CSC411:Machine Learning and Data Mining

Assignment #1

Chen Xu
999543837

Question 1. Logistic Regression

① training set: $D = \{(x^{(1)}, t^{(1)}), \dots, (x^{(N)}, t^{(N)})\}$

input: $x^{(i)} = (x_1^{(i)} \dots x_D^{(i)})$

$t^{(i)} \in \{0, 1\}$.

assume: $p(t=1 \mid x^{(i)}, w, w_0) = \sigma(w^T x^{(i)} + w_0) = \frac{1}{1+\exp(-\sum_{d=1}^{D} w_d x_d^{(i)} - w_0)}$
and data is i.i.d.

Gaussian Prior: $P(w) = \mathcal{N}(w \mid 0, \alpha^{-1} I)$

Multivariate Normal Distribution: $p(w) = (2\pi)^{-\frac{k}{2}} |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)}$

(where k is dimension of data set (features in this case)
$\mu = 0 \quad \Sigma = \alpha^{-1}$)

$$= \frac{1}{(2\pi)^{\frac{k}{2}} \alpha^{(-\frac{k}{2})}} \exp\left(-\frac{\alpha}{2} x^2\right) \quad x \text{ is weights in this case}$$

$$= \frac{1}{(2\pi)^{\frac{k}{2}} \alpha^{(-\frac{k}{2})}} \exp(W^T I W)^{-\frac{\alpha}{2}}$$

loss function: $L(w) = \prod_{i=1}^{N} p(t^{(i)} \mid x^{(i)}) \cdot P(w)$

$$= \prod_{i=1}^{N} P(t=1 \mid x^{(i)})^{\cdot t^{(i)}} \cdot P(t=0 \mid x^{(i)})^{(1-t^{(i)})} \cdot \frac{1}{(2\pi)^{\frac{k}{2}} \alpha^{(-\frac{k}{2})}} \exp(W^T I W)^{-\frac{\alpha}{2}}$$

$$-\log L(w) = -\sum_{i=1}^{N} t^{(i)} \log(P(t=1 \mid x^{(i)}) - \sum_{i=1}^{N} (1-t^{(i)}) \log(P(t=0 \mid x^{(i)}) - \sum_{i=1}^{N} (\log \exp(W^T I W)^{-\frac{\alpha}{2}} + \frac{k}{2}(\log(\frac{\alpha}{2\pi}))$$

where. $P(t=1 \mid x^{(i)}) = \frac{1}{1+\exp(-z^{(i)})} \quad P(t=0 \mid x^{(i)}) = \frac{\exp(-z^{(i)})}{1+\exp(-z^{(i)})}$

$$= -\sum_{i=1}^{N} t^{(i)} \log\left(\frac{1}{1+\exp(-z^{(i)})}\right) - \sum_{i=1}^{N} (1-t^{(i)}) \log\left(\frac{\exp(-z^{(i)})}{1+\exp(-z^{(i)})}\right) + \sum_{i=1}^{N} \left(\frac{\alpha}{2}(W^T I W) - \frac{k}{2} \log\frac{\alpha}{2\pi}\right)$$

$$= \sum_{i=1}^{N} t^{(i)} \log\left(\frac{1}{1+\exp(-z^{(i)})}\right) - \sum_{i=1}^{N} \left[-z - \log(1+\exp(-z^{(i)})) + t^{(i)} z^{(i)}\right] + t^{(i)} \log(1+\exp(-z^{(i)})) + \frac{\alpha}{2}(W^T I W) - \frac{k}{2} \log\frac{\alpha}{2\pi}$$

$$= \sum_{i=1}^{N} z^{(i)} + \log(1+\exp(-z^{(i)})) - t^{(i)} z^{(i)} + \frac{\alpha}{2}(W^T I W) - \frac{k}{2} \log\frac{\alpha}{2\pi}$$

$$\boxed{\text{Loss function} = \sum_{i=1}^{N} \log(1+\exp(-z^{(i)})) + z(1-t^{(i)}) + \frac{\alpha}{2}(W^T I W) - \frac{k}{2} \log\frac{\alpha}{2\pi}}$$

