Coursera-Stanford-ML-Notes

Quentin Truong

20 June 2017 - ? July 2017

Contents

1 Week 1: Introduction

1.1 Overview

- Machine Learning: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

- Supervised Learning: know what our correct output looks like
  - Regression: want continuous output
  - Classification: want discrete output

- Unsupervised Learning: little or no idea what our results should look like
  - Clustering: find groups according to similarity in various variables
  - Nonclustering: find structure in chaos

2 Week 2: Linear Regression with Multiple Variables

2.1 Overview

– Use linear regression for continuous output

– Choose gradient descent if many features (million+) because the inverse matrix required for the normal equation can become expensive to compute

– Normal equation will directly compute theta

– Normalize features if using gradient descent

2.2 Symbols

$$
\begin{aligned}
m &= number\ of\ samples \\
n &= number\ of\ feature \\
x &= (n \times 1) \\
X &= (m \times n) \\
X_j &= (m \times 1) \\
\theta &= (n \times 1) \\
\theta_j &= (1 \times 1)
\end{aligned}
$$

2.3 Gradient Descent

$$
\begin{aligned}
\text{Hypothesis Function} && h_\theta(x) &= \theta^\intercal \times x \\
\text{Vectorized Hypothesis Function} && h_\theta(X) &= X \cdot \theta \\
\text{Linear Regression Cost Function} && J(\theta) &= \frac{1}{2m}\sum(h_\theta(X) - y)^2 \\
\text{Derivative of Linear Regression CF wrt } \theta_j && \frac{\partial}{\partial\theta_j}J(\theta) &= \frac{1}{m}\sum(h_\theta(X) - y)\ .*\ X_j \\
\text{Change in } \theta_j && \theta_j &= \theta_j - \alpha\frac{\partial}{\partial\theta_j} \\
&& &= \theta_j - \alpha\frac{1}{m}\sum(h_\theta(X) - y)\ .*\ X_j \\
\text{Vectorized Change in } \theta && \theta &= \theta - \alpha\frac{1}{m}X^\intercal(X \cdot \theta - y)
\end{aligned}
$$

2.4 Normal Equation

$$\theta = (X^\intercal \cdot X)^{\text{-1}} \cdot X^\intercal \cdot y$$

3 Week 3: Logistic Regression

3.1 Overview

– Use logistic regression for discrete output (classification)
  - $h_\theta(x) = (y = 1|x;\theta)$; gives probability that the output is 1
  - For multi-class classification, use one-vs-all
  - Sigmoid/Logistic function maps any real number to (0, 1)
  - Pick class i that maximizes $h_\theta^i(x)$

– Overfitting is when learned hypothesis fits training data well but fails to generalize
  - Underfitting is when doesn't fit training data

– Address overfitting by reducing number of features, model selection, and regularization
  - Regularization results in simpler hypothesis and less overfitting
  - Extremely large λ will result in underfitting and gradient descent will fail to converge
  - Do not regularize λ_0

– Use other prewritten optimization algorithims (conjugate gradient, BFGS, L-BFGS) because they are faster

3.2 Logistic Regression Hypothesis Function

$$\begin{aligned}
\text{Sigmoid/Logistic Function} \qquad & g(z) = \frac{1}{1+e^{-z}} \\
\text{Hypothesis Function} \qquad & h_\theta(x) = g(\theta^\intercal x) \\
& = \frac{1}{1+e^{-\theta^\intercal x}}
\end{aligned}$$

3.3 Logistic Regression Cost Function

$$\begin{aligned}
Cost(h_\theta(x), y) &= \begin{cases} -\log(h_\theta(x)) \text{ if y} = 1 \\ -\log(1 - h_\theta(x)) \text{ if y} = 0 \end{cases} \\
&= -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x)) \\
J(\theta) &= \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^i), y^i) \\
J(\theta) &= \frac{-1}{m}\sum_{i=1}^{m} \left[y^i \log(h_\theta(x^i)) + (1-y^i)\log(1-h_\theta(x^i))\right]
\end{aligned}$$

