CSC2515 HW 1

1. **Logistic Regression with Regularization**

(a) In the case of regularized logistic regression the likelihood function is :

$$L(\mathbf{w}) = p(\mathbf{w}) \prod_{i=1}^{N} p(t^{(i)}|\mathbf{x}^{(i)}; \mathbf{w}) = p(\mathbf{w}) \prod_{i=1}^{N} p(t=1|\mathbf{x}^{(i)})^{t^{(i)}} \cdot p(t=0|\mathbf{x}^{(i)})^{1-t^{(i)}}$$

the loss function is $\ell(\mathbf{w}) = -\ln L(\mathbf{w})$ also $\mathbf{w} \sim N(\mathbf{w}|0, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{D/2} \exp\left(-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}\right)$ hence

$$\ell(\mathbf{w}) = \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} t^{(i)} \ln p(t=1|\mathbf{x}^{(i)}) - \sum_{i=1}^{N}(1-t^{(i)}) \ln p(t=0|\mathbf{x}^{(i)}) + C(\alpha)$$

where $C(\alpha) = -\frac{D}{2}\ln\left(\frac{\alpha}{2\pi}\right)$ now set $z^{(i)} = \mathbf{w}^T\mathbf{x}^{(i)} + w_0$ so that :

$$p(t=0|\mathbf{x}^{(i)}, \mathbf{w}) = \frac{e^{-z^{(i)}}}{1+e^{-z^{(i)}}}, \quad p(t=1|\mathbf{x}^{(i)}, \mathbf{w}) = \frac{1}{1+e^{-z^{(i)}}}$$

now sub these into $\ell(\mathbf{w})$ and simplify :

$$\begin{aligned}\ell(\mathbf{w}) &= \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N} t^{(i)} \ln\left(1+e^{-z^{(i)}}\right) + \sum_{i=1}^{N}(1-t^{(i)})z^{(i)} + \sum_{i=1}^{N}(1-t^{(i)}) \ln\left(1+e^{-z^{(i)}}\right) + C(\alpha) \\ &= \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N}\left(\ln\left(1+e^{-z^{(i)}}\right) + (1-t^{(i)})z^{(i)}\right) + C(\alpha)\end{aligned}$$

(b) the derivatives are :

$$\begin{aligned}\frac{\partial \ell(\mathbf{w})}{\partial w_j} &= \alpha w_j + \sum_{i=1}^{N} \frac{-x_j^{(i)} e^{-z^{(i)}}}{1+e^{-z^{(i)}}} + \sum_{i=1}^{N}(1-t^{(i)})x_j^{(i)} = \alpha w_j + \sum_{i=1}^{N} x_j^{(i)}\left(p(t=1|\mathbf{x}^{(i)}, \mathbf{w}) - t^{(i)}\right) \\ \frac{\partial \ell(\mathbf{w})}{\partial w_0} &= \sum_{i=1}^{N} t^{(i)}\left(\frac{-e^{-z^{(i)}}}{1+e^{-z^{(i)}}} + (1-t^{(i)})\right) = \sum_{i=1}^{N}\left(p(t=1|\mathbf{x}^{(i)}, \mathbf{w}) - t^{(i)}\right)\end{aligned}$$

(c) The pseudo code for gradient descent with a size step is λ

initialize $\mathbf{w}^0 = (w_1, \ldots, w_D, w_0)$;
while $\mathbf{w}^n$ *does not min* $\ell(\mathbf{w})$ **do**
  $\mathbf{w}^{n+1} \leftarrow \mathbf{w}^n - \lambda \nabla \ell(\mathbf{w}^n)$;
end

where $\nabla \ell(\mathbf{w}^n) = \left(\frac{\partial \ell(\mathbf{w}^n)}{\partial w_1}, \ldots, \frac{\partial \ell(\mathbf{w}^n)}{\partial w_D}, \frac{\partial \ell(\mathbf{w}^n)}{\partial w_0}\right)^T$

2. **Decision Trees**

Set : Is Pasta Overcooked? $\rightarrow O = \{yes, no\}$, Wait time? $\rightarrow W = \{long, short\} = \{l, s\}$, Rude Waiter? $\rightarrow R = \{yes, no\}$ and Satisfaction$\rightarrow$ $S = \{yes, no\}$, We want to calculate : $IG(S|W) = H(S) - H(S|W), IG(S|O) = H(S) - H(S|O), IG(S|R) = H(S) - H(S|R)$ so first $H(S) = -\sum_S p(S)\log_2 p(S) = -p(S = yes)\log_2 p(S = yes) - p(S = no) = \log_2 p(S = no) - \frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} \approx 0.971$