② Gradient:

$$\frac{\partial \text{loss}}{\partial w_j} = \sum_{i=1}^{N} \frac{\exp(-z^{(i)})}{1+\exp(-z^{(i)})}(-x^{(i)}) + x^{(i)}(1-t^{(i)}) \text{ to } w = \sum_{i=1}^{N} x^{(i)}\left(\frac{1}{1+\exp(-z^{(i)})} - t^{(i)}\right) + \alpha w_i$$

$$\boxed{\frac{\partial \text{loss}}{\partial w_j} = \sum_{i=1}^{N} \left[x_j^{(i)}\left(\frac{1}{1+\exp(-z^{(i)})} - t^{(i)}\right)\right] + \alpha w_j}$$

$$\frac{\partial \text{loss}}{\partial w_0} = \sum_{i=1}^{N} \frac{\exp(-z^{(i)})}{1+\exp(-z^{(i)})}(-1) + (1-t^{(i)}) \quad = \sum_{i=1}^{N} \left(\frac{1}{1+\exp(-z^{(i)})} - t^{(i)}\right)$$

$$\boxed{\frac{\partial \text{loss}}{\partial w_0} = \sum_{i=1}^{N} \left(\frac{1}{1+\exp(-z^{(i)})} - t^{(i)}\right)}$$

③ Pseudocode:

λ: learning rate n: number of iterations k: number of features

w^i: ith iteration w.

for i in range n:

$$w_0^{i+1} = w_0^i - \lambda \frac{\partial loss}{\partial w_0} \quad \text{where } \frac{\partial loss}{\partial w_0} = \sum_{i=1}^{N} \left(\frac{1}{1+\exp(-z^{(i)})} - t^{(i)} \right) \quad i \text{ here index \# of data}$$

for j in range k:

$$w_j^{i+1} = w_j^i - \lambda \frac{\partial loss}{\partial w_j} \quad \text{where } \frac{\partial loss}{\partial w_j} = \sum_{i=1}^{N} \left[\left(\frac{1}{1+\exp(-z^{(i)})} - t^{(i)} \right) x_j^{(i)} \right] + \alpha w_j$$

Question 2: Decision Tree:

①.

overcooked pasta.

$\frac{3}{5}$ Yes → $\frac{1}{3}$ Yes, $\frac{2}{3}$ No

$\frac{2}{5}$ No → 1 Yes, 0 No.

$$IG(Y|X) = H(Y) - H(Y|X) = H(Y) - \Sigma P(x) H(Y|X=x)$$

$$H(Y) = -\sum_{x \in X} P(x) \log_2 P(x)$$

$$= -\left(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}\right) = \boxed{0.97}$$

$$IG_{overcooked} = 0.97 - \frac{3}{5}\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right) - \frac{2}{5}(-\log_2 1 - 0\log_2 0)$$

$$= 0.97 - 0.55 = \boxed{0.42.}$$

rude waiter:

$\frac{1}{5}$ Yes → 1 Yes, 0 No

$\frac{4}{5}$ No → $\frac{1}{2}$ Yes, $\frac{1}{2}$ No

$$IG_{rude\ waiter} = 0.97 - \frac{4}{5}\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)$$

$$= 0.97 - 0.8 = \boxed{0.17}$$

waiting time.

$\frac{3}{5}$ Yes → $\frac{2}{3}$ Yes, $\frac{1}{3}$ No

$\frac{2}{5}$ No → $\frac{1}{2}$ Yes, $\frac{1}{2}$ No.

$$IG_{waiting\ time} = 0.97 - \frac{3}{5}\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right) - \frac{2}{5}\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)$$

$$= 0.97 - 0.55 - 0.4 = \boxed{0.02.}$$

⇒ pick overcooked pasta as first node. and split on overcooked pasta.

waiting time

$\frac{2}{3}$ Yes → $\frac{1}{2}$ Yes, $\frac{1}{2}$ No.

$\frac{1}{3}$ No → 1 No 0 Yes

$$IG_{waiting\ time\ |\ overcooked} =$$

$$-\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) - \frac{2}{5}\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right) = 0.918 - 0.667$$

$$= \boxed{0.25}$$

rude waiter

$\frac{2}{3}$ Yes → 2 No

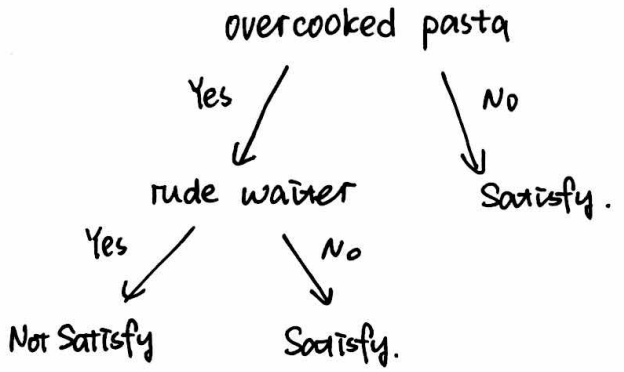
$\frac{1}{3}$ No → 1 Yes

IG(rude waiter | overcooked pasta)

$= 0.918 - 0 = \boxed{0.918}$

⇒ pick rude waiter as second node.

Construct the decision tree:



② Prediction:

Person ID	Satisfied?
6	Yes
7	No
8	Yes.

