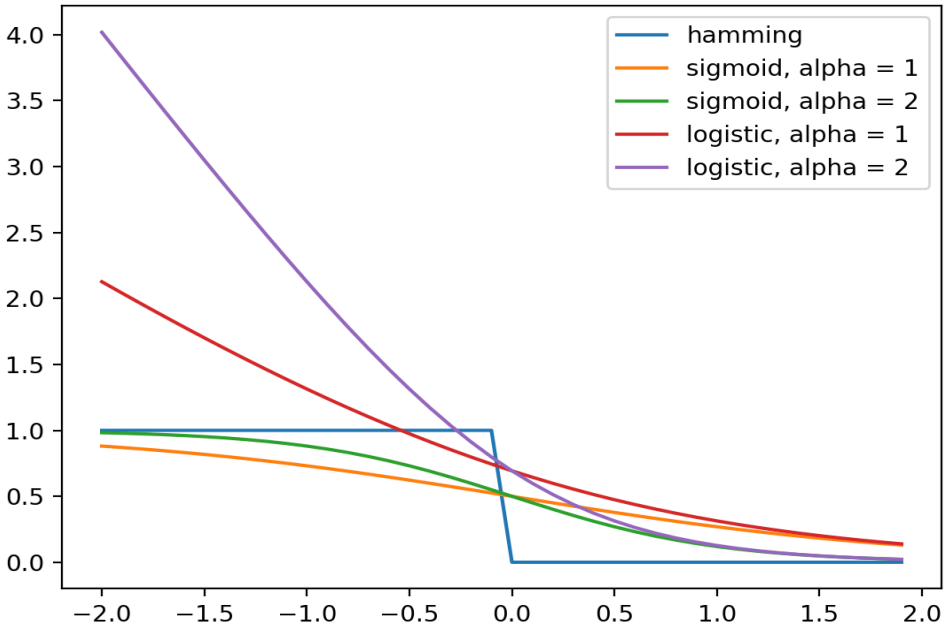
Machine Learning
Computer Science 158
Spring 2017
Problem Set 7
Due: March 21 at 11:59 PM

Name: Madison Hobbs and Shota Yasunaga

1 Feature Extraction [2 pts]

(a) We implemented `generate_output_codes`.

(b) Below is a plot of the three loss functions $g(z)$ for $z \in [-2, 2]$.



(c) We implemented `MulticlassSVM.fit(...)`.

(d) We implemented `MulticlassSVM.predict(...)`.

(e) i. Note the following table summarizing our results:

	hamming loss	sigmoid loss	logistic loss
one versus all	54	46	46
one versus one	41	48	42
random (R1)	38	38	33
random (R2)	34	35	34

ii. For each output code, how do the different loss functions affect the classification results, and why? (You may want to refer back to your plots of the various loss functions.) You do not have to comment on every combination of output code and classifier. Pick the cases that stand out to you, and write a short paragraph (about 5 sentences).

For OVA output code, hamming loss is worst, while sigmoid and logistic losses give equally lower error. Hamming loss says: I don't care much about the magnitude of the error, just whether there's an error or not. However, sigmoid and logisitc loss both penalize more eggregious errors more. The difference between sigmoid and logistic loss is that with logistic loss, we penalize worse errors increasingly more (absolute value of slope increases) whereas with sigmoid loss, we penalize worse errors more but at a decreasing rate (absolute value of slope decreases). We might expect, therefore, for logistic loss to yield a classifier which lets the fewest errors through (lowest error rate). We see that for all the output codes, logistic loss yields as good or better results than the other two loss functions. Only with OVO does hamming loss produce 1 fewer errors than logisitic loss. This could be an artifact of the dataset, and in essense, hamming and logistic losses tie here. It's interesting how hamming loss and logistic loss outperform sigmoid loss in OVO while in all other cases sigmoid and logistic outperform hamming loss (or essentially tie with it).

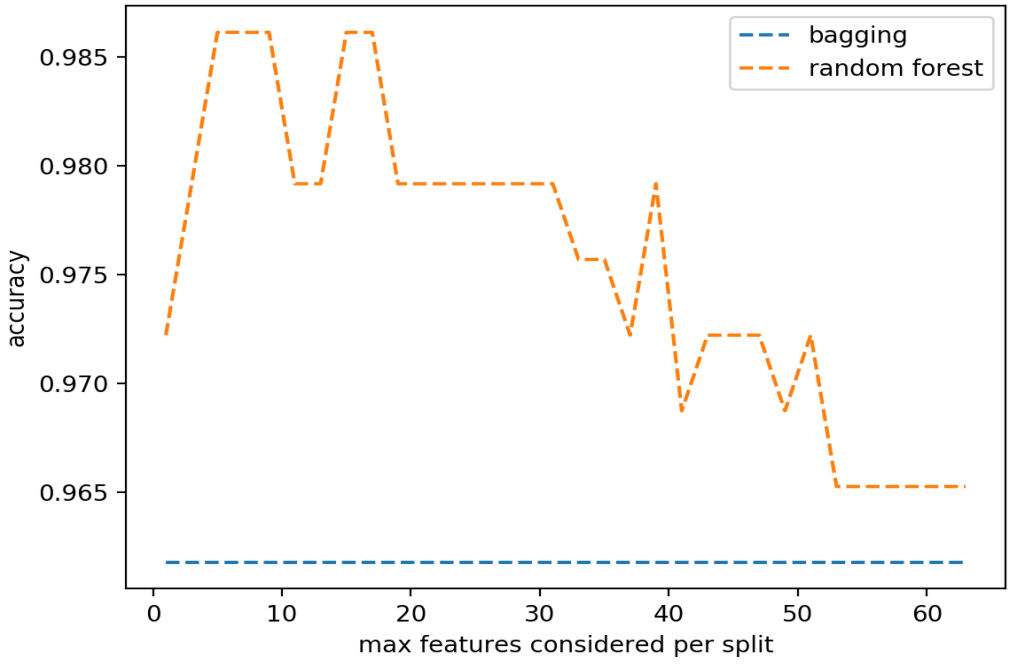
iii. What kind of output code do you think is most suitable for this problem? Why do you think this is the case? Write a short paragraph (about 5 sentences).

