Sound object labeling

EECS 352 Machine perception of Music and Audio
Bongjun Kim
Winter, 2019
Sound object labeling

Dog barking
Goal

• Building a system that automatically labels an audio event

An array of real values

Dog barking
Tasks

• Audio classification

![Audio classification diagram]

• Sound Event Detection (SED)

![Sound Event Detection (SED) diagram]
MACHINE LEARNING: CLASSIFICATION
Supervised learning from data

Function we want to learn (Target function)

\[ Y = f(X) \]

Find a hypothesis function \( h \) such that \( h(X) \approx f(X) \)

- On the training data \( D = \{ <x_1, y_1>, ..., <x_n, y_n> \} \)
Supervised Learning

• Regression
  – A target function maps $X$ onto continuous real values $Y$.

• Classification
  – A target function maps $X$ onto discrete class labels $Y$. 
Overview of general classification tasks

Input data → Feature representation → Classifier → Label

A vector of numbers
\( \tilde{x} = <a_1, a_2, ..., a_n> \)

...that represent attributes of the example, like fundamental frequency, or amplitude.

- Decision Tree
- Nearest Neighbor
- Neural Networks

“Cat image”
“Piano sound”
Overview of general classification tasks

• Example: Classifying a customer to **Good** or **Bad**

Input data → Feature representation → Classifier → Label

Customer

\( \hat{x} = <a_1, a_2> \)

\( a_1 \): # of accounts  
\( a_2 \): salary

**Feature space**

Good  
Bad

\( a_1 \)  
\( a_2 \)
Different Classifiers

• Different classifications need different classifiers.

Bryan Pardo, EECS 352 Spring 2012
Feature selection is important

- How things cluster depend on what you are measuring.
Which of these go together?
Which of these go together?

Length of legs

Body size
Which of these go together?

Furry

# of legs
Nearest Neighbor (NN) Classifier

• When you see a new instance $x$ to classify, find the most similar training example and assign its label to the instance.

• How do you tell what things are similar?
  1. Extract proper features.
  2. Measure distance / similarity in the feature space.
Nearest Neighbor (NN) Classifier

A new instance to classify

The nearest neighbor

\( \mathbf{x} \) is classified into class-1
Nearest Neighbor (NN) Classifier

The nearest neighbor

A new instance to classify

The nearest neighbor

X is classified into class-2
Nearest Neighbor (NN) Classifier

The decision boundary
How do we measure distance?

• Euclidian distance
  – what people intuitively think of as “distance”

\[ d(A, B) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2} \]
**L^p norms**

- L^p norms are all special cases of this function:

\[
        d(\vec{x}, \vec{y}) = \left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}
\]

- L^1 norms = Manhattan Distance: \( p=1 \)

- L^2 norms = Euclidean Distance: \( p=2 \)
Cosine Similarity

• Measure of similarity between two vectors
  – Range from -1 (opposite) to 1 (same)
  – Cosine distance = 1 – cosine similarity

• Cosine similarity between vector $A$ and $B$:

$$sim(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

$$A \cdot B = \sum_{i=1}^{n} A_i B_i$$
$$\|A\|\|B\| = \sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}$$
Feature Scaling

- Different scales of features can mislead distance measure.

  E.g., Measuring distance between humans
  - Feature 1: Height (0-7 feet)
  - Feature 2: weight (0-150 kg)

  In this Euclidean space, the second feature dominates the distance, which might lead to mis-clustering.

  Scaling each feature such that it ranges from 0 to 1 can help.
K-Nearest Neighbor (KNN) Classifier

- Consider multiple neighbors
- Assign most popular label among K nearest neighbors
- More robust to noisy data than NN (k=1)
Choosing K

• Making K too small fits the output to the noise in the dataset (overfitting)

• Making K too large can make decision boundaries in classification indistinct (underfitting)

• Choose K empirically using cross-validation
Choosing K

K=1

K=20
Choosing K

K=1

K=20

Overfitting
Choosing K

K=1

K=20

feature 2

feature 1

Overfitting

Underfitting
Choosing K

K=10

feature 1

feature 2
N-fold cross validation

1) Split data into N groups
2) Train on N-1 groups
3) Validate on the Nth
4) Rotate, repeat
N-fold cross validation

1) Split data into N groups
2) Train on N-1 groups
3) Validate on the Nth
4) Rotate, repeat
N-fold cross validation

1) Split data into N groups
2) Train on N-1 groups
3) Validate on the Nth
4) Rotate, repeat
N-fold cross validation

1) Split data into N groups
2) Train on N-1 groups
3) Validate on the Nth
4) Rotate, repeat
N-fold cross validation

1) Split data into N groups
2) Train on N-1 groups
3) Validate on the Nth
4) Rotate, repeat
Evaluation: Classification accuracy

• Evaluation on a dataset that has NOT been used in model building.

