learning subjective matter from texture: a music genre classification experiment

Noah Murad
CS4437 Introduction to Data Science, Western University

Abstract

We present a novel solution to classifying music by genre based on audio features. We started with a dataset of one million song samples containing metadata features and audio analysis features. We cleansed the data by only using samples with genre tags which do not overlap. We primarily used timbre data as input to train our model. We trained an SVC with an RBF kernel to classify a ten-genre experiment and a six-genre experiment. Our SVC had an accuracy of 59% on the ten-genre experiment and an accuracy of 68% on the six-genre experiment. The subjectivity of genre yielded some inconclusiveness of the source of misclassification.

The Problem

Every musical piece can be classified as many different genres. Identifying the genre of a song or artist may be a trivial matter for a human, but this topic has proven to be interesting for a machine. It is apparent that some genres are more easily identifiable than others. Most studies report the highest accuracy in identifying classical music[4] over more modern genres. Machine learning takes an interesting role when teaching a computer to identify what is ultimately a subjective matter. The goal of our experiments is to train a classifier to be as accurate as possible in classifying the genre of music.

The Dataset

We used the Million Song Dataset (MSD) for our experiments. "The MSD is a freely-available collection of audio features and metadata for a million contemporary popular music tracks"[5]. This dataset is presented by a collaboration between LabROSA and The Echo Nest. The dataset only contains textual data and is 300 GB in size. Being such a large dataset makes it expensive to distribute, which makes it difficult to acquire. We accessed the database through an Amazon Web Services EC2 instance. The MSD also offers a 1% subset of the entire dataset which contains 10,000 songs. We used this subset to test techniques more quickly.

The MSD's features can be classified as metadata features and audio analysis features. Metadata features consist of song title, artist, album, release year, popularity, related artists, and artist tags. Artist tags are social descriptors which are mapped to the artist of the song and generally include music genres, instruments, and demographic tags.

The audio analysis features provide descriptions of the shape and sound of the song. There are two types of audio analysis features: summaries, and temporal features. Audio analysis summaries include duration, tempo, time signature, key, musical mode, and loudness. The temporal features break down each song into layers of sections (chorus, verse, ...), bars, beats, and segments. Segments are the smallest temporal unit and are described as a musical event such as a note or onset. Along with temporal information, each segment has additional data about its loudness, pitches, and timbre (musical colour/texture). These features make up the core of the audio analysis features available.

The timbre and pitches features are available as 12-dimensional vectors for each segment. Each scalar in the pitch vector represents the presence of each musical note at any octave. Each scalar in the timbre vector represents the presence of a timbre constant (Figure 1) for its corresponding segment. Timbre is the quality of a musical sound (or set of musical sounds) which makes it sound distinct from it's pitch or intensity. For example, timbre is the reason why two different instruments playing the same note sound different. This is the reason why we use timbre as our primary feature – it is a descriptor of the mood, colour, and texture of the sound.

We use a combination of metadata and audio analysis features in our experiments. Just as a human identifies the era or genre of a song by listening to it, we classify metadata based on audio features.

Pros and cons. There are no actual audio files, all of the audio analysis is already provided. This is good for our project since we are focusing on music classification and not music analysis. The dataset is very large so you can throw away lots of it while cleansing and still have a lot to work with.

Preprocessing and Exploration

Each instance of the dataset is stored in an .h5 file in directory tree structure with a depth of 4, and branching factor of 26. Because of the large size of the dataset and hardware limitations, we pull, process, and aggregate all relevant data out of the dataset, into one .csv file. Each line contains data for one sample. We store genre as the first element for easy access, followed by any other metadata, followed by audio analysis features. Gathering all of the audio features (timbre, pitch) for each segment is not necessary. Because there are a variable number of segments for each sample, we cannot use the raw audio features; we must summarize this data to have a constant number of features per sample. LabROSA and The Echo Nest suggest that using the averages and covariances of these features is sufficient[1].

