

Machine Learning

Personal Formula Collection

Patrick Bucher

December 19, 2020

Contents

1	Linear Regression	1
1.1	One Variable	1
1.2	Multiple Variables	2
1.3	Normalization	2
2	Classification	2
2.1	Logistic Regression	3
2.1.1	Regularization (Gradient Descent)	3
2.1.2	Regularization (Normal Equation)	4
3	Neural Networks	4
3.1	Activation	4
3.1.1	Vectorization	5
3.2	Cost Function	5
3.3	Forward Propagation	5
3.4	Backpropagation	6
3.5	Gradient Checking	6
3.6	Random initialization	7
4	Error Metrics	7

1 Linear Regression

1.1 One Variable

Prediction:

$$y = h(x) = \theta_0 + \theta_1 x = \theta^T x$$

Cost Function (Squared Error):

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Gradient Descent:

$$\theta = \theta - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)}$$

1.2 Multiple Variables

Prediction:

$$y = h(x) = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n = \theta^T x$$

Cost Function:

$$J(\theta_j) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x_j^{(i)}) - y_j^{(i)})^2$$

Gradient Descent:

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}, j := 0 \dots n$$

An additional feature $x_0 = 1$ is introduced, so that the vector x becomes $n + 1$ dimensional, which simplifies the matrix calculations.

Normal Equation:

$$\theta = (X^T X)^{-1} X^T y$$

Octave (Complexity with n features: $O(n^3)$):

```
theta = pinv(X'*X)*X'*y
```

1.3 Normalization

$$x_i = \frac{x_i - \mu_i}{s_i}$$

Octave:

```
X = (X - mean(X)) ./ std(X)
```

2 Classification

Binary Classification: $y \in \{0, 1\}$, where 0 signifies negative or absent, and 1 signifies positive or present.

2.1 Logistic Regression

$$0 \leq h_{\theta}(x) \leq 1$$

Sigmoid Activation Function g (with asymptotes at y 0 and 1, to be interpreted as probabilities):

$$h_{\theta} = g(\theta^T x), g(z) = \frac{1}{1 + e^{-z}}$$

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Cost Function: $-\log(h_{\theta}(x))$ for $y = 1$ and $-\log(1 - h_{\theta}(x))$ for $y = 0$, combined:

$$C(h_{\theta}(x), y) = -y \cdot \log(h_{\theta}(x)) - (1 - y) \cdot \log(1 - h_{\theta}(x))$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m C(h_{\theta}(x^{(i)}), y^{(i)})$$

With maximum likelihood estimation:

$$J(\theta) = \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \cdot \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \cdot \log(1 - h_{\theta}(x^{(i)})) \right]$$

Prediction:

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Gradient Descent (for each j in θ):

$$\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Vectorized:

$$\begin{aligned} \theta &:= \theta - \frac{\alpha}{m} \sum_{i=1}^m \left[(h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)} \right] \\ \theta &:= \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y}) \end{aligned}$$

2.1.1 Regularization (Gradient Descent)

Regularization mitigates the problem of overfitting for higher-order polynomials.

Regularization term (only regularize θ_j for $j \geq 1$, but not θ_0):

$$\lambda \sum_{j=1}^m \theta_j^2$$

Regularized Cost Function:

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

Regularized Gradient Descent (for θ_j with $j \geq 1$):

$$\begin{aligned} \theta_0 &:= \theta_0 - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} \right] \\ \theta_j &:= \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^m ((h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}) + \frac{\lambda}{m} \theta_j \right] \end{aligned}$$

2.1.2 Regularization (Normal Equation)

To regularize using the normal equation, $(n + 1)(n + 1)$ matrix L with i rows and j columns and the values 1 (for $i = j \wedge i \geq 1 \wedge j \geq 1$) and 0 (all the other indices), respectively, has to be created. (This is an identity matrix of size $n + 1$ with the value at $(0, 0)$ set to 0.)

$$\theta = (X^T X + \lambda L)^{-1} X^T y$$

With regularization, the matrix is always invertible.

3 Neural Networks

Definitions:

- x_0 : bias unit
- $a_i^{(j)}$: activation unit i of layer j
- $\Theta^{(j)}$: weight matrix between layer j and $j + 1$

Given layer j with s_j units, and layer $j + 1$ with s_{j+1} units, the matrix $\Theta^{(j)}$ has the dimensions $s_{j+1} \times (s_j + 1)$.