• Classification accuracy
  – # of correct classifications / total # of examples

• Example: comparing two classifiers
  – Classifier 1: 80% of accuracy
  – Classifier 2: 78% of accuracy
  – Which one would you pick for your system?

• Classification accuracy might hide the details of the performance of your model.
Evaluation: Confusion matrix

- Confusion matrix gives you a better understanding of the behavior of your classifier.

<table>
<thead>
<tr>
<th>True label</th>
<th>Piano</th>
<th>violin</th>
<th>Guitar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piano</td>
<td>19</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>violin</td>
<td>0</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>Guitar</td>
<td>1</td>
<td>3</td>
<td>16</td>
</tr>
</tbody>
</table>
Evaluation: Confusion matrix

- Confusion matrix gives you a better understanding of the behavior of your classifier.

<table>
<thead>
<tr>
<th>True label</th>
<th>Predicted label</th>
<th>Piano</th>
<th>violin</th>
<th>Guitar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piano</td>
<td>19</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>violin</td>
<td>0</td>
<td>15</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Guitar</td>
<td>1</td>
<td>3</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

Classification accuracy: 50/60 = 83%

<table>
<thead>
<tr>
<th>True label</th>
<th>Predicted label</th>
<th>Piano</th>
<th>violin</th>
<th>Guitar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piano</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>violin</td>
<td>7</td>
<td>11</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Guitar</td>
<td>1</td>
<td>0</td>
<td>19</td>
<td></td>
</tr>
</tbody>
</table>

Classification accuracy: 50/60 = 83%
Now that we know...

• How to build a KNN classifier
• How to evaluate it

We need to learn how to extract feature representations from audio input to build audio classification model.
AUDIO EVENT CLASSIFICATION
Audio event classification

- We need to convert waveform to feature representations to feed in a classifier.

- We have already learned one of feature representations: Spectrogram
Why not use the waveform as a feature?

- It is hard to find meaningful patterns
Why not use the waveform as a feature?

- It is hard to find meaningful patterns
Why not use the waveform as a feature?

• It is hard to find meaningful patterns
  – It needs a very powerful model such as deep neural networks which require millions of training data.

• Waveform is too big.
  – 1 second of audio at 44.1kHZ → 44,100 values
Commonly used audio features

• Zero-crossing rate
  – Time-domain feature
  – Rate of sign changes in a signal
  – Low for harmonic sounds, high for noisy sounds

* Figure: https://en.wikipedia.org/wiki/Zero_crossing
Commonly used audio features

- Zero-crossing rate

Graph showing the zero-crossing rate for different sound sources:
- Guitar
- Snare drum
- White noise
Commonly used audio features

- **Spectral centroid**
  - Frequency domain feature
  - The weighted mean of the frequencies in the signal
  - Known as a predictor of the “brightness” of a sound

* figure: https://librosa.github.io/librosa/generated/librosa.feature.spectral_centroid.html
Commonly used audio features

- Spectral centroid

Kick drum

Snare drum
Automatic drum transcription

• Let’s build a drum transcription machine only using spectral centroid features
Automatic drum transcription

• Onset detection
  – librosa.onset.onset_detect
Automatic drum transcription

• Segmentation
  – Cutting the recording every $<\text{onset−2048 samples}>$

(Onset[t – 1] – 2048, Onset[t] – 2048)
Automatic drum transcription

• Extracting spectral centroid from each segment
Automatic drum transcription
Automatic drum transcription - 2

• More challenging example
Automatic drum transcription-2

- Onset detection might not work that well on this example, but let’s assume we have perfect onset info
Automatic drum transcription-2

- Segmentation and feature extraction

- The previous example
Automatic drum transcription-2

• More challenging example

You can find more feature extraction functions in the Librosa package
Feature summarization