The genre tags of the MSD have large overlaps because of genre ambiguity and the fact that genres are tagged to artists and not songs. To illustrate, by looking at Table 1 we can calculate that only 390 of 3885 artists do not share the "pop" and "rock" tags. Our initial plan to overcome this problem was to systematically classify each artist to one of the ten most general genres. This, however, would poison our inputs since an artist can produce multiple genres of music, and genre is mapped to artist and not to song. Since the dataset is so large, the best solution is to discard any song with overlapping genres. This technique yielded 4,000 songs per genre for the ten genre case, and 10,000 songs per genre for the six genre case.

For our base classifier, we used SVC for speed and because this is a supervised learning problem. To select hyperparameters, we performed a grid search on K-fold cross validation for the kernel, cost, and gamma parameters on the subset (to speed things up). We tested on linear, polynomial, and RBF kernels. The best combinations of hyperparameters for all kernels yielded 53% accuracy with a low standard deviation (Output 1). The polynomial kernel displayed high sensitivity to the hyperparameters. The linear kernel had unusually high processing times for large cost values. Because of these inconsistent behaviours, we chose to work with an RBF kernel. After further experimentation with the RBF kernel, our final hyperparameters were C=5,000 and gamma=1/n_features.

Some of the MSD's metadata can help classify genres. We explored song year, duration, and tempo by creating box plots and an SVC to analyze distinguishable characteristics (Figure 2). These features can make for slightly better classification, however, 63% of the songs do not have release years defined. Using year as a feature would limit our sample space too much. Since tempo and duration is defined for every sample, we will be using these features.

In order to get detailed descriptions of each song, we need to analyze at the smallest distinguishable time-step – segments. Among the per-segment audio analysis features to choose from, there are loudness, pitches, and timbre. Loudness is a one dimensional feature which does not provide a rich feature space. Of timbre and pitches, timbre represents the mood of the piece, whereas pitches represents specific notes played. Some genres may use certain combinations of notes more than others, but intuitively, timbre, giving us a musical texture, should be more closely related to genre. Regardless, we trained classifiers with only timbre, only pitches, and both. From the MSD subset, we pulled 212 songs for 6 distinct genres. The classifiers with only timbre and with both pitches and timbre tied in accuracy, beating the classifier with only pitches (Figure 3).

As a secondary project, we also trained classifiers to learn the year of a song. For this, we used timbre averages and timbre covariances, and an SVR since songs in adjacent years are expected to be similar. Less experimentation has been exhausted into this project but we still present preliminary results in Figure 5.

System Architecture and Results

Our final system architecture is an SVC with an RBF kernel, cost of 5,000, and gamma of 1/n_features. We did not perform cross validation in the final training because of the amount of data we have as input. All features are normalized before training and we input the same number of samples per class to reduce bias. We trained our classifier for two different scenarios: 6-genre classification, and 10-genre classification. Our 6-genre classification has 5000 samples per genre, 500 of which are used for testing, and our 10-genre classification has 3500 samples per genres, 350 of which are used for testing. The results are 59% accuracy for the 10-genre experiment and 67% accuracy for the 6-genre experiment (Figure 4).

Analysis

Looking at the confusion matrix, it is apparent that pop music is the hardest to classify. From experience listening to music and identifying genre, the term "pop" is often applied to any popular music, and was used interchangeably with rock music in the past. The misclassifications in pop either stem from improper tagging of "pop" on the artist tags in the MSD and from the fact that the pop genre is the most vague and general genre, which usually overlaps with other big genres such as rock, folk, R&B, and country. To be fair, all of these genres do include many of the same instrumental setups yielding similar musical textures (timbres). This causes multi-way misclassifications between these genres, lowering all of their accuracies, but lowering pop's accuracy the most, being the most general of the genres. Another interesting and expected misclassification are rock and metal. This occurs because metal is a derivative of rock.

The genres with the highest performance are hip hop, electronic, and metal. Hip hop probably had the best performance because of it's unique timbre formed by rapping, which isn't a major part of any other genres in the experiment. We were originally going to include "rap" as a separate genre, however, the “rap” tag had too much overlap with the “hip hop” tag to keep a significant number of samples. This is similar to how other studies reported highest accuracy with classical music[4]. Classical music has a very distinguishable musical texture from most modern music.

One underlying issue with this experiment is that the genre tags are linked to artists, not to individual songs. Our data cleansing techniques have seemed to masked out this problem, but this has limited our music selection. The limits imposed on our music selection may have simplified the classification problem.