3.4 Proof of Logistic Regression Cost Function Derivative

$$
\begin{aligned}
J(\theta) &= \frac{-1}{m}\sum_{i=1}^{m}[y^i \log(h_\theta(x^i)) + (1-y^i)\log(1-h_\theta(x^i))] \\
\log(h_\theta(x^i)) &= \log(\frac{1}{1+e^{-\theta x^i}}) = -\log(1+e^{-\theta x^i}) \\
\log(1-h_\theta(x^i)) &= \log(1-\frac{1}{1+e^{-\theta x^i}}) = \log(e^{-\theta x^i}) - \log(1+e^{-\theta x^i}) = -\theta x^i - \log(1+e^{-\theta x^i}) \\
J(\theta) &= -\frac{1}{m}\sum_{i=1}^{m}\left[-y^i(\log(1+e^{-\theta x^i})) + (1-y^i)(-\theta x^i - \log(1+e^{-\theta x^i}))\right] \\
&= -\frac{1}{m}\sum_{i=1}^{m}\left[y^i\theta x^i - \theta x^i - \log(1+e^{-\theta x^i})\right] \\
&= -\frac{1}{m}\sum_{i=1}^{m}\left[y^i\theta x^i - \log(e^{\theta x^i}) - \log(1+e^{-\theta x^i})\right] \\
&= -\frac{1}{m}\sum_{i=1}^{m}\left[y^i\theta x^i - \log(1+e^{\theta x^i})\right] \\
\frac{\partial}{\partial\theta_j} y^i\theta x^i &= y^i x^i_j \\
\frac{\partial}{\partial\theta_j}\log(1+e^{\theta x^i}) &= \frac{x^i_j e^{\theta x^i}}{1+e^{\theta x^i}} \\
&= \frac{x^i_j}{1+e^{-\theta x^i}} \\
&= x^i_j h_\theta(x^i) \\
\frac{\partial}{\partial\theta_j}J(\theta) &= -\frac{1}{m}\sum_{i=1}^{m}\left[y^i x^i_j - x^i_j h_\theta(x^i)\right] \\
\frac{\partial}{\partial\theta_j}J(\theta) &= \frac{1}{m}\sum_{i=1}^{m}\left[h_\theta(x^i) - y^i\right] x^i_j
\end{aligned}
$$

3.5 Regularization

Regularizing Term $\quad \lambda\sum_{j=1}^{n}\theta_j^2$

Regularized Linear Regression CF $\quad J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i)-y^i)^2 + \lambda\sum_{j=1}^{n}\theta_j^2$

Regularized Logistic Regression CF $\quad J(\theta) = \frac{-1}{m}\sum_{i=1}^{m}\left[y^i\log(h_\theta(x^i)) + (1-y^i)\log(1-h_\theta(x^i))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$

Regularized GD (Lin/Log Regression) $\quad \begin{cases} \theta_0 = \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i)-y^i)x^i_0\right] \\ \theta_j = \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i)-y^i)x^i_j + \frac{\lambda}{m}\theta_j\right] \text{ (j=1,2,...,n)} \end{cases}$

Regularized Normal Equation $\quad \theta = (X^\intercal X + \lambda \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}_{n+1,n+1})^{-1} X^\intercal y$

4 Week 4: Artificial Neural Networks

4.1 Overview

– Non linear classification of problems with many features
  - Necessary b/c 100 features at 3rd level polynomials generates 170k features; infeasible to deal with
  - "One learning algorithm" hypothesis

4.2 Structure and symbols

– Three Layer System
  - Layer 1: Input nodes
  - Layer 2: Hidden/intermediate layer
  - Layer 3: Output layer

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^2 \\ a_2^2 \\ a_3^2 \end{bmatrix} \rightarrow h_\Theta(x) \tag{1}$$

$$\begin{aligned} a_1^2 &= g(\Theta_{10}^1 x_0 + \Theta_{11}^1 x_1 + \Theta_{12}^1 x_2 + \Theta_{13}^1 x_3) \\ a_2^2 &= g(\Theta_{20}^1 x_0 + \Theta_{21}^1 x_1 + \Theta_{22}^1 x_2 + \Theta_{23}^1 x_3) \\ a_3^2 &= g(\Theta_{10}^1 x_0 + \Theta_{31}^1 x_1 + \Theta_{32}^1 x_2 + \Theta_{33}^1 x_3) \\ h_\Theta(x) &= a_1^3 \\ &= g(\Theta_{10}^2 a_0^2 + \Theta_{11}^2 a_1^2 + \Theta_{12}^2 a_2^2 + \Theta_{13}^2 a_3^2) \end{aligned}$$

– Symbols
  - a_i^j: "activation" of unit i in layer j
  - Θ^j: matrix of weights controlling function mapping from layer j to layer j+1; each layer gets own Θ^j

– General System
  - Can have multiple hidden layers
  - Can have multiple outputs (one-vs-all for multi-class classification)
  - If network has s_j units in layer j and s_{j+1} units in layer $j+1$, then Θ^j will be of dimension $s_{j+1} \times s_{j+1}$
    – The +1 comes from the addition in Θ^j of the bias node, x_0 and Θ_0^j