Recitation 4
Logisitc Regression

10-601: Introduction to Machine Learning

02/15/2019

1 Binary Logistic Regression

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

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

   (a) Write down the negative conditional log-likelihood of data denoted by $J(\boldsymbol{\theta})$ in terms of N and $p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}\right)$ where $\boldsymbol{\theta} \in \mathbb{R}^M$.

   (b) Given $\sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^T \mathbf{x}^{(i)})}$. Using the i^{th} training example, show that the partial derivative of $J_i(\boldsymbol{\theta})$ with respect to the jth parameter θ_j is as follows:

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

2. Let's go through a toy problem.

Y	X_1	X_2	X_3
1	1	2	4
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 $\sigma(6) \approx 1$.

(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 $\boldsymbol{\theta}$ after seeing 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?

2 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 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 code to naively represent such a vector in your language of choice.

   (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.

   (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

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.

   (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.

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

3. *Sparse perceptrons*

   (a) Suppose we have this dataset of 10 features:

   $$\mathcal{D} = \{([x_0 : 3, x_5 : 4], 0), [x_3 : 1, x_8 : 4], 1), [x_1 : 3, x_9 : 1], 1)\}$$

   Where we only show the features that are nonzero.

   First define an add function that adds a sparse vector to a dense vector

   (b) Now write a program to train a perceptron to classify the dataset correctly, taking advantage of the sparsity of the features.