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 ith training example

Linear Regression

- Process flow
  - Training set -> learning algorithm -> h
  - h is the hypothesis, function which takes input x and outputs estimated y
    - Maps $x \to y$
- $h_\theta(x) = \theta_0 + \theta_1 x$ 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 θ_0, θ_1 distance to true values which is minimizing $\boxed{J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2}$

  or the average of training set residuals
  - 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 θ_0, θ_1 -> 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
- Gradient $-\nabla J$ points in direction of steepest *descent*

$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J\left(\theta_0, \theta_1\right) \quad (\text{ for } j=0 \text{ and } j=1) \\
&\}
\end{aligned}
$$