Part I.
Reviewed notation

1. Multivariable Regression

- $y : [m \times 1]$ - Output variable (dependent variable)
- $X : [m \times (n+1)]$ - Input variables (independent variables)
- $\theta : [(n+1) \times 1]$ - parameters
- m : number of observations
- n : number of parameters
- $h_\theta = X\theta =$ regression output, should be as close as possible to y

1.1. Normal resolution

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

1.2. Gradient Descent

Gradient descent (some sort of Newton), used as a numerical method to iteratively find the optimum of a function (J in this case)

$$\begin{aligned} \theta &:= \theta - \alpha \Delta J(\theta) && (2) \\ &:= \theta - \alpha \frac{1}{m} X^T (X\theta - y) && (3) \end{aligned}$$

2. Multivariable regression with Regularization

To automatically choose optimal parameters and avoid from overfitting, an additional feature is introduced λ.

The Normal Method thus becomes

$$\theta = \left(\theta^T \theta + \lambda \begin{pmatrix} 0 & 1 & \dots & 1 \\ \vdots & 1 & \dots & \vdots \\ 0 & 1 & \dots & 1 \end{pmatrix} \right)^{-1} X^T y \tag{4}$$

and the gradient descent algorithm

$$\theta := \theta - \alpha \frac{1}{m} X^T (X\theta - y) + \alpha \frac{\lambda}{m} \theta \tag{5}$$

3. Logistic Regression

3.1. Variables for one output

- $y \in [0, 1]^{m \times 1}$
- $h_\theta = \mathbb{P}(y = 1 | X; \theta) = g(X\theta)$

3.2. Sigmoid

Map $z \in \mathbb{R}$ to $g(z) \in [0,1]$

$$g(z) = \frac{1}{1+e^{-z}} \tag{6}$$

3.3. Gradient Descent

$$\theta \;:=\; \theta - \alpha \frac{1}{m} X^T (g(X\theta) - y) \tag{7}$$

3.4. Variables for p outputs - multi-classifier

- $y \in [0,1]^{m \times p}$
- $\theta : [(n+1) \times p]$
- $X : [m \times (n+1)]$
- $h_\theta = \begin{pmatrix} h_{\theta_1} \\ \vdots \\ h_{\theta_p} \end{pmatrix}$

4. Neural Networks

4.1. Forward Propagation

- m : number of observations
- $y : [m \times s_L]$
- $\theta^{(l)} : [(s_l + 1) \times s_{(l+1)}]$

Initialization ($l = 1, s_1 = n$):

$$\begin{aligned}
a^{(1)} &= x && (8)\\
[m \times s_1] &= [m \times n] && (9)\\
\tilde{a}^{(1)} &= \tilde{x} && (10)\\
[m \times (s_1 + 1)] &= [m \times (n+1)]
\end{aligned}$$

$l < L$:

$$\begin{aligned}
\tilde{a}^{(l+1)} &= g(a^{(l)}\theta^{(l)}) = g(z^{(l+1)}) && (11)\\
[m \times (s_l + 1)] &= [m \times (s_l + 1)][(s_l + 1) \times (s_{l+1})] && (12)
\end{aligned}$$

$$a^{(l)} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} : + \tilde{a}^{(l)} \tag{13}$$

$\rightarrow$ add a column of ones at the beginning of the matrix

4.2. Backward Propagation (in order to find ΔJ)

Initialization $l = L$:

$$\begin{aligned} \delta^{(L)} &= a^{(L)} - y^{(L)} \qquad (14) \\ [m \times s_L] &= [m \times s_L] - [m \times s_L] \end{aligned}$$

$l < L$

$$\begin{aligned} \delta^{(l)} &= \tilde{\delta}^{(l+1)} \theta^{T(l)} \circ g'(z^{(l)}) \qquad (15) \\ &= \tilde{\delta}^{(l+1)} \theta^{T(l)} \circ a^{(l)} \circ (1 - a^{(l)}) \qquad (16) \\ [m \times (s_l + 1)] &= [m \times s_{l+1}] - [s_{l+1} \times (s_l + 1)] \end{aligned}$$

4.3. Compute/train a Neural Network

A Neural network is defined by the following dimensions:

- m: number of observations
- s_l: number of node for layer l
- n: number of incoming variables $n = s_1$ ($n + 1$ for vector of ones)
- p: number of output variables ($p = s_L$)
- L: Number of nodes