3.1 Activation

Neural network with three units in the one hidden layer:

$$\begin{aligned} a_1^{(2)} &= g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \dots) \\ a_2^{(2)} &= g(\Theta_{20}^{(2)} x_0 + \Theta_{21}^{(2)} x_1 + \dots) \\ a_3^{(2)} &= g(\Theta_{30}^{(3)} x_0 + \Theta_{31}^{(3)} x_1 + \dots) \\ h_{\Theta}(x) &= a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \dots) \end{aligned}$$

3.1.1 Vectorization

With (forward propagation):

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

And:

$$z_1^{(2)} = \Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3$$

Follows:

$$a_1^{(2)} = g(z_1^{(2)})$$

So that:

$$z^{(2)} = \Theta^{(1)}x = \Theta^{(1)}a^{(1)}$$

Output layer:

$$h_{\Theta} = a^{(3)} = g(z^{(3)})$$

3.2 Cost Function

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

With $(h_{\Theta}(x))_i$ being the i^{th} output. Note that regularization is *not* added to the bias unit, i.e. only for $j \geq 1$.

3.3 Forward Propagation

With a single training example (x, y) . The first activation is the input (a bias unit $a_0^{(1)} = 1$ must be added before):

$$a^{(1)} = x$$

The second activation is computed using Θ and the sigmoid function $g(z)$:

$$z^{(2)} = \Theta^{(2)}a^{(2)}$$

$$a^{(2)} = g(z^{(2)})$$

The bias unit $a_0^{(2)} = 1$ must be added again, then the further activations (l) are computed:

$$z^{(l)} = \Theta^{(l)}a^{(l)}$$

$$a^{(l)} = g(z^{(l)})$$

Finally, the output (layer L) is computed:

$$z^{(L)} = \Theta^{(L)} a^{(L)}$$

$$a^{(L)} = g(z^{(L)}) = h_{\Theta}(x)$$

3.4 Backpropagation

The δ for the rightmost layer L is computed as:

$$\delta^L = a^{(L)} - y$$

The further δ values are computed from right to left, down to $l = 2$ (no δ for the input layer):

$$\delta^{(l)} = \delta^{(l+1)} \Theta^{(l)} g'(z^{(l)})$$

With (bias unit included in $a^{(l)}$):

$$g'(z^{(l)}) = a^{(l)}(1 - a^{(l)})$$

The Δ values are computed as ($a^{(l)}$ *without* bias unit):

$$\Delta^{(l)} = (\delta^{(l+1)})^T a^{(l)}$$

Finally, the gradients D for $j \geq 1$ are computed as follows:

$$D_{ij}^{(l)} = \frac{1}{m} (\Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)})$$

And without regularization for $j = 0$, respectively:

$$D_{ij}^{(l)} = \frac{1}{m} \Delta_{ij}^{(l)}$$

Which is the partial derivative of the cost function:

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

3.5 Gradient Checking

Estimate the derivative of $J(\Theta)$ with $\varepsilon \approx 10^{-4}$ (two-sided difference):

$$\frac{d}{d\Theta} J(\Theta) \approx \frac{J(\Theta + \varepsilon) - J(\Theta - \varepsilon)}{2\varepsilon}$$

The result should only deviate from the D values by a rounding margin.

3.6 Random initialization

When working with neural networks, Θ must be initialized to a random value symmetrically around 0. A (10×11) matrix is initialized as follows (Octave):

```
init_epsilon = 0.1;  
Theta = rand(10,11) * (2 * init_epsilon) - init_epsilon;
```

4 Error Metrics

Confusion Matrix:

		actual	
		1	0
prediction	1	true positive	false positive
	0	false negative	true negative

Precision ($0 \leq P \leq 1$):

$$P = \frac{tp}{tp + fp}$$

Recall ($0 \leq R \leq 1$):

$$R = \frac{tp}{tp + fn}$$

F_1 Score ($0 \leq F_1 \leq 1$):

$$F_1 = 2 \frac{PR}{P + R}$$

Some rules of thumb:

- A higher classification threshold leads to a higher precision and a lower recall.
- A lower classification threshold leads to a lower precision and a higher recall.
- Many features can help to lower the bias.
- Many training examples can help to lower the variance.
- If a human expert can predict y based on x , more training data can help.