# HW3 - DVA

*mmendiola3*

## 1. Theory

**a. Write down the formula for computing the gradient of the loss function used in Logistic Regression. Specify what each variable represents in the equation.**

Cost function:

$$J(\theta) = \sum_{i=1}^{n} log(1 + \exp(y^i < \theta, x^i >)$$

We'll add a constant $1/n$ to scale the update by the number of training samples. Update function:

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} \sum_{i=1}^{n} log(1 + \exp(y^i < \theta, x^i >))$$

$$= \theta_j - \alpha \sum_{i=1}^{n} \frac{1}{1 + \exp(y^i < \theta, x^i >)} \cdot \frac{\partial}{\partial \theta_j} \exp(y^i < \theta, x^i >)$$

$$= \theta_j - \alpha \sum_{i=1}^{n} \frac{\exp(y^i < \theta, x^i >) \cdot \frac{\partial}{\partial \theta_j} y^i < \theta, x^i >}{1 + \exp(y^i < \theta, x^i >)}$$

$$= \theta_j - \alpha \sum_{i=1}^{n} \frac{\exp(y^i < \theta, x^i >) \cdot y^i x_j^i}{1 + \exp(y^i < \theta, x^i >)}$$

$$= \theta_j - \alpha \sum_{i=1}^{n} \frac{y^i x^i}{1 + \exp(-y^i < \theta, x^i >)}$$

note: $< \theta, x^i >$ is constant with the exception of $\theta_j \cdot x_j^i$

**Terms:**

- $\theta_j$ : The value of the current parameter vector at feature index j
- $\alpha$ : The learning rate, which decreases over each training iteration
- $n$ : The number of training samples
- $y^i$ : The classification label for training sample index $i$
- $x^i$ : The feature vector for training sample index $i$
- $< \theta, x^i >$ : The dot product of the parameter vector and the training sample at index $i$

**b. Write pseudocode for training a model using Logistic Regression.**

```
Calculate negative log likelyhood for a given x, y, and theta
function calc_cost(x, y, theta):
  return 1/(log(1 + exp(y *<theta, x>)))
```

```
x = training matrix (n examples x d features)
y = training labels (n labels of -1 or 1)

set x^i_0 = 1 # for bias term
alpha = <some learning rate>
epsilon = <some stopping threshold value>
theta = generate random vector of size d+1 (feature count + bias)
cost = calc_cost(theta)
delta_cost = cost # tracks change in the cost function

while delta_cost > epsilon:
  theta = theta - alpha * sum((y*x) / (1 + exp(-y * <theta, x>)))
  new_cost = calc_cost(x, y, theta)
  delta_cost = cost - new_cost
  cost = new_cost

theta is now trained against the sample set
```

**c. Calculate the number of operations per gradient descent iteration. (Hint: Use variable n for number of examples and d for dimensionality.)**

$$\sum_{i=1}^{n} f(i) = O(n \cdot f(i))$$

$$y^i \cdot x^i = O(d)$$

$$< \theta, x^i > = O(d)$$

$$y^i \cdot h = O(d)$$

$$\frac{f() \cdot g(d)}{1 + exp(h(d))} = O(d)$$

$$\therefore$$

$$\sum_{i=1}^{n} \frac{y^i x^i}{1 + \exp(-y^i < \theta, x^i >)} = O(nd)$$

There are a number of computations that are linear wrt d. The summation across samples adds the multiplicative factor n.