

Assignment 3 Questions

Pascale Walters

Q1: In machine learning, what are bias and variance? When we evaluate a classifier, what are overfitting and underfitting, and how do these relate to bias and variance?

Bias refers to inherent error in the selected model. The average error between models trained over several training datasets and a true model that adequately represents the model is bias. Variance refers to the error between models trained on different datasets.

When evaluating a classifier, overfitting occurs when the model fits the training data too closely and does not generalize well to testing data. This is similar to having a high variance, because models trained on different training sets would give very different results, but low bias. Underfitting occurs when the selected model does not capture the training data well, which would result in high training and testing error. Low variance and high bias occur in underfitting.

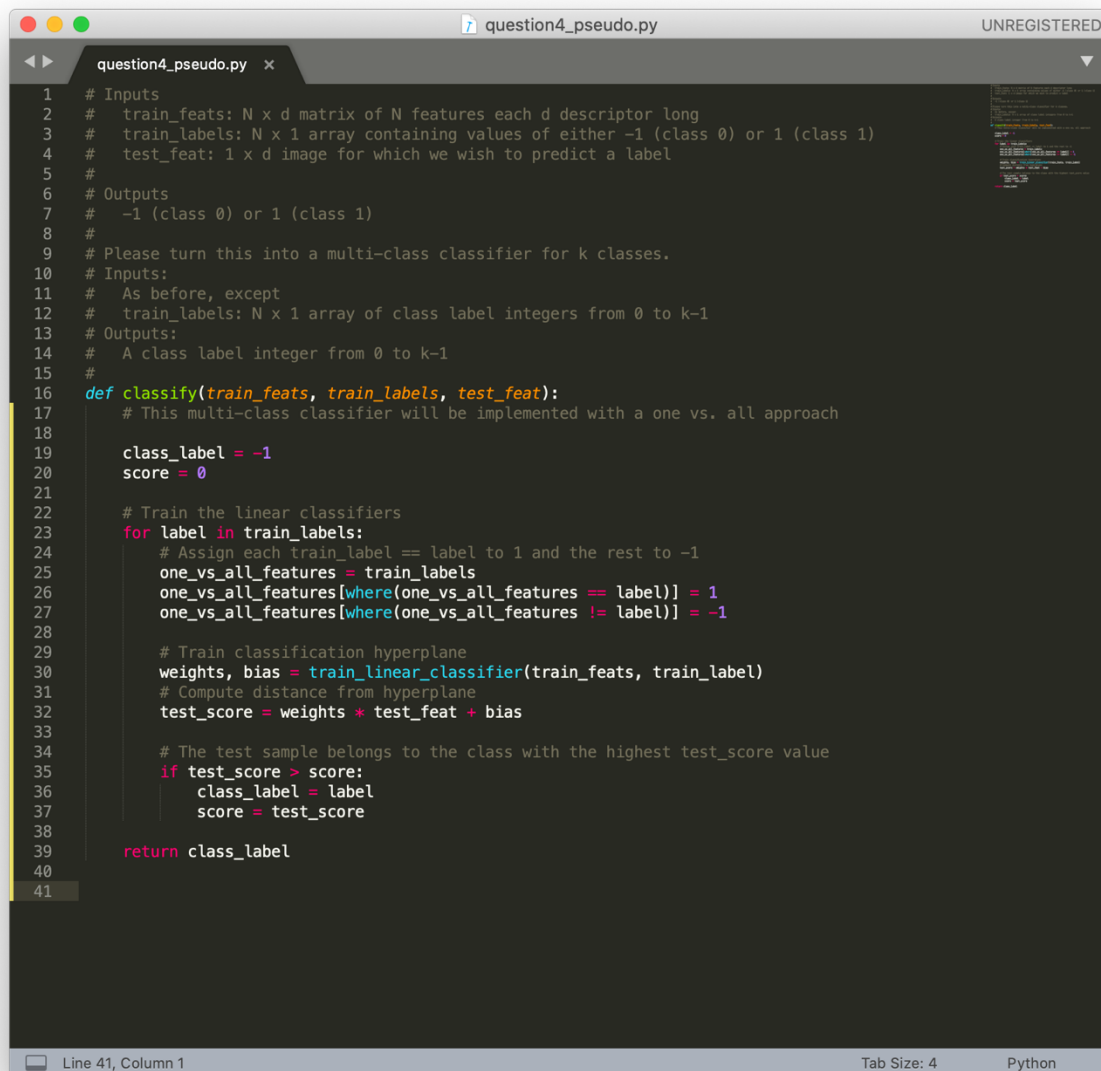
Q2: Given a linear classifier like SVM, how might we handle data that are not linearly separable? How does the kernel trick help in these cases? (See hidden slides in supervised learning crash course deck, plus your own research.)