Our year classification has an unexpectedly high accuracy of 13.75%, but the MSE is also quite high. Year is difficult to predict as many genres (timbres) of music are created every year.

Classifying a subjective matter can be limited by choice of label, so it is difficult to make a definite statement of how well our classifier performed. We believe that most of the misclassifications are caused by the subjectivity of genre but do believe that our classifier can be further developed.

Going further

Let us present ideas of how these experiments can be extended by making broader use of the MSD. Along with segment data, using more temporal features such as section, bar, and beat features, we can generate more rich features. The simplest example of this extension is to calculate timbre averages and timbre covariances by beat number per bar. This may raise some issues because not every song has the same time signature (number of beats per bar) and some songs even have variable time signatures. Although they are less common, variable time signatures would be highly difficult to detect with only MSD features.

Conclusion

The field of music information retrieval has provided us with an interesting introduction to data science. Working with subjective labels does impose limitations on experiments. The ability to detect subjective classes with high accuracy using a relatively simple model displays both the power of machine learning and the importance of timbre in music information retrieval.

References

[1] T. Bertin-Mahieux, D. Ellis, B. Whitman, P. Lamere. The Million Song Dataset. *International Society for Music Information Retrieval Conference 2011*. http://ismir2011.ismir.net/papers/OS6-1.pdf.

[2] Tripp C., Hung H., Pontikakis M. “Waveform-Based Musical Genre Classification” http://cs229.stanford.edu/proj2011/HaggbladeHongKao-MusicGenreClassification.pdf

[3] Liang D., Haijie G., O’Connor B. “Music Genre Classification with the Million Song Dataset” http://www.ee.columbia.edu/~dliang/files/FINAL.pdf

[4] Haggblade M., Hong Y., Kao K. “Music genre classification” *http://cs229.stanford.edu/proj2006/TrippHungPontikakisWaveformBasedMusicalGenreClassification.pdf*

[5] LabROSA. “Million Song Dataset” *http://labrosa.ee.columbia.edu/millionsong/*

Tables, Figures, and Outputs

N Most Common Tags	**Artist Coverage**
1	2346
2	2736
3	3049
4	3202
5	3287
10	3489
100	3751
3502 (all)	3885

Table 1: The number of artists covered by the n most commonly used tags in the subset of 10,000 songs. For example, "rock," the most common tag, covers 2346 artists. "Rock" and/or "pop," the two most common tags, appear in 2736 artists.

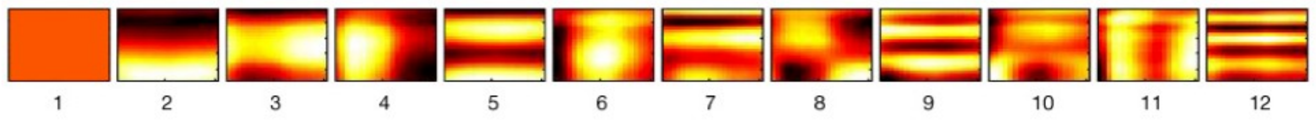

Figure 1: Timbre constants used by The Echo Nest's analyzer to represent timbre as a 12-dimensional vector. Multiplying the timbre vector by these constants should produce a sound with a similar mood to the analyzed segment.