Next using $H(Y|X) = \sum_{x\in X} p(x)H(Y|X = x)$ we see that

$$\begin{aligned}
H(S|R) &= p(R = no)H(S|R = no) + p(R = yes)H(S|R = yes)\\
&= (1/5))H(S|R = no) + (4/5))H(S|R = yes)\\
H(S|O) &= p(O = no)H(S|O = no) + p(O = yes)H(S|O = yes)\\
&= (2/5))H(S|O = no) + (3/5))H(S|O = yes)\\
H(S|W) &= p(W = s)H(S|W = s) + p(W = l)H(S|W = l)\\
&= (2/5))H(S|W = s) + (3/5))H(S|W = l)
\end{aligned}$$

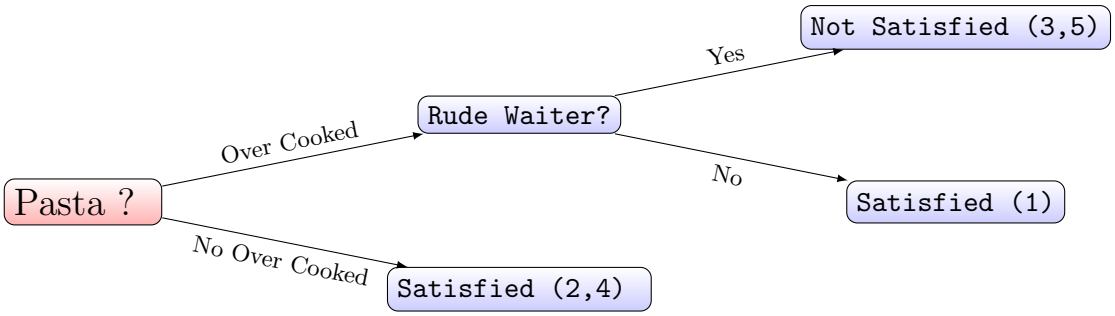
Next using $H(Y|X = x) = -\sum_{y\in Y} p(y|x)\log_2 p(y|x)$ we see that

$$\begin{aligned}
H(S|R = yes) &= -p(S = yes|R = yes)\log_2 p(S = yes|R = yes) - p(S = no|R = yes)\log_2 p(S = no|R = yes)\\
&= -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1\\
H(S|R = no) &= -p(S = yes|R = no)\log_2 p(S = yes|R = no) - p(S = No|R = no)\log_2 p(S = no|R = no) = 0\\
H(S|O = yes) &= -p(S = yes|O = yes)\log_2 p(S = yes|O = yes) - p(S = no|O = yes)\log_2 p(S = no|O = yes)\\
&= -(2/3)\log_2(2/3) - (1/3)\log_2(1/3) \approx 0.92\\
H(S|O = no) &= -p(S = yes|O = no)\log_2 p(S = yes|O = no) - p(S = no|O = no)\log_2 p(S = no|O = no) = 0\\
H(S|W = l) &= -p(S = yes|W = l)\log_2 p(S = yes|W = l) - p(S = no|W = l)\log_2 p(S = no|W = l)\\
&= -(2/3)\log_2(2/3) - (1/3)\log_2(1/3) \approx 0.92\\
H(S|W = s) &= -p(S = yes|W = s)\log_2 p(S = yes|W = s) - p(S = no|W = s)\log_2 p(S = no|W = s)\\
&= -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1
\end{aligned}$$

Hence we can see $H(S|W) = 0.952, \quad H(S|O) = 0.5513, \quad H(S|R) = 0.8 \Longrightarrow$

$$\boxed{IG(S|W) = 0.02, \quad IG(S|R) = 0.17, \quad IG(S|O) = 0.42}$$

thus I have constructed the decision tree:



Using this tree one can easily see that :