• Using summary statistics over time to represent an audio expert as a single vector

\[ \vec{x} = \langle a_1, a_2, ... , a_n \rangle \]

\[ \text{Mean}([SC_1, SC_2, SC_3, ..., SC_t]) = a_1 \]

\[ \text{Variance}([SC_1, SC_2, SC_3, ..., SC_t]) = a_2 \]

\[ \text{Delta-mean}([SC_1, SC_2, SC_3, ..., SC_t]) \]
\[ = \text{mean}([SC_2 - SC_1, SC_3 - SC_2, ... , SC_t - SC_{t-1}]) = a_3 \]

\[ \text{Delta-var}([SC_1, SC_2, SC_3, ..., SC_t]) \]
\[ = \text{var}([SC_2 - SC_1, SC_3 - SC_2, ... , SC_t - SC_{t-1}]) = a_4 \]

*SC: Spectral Centroid
Feature summarization

• Example for multi dimensional features

![Diagram showing summarization process]

- Summarize over time
- Concatenate
- Mean, Variance
Example using a TINY spectrogram

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Spectrogram</th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 0 3 1 5</td>
<td>1</td>
<td>3.2</td>
</tr>
<tr>
<td></td>
<td>0 .4 0 .4 .2</td>
<td>.2</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>0 29 1 20 10</td>
<td>12</td>
<td>124.4</td>
</tr>
<tr>
<td></td>
<td>10 10 10 10 10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0 0 0 50 0</td>
<td>10</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td>0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time
Example using a TINY spectrogram

<table>
<thead>
<tr>
<th>Spectrogram</th>
<th>Delta</th>
<th>Delta-mean</th>
<th>Delta-variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 3 1 5</td>
<td>-1 3 -2 4</td>
<td>1</td>
<td>6.5</td>
</tr>
<tr>
<td>0 .4 0 .4 .2</td>
<td>.4 -.4 .4 -.2</td>
<td>.05</td>
<td>.13</td>
</tr>
<tr>
<td>0 29 1 20 10</td>
<td>29 -28 19 -10</td>
<td>2.5</td>
<td>515.3</td>
</tr>
<tr>
<td>10 10 10 10 10</td>
<td>0 0 0 0 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 0 50 0</td>
<td>0 0 50 -50</td>
<td>0</td>
<td>1250</td>
</tr>
</tbody>
</table>

Delta = frame at (t+1) – frame at t
Example using a TINY spectrogram

<table>
<thead>
<tr>
<th>Mean</th>
<th>Variance</th>
<th>Delta -Mean</th>
<th>Delta -Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.2</td>
<td>1</td>
<td>6.5</td>
</tr>
<tr>
<td>.2</td>
<td>0.03</td>
<td>.05</td>
<td>.13</td>
</tr>
<tr>
<td>12</td>
<td>124.4</td>
<td>2.5</td>
<td>515.3</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>400</td>
<td>0</td>
<td>1250</td>
</tr>
</tbody>
</table>

The final feature vector (concatenating them all):
[1, .2, 12, 10, 10, 3.2, 0.03, 124.4, 0, 400, 1, .05, 2.5, 0, 0, 6.5, .13, 515.3, 0, 1250]
Sound Event Detection by Classification

Context-window

Classification on each context window

Dog barking
Car engine
Door knock
Challenges

• Polyphonic environment, background noise

• Noisy labels

• Using a hierarchical relationship between audio labels

• Weakly labeled training dataset

• A small amount of labeled training dataset

• A large amount of unlabeled training dataset
Datasets for sound object labeling

- Urban sound dataset: https://urbansounddataset.weebly.com/
- AudioSet: https://research.google.com/audioset/
- ESC: https://github.com/karoldvl/ESC-50
- IRMAS: https://www.upf.edu/web/mtg/irmas
- Vocal Imitation Set: https://zenodo.org/record/1340763#.XEtAJs9KiRs
EXAMPLE: DOOR KNOCKING / PHONE RINGING CLASSIFICATION
Training data
Feature extraction and summarization

• Zero-crossing rate and Spectral centroid
  – window length = 2048, hop length = 1024
  – Both features are represented as a single number for each time frame. So we get two feature values for each time frame (2-dimensional space)
  – The number of time frames vary with the length of each signal.

• To represent all the signals as the same size of feature vectors, we do summarization.
  – In this tutorial, I will take mean over frames.
Feature extraction and summarization

Now we can map all the signals into 2-dimensional feature space
Plotting them in the feature space
Feature scaling

![Graph showing Feature scaling with Knock and Phone categories. The x-axis represents Zero Crossing Rate, and the y-axis represents Spectral Centroid. The data points are scattered, with Knock data points in blue and Phone data points in red.]
Testing examples
Plotting test examples

Nearest Neighbor classifier would perfectly work in this testing case