Machine Learning Course Notes
Stanford – Coursera

Bartol Freskura

29-12-2015

Contents

1 Week 5

1.1 Cost Function and Backpropagation

1.1.1 Cost Function for neural networks

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} = y_k^{(i)}\log(h(x^{(i)})) + (1-y_k^{(i)})\log\left(1-(h(x^{(i)}))_k\right)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\theta_{ji}^{(l)}\right)^2$$

1.1.2 Calculating cost error

$$\delta_j^{(L)} = a_j^{(L)} - y_j \text{ — error of node } j \text{ in layer } L$$
$$g'(z) = \tfrac{1}{1+e^{-z}} .* \left(1 - \tfrac{1}{1+e^{-z}}\right) \text{ — sigmoid function derivation}$$
$$\delta^{(L-1)} = (\theta^{(L-1)})^T\delta^{(L)} .* g'(z^{(L-1)}) \text{ — error vector for level } L-1$$

1.1.3 Backpropagation algorithm

Algorithm 1: Backpropagation

Training set — $\left\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\right\}$
Set gradient — $\Delta_{ij}^{(l)} = 0$ for all l, i, j

for $i = 1$ *to* m **do**
  Set $a^{(1)} = x^{(i)}$
  Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$
  Using $y^{(i)}$ compute $\delta^{(i)} = a^{(L)} - y^{(i)}$
  Compute $\delta^{(L-1)}, \delta^{(L-2)}, \delta^{(L-3)}, \ldots, \delta^{(2)}$
  $\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)}\delta_i^{(l+1)}$
end

1.1.4 Gradient Checking

Suppose you have a funciton $f_i(\theta)$ that computes $\frac{\partial}{\partial\theta_i}J(\theta)$; you'd like to check if f_i is outputting correct derivative values.

$$\text{Let } \theta^{(i+)} = \theta + \begin{bmatrix}0\\0\\\vdots\\\epsilon\\\vdots\\0\end{bmatrix} \text{ and } \theta^{(i-)} = \theta - \begin{bmatrix}0\\0\\\vdots\\\epsilon\\\vdots\\0\end{bmatrix}$$

You can now numerically verify $f_i(\theta)$'s correctness by checking, for each i, that:

$$f_i(\theta) = \frac{J(\theta^{(i+)}) - J(\theta^{(i-)})}{2\epsilon}$$

2 Week 6

2.1 Evaluating a Learning Algorithm

What to do after you have implemented the algorithm and the algorithm gives bad results

- Get more training examples
- Try smaller sets of features
- Try gettting additional features
- Try adding polynomial features ($x_1^2, x_2^2, x_1x_2, etc.$)
- Try increasing or decreasing λ

Machine learning diagnostic: A test that you can run to gain insight what is or isn't working with a learning algorithm, and gain guidance as to how best to improve its performance

One way to evaluate your hypothesis is to split the training dataset into two groups.
First group will contain 70% of the training set and it will be called *Training Set*, while the other 30% will be used as the *Test Set*.
We will denote test set examples as: $(x_{test}^{(i)}, y_{test}^{(i)})$

Testing procedure for linear regression

- Learn paramater θ from trianing data (minimizing $J(\theta)$)
- Compute test set error:

$$J_{test}(\theta) = \frac{1}{2*m_{test}} \sum_{i=1}^{m_{test}} = \left(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)}\right)$$

Testing procedure for logistic regression

- Learn paramater θ from trianing data (minimizing $J(\theta)$)
- Compute test set error:

$$J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} = y_{test}^{(i)} \log(h(x_{test}^{(i)})) + (1 - y_{test}^{(i)}) \log h(x_{test}^{(i)})$$

Error with model selection with degree d: $J_{test}(\theta^{(j)})$ is likely to be an optimistic estimate of generilazation error. I.e. our extra parameter (d = degree of polynomial) is fit to test set.

Solution to this problem: Split the data set into three subsets: *Training set*(60%), *Cross validation set*(20%) and *Test set*(20%)
We will denote cross validation set examples as: $(x_{cv}^{(i)}, y_{cv}^{(i)})$

Final procedure for model selection:

1. Choose the model that has the lowest J_{cv}, e.g. $J_{cv}(\theta^{(4)})$ — 4 denotes the polynomial degree
2. Calculate the generalization error for test set $J_{test}(\theta^{(4)})$

2.2 Bias vs. Variance

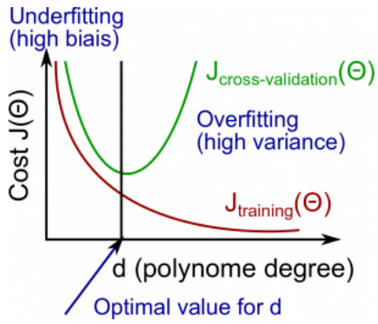
High Bias: both $J_{train}(\theta)$ and $J_{CV}(\theta)$ will be high. Also $J_{train}(\theta) \approx J_{CV}(\theta)$

High Variance: $J_{train}(\theta)$ will be low and $J_{CV}(\theta)$ will be much greater than $J_{train}(\theta)$

A large λ heavily penalizes all the θ parameters, which greatly simplifies the line of our resulting function, so causes underfitting.

The relationship of λ to the training set and the variance set is as follows:

- Low λ: $J_{train}(\theta)$ is low and $J_{CV}(\theta)$ is high (high variance/overfitting).
- Intermediate λ: $J_{train}(\theta)$ and $J_{CV}(\theta)$ are somewhat low and $J_{train}(\theta) \approx J_{CV}(\theta)$
- Large λ: both $J_{train}(\theta)$ and $J_{CV}(\theta)$ will be high (underfitting/high bias)



Our decision process can be broken down as follows:

- Getting more training examples — Fixes high variance
- Trying smaller sets of features — Fixes high variance
- Adding features — Fixes high bias
- Adding polynomial features — Fixes high bias
- Decreasing λ — Fixes high bias
- Increasing λ — Fixes high variance

2.2.1 Diagnosing neural networks

A neural network with fewer parameters is prone to underfitting. It is also computationally cheaper.

A large neural network with more parameters is prone to overfitting. It is also computationally expensive. In this case you can use regularization (increase λ) to address the overfitting.

2.3 Handling Skewed Data

Precision: $\frac{True\ positive}{True\ positive + False\ positive}$

Recall: $\frac{True\ positive}{True\ positive + False\ negative}$

	1	0
1	True Positive	False positive
0	False Negative	True negative