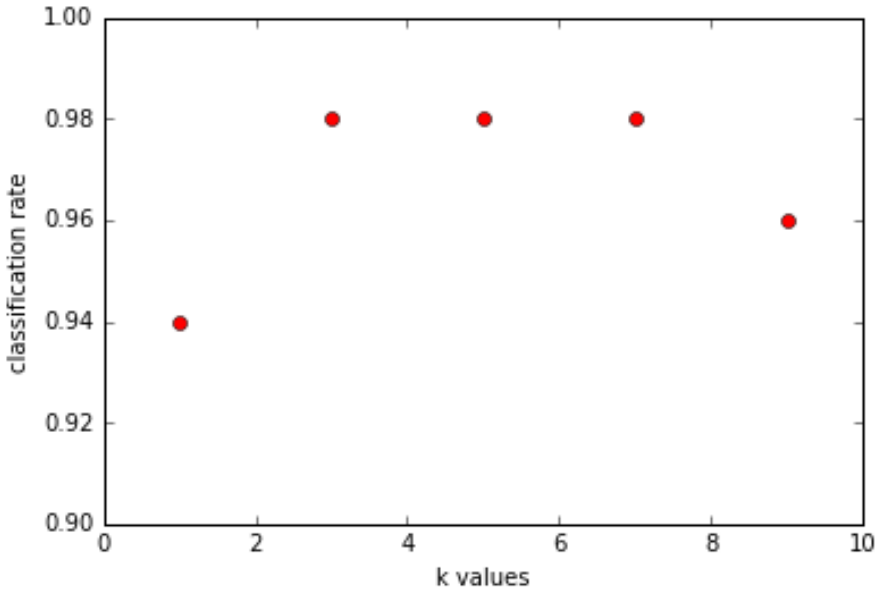
Person 6 : doesn't have Over Cooked pasta so is **Satisfied**.

Person 7 : has Overcooked Pasta and a Rude Waiter so is **Not Satisfied**.

Person 8 : has Overcooked pasta but a waiter that isn't rude so they are **Satisfied**.

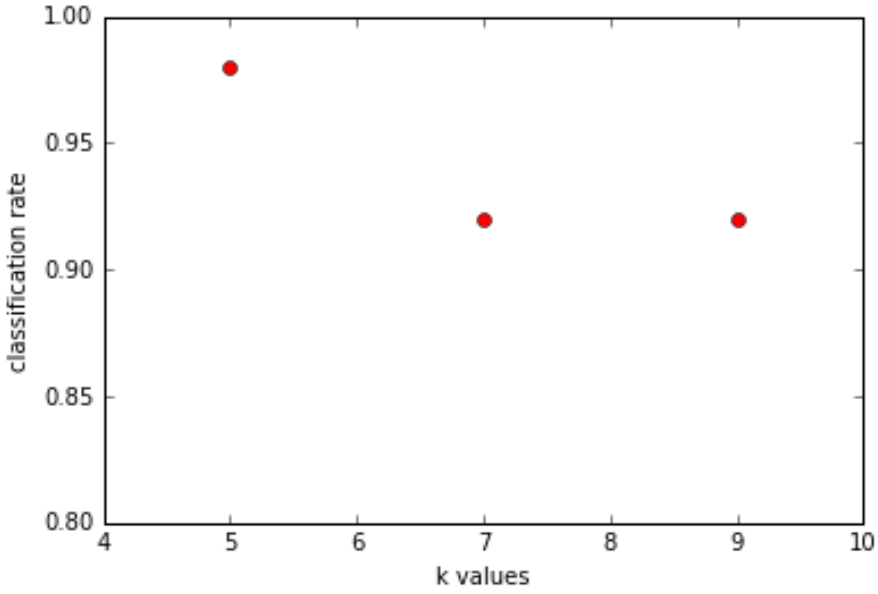
3. **k Nearest Neighbours**

Here is my plot of the classification rate vs k values



(a) The classifier performs well and generally it seems to converge to a max classification rate of 0.98 for $k = 3, 5, 7$. I would use $k^* = 7$ with a classification rate of 0.98 to maximize k and the classification rate; this tells us more about the vectors near to the data we are trying to classify as well as minimizes variance in the model. However it may cause under fitting problems due to high bias. However I don't believe this k suffers from this as its classification rate is pretty high however the classification rate for $k^* + 2 = 9$ drops off to 0.96 so at this value the kNN model may encounter bias-variance balance problems. Also for $k^* - 2 = 5$ the rate is 0.98 : also very good.

(b) The test performances for these values of k still have a reasonably high classification rate, but there are significant differences between the two performances. For example $k^* - 2 = 5$ performs better (with a rate of 0.98) than k^* which matches the rate for $k^* + 2 = 9$ at 0.92 ($< 0.96 =$ its rate for the validation set). This which means that the model does not generalize to the test data and there are bias variance balancing issues as the model with $k^* = 7$ is too biased and we are under fitting : A better choice for k^* would have been $k = 5$ which has a higher classification rate. See the following plot of the test performance for the relevant k values.