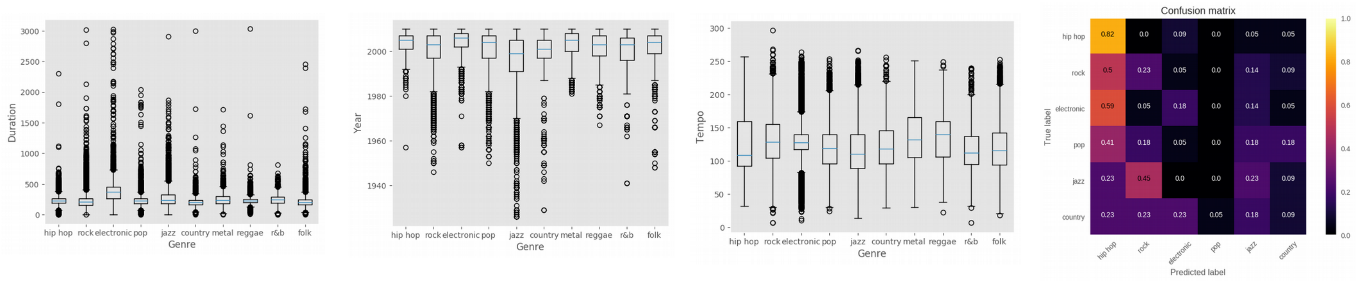

Figure 2: Distributions of duration, year, and tempo per genre and results of a SVC classifier with with an RBF kernel using only duration, year, and tempo as input features on 212 instances per genre. The accuracy for this classifier is 25.76%, which is better than, but a little close to 16.66%

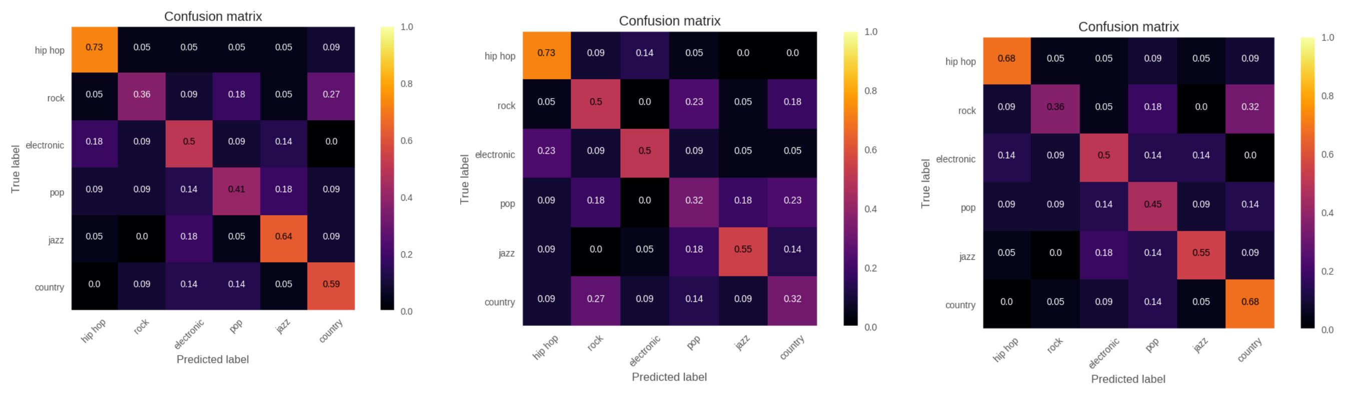

Figure 3: Confusion matrices for SVC classifiers with an RBF kernel using (from left to right) only pitches, only timbre, and both pitches and timbre as input features on 212 instances per genre. Accuracy is (from left to right) 48.48%, 53.79%, 53.79%.

