7. Classification and Accuracy

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

Now we need a way to actually use our model to classify the data points. In this section, you will implement a way to classify the data points using your model parameters, and then measure the accuracy of your model.

Classification

1.0/1 point (graded)

Implement a classification function that uses θ and θ_0 to classify a set of data points. You are given the feature matrix, θ, and θ_0 as defined in previous sections. This function should return a numpy array of -1s and 1s. If a prediction is **greater than** zero, it should be considered a positive classification.

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

Tip:: As in previous exercises, when x is a float, “$x = 0$" should be checked with $|x| < \epsilon$.

```
    Args:
        feature_matrix - A numpy matrix describing the given data. Each row
            represents a single data point.
                theta - A numpy array describing the linear classifier.
        theta - A numpy array describing the linear classifier.
        theta_0 - A real valued number representing the offset parameter.

    Returns: A numpy array of 1s and -1s where the kth element of the array is
    the predicted classification of the kth row of the feature matrix using the
    given theta and theta_0. If a prediction is GREATER THAN zero, it should
    be considered a positive classification.
    """
    epsilon = 10**(-8)
    prediction = feature_matrix @ theta + theta_0
    return np.where(prediction > epsilon, 1, -1)
```

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

Correct

```
def classify(feature_matrix, theta, theta_0):
    """
    A classification function that uses theta and theta_0 to classify a set of
    data points.

    Args:
        feature_matrix - A numpy matrix describing the given data. Each row
            represents a single data point.
                theta - A numpy array describing the linear classifier.
        theta - A numpy array describing the linear classifier.
        theta_0 - A real valued number representing the offset parameter.

    Returns: A numpy array of 1s and -1s where the kth element of the array is
    the predicted classification of the kth row of the feature matrix using the
    given theta and theta_0. If a prediction is GREATER THAN zero, it should
    be considered a positive classification.
    """
    (nsamples, nfeatures) = feature_matrix.shape
    predictions = np.zeros(nsamples)
    for i in range(nsamples):
        feature_vector = feature_matrix[i]
        prediction = np.dot(theta, feature_vector) + theta_0
        if (prediction > 0):
            predictions[i] = 1
        else:
            predictions[i] = -1
    return predictions
```

Test results

See full output

CORRECT

See full output

Solution:

See above for expected answer.

Another possible solution is:

```
def classify(feature_matrix, theta, theta_0):
        return (feature_matrix @ theta + theta_0 > 1e-7) * 2.0 - 1
```

Here, we use the fact that a boolean will be implicitly casted by NumPy into 0 or 1 when mutiplied by a float.

Again, note that we identified 0 to the range $[-\varepsilon, +\varepsilon]$ for numerical reasons.

Submit You have used 1 of 20 attempts

Answers are displayed within the problem

Accuracy

1.0/1 point (graded)

We have supplied you with an `accuracy` function:

```
def accuracy(preds, targets):
    """
    Given length-N vectors containing predicted and target labels,
    returns the percentage and number of correct predictions.
    """
    return (preds == targets).mean()
```

The `accuracy` function takes a numpy array of predicted labels and a numpy array of actual labels and returns the prediction accuracy. You should use this function along with the functions that you have implemented thus far in order to implement `classifier_accuracy`.

The `classifier_accuracy` function should take 6 arguments:

- a classifier function that, itself, takes arguments `(feature_matrix, labels, **kwargs)`
- the training feature matrix
- the validation feature matrix
- the training labels
- the valiation labels
- a `**kwargs` argument to be passed to the classifier function

This function should train the given classifier using the training data and then compute compute the classification accuracy on both the train and validation data. The return values should be a tuple where the first value is the training accuracy and the second value is the validation accuracy.

Implement classifier accuracy in the coding box below:

Available Functions: You have access to the NumPy python library as `np`, to `classify` which you have already implemented and to `accuracy` which we defined above.

```

    Returns: A tuple in which the first element is the (scalar) accuracy of the
    trained classifier on the training data and the second element is the
    accuracy of the trained classifier on the validation data.
    """
    theta, theta_0 = classifier(train_feature_matrix, train_labels, **kwargs)

    training_preds = classify(train_feature_matrix, theta, theta_0)
    validation_preds = classify(val_feature_matrix, theta, theta_0)

    training_accu = (training_preds == train_labels).mean()
    validation_accu = (validation_preds == val_labels).mean()

    return (training_accu, validation_accu)

```

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

Correct

```
def classifier_accuracy(
        classifier,
        train_feature_matrix,
        val_feature_matrix,
        train_labels,
        val_labels,
        **kwargs):
    """
    Trains a linear classifier and computes accuracy.
    The classifier is trained on the train data. The classifier's
    accuracy on the train and validation data is then returned.

    Args:
        classifier - A classifier function that takes arguments
            (feature matrix, labels, **kwargs) and returns (theta, theta_0)
        train_feature_matrix - A numpy matrix describing the training
            data. Each row represents a single data point.
        val_feature_matrix - A numpy matrix describing the training
            data. Each row represents a single data point.
        train_labels - A numpy array where the kth element of the array
            is the correct classification of the kth row of the training
            feature matrix.
        val_labels - A numpy array where the kth element of the array
            is the correct classification of the kth row of the validation
            feature matrix.
        **kwargs - Additional named arguments to pass to the classifier
            (e.g. T or L)

    Returns: A tuple in which the first element is the (scalar) accuracy of the
    trained classifier on the training data and the second element is the
    accuracy of the trained classifier on the validation data.
    """
    theta, theta_0 = classifier(train_feature_matrix, train_labels, **kwargs)
    train_predictions = classify(train_feature_matrix, theta, theta_0)
    val_predictions = classify(val_feature_matrix, theta, theta_0)
    train_accuracy = accuracy(train_predictions, train_labels)
    validation_accuracy = accuracy(val_predictions, val_labels)
    return (train_accuracy, validation_accuracy)
```

Test results

See full output

CORRECT

See full output

Solution:

See above for expected answer.

In this code, `**kwargs` stands for keyword-arguments. If you are not familiar with the `**` syntax, you can take a look at this tutorial.

Submit   You have used 1 of 20 attempts

Answers are displayed within the problem

Baseline Accuracy

3/3 points (graded)

Now, uncomment the relevant lines in **main.py** and report the training and validation accuracies of each algorithm with T = 10 and λ = 0.01 (the λ value only applies to Pegasos).

Please enter the **validation accuracy** of your Perceptron algorithm.

0.7160   ✔ **Answer:** 0.7160

Please enter the **validation accuracy** of your Average Perceptron algorithm.

0.7980

Answer: 0.7980

Please enter the **validation accuracy** of your Pegasos algorithm.

0.7900

Answer: 0.7900

Solution:

- The Perceptron validation accuracy should be 0.7160
- The Average Perceptron validation accuracy should be 0.7980
- The Pegasos validation accuracy should be 0.7900

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 / 7. Classification and Accuracy