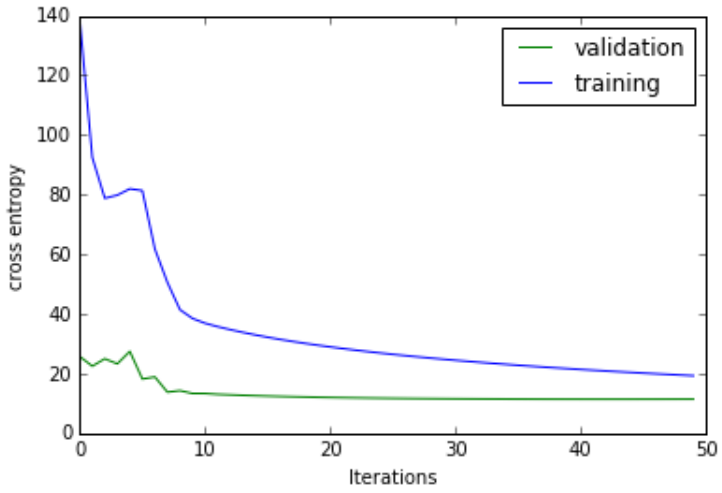
4. **Logistic Regression**

(a) I found that using a size step learning rate of $\lambda = 0.3$ and 50 iterations the outcomes :

```
TRAIN CE:19.298549  TRAIN FRAC:97.50
VALID CE:11.451350  VALID FRAC:92.00
TEST  CE:7.023357   TEST  FRAC:96.00
```

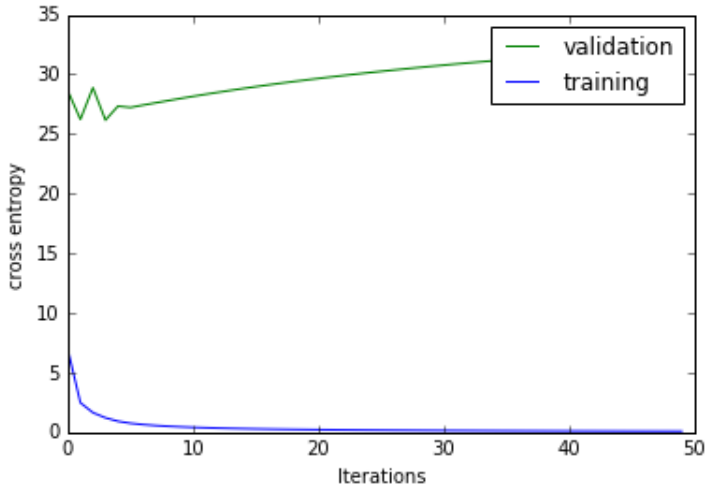
meaning we have error rate 2.5 , 8.0 , 4 percent for the training, validation and test data.

(b) Using the regular training set the relevant plot of cross entropy vs iterations is



In the training curve we see that there is a fairly rapid (10-20 iterations) descent while gradient descent begins to convergence to the weights that minimize the cross entropy then a more gradual convergence to its final min value. which thus also converges to its min value from a fairly large initial value. Similarly in the validation curve there is an even quicker, immediate convergence (< 10 iterations) to the weights that minimize cross entropy which also converges to its min value.

Using the smaller training set the relevant plot of cross entropy vs iterations is



In the training curve we see that there is an immediate rapid (< 5 iterations) convergence to the optimal weights that minimize the cross entropy (~ 0). This is expected since classifying a smaller amount of input data is simpler computationally. However because there is a small amount of data the model does not generalize well and the validation data seems to not converge well as it initializes at a smaller cross entropy value then gradually converges upwards to some value.

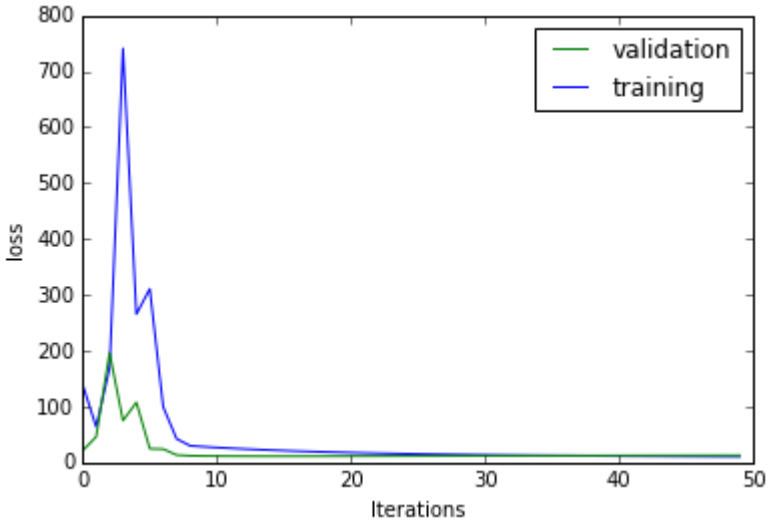
Running the code several times does not change the plots at all, as none of the values in them will not be computed differently in repeated trials.