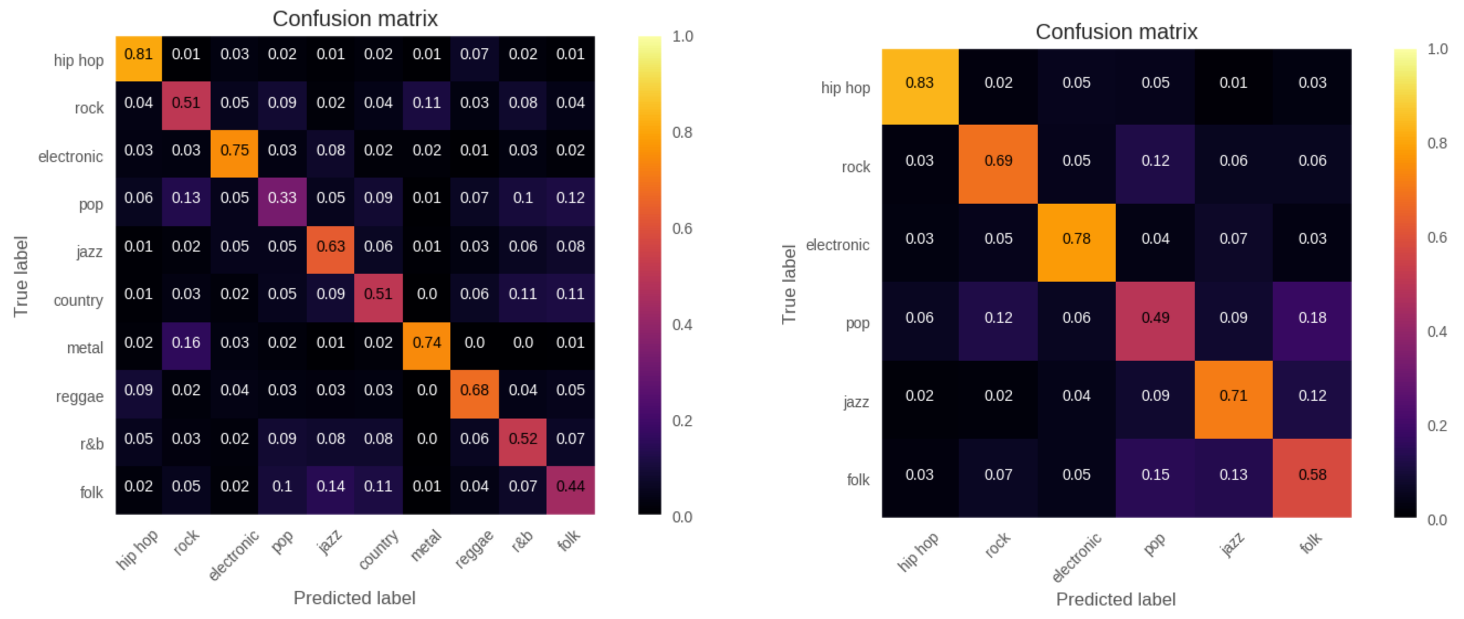

Figure 4: Final confusion matrices for 10-genre and 6-genre experiments.
Accuracy is 59.06% for 10-genre and 68.14% for 6-genre.
SVC classifiers trained with RBF kernel, C=5000, and gamma=1/n_features=1/92.

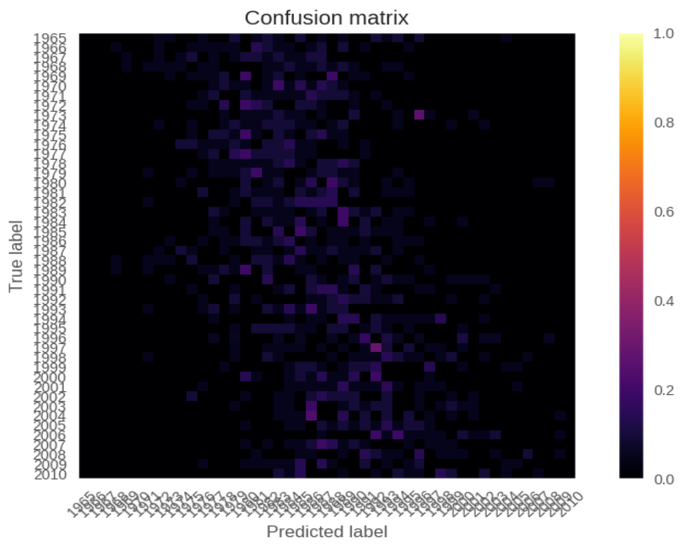

Figure 5: Year confusion matrix. Accuracy of 13.75%, MSE of 79.44. Trained on an SVR with cost of 20,000.