1. $a^{(1)} = x$
2. forward: get $a = (l)$
3. backward: get $\delta^{(l)}$
4. $\Delta^{(l)} = a^{T(l)} \tilde{\delta}^{(l+1)}$
5. $\nabla^{(l)} J(\theta) = D^{(l)} = \frac{1}{m} \Delta^{(l)}$
6. Optimization to find $\hat{\theta}^{(l)}$: $\min_\theta J$

5. Support Vector Machine (SVM)

Also referred to as Large Margin Classifiers.

$$J(\theta) = \min_\theta C \sum_{i=1}^{m} \left[y^{(i)} \text{cost}_1(\theta^T x^i) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2. \qquad (17)$$

$$C = \frac{1}{\lambda} \qquad (18)$$

Part II.
First Version - as in Coursera Course

Cost Function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}(h_\theta(x^{(i)} - y^{(i)}))^2 \tag{19}$$

Hypothesis

$$h_\theta(\mathbf{x}) = \theta^T\mathbf{x} \tag{20}$$

with $\mathbf{x} = (1, x_1, ..., x_n)$ and $\theta = (\theta_0, \theta_1, ..., \theta_n)$

Gradient Descent Minimize $J(\theta)$ for θ, the algorithm is

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta) \tag{21}$$

where α is the *learning rate*.
and

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \tag{22}$$

alternatives to gradient descent

- Conjugate gradient
- BFGS
- L-BFGS

Analytical Solution

$$\theta = (X^TX)^{-1}X^Ty \tag{23}$$

6. Logistic Regression

classification: $y = 0$ or $y = 1$
Want : $0 \leq h_\theta(x) \leq 1$

$$h_\theta(x) = g(\theta^Tx) \tag{24}$$

with logistic (sigmoid) function $z \in \mathbb{R}$:

$$g(z) = \frac{1}{1+e^{-z}} \in (0, 1) \tag{25}$$

output can be read as $h_\theta(x) = \mathbb{P}(y = 1|x;\theta)$: probabability that $y = 1$ given x parametrized by θ is h.

6.1. Decision Boundary

$$y = \begin{cases} 1 & z > 0 \\ 0 & z < 0 \end{cases} \tag{26}$$

where z is the argumen of g and is usally of the form $\theta^T x$

$$\text{Cost}(h_\theta(x, y)) = \frac{1}{2}(h_\theta(x) - y)^2 \tag{27}$$

6.2. Cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) \tag{28}$$

$$\text{Cost}(h_\theta(x, y)) = \begin{cases} -\log\left(h_\theta(x)\right) & y = 1 \\ -\log\left(1 - h_\theta(x)\right) & y = 0 \end{cases} \tag{29}$$

since $y = 1$ or $y = 0$:

$$\text{Cost}(h_\theta(x), y) = -y\log\left(h_\theta(x)\right) - (1-y)\log\left(1 - h_\theta(x)\right) \tag{30}$$

allows to have

y	$h_\theta(\mathbf{x})$	$\text{Cost}(h, y)$
0	0	0
1	0	∞
0	1	∞
1	1	0

7. Regularization

7.1. Problem of overfitting

If too many features (θ), the learned hypothesis may fit the training examples very well ($J(\theta) \approx 0$), but fail to generalize to new examples.

7.1.1. Addressing overfitting

- Reduce the number of features
  - Manually select features to be kept
  - Model selection algorithm
- Regularization

7.2. Cost function

add λ to reduce the number of allowed parameters.

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^i) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right] \tag{31}$$

If λ is too large, **underfit** occurs.

7.3. Regularized Logistic regression

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\log\left(h_\Theta(x^{(i)})\right) + (1-y^{(i)})\log\left(1-h_\Theta(x^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{m}\Theta_j^2. \qquad (32)$$

The partial derivative are

$$\frac{\partial J(\theta)}{\partial\theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \qquad \text{for } j=0 \qquad (33)$$

$$\frac{\partial J(\theta)}{\partial\theta_j} = \left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j \qquad \text{for } j\geq 1. \qquad (34)$$

8. Neural Networks: Representation

8.1. Non-linear hypotheses

8.2. Neurons and the brain

8.3. Model representation I

Neuron model: Logistic unit
$\mathbf{x} = (1, x_1, ..., x_n)$ and $\theta = (\theta_0, \theta_1, ...\theta_n)$

