4. Pegasos Algorithm

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

Now you will implement the Pegasos algorithm. For more information, refer to the original paper at original paper.

The following pseudo-code describes the Pegasos update rule.

Pegasos update rule $\left(x^{(i)}, y^{(i)}, \lambda, \eta, \theta\right)$:
if $y^{(i)}\left(\theta \cdot x^{(i)}\right) \leq 1$ then
    update $\theta = (1 - \eta\lambda)\,\theta + \eta y^{(i)} x^{(i)}$
else:
    update $\theta = (1 - \eta\lambda)\,\theta$

The η parameter is a decaying factor that will decrease over time. The λ parameter is a regularizing parameter.

In this problem, you will need to adapt this update rule to add a bias term (θ_0) to the hypothesis, but take care not to penalize the magnitude of θ_0.

Pegasos Single Step Update

1.0/1 point (graded)
Next you will implement the single step update for the Pegasos algorithm. This function is very similar to the function that you implemented in **Perceptron Single Step Update**, except that it should utilize the Pegasos parameter update rules instead of those for perceptron. The function will also be passed a λ and η value to use for updates.

Available Functions: You have access to the NumPy python library as `np`.

```
            algorithm before this update.
        current_theta_0 - The current theta_0 being used by the
            Pegasos algorithm before this update.

    Returns: A tuple where the first element is a numpy array with the value of
    theta after the current update has completed and the second element is a
    real valued number with the value of theta_0 after the current updated has
    completed.
    """
    if label * (feature_vector @ current_theta + current_theta_0) <= 1:
        current_theta =  (1 - eta*L) * current_theta + eta * feature_vector * label
        current_theta_0 = current_theta_0 + eta * label
    else:
        current_theta = (1 - eta*L) * current_theta
    return (current_theta, current_theta_0)
```

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

Correct

```
def pegasos_single_step_update(
        feature_vector,
        label,
        L,
        eta,
        current_theta,
        current_theta_0):
    """
    Properly updates the classification parameter, theta and theta_0, on a
    single step of the Pegasos algorithm

    Args:
        feature_vector - A numpy array describing a single data point.
        label - The correct classification of the feature vector.
        L - The lamba value being used to update the parameters.
        eta - Learning rate to update parameters.
        current_theta - The current theta being used by the Pegasos
            algorithm before this update.
        current_theta_0 - The current theta_0 being used by the
            Pegasos algorithm before this update.

    Returns: A tuple where the first element is a numpy array with the value of
    theta after the current update has completed and the second element is a
    real valued number with the value of theta_0 after the current updated has
    completed.
    """
    mult = 1 - (eta * L)
    if label * (np.dot(feature_vector, current_theta) + current_theta_0) ≤ 1:
        return ((mult * current_theta) + (eta * label * feature_vector),
                (current_theta_0) + (eta * label))
    return (mult * current_theta, current_theta_0)
```

Test results

See full output

```
CORRECT
```

See full output

Solution:

See above for expected answer.

The Pegasos algorithm mixes together a few good ideas: regularization, hinge loss, subgradient updates, and decaying learning rate.

When using a bias, the bias update for a mistake becomes:

$$\theta_0 = \theta_0 + \eta y^{(i)}$$

Submit You have used 1 of 20 attempts

Answers are displayed within the problem

Full Pegasos Algorithm

1.0/1 point (graded)

Finally you will implement the full Pegasos algorithm. You will be given the same feature matrix and labels array as you were given in **Full Perceptron Algorithm**. You will also be given T, the maximum number of times that you should iterate through the feature matrix before terminating the algorithm. Initialize θ and θ_0 to zero. For each update, set $\eta = \frac{1}{\sqrt{t}}$ where t is a counter for the number of updates performed so far (between 1 and nT inclusive). This function should return a tuple in which the first element is the final value of θ and the second element is the value of θ_0.

Note: Please call `get_order(feature_matrix.shape[0])`, and use the ordering to iterate the feature matrix in each iteration. The ordering is specified due to grading purpose. In practice, people typically just randomly shuffle indices to do stochastic optimization.

Available Functions: You have access to the NumPy python library as `np` and `pegasos_single_step_update` which you have already implemented.

```
    n_para = feature_matrix.shape[1]
    theta = np.zeros(n_para)
    theta_0 = 0
    count = 1
    for t in range(T):
        for i in get_order(n_data):
            eta = 1/(count**0.5)
            theta, theta_0 = pegasos_single_step_update(feature_matrix[i],
                                                        labels[i],
                                                        L,
                                                        eta,
                                                        theta,
                                                        theta_0)
            count += 1
    return (theta, theta_0)
```

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

Correct

```
def pegasos(feature_matrix, labels, T, L):
    """
    Runs the Pegasos algorithm on a given set of data. Runs T
iterations through the data set, there is no need to worry about
stopping early.

For each update, set learning rate = 1/sqrt(t),
where t is a counter for the number of updates performed so far (between 1
and nT inclusive).

NOTE: Please use the previously implemented functions when applicable.
Do not copy paste code from previous parts.

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.
    T - An integer indicating how many times the algorithm
        should iterate through the feature matrix.
    L - The lamba value being used to update the Pegasos
        algorithm parameters.

Returns: A tuple where the first element is a numpy array with the value of
the theta, the linear classification parameter, found after T
iterations through the feature matrix and the second element is a real
number with the value of the theta_0, the offset classification
parameter, found after T iterations through the feature matrix.
"""
(nsamples, nfeatures) = feature_matrix.shape
theta = np.zeros(nfeatures)
theta_0 = 0
count = 0
for t in range(T):
    for i in get_order(nsamples):
        count += 1
        eta = 1.0 / np.sqrt(count)
        (theta, theta_0) = pegasos_single_step_update(
            feature_matrix[i], labels[i], L, eta, theta, theta_0)
return (theta, theta_0)
```

Test results

See full output

CORRECT

See full output

Submit

You have used 2 of 20 attempts