Output 1: Grid search on 5-fold cross validation for the kernel, cost, and gamma parameters on the subset of 212 samples per genre.

```
Accuracy: 0.33 (+/- 0.00) for rbf c=0.001000 gamma=1.000000 in 3.38 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.001000 gamma=0.100000 in 3.24 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.001000 gamma=0.001000 in 3.22 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.001000 gamma=0.000100 in 3.24 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.001000 gamma=0.000010 in 3.30 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.010000 gamma=1.000000 in 3.54 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.010000 gamma=0.100000 in 3.41 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.010000 gamma=0.001000 in 3.24 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.010000 gamma=0.000100 in 3.23 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.010000 gamma=0.000010 in 3.24 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.100000 gamma=1.000000 in 3.49 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.100000 gamma=0.100000 in 3.49 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.100000 gamma=0.001000 in 3.25 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.100000 gamma=0.000100 in 3.23 s
Accuracy: 0.33 (+/- 0.00) for rbf c=0.100000 gamma=0.000010 in 3.22 s
Accuracy: 0.48 (+/- 0.05) for rbf c=1.000000 gamma=1.000000 in 3.09 s
Accuracy: 0.33 (+/- 0.00) for rbf c=1.000000 gamma=0.100000 in 3.46 s
Accuracy: 0.33 (+/- 0.00) for rbf c=1.000000 gamma=0.001000 in 3.39 s
Accuracy: 0.33 (+/- 0.00) for rbf c=1.000000 gamma=0.000100 in 3.23 s
Accuracy: 0.33 (+/- 0.00) for rbf c=1.000000 gamma=0.000010 in 3.22 s
Accuracy: 0.53 (+/- 0.03) for rbf c=10.000000 gamma=1.000000 in 2.84 s
Accuracy: 0.48 (+/- 0.06) for rbf c=10.000000 gamma=0.100000 in 3.06 s
Accuracy: 0.33 (+/- 0.00) for rbf c=10.000000 gamma=0.001000 in 3.50 s
Accuracy: 0.33 (+/- 0.00) for rbf c=10.000000 gamma=0.000100 in 3.41 s
Accuracy: 0.33 (+/- 0.00) for rbf c=10.000000 gamma=0.000010 in 3.24 s
Accuracy: 0.51 (+/- 0.03) for rbf c=100.000000 gamma=1.000000 in 3.78 s
Accuracy: 0.52 (+/- 0.06) for rbf c=100.000000 gamma=0.100000 in 3.01 s
Accuracy: 0.33 (+/- 0.00) for rbf c=100.000000 gamma=0.001000 in 3.54 s
Accuracy: 0.33 (+/- 0.00) for rbf c=100.000000 gamma=0.000100 in 3.53 s
Accuracy: 0.33 (+/- 0.00) for rbf c=100.000000 gamma=0.000010 in 3.64 s
Accuracy: 0.48 (+/- 0.05) for rbf c=1000.000000 gamma=1.000000 in 4.77 s
Accuracy: 0.53 (+/- 0.06) for rbf c=1000.000000 gamma=0.100000 in 4.09 s
Accuracy: 0.48 (+/- 0.05) for rbf c=1000.000000 gamma=0.001000 in 3.16 s
Accuracy: 0.33 (+/- 0.00) for rbf c=1000.000000 gamma=0.000100 in 3.53 s
Accuracy: 0.33 (+/- 0.00) for rbf c=1000.000000 gamma=0.000010 in 3.49 s
Accuracy: 0.48 (+/- 0.05) for rbf c=10000.000000 gamma=1.000000 in 4.60 s
Accuracy: 0.49 (+/- 0.02) for rbf c=10000.000000 gamma=0.100000 in 11.24 s
Accuracy: 0.51 (+/- 0.04) for rbf c=10000.000000 gamma=0.001000 in 2.96 s
Accuracy: 0.49 (+/- 0.05) for rbf c=10000.000000 gamma=0.000100 in 3.06 s
Accuracy: 0.33 (+/- 0.00) for rbf c=10000.000000 gamma=0.000010 in 3.46 s
Accuracy: 0.48 (+/- 0.05) for rbf c=100000.000000 gamma=1.000000 in 4.67 s
