

RECITATION 5

LOGISTIC REGRESSION

10-601: INTRODUCTION TO MACHINE LEARNING

10/09/2020

This recitation consists of 3 parts: In part 1, we will go over how to **represent data features using dense and sparse representation**. Part 2 will go over the **negative log likelihood** and **gradient derivations** for **binary logistic regression**, as well as a small toy example. Part 3 will focus on **multinomial logistic regression**. The materials were designed to help you with Homework 4.

1 Feature Vector Representation

In many machine learning problems, we will want to find the set of parameters that optimize our objective function. Usually, a naive (dense) representation will suffice, but sometimes careful consideration must be taken to afford tenable run times.

1. A Naive Representation

- Consider a feature vector x defined by $x_0 = 1, x_1 = 0, x_2 = 2, x_3 = 0, x_4 = 1$. Write the pseudo code to naively represent such a vector in Python.
- One thing we often want to do in many machine learning algorithms is take the dot product of the feature vector with a parameter vector. Given the naive representation above, write a function that takes the dot product between two vectors.

```
def dot(X, W):  
    product = 0.0  
    # TODO: Implement dot product  
  
    return product
```

- Now let our parameter vector w be defined by $w_0 = 0, w_1 = 1, w_2 = 2, w_3 = 3, w_4 = 4$. Time how long it takes to take the dot product $x \cdot w$. What if you append 10,000 zeros on the end of both x and w

2. Take Advantage of Nothing

- Something key to notice in the larger x and w is that they have a large amount of zeros. This is called being sparse (as opposed to being dense). We can hope to take advantage of this. Write a better representation of x in code that takes advantage of sparsity.

- (b) Like in the question before, write a function that takes the dot product between two vectors x and w , this time taking advantage of the fact that x is sparse.

```
def sparse_dot(X, W):  
    product = 0.0  
    # TODO: Implement sparse dot product  
  
    return product
```

- (c) Now time this new dot product function on extremely sparse inputs and compare to the naive representation.

3. Sparse Vector Operations

Define an add function that adds a sparse vector to a dense vector

```
def sparse_add(X, W):  
    # TODO: Implement updating W by adding values in X  
  
    return W  
  
def sparse_sub(X, W):  
    # TODO: Implement updating W by subtracting values in X  
  
    return W
```

2 Binary Logistic Regression

1. For binary logistic regression, we have the following dataset:

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(N)}, y^{(N)})\} \text{ where } \mathbf{x}^{(i)} \in \mathbb{R}^M, y^{(i)} \in \{0, 1\}$$

A couple of reminders from lecture

- 1.

$$\sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x}^{(i)})} = \frac{\exp(\boldsymbol{\theta}^T \mathbf{x}^{(i)})}{1 + \exp(\boldsymbol{\theta}^T \mathbf{x}^{(i)})}$$

- 2.

$$\begin{aligned} p(y^{(i)} | \mathbf{x}^{(i)}, \boldsymbol{\theta}) &= \begin{cases} \sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)}) & y^{(i)} = 1 \\ 1 - \sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)}) & y^{(i)} = 0 \end{cases} \\ &= \sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)})^{y^{(i)}} (1 - \sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)}))^{(1-y^{(i)})} \end{aligned}$$

- 3.

$$\phi^{(i)} = \sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)})$$

- 4.

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z))$$

5. if $z = f(\boldsymbol{\theta})$ then

$$\frac{\partial \sigma(f(\boldsymbol{\theta}))}{\partial \theta_j} = \sigma(f(\boldsymbol{\theta}))(1 - \sigma(f(\boldsymbol{\theta}))) \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_j}$$

In binary logistic regression, this is

$$\frac{\partial \phi^{(i)}}{\partial \theta_j} = \phi^{(i)} * (1 - \phi^{(i)}) * \frac{\partial \boldsymbol{\theta}^T \mathbf{x}^{(i)}}{\partial \theta_j}$$

6. remember that

$$\frac{\partial \log(f(z))}{\partial z} = \frac{1}{f(z)} \frac{\partial f(z)}{\partial z}$$

2. (a) Write down our objective function, $J(\boldsymbol{\theta})$, which is $\frac{1}{N}$ times the negative conditional log-likelihood of data, in terms of N and $p(y^{(i)} | \mathbf{x}^{(i)}, \boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \mathbb{R}^M$. As usual, assume $y^{(i)}$ are independent and identically distributed.

