

Introduction

- Supervised learning - right answers are given
 - Algorithm is given the correct/expected answer
 - Regression - predict continuous valued output
 - Classification - predict results to a discrete output
 - * Can have more than 2 classifications
 - * Models may require infinite number of attributes or features, is done with SVM (support vector machine)
- Unsupervised learning - dataset is not classified, a structure must be predicted
 - Example - cluster classification of news articles
 - Approach problems without an idea of the results or knowledge of effect of variables
 - Derived from clustering data based on variable relationships
 - No feedback

Model and Cost Function

- Linear regression algorithm
 - Fitting a line to data - supervised learning
 - * Predicting a real-valued output
- Notation
 - m is number of training examples
 - x represents the input variable/features
 - y represents output/target variable
 - (x, y) represents a single training example
 - (x_i, y_i) refers to i th training example

Linear Regression

- Process flow
 - Training set \rightarrow learning algorithm $\rightarrow h$
 - h is the hypothesis, function which takes input x and outputs estimated y
 - * Maps $x \rightarrow y$
- $h_\theta(x) = \theta_0 + \theta_1 x$ is the cost function
 - θ_i are parameters that correspond to the regression line
 - * Choose θ_0, θ_1 so $h_\theta(x)$ is close to y for examples in training data (x, y)
- Minimizing the average of training set residuals

θ_0, θ_1 distance to true values which is minimizing

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x_i) - y_i)^2$$

- m is the training set size
- Is the squared error cost function - goal is to minimize
- Halving the mean is for convenience as derivative will cancel it
- Hypothesis is a function of x for some fixed θ_1 and $J(\theta_1)$ is a function of θ_1

Contour plots

- 2 parameters $\theta_0, \theta_1 \rightarrow$ 3D plot paraboloid
 - Height is J
- Can find minimum from contour plot
 - Closer to minimum on a contour plot means better fit

Gradient Descent Algorithm

- Outline
 - Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$
 - Start with some θ_0, θ_1
 - Keep changing θ_0, θ_1 to reduce J until a minimum is reached (for any cost function J)
- Gradient $-\nabla J$ points in direction of steepest *descent*

repeat until convergence {
 $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ (for $j = 0$ and $j = 1$)
 }

- Assignment $:=$ is not same as $=$
 - Can do $a := a + 1$ but not $a = a + 1$
- Simultaneous update must be used

temp 0 $:= \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
 temp 1 $:= \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
 $\theta_0 :=$ temp 0
 $\theta_1 :=$ temp 1

- $\alpha > 0$ is the learning rate and $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ is derivative in direction of θ_j
 - A large α means minimum can be overshoot
 - * Fail to converge -> even diverge
 - Small α means slow descent
- Convergence can occur even with a fixed α since the partial derivative term decreases when minima is approached over time

Gradient Descent for linear regression

- Need to minimize square error cost function

$$\begin{aligned} j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) &= \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \\ j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) &= \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \cdot x^{(1)} \end{aligned}$$

- Linear regression always results in a convex (bow) cost function J
 - Will always converge to the global minimum
- Batch gradient descent - each step uses all training examples

Linear Algebra Review

- Dimension of a matrix is row \times col or $\mathbb{R}^{\text{row} \times \text{col}}$
- A_{ij} is i th row and j th column entry of matrix A
- A vector is a $n \times 1$ matrix of dimension n
 - y_i is the i th element in the vector \vec{y}
 - Can be 0 or 1-indexed
- Given $h_{\theta}(x) = \theta_0 + \theta_1 x$, can use $\vec{p} = D \times \vec{\theta}$ where \vec{p} is the predicted regression values vector of dimension 4, D is $n \times 2$ matrix with column 1 = $\vec{1}$, and $\vec{\theta}$ is the parameter matrix of dimension 2
 - More computationally efficient
- If $C = A \times B$, the i th column of C is $A \times \vec{B}_i$ where \vec{B}_i is the i th column of B
- If applying multiple hypotheses to a data set, use a $2 \times n$ matrix where there are n hypotheses and 2 parameters
- No commutative matrix/vector multiplication but associativity works

- Identity matrix: $A \times I = I \times A = A$ but $AB \neq BA$ if $B \neq A \neq I$
- If A is $m \times m$ and has inverse A^{-1} , then $AA^{-1} = A^{-1}A = I$
- A transpose makes the the i th row the i th column and is reversible
 - If A is an $m \times n$ matrix and $B = A^T$, then B is an $n \times m$ matrix and $B_{ij} = A_{ji}$