1 Machine Learning Formulae

Note: Lecture notes and slides available here:
`https://vkosuri.github.io/CourseraMachineLearning/`

1.1 Notation

1.1.1 Linear and logistic regression

Feature:

$$x_j \tag{1}$$

Single data point in a feature:

$$x_j^{(i)} \tag{2}$$

Feature vector:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \tag{3}$$

Matrix of training examples, stored row-wise:

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_n \end{bmatrix} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ x_0^{(3)} & x_1^{(3)} & x_2^{(3)} & \cdots & x_n^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_0^{(m)} & x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \tag{4}$$

Theta:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \tag{5}$$

Feature scaling:

$$x_j := \frac{x_j}{s_j} \quad \text{where} \quad s_j = max(x_j) - min(x_j) \tag{6}$$

Mean normalisation:

$$x_j := x_j - \mu_j \quad \text{where} \quad \mu_j = \frac{\sum_{i=1}^{m} x_j^{(i)}}{m} \tag{7}$$

Feature normalisation (feature scaling *and* mean normalisation):

$$x_j := \frac{x_j - \mu_j}{s_j} \tag{8}$$

1.1.2 Neural networks

General structure:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \\ a_2^{(1)} \\ \vdots \\ a_n^{(1)} \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \\ \vdots \\ a_n^{(2)} \end{bmatrix} \tag{9}$$

1.2 Linear regression

1.2.1 Basic equations

Hypothesis function:

$$h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{10}$$

Cost function:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta^{(i)} - y^{(i)})^2 \tag{11}$$

$$J(\theta) = \frac{1}{2m} (X\theta - y)^T (X\theta - y) \tag{12}$$

Gradient descent:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \tag{13}$$

$$\theta := \theta - \alpha \nabla J(\theta) \tag{14}$$

Gradient descent (update rule):

$$\begin{aligned} &\mathit{repeat\ until\ convergence} : \{ \\ &\qquad \theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j \\ &\} \end{aligned} \tag{15}$$

$$\begin{aligned} &\mathit{repeat\ until\ convergence} : \{ \\ &\qquad \theta := \theta - \frac{\alpha}{m} X^T (X\theta - y) \\ &\} \end{aligned} \tag{16}$$

Normal equation:

$$\theta = (X^T X)^{-1} X^T y \tag{17}$$

1.2.2 Polynomial regression

Hypothesis function:

$$h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{18}$$

But where some of the higher-order features are more complex functions of lower order features e.g.:

$$x_3 = x_1 \sqrt{x_2} \tag{19}$$

1.2.3 Equations with regularisation

Cost function:

$$J(\theta) = \frac{1}{2m}[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2] \tag{20}$$

$$J(\theta) = \frac{1}{2m}[(X\theta - y)^T(X\theta - y) + \lambda\theta^T\theta] \tag{21}$$

Gradient descent (update rule):

$$\begin{aligned}
&\textit{repeat until convergence}: \{ \\
&\qquad \theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0 \\
&\qquad \theta_j := \theta_j - \alpha[(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j) + \frac{\lambda}{m}\theta_j] \\
&\}
\end{aligned} \tag{22}$$

Note: By convention, the θ_0,x_0 update term is not regularised.

The update rule can also be re-arranged to give the original update rule, plus an additional term at the front:

$$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j \tag{23}$$

Normal equation:

$$\theta = (X^TX + \lambda \cdot L)^{-1}X^Ty \tag{24}$$

Where:

$$L = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{25}$$

1.3 Logistic regression

1.3.1 Basic equations

Hypothesis function:

$$h_\theta(x) = g(\theta^T x) \tag{26}$$

$$h_\theta(x) = g(X\theta) \tag{27}$$

Where:

$$g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \tag{28}$$

Hypothesis function interpretation:

$$h_\theta(x) = P(y = 1|x;\theta) = 1 - P(y = 0|x;\theta) \tag{29}$$

Decision boundary:

$$h_\theta(x) \geq 0.5 \rightarrow y = 1 \tag{30}$$

$$h_\theta(x) < 0.5 \rightarrow y = 0 \tag{31}$$

Decision boundary (2):

$$\theta^T x \geq 0 \rightarrow y = 1 \tag{32}$$

$$\theta^T x < 0 \rightarrow y = 0 \tag{33}$$

Cost function:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} log(h_\theta(x^{(i)})) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)}))] \tag{34}$$

$$J(\theta) = \frac{1}{m}[-y^T log(h) - (1 - y)^T log(1 - h)] \tag{35}$$

Gradient descent (update rule):

$$\begin{aligned} & repeat\ until\ convergence : \{ \\ & \qquad \theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j \\ & \} \end{aligned} \tag{36}$$

$$
\begin{aligned}
&repeat\ until\ convergence: \{ \\
&\qquad \theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - y) \\
&\}
\end{aligned}
\tag{37}
$$

Multi-class classification:

$$
\begin{aligned}
& y \in \{0, 1, 2, ..., n\} \\
& h_\theta^{(0)}(x) = P(y = 0|x;\theta) \\
& h_\theta^{(1)}(x) = P(y = 1|x;\theta) \\
& h_\theta^{(2)}(x) = P(y = 2|x;\theta) \\
& \vdots \\
& h_\theta^{(n)}(x) = P(y = n|x;\theta) \\
& prediction = max(h_\theta^{(i)}(x))
\end{aligned}
$$

1.3.2 Equations with regularisation

Cost function:

$$
J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(h_\theta(x^{(i)})) + (1-y^{(i)})log(1-h_\theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{38}
$$

$$
J(\theta) = \frac{1}{m}[-y^T log(h) - (1-y)^T log(1-h)] + \frac{\lambda}{2m}\theta^T\theta \tag{39}
$$

Gradient descent (update rule):

$$
\begin{aligned}
&repeat\ until\ convergence: \{ \\
&\qquad \theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0 \\
&\qquad \theta_j := \theta_j - \alpha[(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j) + \frac{\lambda}{m}\theta_j] \\
&\}
\end{aligned}
\tag{40}
$$

Note: This is the same as the update rule for linear regression. By convention, the θ_0,x_0 update term is not regularised.

As with linear regression, the update rule can also be re-arranged to give the original update rule, plus an additional term at the front:

$$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j \tag{41}$$

1.4 Neural networks

1.4.1 Notation

L = number of layers
l = specific layer number
s_l = number of units/nodes (not including bias unit) in layer l
$a_j^{(l)}$ = unit/node j in layer l
$a_j^{(t)(l)}$ = training example in unit/node j in layer l
Θ^l = matrix of weights for moving between layer l and $l+1$
$\Theta_{i,j}^l$ = row i, column j in matrix of weights l
$a^{(1)} = X$ = input layer
$a^{(L)} = h_\Theta(x)$ = output layer
K = number of classes in the output layer = number of classes in y (including 0)

1.4.2 Matrix dimensions

The input layer $a^{(1)}$ is the same as the matrix of training examples X:

$$a^{(1)} = X = \begin{bmatrix} & \leftarrow \quad n+1 \quad \rightarrow & \\ \uparrow & & \\ m & & \\ \downarrow & & \\ & & \end{bmatrix} = \begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_n \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \tag{42}$$

Where:

$$x_0 = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \tag{43}$$

The hidden layers have $(s_l + 1)$ columns (the '+1' is the bias unit), each with m training examples:

$$a^{(l)} = \begin{bmatrix} & \leftarrow \quad (s_l + 1) \quad \rightarrow & \\ \uparrow & & \\ m & & \\ \downarrow & & \\ & & \end{bmatrix} = \begin{bmatrix} a_0^{(l)} & a_1^{(l)} & a_2^{(l)} & \cdots & a_n^{(l)} \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \tag{44}$$

Where:

$$a_0^{(l)} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \tag{45}$$

The output layer $a^{(L)}$ is the model hypothesis $h_\Theta(x)$.