Accuracy: 0.48 (+/- 0.04) for rbf c=100000.000000 gamma=0.100000 in 16.42 s
Accuracy: 0.53 (+/- 0.07) for rbf c=100000.000000 gamma=0.001000 in 4.60 s
Accuracy: 0.51 (+/- 0.04) for rbf c=100000.000000 gamma=0.000100 in 2.83 s
Accuracy: 0.49 (+/- 0.05) for rbf c=100000.000000 gamma=0.000010 in 3.05 s

Accuracy: 0.33 (+/- 0.00) for poly c=0.001000 gamma=1.000000 in 2.41 s
[32/591]
Accuracy: 0.33 (+/- 0.00) for poly c=0.001000 gamma=0.100000 in 2.56 s
Accuracy: 0.33 (+/- 0.00) for poly c=0.001000 gamma=0.001000 in 2.32 s
Accuracy: 0.33 (+/- 0.00) for poly c=0.001000 gamma=0.000100 in 2.30 s
Accuracy: 0.33 (+/- 0.00) for poly c=0.001000 gamma=0.000010 in 2.59 s
Accuracy: 0.33 (+/- 0.00) for poly c=0.010000 gamma=1.000000 in 2.77 s
Accuracy: 0.33 (+/- 0.00) for poly c=0.010000 gamma=0.100000 in 2.32 s
Accuracy: 0.33 (+/- 0.00) for poly c=0.010000 gamma=0.001000 in 2.31 s
Accuracy: 0.33 (+/- 0.00) for poly c=0.010000 gamma=0.000100 in 2.37 s
Accuracy: 0.33 (+/- 0.00) for poly c=0.010000 gamma=0.000010 in 2.30 s
```

```
Accuracy: 0.33 (+/- 0.00) for poly c=0.100000 gamma=1.000000 in 2.48 s
Accuracy: 0.33 (+/- 0.00) for poly c=0.100000 gamma=0.100000 in 2.34 s
Accuracy: 0.33 (+/- 0.00) for poly c=0.100000 gamma=0.001000 in 2.31 s
Accuracy: 0.33 (+/- 0.00) for poly c=0.100000 gamma=0.000100 in 2.30 s
Accuracy: 0.33 (+/- 0.00) for poly c=0.100000 gamma=0.000010 in 2.30 s
Accuracy: 0.51 (+/- 0.04) for poly c=1.000000 gamma=1.000000 in 2.21 s
Accuracy: 0.33 (+/- 0.00) for poly c=1.000000 gamma=0.100000 in 2.74 s
Accuracy: 0.33 (+/- 0.00) for poly c=1.000000 gamma=0.001000 in 2.29 s
Accuracy: 0.33 (+/- 0.00) for poly c=1.000000 gamma=0.000100 in 2.39 s
Accuracy: 0.33 (+/- 0.00) for poly c=1.000000 gamma=0.000010 in 2.58 s
Accuracy: 0.52 (+/- 0.03) for poly c=10.000000 gamma=1.000000 in 2.12 s
Accuracy: 0.33 (+/- 0.00) for poly c=10.000000 gamma=0.100000 in 2.50 s
Accuracy: 0.33 (+/- 0.00) for poly c=10.000000 gamma=0.001000 in 2.29 s
Accuracy: 0.33 (+/- 0.00) for poly c=10.000000 gamma=0.000100 in 2.32 s
Accuracy: 0.33 (+/- 0.00) for poly c=10.000000 gamma=0.000010 in 2.31 s
Accuracy: 0.49 (+/- 0.04) for poly c=100.000000 gamma=1.000000 in 3.24 s
Accuracy: 0.33 (+/- 0.00) for poly c=100.000000 gamma=0.100000 in 2.69 s
Accuracy: 0.33 (+/- 0.00) for poly c=100.000000 gamma=0.001000 in 2.29 s
Accuracy: 0.33 (+/- 0.00) for poly c=100.000000 gamma=0.000100 in 2.31 s
Accuracy: 0.33 (+/- 0.00) for poly c=100.000000 gamma=0.000010 in 2.41 s
Accuracy: 0.46 (+/- 0.03) for poly c=1000.000000 gamma=1.000000 in 3.84 s
Accuracy: 0.51 (+/- 0.04) for poly c=1000.000000 gamma=0.100000 in 2.15 s
Accuracy: 0.33 (+/- 0.00) for poly c=1000.000000 gamma=0.001000 in 2.29 s
Accuracy: 0.33 (+/- 0.00) for poly c=1000.000000 gamma=0.000100 in 2.30 s
Accuracy: 0.33 (+/- 0.00) for poly c=1000.000000 gamma=0.000010 in 2.31 s
Accuracy: 0.46 (+/- 0.03) for poly c=10000.000000 gamma=1.000000 in 3.77 s
Accuracy: 0.52 (+/- 0.03) for poly c=10000.000000 gamma=0.100000 in 2.11 s
Accuracy: 0.33 (+/- 0.00) for poly c=10000.000000 gamma=0.001000 in 2.30 s
Accuracy: 0.33 (+/- 0.00) for poly c=10000.000000 gamma=0.000100 in 2.29 s
Accuracy: 0.33 (+/- 0.00) for poly c=10000.000000 gamma=0.000010 in 2.43 s