5. **Regularized Logistic Regression**

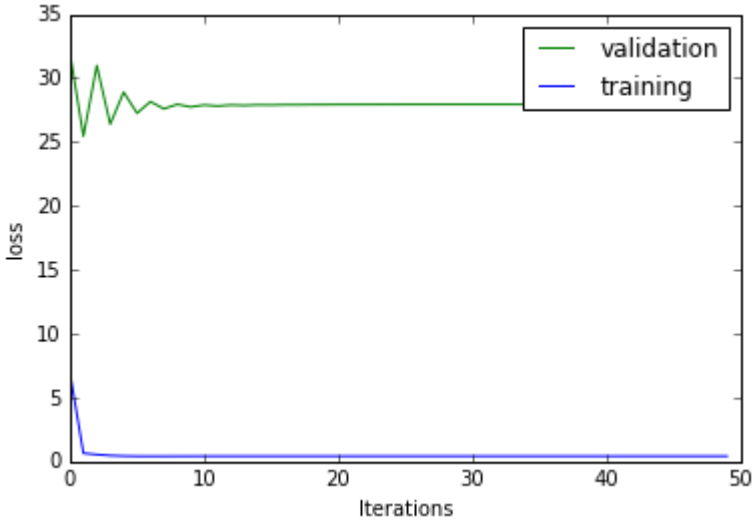
(a) I found using a learning rate of $\lambda = 0.75$, decay rate of $\alpha = 1.0$ and 21 iterations :

```
TRAIN CE:17.497546 TRAIN FRAC: 97.00
VALID CE:11.200411 VALID FRAC: 92.00
TEST  CE: 4.824239  TEST FRAC: 100.00
```

meaning we have error rate 3.0 , 8.0 , 0.0 percent for the training, validation and test data.



Using the smaller training set the relevant plot of cross entropy vs iterations is



(b) Here are subsequent losses and rates for various increases of alpha

```
ALPHA:  2.5  TRAIN CE:17.742711 TRAIN FRAC:97.00  VALID CE:11.042139  VALID FRAC:92.00
ALPHA:  5.0  TRAIN CE:18.514785 TRAIN FRAC:97.00  VALID CE:10.921169  VALID FRAC:90.00
ALPHA:  7.5  TRAIN CE:19.700053 TRAIN FRAC:97.00  VALID CE:10.961772  VALID FRAC:90.00
ALPHA: 10.0  TRAIN CE:21.236965 TRAIN FRAC:97.00  VALID CE:11.137368  VALID FRAC:92.00
```

Given other hyperprameters fixed, the loss and classification rate decrease then increase when alpha is increased. This occurs because when we increase alpha we are increasing the term that penalizes model complexity in the loss function thus the loss function will shift to a new smaller minimum value however if we penalize too much then the model becomes biased and we are under fitting so then the loss minimum will subsequently increase. Thus alpha should be 5 for the new minimum of the loss function.

(c) 3. Here are the relevant losses and rates *without regularization* :

```
TRAIN CE:17.430209 TRAIN FRAC:97.00
VALID CE:11.343668 VALID FRAC:92.00
TEST  CE:4.608680  TEST  FRAC:100.00
```

Clearly the performance of both models is very close. The non-regularized model has slightly smaller loss on the test set while the regularized model has a slightly smaller loss value on the validation set. Essentially this means that the model has a sufficient balance between bias and variance thus we do not need to penalize the model complexity or correct for over fitting by using regularization.

(d) By Comparing the classification rates for the three classifier we see that the logistic regression (regularized and not) classifier performs better on the test data ie 100% (see above) versus 92% for $k = 7$ and 98% for $k = 5$ in the kNN classifier.

In general, I would choose the classifier that had its strengths maximized and weaknesses minimized in the context of the specific classification problem and its input data. For the particular classifiers we list these pros and cons:

Logistic Regression :

- Pros : Can classify multiple classes using an intuitive probabilistic model for class predictions, models are quick to train and classify, can use regularization to correct over fitting
- Cons : has a linear decision boundary $\mathbf{w}^T\mathbf{x} + w_0$ which will poorly classify more complex classes in the input space

k Nearest Neighbours:

- Pros : Can classify complex decision boundaries and works well with lots of data
- Cons : Sensitive to noise from classes and scales of attributes. Also distances are meaningless in higher dimensions, Scales linearly with example number