For a single class neural network, this is a vector whose values correspond to the hypothesis value for each entry in the training data:

$$a^{(L)} = h_\Theta(x) = \begin{bmatrix} h_{(\Theta)}(x^{(1)}) \\ h_{(\Theta)}(x^{(2)}) \\ h_{(\Theta)}(x^{(3)}) \\ \vdots \\ h_{(\Theta)}(x^{(m)}) \end{bmatrix} \tag{46}$$

For a multiclass neural network, this is a matrix of values whose columns correspond to the different classes, and whose rows correspond to each entry in the training data:

$$a^{(L)} = h_\Theta(x) = \begin{bmatrix} h_\Theta(x)_1 \\ h_\Theta(x)_2 \\ h_\Theta(x)_3 \\ \vdots \\ h_\Theta(x)_K \end{bmatrix}$$

$$= \begin{bmatrix} & \leftarrow & K & \rightarrow & \\ \uparrow & & & & \\ m & & & & \\ \downarrow & & & & \\ & & & & \end{bmatrix} = \begin{bmatrix} h_\Theta(x^{(t)})_1 & h_\Theta(x^{(t)})_2 & h_\Theta(x^{(t)})_3 & \cdots & h_\Theta(x^{(t)})_K \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \tag{47}$$

Remember, the hypothesis values correspond to the probability of training example t being in class K:

$$h_\Theta(x^{(t)})_k = P(x^{(t)} = k|x; \Theta) \tag{48}$$

The matrix of weights $\Theta_{i,j}^l$ has $s_l + 1$ rows (the $+1$ from the bias unit) and

(s_{l+1}) columns:

$$\Theta^{l}_{i,j} = \begin{bmatrix} & \leftarrow \;\; s_l + 1 \;\; \rightarrow & \\ \uparrow & & \\ s_{l+1} & & \\ \downarrow & & \\ & & \end{bmatrix} \tag{49}$$

The data classifications in y are represented as a binary $m \times k$ matrix, with each row corresponding to an entry in the training data, and each column corresponding to a different class.

$$y = \begin{bmatrix} y_0 & y_1 & y_2 & \cdots & y_K \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} = \begin{bmatrix} & \leftarrow \;\; K \;\; \rightarrow & \\ \uparrow & & \\ m & & \\ \downarrow & & \\ & & \end{bmatrix} \tag{50}$$

And the mapping is as follows:

$$y = \begin{bmatrix} 1 \\ 5 \\ 2 \\ 8 \\ 9 \\ 0 \\ 5 \\ \vdots \\ 9 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{51}$$

1.4.3 Basic representation

Single class neural network:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \\ \cdots \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(3)} \\ a_1^{(3)} \\ a_2^{(3)} \\ \cdots \end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix} a_0^{(j)} \\ a_1^{(j)} \\ a_2^{(j)} \\ \cdots \end{bmatrix} \rightarrow \cdots \rightarrow h_{\Theta}(x) \tag{52}$$

Multiclass neural network:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \\ \cdots \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(3)} \\ a_1^{(3)} \\ a_2^{(3)} \\ \cdots \end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix} a_0^{(j)} \\ a_1^{(j)} \\ a_2^{(j)} \\ \cdots \end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix} h_\Theta(x)_1 \\ h_\Theta(x)_2 \\ h_\Theta(x)_3 \\ \vdots \\ h_\Theta(x)_k \end{bmatrix} \tag{53}$$

1.4.4 Forward propagation

The role of feed forward propagation is to calculate the hypothesis $h_\Theta(x)$ and - if tracking - the overall cost $J(\Theta)$. The values of $h_\Theta(x)$ feed directly into the back propagation algorithm.

