# Parameter Tying by Quantization - Applied to Logistic Regression

***Abstract*—————————-**

## I. Introduction

——————————————-

## II. Notation

x=$\{x_1, x_2, \ldots, x_n\}$ denote the variables. $\bar{x} = \{\bar{x_1}, \ldots, \bar{x_n}\}$ denote an assignment of values to the variables and y denote the target values.

Let $g_\theta : \mathbb{R}^n \to \mathbb{R}$ be a function that maps the input vector $x$ to the Real space. For example, for a linear decision boundary, $g_\theta(x) = \theta^T x + b$ where $\theta = \{\theta_1, \theta_2, \ldots, \theta_m\}$ are the parameters to be learnt and b is the bias.

Since we are concerned with Logistic Regression, we define $\sigma : \mathbb{R} \to \mathbb{R}$ as the sigmoid function,

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

where $z = g_\theta(x)$. The vector of parameters $\theta$ is to be learnt. Thus here the sigmoid function is denoted as

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

Let D be the number of samples in the training data denoted by $\bar{X}_{D\times n}$, the rows of which be denoted by $\bar{x}^{(i)}$, $1 \le i \le D$. Let $\bar{Y}$ be a column vector of the target values $\{\bar{y}^{(1)}, \ldots, \bar{y}^{(D)}$

Define the cost function $J : \mathbb{R}^m \to \mathbb{R}$ for Logistic Regression as

$$J(\theta) = \sum_{i=1}^{D} \bar{y}^{(i)} log(\sigma_\theta(\bar{x}^{(i)}) + (1-\bar{y}^{(i)}) log(1-\sigma_\theta(\bar{x}^{(i)})$$

Let $k \le m$ denote the number of equivalence classes (clusters) into which the $m$ parameters are to be divided into (to form $k$ clusters).

## III. The algorithm

Initially Logistic Regression can be implemented using any standard minimization algorithm to do so, For example, BFGS. The weights so obtained may fit the input data very well, however these may be prone to over-fitting. Regularization may be used to overcome this problem, instead we impose certain equality constraints (through equivalence relations) on the parameters and re-learn the model (according to the paper).

One can use one dimensional k-means clustering on the $m$ parameters, where k is specified apriori (i.e. k is not learnt). This can be achieved in $O(m^2 k)$ time using dynamic programming (Wang and Song, 2011).

Let $S_1, S_2, \ldots, S_k$ be the k clusters thus obtained. Define a quantization $\mu = \{\mu_1, \mu_2, \ldots, \mu_k\}$ of $\theta$ and $\mathbb{Q} : \bar{\theta} \to \mu$ be the quantizer between $x$ and $\mu$, so that $\mathbb{Q}(\theta_j) = \mu_i$ iff $\theta_j \in S_i$. Further, let the notation $\mathbb{Q}(\theta)$ denote the m dimensional vector $\{\mathbb{Q}(\theta_1), \mathbb{Q}(\theta_2), \ldots, \mathbb{Q}(\theta_m)\}$ A new cost function can then be defined based on these constraints as follows:

$$J_2(\theta) = \sum_{i=1}^{D} \bar{y}^{(i)} log(\sigma_{\mathbb{Q}(\theta)}(\bar{x}^{(i)}) + (1-\bar{y}^{(i)}) log(1-\sigma_{\mathbb{Q}(\theta)}(\bar{x}^{(i)})$$

The k parameters $\{\mu_1, \ldots, \mu_k\}$ are to be learnt so that $j_2(\theta)$ is minimized and this can be done using any standard minimization technique like BFGS. The initial values for $\{\mu_1, \ldots, \mu_k\}$ can be taken as the means of clusters $S_1, \ldots, S_k$.

It is better to normalize the data appropriately so that the exponential term in the sigmoid function does not become very large or very small. An easy way to normalize is to divide $\theta^T \bar{x}^{(i)}$ by $max_{1\le i\le D}(\theta^T \bar{x})$.

## IV. Experiments and results

In all the experiments, full-batch training was used.

### A. SPAM Dataset

On the SPAM database which contains data about classifying emails as spam/ham (not spam), this algorithm significantly outperforms L2 regularization. The model was trained on 123 spam mails and 340 ham mails. Using L2 regularization built-in with $sklearn.linear\_model.LogisticRegression$ class, an accuracy of about 76.987. The results are tabulated in

Table-**??**.

This, I think, is remarkable as even for k=252 we get better accuracy on the test set, i.e. equivalent to classification without regularization. *Is this happening because I am using Full Batch training?

## *B. Semeion Digit Recognition Database*

On the Semeion database (from UCI Repository) for handwritten digit recognition, L2 regularization outperforms quantization. The training data and the testing data samples were randomly selected from the dataset. (This gives slightly different results every time the algorithm is run). The results for D=400 and D=700 (number of training samples) are tabulated in Table I and Table II respectively. The average accuracy with L2 regularization is 89.308 percent for D=700 and 86.930 percent for D=400.

TABLE I
SEMEION DATASET RESULTS: ACCURACY VS K, NUMBER OF TRAINING SAMPLES=400, NUMBER OF PARAMETERS=257

| **k** | **Accuracy on Training set** | **Accuracy on Test set** |
|---|---|---|
| 4 | 8.750 | 10.059 |
| 130 | 40.750 | 31.014 |
| 145 | 64.000 | 53.562 |
| 148 | 64.250 | 53.814 |
| 150 | 72.000 | 59.933 |
| 152 | 80.000 | 63.034 |
| 155 | 84.500 | 69.237 |
| 158 | 90.750 | 71.752 |
| 160 | 93.000 | 77.536 |
| 195 | 100.000 | 79.631 |
| 200 | 100.000 | 80.469 |
| 210 | 100.000 | 78.793 |
| 250 | 100.000 | 79.799 |

TABLE II
SEMEION DATASET RESULTS: ACCURACY VS K, NUMBER OF TRAINING SAMPLES=700, NUMBER OF PARAMETERS=257

| **k** | **Accuracy on Training set** | **Accuracy on Test set** |
|---|---|---|
| 4 | 9.714 | 9.742 |
| 130 | 58.714 | 47.592 |
| 145 | 94.571 | 76.708 |
| 148 | 92.857 | 75.252 |
| 150 | 99.857 | 81.635 |
| 152 | 100.0 | 79.843 |
| 155 | 100.0 | 81.523 |
| 158 | 100.0 | 80.963 |
| 160 | 100.0 | 82.082 |
| 200 | 100.0 | 83.0907 |
| 250 | 100.0 | 83.0907 |

## *C. Iris Dataset*

On a very basic level, tying parameters by quantization works better than L2 regression on the iris dataset for k=2 and k=3 (not for k=1,4). The algorithm was tested as follows: Of the 150 samples randomly put 75 samples in the training set and the rest in the testing set.

Out of 200 such random samples, for k=2, the quantization algorithm outperformed L2 regularization in 151 of the samples i.e. 75.5 percent and the accuracies were equal in 28 samples. Thus L2 regularization was better only in 10.5 percent of the samples. The mean difference in the accuracies of the two algorithms was 3.053 percent.

Out of the 200 random samples for k=3, L2 regularization was outperformed in 137 samples, and performed equally well in 24 samples. Thus L2 regularization was better only in 19.5 percent of the cases. The mean difference in the accuracies was 2.513 percent.

## ACKNOWLEDGMENT

## REFERENCES

[1]