2. Hinge Loss

Extension Note: Project 1 due date has been extended by 2 days to **July 4 23:59UTC** (Note the UTC time zone).

In this project you will be implementing linear classifiers beginning with the Perceptron algorithm. You will begin by writing your loss function, a hinge-loss function. For this function you are given the parameters of your model θ and θ_0. Additionally, you are given a feature matrix in which the rows are feature vectors and the columns are individual features, and a vector of labels representing the actual sentiment of the corresponding feature vector.

Hinge Loss on One Data Sample

1.0/1 point (graded)

First, implement the basic hinge loss calculation on a single data-point. Instead of the entire feature matrix, you are given one row, representing the feature vector of a single data sample, and its label of +1 or -1 representing the ground truth sentiment of the data sample.

Reminder: You can implement this function locally first, and run `python test.py` in your `sentiment_analysis` directory to validate basic functionality before checking against the online grader here.

Available Functions: You have access to the NumPy python library as np; No need to import anything.

```

    Args:
        feature_vector - A numpy array describing the given data point.
        label - A real valued number, the correct classification of the data
            point.
        theta - A numpy array describing the linear classifier.
        theta_0 - A real valued number representing the offset parameter.


    Returns: A real number representing the hinge loss associated with the
    given data point and parameters.
    """
    agreement = label * (np.dot(theta,feature_vector) + theta_0)
    return max(0, 1 - agreement)

```

Press ESC then TAB or click outside of the code editor to exit

Correct

```
def hinge_loss_single(feature_vector, label, theta, theta_0):
    """
    Finds the hinge loss on a single data point given specific classification
    parameters.

    Args:
        feature_vector - A numpy array describing the given data point.
        label - A real valued number, the correct classification of the data
            point.
        theta - A numpy array describing the linear classifier.
        theta_0 - A real valued number representing the offset parameter.


    Returns: A real number representing the hinge loss associated with the
    given data point and parameters.
    """
    y = np.dot(theta, feature_vector) + theta_0
    loss = max(0.0, 1 - y * label)
    return loss
```

Test results

```
                                                                        See full output
CORRECT
                                                                        See full output
```

Solution:

See above for expected answer.

Another possible solution is:

```
def hinge_loss_single(feature_vector, label, theta, theta_0):
        y = theta @ feature_vector + theta_0
        return max(0, 1 - y * label)
```

Here, the `@` operator is shorthand for dot product.

Submit You have used 1 of 20 attempts

Answers are displayed within the problem

The Complete Hinge Loss

1.0/1 point (graded)
Now it's time to implement the complete hinge loss for a full set of data. Your input will be a full feature matrix this time, and you will have a vector of corresponding labels. The k^{th} row of the feature matrix corresponds to the k^{th} element of the labels vector. This function should return the appropriate loss of the classifier on the given dataset.

Available Functions: You have access to the NumPy python library as np, and your previous function as `hinge_loss_single`

```
            represents a single data point.
        labels - A numpy array where the kth element of the array is the
            correct classification of the kth row of the feature matrix.
        theta - A numpy array describing the linear classifier.
        theta_0 - A real valued number representing the offset parameter.


    Returns: A real number representing the hinge loss associated with the
    given dataset and parameters. This number should be the average hinge
    loss across all of the points in the feature matrix.
    """
    agreement = labels * (np.dot(feature_matrix, theta) + theta_0)
    hinge_loss = np.maximum(0, 1 - agreement)
    return np.mean(hinge_loss)

```

Press ESC then TAB or click outside of the code editor to exit

Correct

```
def hinge_loss_full(feature_matrix, labels, theta, theta_0):
    """
    Finds the total hinge loss on a set of data given specific classification
    parameters.

    Args:
        feature_matrix - A numpy matrix describing the given data. Each row
            represents a single data point.
        labels - A numpy array where the kth element of the array is the
            correct classification of the kth row of the feature matrix.
        theta - A numpy array describing the linear classifier.
        theta_0 - A real valued number representing the offset parameter.


    Returns: A real number representing the hinge loss associated with the
    given dataset and parameters. This number should be the average hinge
    loss across all of the points in the feature matrix.
    """
    loss = 0
    for i in range(len(feature_matrix)):
        loss += hinge_loss_single(feature_matrix[i], labels[i], theta, theta_0)
    return loss / len(labels)
```

Test results

See full output

```
CORRECT
```

See full output

Solution:

See above for expected answer: we simply sum and take the average.

Another possible solution is:

```
def hinge_loss_full(feature_matrix, labels, theta, theta_0):
        ys = feature_matrix @ theta + theta_0
        loss = np.maximum(1 - ys * labels, np.zeros(len(labels)))
        return np.mean(loss)
```

Here, we use the fact that matrix multiplication is equivalent to stacking the result of dot products to from `ys`, an array of outputs. We then use `np.maximum` to take the element-wise maximum of two arrays.

This solution is more efficient because it makes use of NumPy's fast matrix multiplication capability.

Submit You have used 1 of 20 attempts

Answers are displayed within the problem

Discussion

Show Discussion

Topic: Unit 1 Linear Classifiers and Generalizations (2 weeks):Project 1: Automatic Review Analyzer / 2. Hinge Loss