There are several techniques that can be used when the data are not linearly separable. Hinge loss can be used to tolerate data points that are misclassified or on the boundary. Another technique would be to use the kernel trick. In this method, the data points are mapped to a higher dimensional space where the datapoints are linearly separable. This will give a decision boundary that is not linear in the original dimensionality of the training data.

Q3: Given a linear classifier such as SVM which separates two classes (binary decision), how might we use multiple linear classifiers to create a new classifier which separates k classes?

Below, we provide pseudocode for a linear classifier. It trains a model on a training set, and then classifies a new test example into one of two classes. Please convert this into a multi-class classifier. You can take either the one vs. all (or one vs. others) approach or the one vs. one approach in the slides; please declare which approach you take.

A classifier for multiple classes using a linear SVM can be handled in two different ways: one vs. all or one vs. one. In one vs. all, linear SVMs are learned to separate each class from the rest of the training samples. During testing, the sample is assigned to the class that returns the highest decision value. In one vs. one, linear SVMs are learned to separate all possible pairs of classes. During testing, samples are classified based on voting from all the pairwise comparisons. Pseudocode of a one vs. all approach is in the following figure:



```
1 # Inputs
2 # train_feats: N x d matrix of N features each d descriptor long
3 # train_labels: N x 1 array containing values of either -1 (class 0) or 1 (class 1)
4 # test_feat: 1 x d image for which we wish to predict a label
5 #
6 # Outputs
7 # -1 (class 0) or 1 (class 1)
8 #
9 # Please turn this into a multi-class classifier for k classes.
10 # Inputs:
11 # As before, except
12 # train_labels: N x 1 array of class label integers from 0 to k-1
13 # Outputs:
14 # A class label integer from 0 to k-1
15 #
16 def classify(train_feats, train_labels, test_feat):
17     # This multi-class classifier will be implemented with a one vs. all approach
18
19     class_label = -1
20     score = 0
21
22     # Train the linear classifiers
23     for label in train_labels:
24         # Assign each train_label == label to 1 and the rest to -1
25         one_vs_all_features = train_labels
26         one_vs_all_features[where(one_vs_all_features == label)] = 1
27         one_vs_all_features[where(one_vs_all_features != label)] = -1
28
29         # Train classification hyperplane
30         weights, bias = train_linear_classifier(train_feats, train_label)
31         # Compute distance from hyperplane
32         test_score = weights * test_feat + bias
33
34         # The test sample belongs to the class with the highest test_score value
35         if test_score > score:
36             class_label = label
37             score = test_score
38
39     return class_label
40
41
```

Q4: Suppose we are creating a visual word dictionary using SIFT and k-means clustering for a scene recognition algorithm. Examining the SIFT features generated from our training database, we see that many are almost equidistant from two or more visual words. Why might this affect classification accuracy?

Given the situation, describe two methods to improve classification accuracy, and explain why they would help.

If many SIFT features generated from the training data are each equidistant from several visual words in a dictionary, the visual words are likely a poor selection. Classification accuracy might be reduced because assignment of new samples to classes would be ambiguous, and therefore has a higher likelihood of being incorrect. Assignment to an incorrect class would be as equally likely as assignment to the correct class.

Assuming k-means clustering was used to generate the visual words, if the classes are too close together for good training and testing accuracy, the SIFT features may not be a good choice. If there is too much ambiguity when using SIFT features, another descriptor, such as histogram of oriented gradients, should be used. These features may be more distinctive and perform better for classification. Another method to improve classification would be to use another classification method, such as linear SVM and also use the kernel trick so that the data can become linearly separable. This new method may allow for better and less ambiguous classification in the feature space.

Q5: The way that the bag of words representation handles the spatial layout of visual information can be both an advantage and a disadvantage. Describe an example scenario for each of these cases, plus describe a modification or additional algorithm which can overcome the disadvantage. How might we evaluate whether bag of words is a good model?

The bag of words representation of an image does not include any spatial information about the locations of the features relative to each other. This can be an advantage as it means that the bag of words representation is robust to changes in orientation and provides a more compact summary of the features than if spatial information were included. A scenario for this advantage would be if the detection algorithm were running in real time or on a storage-limited device. The lack of spatial information in the bag of words representation is a disadvantage as the layout of words can be very important for recognition. For example, in a self-driving car application, the edges of a stop sign or a speed limit sign could be similar without the layout of the edges being considered.

An additional algorithm that can include spatial information in the bag of words model is a spatial pyramid. In this method, features are extracted at several levels of resolution and the location of the feature within the image is considered. Bag of words can be evaluated by determining its performance in situations with changes in illumination, orientation, occlusions, etc.