Recitation 5
Logistic Regression

10-601: Introduction to Machine Learning

3/12/2021

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**

   (a) 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.

   ```
   X = [1, 0, 2, 0, 1]
   ```

   (b) 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
   ```

   ```
   def dot(X, W):
       product = 0.0
       for x_i, w_i in zip(X, W):
           product += x_i * w_i
   ```

```
    return product
```

(c) 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

In a jupyter python3 environment:

```
W = [0, 1, 2, 3, 4]
%time dot(X, W)

X = X + [0] * 10000
W = W + [0] * 10000
%time dot(X, W)
```

As a note to TAs: If you choose to use python during recitation, it might be good to show how even using numpy-backed arrays will not save you from naively taking the dot product between sparse vectors.

2. **Take Advantage of Nothing**

(a) 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.

```
X = {
        0: 1,
        2: 2,
        4: 1,
    }
```

(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
```

```
def sparse_dot(X, W):
    product = 0.0
```

```
    for i, v in X.items():
        product += W[i] * v
    return product
```

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

```
%time dot(X, W)
```

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
```

```
def sparse_add(X, W):
    for i, v in X.items():
        W[i] += v
    return W

def sparse_sub(X, W):
    for i, v in X.items():
        W[i] -= v
    return W
```

2 Binary Logistic Regression

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

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

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

3.
$$\boldsymbol{\phi}^{(i)} = \sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)})$$

4.
$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z))$$

5. if $z = f(\boldsymbol{\theta})$ then
$$\frac{\partial \sigma(f(\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_j} = \sigma(f(\boldsymbol{\theta}))(1 - \sigma(f(\boldsymbol{\theta}))) \frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_j}$$
In binary logistic regression, this is
$$\frac{\partial \boldsymbol{\phi}^{(i)}}{\partial \boldsymbol{\theta}_j} = \boldsymbol{\phi}^{(i)} * (1 - \boldsymbol{\phi}^{(i)}) * \frac{\partial \boldsymbol{\theta}^T \mathbf{x}^{(i)}}{\partial \theta_j}$$

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 $p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}\right)$ where $\boldsymbol{\theta} \in \mathbb{R}^M$. As usual, assume $y^{(i)}$ are independent and identically distributed.

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\log(\prod_{i=1}^{N} p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}\right))$$
$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\log(p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}\right))$$

(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 $\boldsymbol{\phi}^{(i)}$

$$\begin{aligned}
J(\boldsymbol{\theta}) &= -\frac{1}{N}\sum_{i=1}^{N}\log\left(\sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)})^{y^{(i)}}(1-\sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)}))^{(1-y^{(i)})}\right) \\
&= -\frac{1}{N}\sum_{i=1}^{N}(y^{(i)}\log\left(\sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-\sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)})\right)) \\
&= -\frac{1}{N}\sum_{i=1}^{N}(y^{(i)}\log\left(\boldsymbol{\phi}^{(i)}\right) + (1-y^{(i)})\log\left(1-\boldsymbol{\phi}^{(i)}\right))
\end{aligned}$$

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

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

Show that the partial derivative of $J^{(i)}(\boldsymbol{\theta})$ with respect to the jth 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,

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

note

$$\frac{\partial \boldsymbol{\theta}^T \mathbf{x}^{(i)}}{\partial \theta_j} = \mathbf{x}_j^{(i)}$$

$$\begin{aligned}
\frac{\partial J^{(i)}(\boldsymbol{\theta})}{\partial \theta_j} &= -\frac{y^{(i)}}{\boldsymbol{\phi}^{(i)}} \frac{\partial \boldsymbol{\phi}^{(i)}}{\partial \boldsymbol{\theta}_j} - \frac{(1-y^{(i)})}{1-\boldsymbol{\phi}^{(i)}} \frac{\partial (1-\boldsymbol{\phi}^{(i)})}{\partial \boldsymbol{\theta}_j} \\