Question 3:

3.1 k-Nearest Neighbours:

3.1.1

Classification rate on validation set when k = [1,3,5,7,9]:

K	Classification Rate (Validation Set)
1	0.94
3	0.98
5	0.98
7	0.98
9	0.96

k = 9, classification rate drops as the model overfits the data.
k = 1, model could be easily affected by noise or outliers or even mislabel training data.
k = 3 is too close to the margin and could easily underfit the data.
k = 7 is also close to the margin and could overfit the data, especially when we do not have a big enough data set, larger k will result in selecting data point from other classes, besides the computation cost will be higher as k selected to be bigger.
k = 5 will be selected as it is the most stable one while producing satisfying classification rate and the classification rate when k = 5 is 0.98.

With the selected k = 5, results for test set on k = [3,5,7] is shown below:

K	Classification Rate (Test Set)
3	0.98
5	0.98
7	0.92

3.1.2

For k = 3 and k = 5, the performance on test set and validation set matches.
For k = 7, the performance on test set and validation set don't match, because as mentioned in last question, model overfits the data and we do not have enough data points, bigger k value could cause the model to predict wrong results.
Something worth noticing is that when running k = 1 on test set, classification rate is 100% which means test set is very similar to training set data.
To sum up, even though the performance varies a little bit, our pick of k is still stable.

3.2 Logistics Regression:

3.2.1

The relationship among initialized weights, number of iterations and learning rate are complicated.
If initialized weights too random, it may cause more iterations to achieve minimum which would raise computational cost. Here weights is initialized as a normal distribution within [-1,1].
If number of iterations are too big, computational cost will be high while cross entropy and classification rate will not improve much. If number of iterations are small, even the training set classification rate will not reach 100 sometimes. Also, it might not be sufficient enough to find the minimum.
If learning rate is too big, it might over shoot and miss the minimum. If learning rate is too small, it will cost more iterations to reach the minimum.

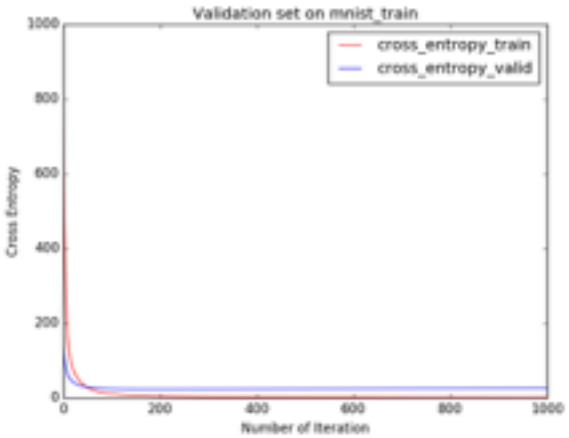
The final parameter set chosen is shown in the table below:

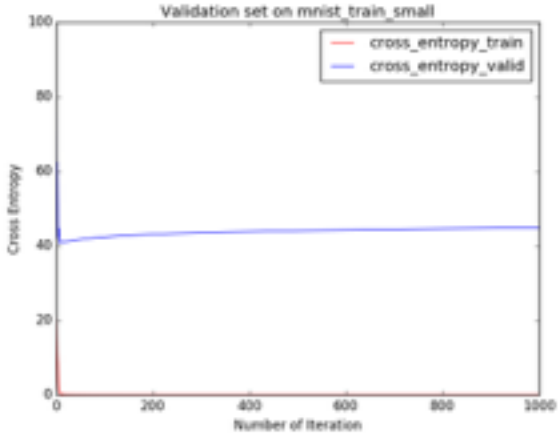
	Learning Rate	# Iterations	Cross Entropy	Classification Rate	Classification Error
mnist_train	0.5	1000	0.844	100	0
validation set	0.5	1000	25.53	90	10
test set	0.5	1000	14.48	92	8

The test set was only computed once after parameter set was picked, the result is satisfying as the classification rate is even higher than validation set. However this might be also because as I mentioned in k-NN section, the test set is more similar to training set.

3.2.2

Run chosen parameters on mnist_train and mnist_train_small, both with validation set. Results are shown below:





	Validation Cross Entropy	Validation Classification Rate
mnist_train	25.53	90
mnist_train_small	46.73	66

Cross entropy for training set decreases fast at the beginning as goal of gradient decent is to minimize the loss function (cross entropy), after numbers of iterations, CE converge to certain value that is very close to zero. Cross entropy for validation set decrease fast at the beginning and then goes up a little and then converge.

The validation classification rate is low when model is trained with train_small data set as it only contains 5 examples which caused the model underfits the validation set.