Note: Feed forward and back propagation are carried out element-wise through the training data, typically using a for-loop:

$$\begin{aligned} &\text{for } i \text{ in range(m):} \\ &\cdots \\ &\cdots \end{aligned} \tag{54}$$

For layer 1:

$$a^{(1)} = X \tag{55}$$

For layer 2:

$$\begin{aligned} z^{(2)} &= \Theta^{(1)T} X \\ a^{(2)} &= g(z^{(2)}) \\ &\text{[Then add bias unit as column of ones]} \end{aligned} \tag{56}$$

For layer $l+1$:

$$\begin{aligned} z^{(l+1)} &= \Theta^{(l)T} a^{(l)} \\ a^{(l+1)} &= g(z^{(l+1)}) \\ &\text{[Then add bias unit as column of ones]} \end{aligned} \tag{57}$$

For output layer:

$$\begin{aligned} z^{(L)} &= \Theta^{(L-1)T} a^{(L-1)} \\ a^{(L)} &= g(z^{(L)}) = h_\Theta(x) \end{aligned} \tag{58}$$

The cost function (unregularised)is then defined as:

$$J_\Theta(x) = \frac{1}{m}\sum_{t=1}^{m}\sum_{k=1}^{K}[y_k^{(i)} log(h_\Theta(x^{(i)})_k) + (1-y_k^{(i)})log(1-h_\Theta(x^{(i)})_k)] \tag{59}$$

And regularised is:

$$J_\Theta(x) = \frac{1}{m}\sum_{t=1}^{m}\sum_{k=1}^{K}[y_k^{(i)} log(h_\Theta(x^{(i)})_k) + (1-y_k^{(i)})log(1-h_\Theta(x^{(i)})_k)] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(\Theta_{j,i}^{(l)})^2 \tag{60}$$

1.4.5 Backpropagation

The aim of back propagation is to calculate the gradient $\frac{\partial}{\partial \Theta_{i,j}^{(l)}} J(\Theta)$, which can then be used in an update rule such as gradient descent.

Back propagation is carried out element-wise through the training data, typically using a for-loop:

$$\begin{aligned} &\text{for } t \text{ in range(m):} \\ &\cdots \\ &\cdots \end{aligned} \tag{61}$$

Calculating δ

We calculate δ for layers L,$(L-1)$,$(L-2)$,$\cdots$,2. Note that we do not calculate δ^1 i.e. δ for the input layer.

For the output later:

$$\delta^{(L)} = a^{(L)} - y \tag{62}$$

For the hidden layers:

$$\delta^{(l)} = ((\Theta^{(l)})^T \delta^{(l+1)}). * g'(z^{(l)}) \tag{63}$$

Where:

$$g'(z^{(l)}) = g(z^{(l)}). * (1 - g(z^{(l)})) = a^{(l)}. * (1 - a^{(l)}) \tag{64}$$

And, as before (although written a little differently):

$$z^{(l)} = (\Theta^{(l-1)})^T a^{(l-1)} \tag{65}$$

And $.*$ is the element-wise multiplication.

Note that δ^l is a 2-D vector with the same number of units as layer l:

$$[\delta^l] = \begin{bmatrix} \\ \uparrow \\ s_l + 1 \\ \downarrow \\ \\ \end{bmatrix} \tag{66}$$

Element-wise, this is implemented as:

$for\ t\ in\ range(m):$

$$\begin{gathered}(\delta^{(L)})^{(t)} = (a^{(L)})^{(t)} - y^{(t)} \\ (\delta^{(l)})^{(t)} = ((\Theta^{(l)})^T \delta^{(l+1)}).*(a^{(l)})^{(t)}.*(1 - (a^{(l)})^{(t)})\end{gathered} \tag{67}$$

Where:

$$[(\delta^{(l)})^{(t)} = ((\Theta^{(l)})^T \delta^{(l+1)})] = \tag{68}$$

Calculating Δ

Note that $\Delta^{(l)}$ has the same dimensions as $\Theta^{(l)}$:

$$[\delta^{(l)}] = [\Theta^{(l)}] = \begin{bmatrix} & \leftarrow \quad s_l + 1 \quad \rightarrow \\ \uparrow & \\ s_{l+1} & \\ \downarrow & \\ & \end{bmatrix} \tag{69}$$