8.4. Feedforward

- Input layer:

$$a^{(1)} = x \qquad (35)$$

- Hidden layer(s)

$$\begin{aligned} z^2 &= \Theta^{(1)}a^{(1)} & (36)\\ a^{(2)} &= g(z^{(2)}) & (37)\\ &\ldots & (38)\\ z^{(k+1)} &= \Theta^{(k)}a^{(k)} & (39)\\ a^{(k+1)} &= g(z^{(k+1)}) & (40) \end{aligned}$$

- Output layer

$$\begin{aligned} z^{(3)} &= \Theta^{(2)}a^{(2)} & (41)\\ h_\Theta = a^{(3)} &= g(z^{(3)}) & (42) \end{aligned}$$

- m total number of sample: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(m)}, y^{(m)})\}$
- L total number of layers indexed by l
- s_l number of unit per layer (without counting bias unit).
- $\Theta^{(l)}$ has size $s_{l+1}\times(s_l+1)$ (The $+1$ is here to account for the bias unit).
- Last layer L
  - Binary Classification: $s_L = 1$
  - Multi-class classification: $s_L = K$.

9. Neural networks: learning

9.1. Cost function

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\Theta(x^{(i)})\right)_k + (1-y^{(i)})\log\left(1-h_\Theta(x^{(i)})_k\right)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ij}^{(l)})^2. \tag{43}$$

9.2. Backpropagation

In order to $\min_\Theta J(\Theta)$, need to compute

- $J(\Theta) \Rightarrow$ by plugging values directly into function
- $\frac{\partial}{\partial\Theta_{ij}^{(l)}} J(\Theta) \Rightarrow$ with backpropagation algorithm

$$\begin{cases} \delta^{(L)} = a^{(L)} - y = h_\Theta - y & \text{for } l = L \\ \delta^{(l)} = (\Theta^{(l)})^T \delta^{(4)} \circ \frac{d}{dz} g(z^{(l+1)}) & \text{for } 1 < l < L \end{cases} \tag{44}$$

[1] The intuition being δ is the "error" at every node that we try to minimize and converges to the gradient.

Note on $\frac{d}{dz}g(z)$ for $g(z) = \frac{1}{1+e^{-z}}$

$$\frac{d}{dz}g(z) = \frac{e^{-z}}{1+e^{-z}} = g(z)(1-g(z)) \tag{46}$$

Backpropagation Algorithm

- For $i = 1$ to m
  - Set $a(1) = x^{(i)}$
  - Perform forward propagation to compute $a^{(l)}$ for $l = 2, .., L$
  - Using $y^{(i)}$ compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
  - Compute $\delta^{(L-1)}, \delta^{L-2}, ..., \delta^{(2)}$
  - $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$ or $\Delta^{(l)} := \Delta^{(l)} + \delta(l+1)\left(a(l)\right)^T$
- $D_{ij}^{(l)} := \frac{1}{m}\Delta_{ij}^{(l)} + \lambda\Theta_{ij}^{(l)}$ if $j \neq 0$
- $D_{ij}^{(l)} := \frac{1}{m}\Delta_{ij}^{(l)}$ if $j = 0$

[1] $\circ$ is the Hadamard product and produces a matrix where all elements ij are a multiplication of the element ij of the two input matrices:

$$\begin{pmatrix} a_{11} & ... & a_{1m} \\ a_{21} & ... & \vdots \\ \vdots & ... & \vdots \\ a_{n1} & ... & a_{nm} \end{pmatrix} \circ \begin{pmatrix} b_{11} & ... & b_{1m} \\ b_{21} & ... & \vdots \\ \vdots & ... & \vdots \\ b_{n1} & ... & b_{nm} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & ... & a_{1m}b_{1m} \\ a_{21}b_{21} & ... & \vdots \\ \vdots & ... & \vdots \\ a_{n1}b_{n1} & ... & a_{nm}b_{nm} \end{pmatrix} \tag{45}$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)} \tag{47}$$

10. Support Vector Machine (SVM)

$$J(\theta) = \min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} \mathrm{cost}_1(\theta^T x^i) + (1 - y^{(i)}) \mathrm{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2. \tag{48}$$

The SVM is a *large margin classifier*