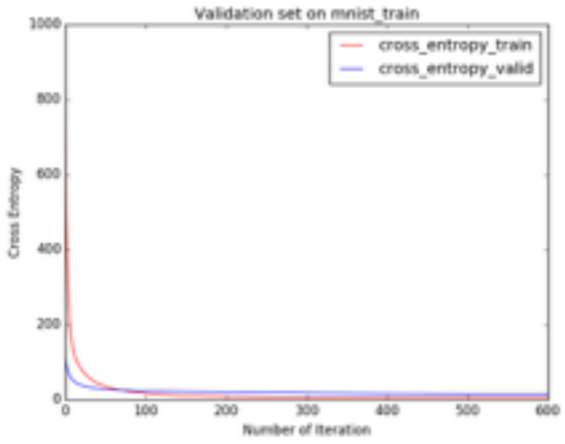
The results change every time when running the code, because weights are randomly initialized. The statistics in the table are average value from 5 runs.

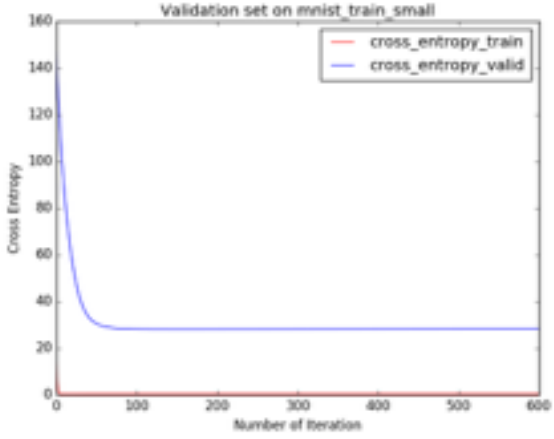
3.3 Regularized Logistic Regression:

3.3.1

Parameters chosen are shown below:

	Learning Rate	Weight Decay	# Iterations	Cross Entropy	Classification Rate	Classification Error
mnist_train	0.5	1	600	5.65	100	0
validation on mnist_train	0.5	1	600	14.93	92	8
mnist_train_small	0.5	1	600	0.41	100	0
validation on mnist_train_small	0.5	1	600	28.42	74	20





3.3.2

Given other parameters fixed, increasing alpha results in a better prediction rate and a lower cross entropy on validation set. According to the graph, cross entropy goes down and then converge. Based on the experiments, alpha equals to 1 produces the best results. This makes sense because 1/alpha is variance of the Gaussian prior added to the loss function, bigger the alpha, weights will be normalized distributed more closely to the origin. However, very big alpha might result in underfitting the data as weights are limited in a small range. But 1 is fine in this case.

Experiment results are shown below:

CE/Classification Rate	alpha = 0.001	alpha = 0.01	alpha = 0.1	alpha = 1
validation on mnist_train	16.6/88	14.7/92	10.4/92	5.19/92
validation on mnist_train_small	139.7/64	89.2/64	34.2/72	27.5/74

3.3.3

Comparing the results with and without regularizer, both cross entropy and classification error deceases for validation test, especially in the mist_train_small case in which the classification rate increased by 8%. Interesting to notice that CE for training set actually increases comparing with non-regularized model, this is because after adding on the Gaussian prior, the model is more generalized.

3.3.4

Summary (mnist_train)	Validation Classification Rate	Validation Cross Entropy
k_NN k = 5	98	N/A
Logistic Regression	90	25.53
Regularized Logistic Regression	92	14.93

Summary (mnist_train_small)	Validation Classification Rate	Validation Cross Entropy
k-NN k = 5	68	N/A
Logistic Regression	66	46.73
Regularized Logistic Regression	74	28.42

Summary (mnist_train)	Test Set Classification Rate	Test Set Cross Entropy
k-NN k = 5	72	N/A
Logistics Regression	92	22.36
Regularized Logistics Regression	94	6.57

Since the weights are initialized by a random normal distribution function, classification rate varies for each run, however, all results other than test set shown above are average over at least 5 runs. In terms of the performance, k-NN has the best classification rate of 98% when picking k = 5. And regularized logistic regression has a classification rate of 92% on validation set. Regularized logistic regression improves classification rate while decreasing the cross entropy. However k-NN performances poorly on test set as it highly depends on data, for example if there are enough data points or is there mislabeled points and noises, while regularized logistics regression tends to have a stable performance. Hence, I think regularized logistics regression is better comparing with other two models.

In general, k-NN is good because it does not require training, simply compute the distance and select a proper K, however the trick is that it is hard to pick a proper k because k-NN is depending on nature of data set such as its distributions and amount, also the algorithm could take up a lot memory as it needs to memorize all data points. On the other hand, logistic regression is good because it could look into the importance of a certain feature and it does not require a lot memory.