = \theta_0 + \theta_1 x \\ J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \end{cases}$ → apply GD to get lowest cost parame

$\theta_j := \theta_j - \alpha \boxed{\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)}$ ↷ $j=0$ ↝ $\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$

↷ $j=1$ ↝ $\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$

$\begin{cases} \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \\ \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)} \end{cases}$

* "Batch" Gradient Descent: each iteration goes through the entire training set

Week 2

↝ Multivariate Linear Regression
- x_n feature contribute to output value Y
- Notation: $x^{(i)}$ is vector containing all features in i-eth example
- Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$
  - ↳ $x_0 = 1$ ↝ $h_\theta(x) = \sum_{i=0}^{n} x_i \theta_i = \theta^T X$
    - θ: vector with all θ_i
    - X: vector with all x_s
- Gradient Descent:

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

↝ Feature Scaling
- Features being on a similar scale can make gradient descent work faster
- get every feature in range $[-1, 1]$ ↝ divide by maximum value
- Mean Normalization: $x_i = \frac{x_i - \mu_i}{\text{range}}$ ↝ average value

↝ Convergence method
- establish a threshold E and if $J(\theta)$ decreases by less than E in one iteration it has converged * hard to determine E
- plot n°iterations x $J(\theta)$ → helpful when choosing α

* It's possible to create new features that enable the fitting of polynomial functions to the data