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

Normal Equations

- Solver for optimum value directly
- Take derivative and equal to zero
- Defining X as a matrix containing all features and Y as the solution vel

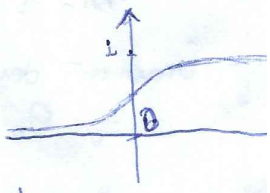
$$\theta = (X^T X)^{-1} X^T y$$

- No need to choose α or to iterate
- Does not scale well to a large number of features

Week 3

Logistic Regression

- Used in classification problems $\rightsquigarrow$ Linear regression is ineffective
- Model (Binary case)
  - $0 \leq h_\theta(x) \leq 1$
  - Logistic function: $g(z) = \frac{1}{1+e^{-z}} \rightsquigarrow h_\theta(x) = g(\theta^T x)$



  - h_θ will output the probability of the correct prediction being 1
- Decision Boundary
  - if $\theta^T x \geq 0 \rightsquigarrow y = 1$ else $y = 0$
  - $\theta^T x$ can represent all kinds of polynomials by adding extra features
- Cost
  - Original cost function is not convex in logist regression $\rightsquigarrow$ gradient descent wouldn't wo
  - Defining:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{case } y=1 \rightarrow Cost = -\log(h_\theta(x))$$

$$\text{case } y=0 \rightarrow Cost = -\log(1 - h_\theta(x))$$

    - In LR, we have:
      - cost zero when prediction correct
      - cost blows up when prediction incorrect
  - Simplified:

$$Cost(h_\theta(x), y) = -y \log(h_\theta(x)) - (1-y)\log(1 - h_\theta(x))$$

- Gradient Descent
  - Same update rule:

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

  - Vectorized:

$$\theta := \theta - \frac{\alpha}{m} X^T (g(\theta X) - \vec{y})$$

- Advanced Optimization Algorithms
  - Conjugate Gradient, BFGS, L-BFGS } provided by language * fminunc()
- Multiclass Classification
  - $\hookrightarrow$ One-vs-all: find h_θ^i for each class and predict whichever maximizes the resu