&= -\frac{y^{(i)}}{\boldsymbol{\phi}^{(i)}} \frac{\partial \boldsymbol{\phi}^{(i)}}{\partial \boldsymbol{\theta}_j} + \frac{(1-y^{(i)})}{1-\boldsymbol{\phi}^{(i)}} \frac{\partial \boldsymbol{\phi}^{(i)}}{\partial \boldsymbol{\theta}_j} \\
&= -\frac{y^{(i)}}{\boldsymbol{\phi}^{(i)}} \boldsymbol{\phi}^{(i)} * (1-\boldsymbol{\phi}^{(i)}) * \frac{\partial \boldsymbol{\theta}^T \mathbf{x}^{(i)}}{\partial \theta_j} + \frac{(1-y^{(i)})}{1-\boldsymbol{\phi}^{(i)}} \boldsymbol{\phi}^{(i)} * (1-\boldsymbol{\phi}^{(i)}) * \frac{\partial \boldsymbol{\theta}^T \mathbf{x}^{(i)}}{\partial \theta_j} \\
&= (-y^{(i)}(1-\boldsymbol{\phi}^{(i)}) + (1-y^{(i)})\boldsymbol{\phi}^{(i)})\mathbf{x}_j^{(i)} \\
&= (-y^{(i)} + y^{(i)}\boldsymbol{\phi}^{(i)} + \boldsymbol{\phi}^{(i)} - y^{(i)}\boldsymbol{\phi}^{(i)})\mathbf{x}_j^{(i)} \\
&= (\boldsymbol{\phi}^{(i)} - y^{(i)})\mathbf{x}_j^{(i)} \\
&= (\sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)}) - y^{i})\mathbf{x}_j^{(i)}
\end{aligned}$$

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 $\boldsymbol{\theta} = \begin{bmatrix} -2 \\ 2 \\ 1 \end{bmatrix}$?

$J(\boldsymbol{\theta}) = -\frac{1}{3}[\log(\sigma(3)) + \log(\sigma(-1)) + \log(1-\sigma(-5))] \approx 0.46$

(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 $\sigma(3) \approx 0.95$.

$$\frac{\partial J^{(1)}(\boldsymbol{\theta})}{\partial \theta_1} = (\sigma(3) - 1)1 = -0.05$$
$$\frac{\partial J^{(1)}(\boldsymbol{\theta})}{\partial \theta_2} = (\sigma(3) - 1)2 = -0.10$$
$$\frac{\partial J^{(1)}(\boldsymbol{\theta})}{\partial \theta_3} = (\sigma(3) - 1)1 = -0.05$$

(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 $\sigma(-1) \approx 0.25$.

$$\frac{\partial J^{(2)}(\boldsymbol{\theta})}{\partial \theta_1} = (\sigma(-1) - 1)1 = -0.75$$
$$\frac{\partial J^{(2)}(\boldsymbol{\theta})}{\partial \theta_2} = (\sigma(-1) - 1)1 = -0.75$$
$$\frac{\partial J^{(2)}(\boldsymbol{\theta})}{\partial \theta_3} = (\sigma(-1) - 1) - 1 = 0.75$$

(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?

$$\begin{bmatrix} -2 \\ 2 \\ 1 \end{bmatrix} - 1 \begin{bmatrix} -0.75 \\ -0.75 \\ 0.75 \end{bmatrix} = \begin{bmatrix} -1.25 \\ 2.75 \\ 0.25 \end{bmatrix}$$

(e) What is the new $J(\boldsymbol{\theta})$ after doing the above update? Should it decrease or increase?
$J(\boldsymbol{\theta}) = 0.09$

It should decrease for logistic classifier to learn.

(f) Given a test example where ($X_1 = 1, X_2 = 3, X_3 = 4$), what will the classifier output following this update?
$\sigma(\boldsymbol{\theta}^T X) > 0.5 \implies Y = 1$

3 Multinomial Logistic Regression (Optional Learning)

1. Definition

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

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

   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.

$$p\left(Y^{(i)} = y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{\Theta}\right) = \frac{\exp\left(\mathbf{\Theta}_{y^{(i)}} \mathbf{x}^{(i)}\right)}{\sum_{j=1}^{K} \exp\left(\mathbf{\Theta}_j \mathbf{x}^{(i)}\right)} = \text{softmax}(\mathbf{\Theta}\mathbf{x}^{(i)})_{y^{(i)}} \tag{1}$$

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

2. Suppose $K = 4$ and $N = 10$, $M = 3$. What could $\mathbf{\Theta}$ look like?

   $\mathbf{\Theta}$ will have K rows because there are K distinct labels. $\mathbf{\Theta}$ will have M+1 columns because there are M features plus a bias term. So any K by (M+1) matrix is a possible candidate for $\mathbf{\Theta}$.

$$\begin{bmatrix} 0.5 & -2 & 5 & 7 \\ 0 & 0.22 & 6 & 1 \\ 9 & 2 & 0.1 & 6 \\ 7 & -0.5 & 0 & 1 \end{bmatrix}$$

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.

   Let $n = 1, \implies \mathbf{c} = [1, 0, 0, 0, 0]^T$
   Let $n = 3, \implies \mathbf{c} = [0, 0, 1, 0, 0]^T$
   Let $n = 4, \implies \mathbf{c} = [0, 0, 0, 1, 0]^T$

4. In multinomial logistic regression, we form the matrix $\mathbf{T}$ where the ith 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.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$