- (b) Write $J(\boldsymbol{\theta})$ in terms of $\sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)})$. simplify as much as possible. Then write in terms of $\phi^{(i)}$

- (c) In stochastic gradient descent, we use only a single $\mathbf{x}^{(i)}$. Given $\phi^{(i)} = \sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)})$ and

$$J^{(i)}(\boldsymbol{\theta}) = -y^{(i)} \log(\phi^{(i)}) - (1 - y^{(i)}) \log(1 - \phi^{(i)})$$

Show that the partial derivative of $J^{(i)}(\boldsymbol{\theta})$ with respect to the j th parameter θ_j is as follows:

$$\frac{\partial J^{(i)}(\boldsymbol{\theta})}{\partial \theta_j} = (\sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

Remember,

$$\frac{\partial \phi^{(i)}}{\partial \theta_j} = \phi^{(i)} * (1 - \phi^{(i)}) * \frac{\partial \boldsymbol{\theta}^T \mathbf{x}^{(i)}}{\partial \theta_j}$$

3. Let's go through a toy problem.

Y	X_1	X_2	X_3
1	1	2	1
1	1	1	-1
0	1	-2	1

(a) What is $J(\boldsymbol{\theta})$ of above data given initial $\boldsymbol{\theta} = \begin{bmatrix} -2 \\ 2 \\ 1 \end{bmatrix}$?

(b) Calculate $\frac{\partial J^{(1)}(\boldsymbol{\theta})}{\partial \theta_1}$, $\frac{\partial J^{(1)}(\boldsymbol{\theta})}{\partial \theta_2}$ and $\frac{\partial J^{(1)}(\boldsymbol{\theta})}{\partial \theta_3}$ for first training example. Note that $\sigma(3) \approx 0.95$.

(c) Calculate $\frac{\partial J^{(2)}(\boldsymbol{\theta})}{\partial \theta_1}$, $\frac{\partial J^{(2)}(\boldsymbol{\theta})}{\partial \theta_2}$ and $\frac{\partial J^{(2)}(\boldsymbol{\theta})}{\partial \theta_3}$ for second training example. Note that $\sigma(-1) \approx 0.25$.

(d) Assuming we are doing stochastic gradient descent with a learning rate of 1.0, what are the updated parameters $\boldsymbol{\theta}$ if we update $\boldsymbol{\theta}$ using the second training example?

(e) What is the new $J(\boldsymbol{\theta})$ after doing the above update? Should it decrease or increase?

-
- (f) Given a test example where $(X_1 = 1, X_2 = 3, X_3 = 4)$, what will the classifier output following this update?

3 Multinomial Logistic Regression

1. Definition

Multinomial logistic regression, also known as softmax regression or multiclass logistic regression, is a generalization of binary logistic regression.

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(N)}, y^{(N)})\} \text{ where } \mathbf{x}^{(i)} \in \mathbb{R}^M, y^{(i)} \in \{1, \dots, K\} \text{ for } i = 1, \dots, N$$

Here N is the number of training examples, M is the number of features, and K is the number of possible classes, which is usually greater than two to be interesting.

$$p(Y^{(i)} = y^{(i)} \mid \mathbf{x}^{(i)}, \Theta) = \frac{\exp(\Theta_{y^{(i)}} \mathbf{x}^{(i)})}{\sum_{j=1}^K \exp(\Theta_j \mathbf{x}^{(i)})} = \text{softmax}(\Theta \mathbf{x}^{(i)})_{y^{(i)}} \quad (1)$$

where Θ is the parameter matrix of size $K \times (M + 1)$, and $\Theta_{y^{(i)}}$ denotes the $y^{(i)}$ th **row** of Θ , which is the parameter vector for the $y^{(i)}$ th class.

2. Suppose $K = 4$ and $N = 10$, $M = 3$. What could Θ look like?

3. A *one-hot encoding* is a vector representation of a one dimensional integer defined as such: a vector \mathbf{c} of length K is a *one-hot encoding* of integer $n \iff |\mathbf{c}| = K$ and for all $j \neq n$, $\mathbf{c}_j = 0$ and $\mathbf{c}_n = 1$. Give some examples of one-hot encodings where $K = 5$.

4. In multinomial logistic regression, we form the matrix \mathbf{T} where the i th row of \mathbf{T} is the one-hot encoding of label $y^{(i)}$. Draw \mathbf{T} if $\mathbf{y} = [1, 3, 1, 4, 4]^T$ and $K = 4$.