Accuracy: 0.46 (+/- 0.03) for poly c=100000.000000 gamma=1.000000 in 3.94 s
Accuracy: 0.49 (+/- 0.04) for poly c=100000.000000 gamma=0.100000 in 3.22 s
Accuracy: 0.33 (+/- 0.00) for poly c=100000.000000 gamma=0.001000 in 2.45 s
Accuracy: 0.33 (+/- 0.00) for poly c=100000.000000 gamma=0.000100 in 2.29 s
Accuracy: 0.33 (+/- 0.00) for poly c=100000.000000 gamma=0.000010 in 2.29 s

Accuracy: 0.33 (+/- 0.00) for linear c=0.001000 gamma=1.000000 in 2.29 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.001000 gamma=0.100000 in 2.28 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.001000 gamma=0.001000 in 2.27 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.001000 gamma=0.000100 in 2.27 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.001000 gamma=0.000010 in 2.27 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.010000 gamma=1.000000 in 2.37 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.010000 gamma=0.100000 in 2.37 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.010000 gamma=0.001000 in 2.41 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.010000 gamma=0.000100 in 2.36 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.010000 gamma=0.000010 in 2.36 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.100000 gamma=1.000000 in 2.38 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.100000 gamma=0.100000 in 2.43 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.100000 gamma=0.001000 in 2.38 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.100000 gamma=0.000100 in 2.38 s
Accuracy: 0.33 (+/- 0.00) for linear c=0.100000 gamma=0.000010 in 2.38 s
Accuracy: 0.46 (+/- 0.05) for linear c=1.000000 gamma=1.000000 in 2.18 s
Accuracy: 0.46 (+/- 0.05) for linear c=1.000000 gamma=0.100000 in 2.19 s
Accuracy: 0.46 (+/- 0.05) for linear c=1.000000 gamma=0.001000 in 2.19 s
Accuracy: 0.46 (+/- 0.05) for linear c=1.000000 gamma=0.000100 in 2.40 s
Accuracy: 0.46 (+/- 0.05) for linear c=1.000000 gamma=0.000010 in 2.17 s
Accuracy: 0.52 (+/- 0.04) for linear c=10.000000 gamma=1.000000 in 1.96 s
Accuracy: 0.52 (+/- 0.04) for linear c=10.000000 gamma=0.100000 in 1.97 s
Accuracy: 0.52 (+/- 0.04) for linear c=10.000000 gamma=0.001000 in 1.97 s
```

```
Accuracy: 0.52 (+/- 0.04) for linear c=10.000000 gamma=0.000100 in 1.97 s
Accuracy: 0.52 (+/- 0.04) for linear c=10.000000 gamma=0.000010 in 1.97 s
Accuracy: 0.52 (+/- 0.04) for linear c=100.000000 gamma=1.000000 in 2.65 s
Accuracy: 0.52 (+/- 0.04) for linear c=100.000000 gamma=0.100000 in 2.65 s
Accuracy: 0.52 (+/- 0.04) for linear c=100.000000 gamma=0.001000 in 2.65 s
Accuracy: 0.52 (+/- 0.04) for linear c=100.000000 gamma=0.000100 in 2.65 s
Accuracy: 0.52 (+/- 0.04) for linear c=100.000000 gamma=0.000010 in 2.65 s
Accuracy: 0.51 (+/- 0.05) for linear c=1000.000000 gamma=1.000000 in 11.28 s
Accuracy: 0.51 (+/- 0.05) for linear c=1000.000000 gamma=0.100000 in 11.40 s
Accuracy: 0.51 (+/- 0.05) for linear c=1000.000000 gamma=0.001000 in 11.51 s
Accuracy: 0.51 (+/- 0.05) for linear c=1000.000000 gamma=0.000100 in 11.38 s
Accuracy: 0.51 (+/- 0.05) for linear c=1000.000000 gamma=0.000010 in 11.79 s
Accuracy: 0.50 (+/- 0.04) for linear c=10000.000000 gamma=1.000000 in 122.10 s
Terminated (processing time too long for subset)
```