0.1 Coursera

Week 1

Machine Learning algorithms : Supervised learning Unsupervised learning Reinforcement Learning Recommender systems

Supervised Learning based on right answers Regression problem: predict continuous valued output (price) Classification problem: output value of 0 or 1 To describe the supervised learning problem slightly more formally, our goal is, given a training set, to learn a function h : X -¿ Y so that h(x) is a good predictor for the corresponding value of y. For historical reasons, this function h is called a hypothesis. Seen pictorially, the process is therefore like this:

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.

We can derive this structure by clustering the data based on relationships among the variables in the data.

With unsupervised learning there is no feedback based on the prediction results. Not told what to do -¿ find some structure in data Clustering Algorithm not give the algorithm the right answers Cocktail party problem: take a data that seems dated together and separate them (two audio sources),

Hypothesis Function

Cost Function Cost Function measures the accuracy of our hypothesis function.

Gradient descent Algorithm Optimize parameters in hypothesis function to minimize the cost function Convex functions has only a global optima, no local optima (good for working with) "Batch" gradient descent: each step of gradient descent uses all the training examples

Learning rate Controls how big the step is in the optimization process

Week 2

Fix a notation thorughout the work m -¿ the number of training examples i n -¿ the number of features j
$x_j^{(i)}$
$i = 1, ..., m$
$j = 0, ..., n$ i are the lines and j the columns. Gradient descent for multiple variables

Feature scale mean normalize data to converge faster (do not apply to inserted x_0 that equals 1 for example). Adjust input values as shown in this

formula.

$$x_i := \frac{x_i - \mu_i}{s_i} \tag{1}$$

where μ_i is the average of all values for feature (i) and s_i is the range of values $(max(x_i) - min(x_i))$ or the standard deviation of a given feature i. get all features between -1 and 1 range or 0 and 1

Gradient descent $\mathcal{O}(n^2)$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{2}$$

works well when n is large

Learning rate how to choose: look graph min J vs No. of iterations. It should decrease at every iteration.

Polynomial regression Create new features to fit a polynomial regression to a linear regression. E.g.

$$x_0 = (feature) \tag{3}$$

$$x_1 = (feature)^2 \tag{4}$$

Analytic solution (faster convergence) The value of $\theta = (X^T X)^{-1} X^T y$ (normal equation) gives the optimal value of θ that minimizes $J(\theta)$ Don't need to iterate nor choose α. Compute the inverse costs a lot $\mathcal{O}(n^3)$. So it is preferably to use if n is small over the iterative method.