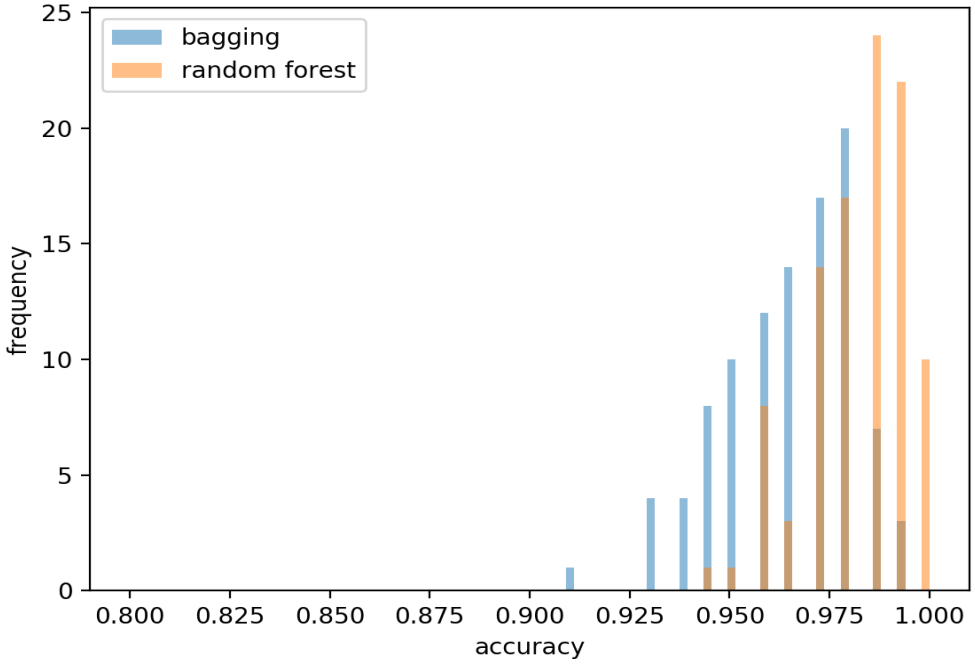
Both random output codes yielded the lowest error, so it seems a random output code is most suitable for this problem. Perhaps this is because OVO and OVA are too systematic in checking each class against the other. Maybe the soybean diseases we are predicting are correlated and a lot of the classifiers produced using OVO or OVA are meaningless or fitting to noise. Random output codes may fix this (although it's possible randomness could yield worse classifiers.) Perhaps random output codes homogenize comparisons.

2 Digit Recognition using Bagging [5+3 pts]

In this problem, we will study bagging and random forest on a handwritten digit dataset. This dataset contains 1797 samples (each having 64 features) of hand-written digits, classified into 10 classes.4 However, for this problem, we will only use 720 examples from 4 classes.

(a) **(3 pts)** Run `main(...)` to compare the performance of these ensemble classifiers using 100 random splits of the digits dataset into training and test sets, where the test set contains roughly 20% of the data. This generates two plots : We vary the number of features considered for the random forest from 1 to 65 (in step sizes of 2). Then, on a single plot, we plot the accuracy of the bagging classifier against that of the random forest with varying number of features. (Note: This code may take several minutes to run.)





Include these plots in your writeup. How does performance vary with the number of features considered per split? Which classifier performs better, and why?

Solution:

Random Forest performance varies with the number of features considered at each split, while the bagged trees remains constant. The Random Forest consistently performed better (higher accuracy) than bagged trees. This is probably because the Random Forest has decorrelated trees (each trained using a different subset of features) whereas bagging pulls from the same pool of features in each tree, making them essentially the same/correlated.

(b) **(2 pts)** One useful application of bagging ensembles is to assess the relative importance of features. In your own words, describe how you think feature importance might be calculated when using Decision Trees as your base classifier.

Now, using scikit-learns examples as a guideline, generate a representation of the relative importance of each individual pixel for digit recognition (include this plot in your writeup). For this problem, you are allowed to use `sklearn.ensemble.RandomForestClassifier`.

What do you observe? Is this surprising?

Solution:
We can add up over all the trees how much each feature, when used to make a split, increases information gain. This can measure the importance of that feature by effectively measuring how much it contributes to information gain (how important it is in terms of gaining information about predicting the labels).
Below, we see a heat map of the feature importance for this problem. We notice there is just one single pixel in the center that is vastly important, while many pixels (on the periphery) are not that important. It makes sense that pixels on the periphery are not as important (they may not even contain the digit) while pixels towards the center are more important. It is a bit shocking however just how important that one pixel in the center is.