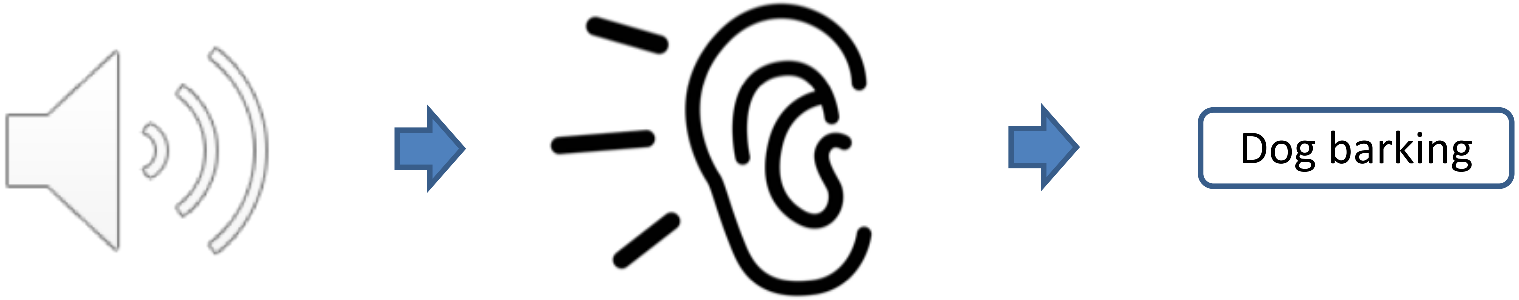
Sound object labeling

EECS 352 Machine perception of Music and Audio

Bongjun Kim

Winter, 2019

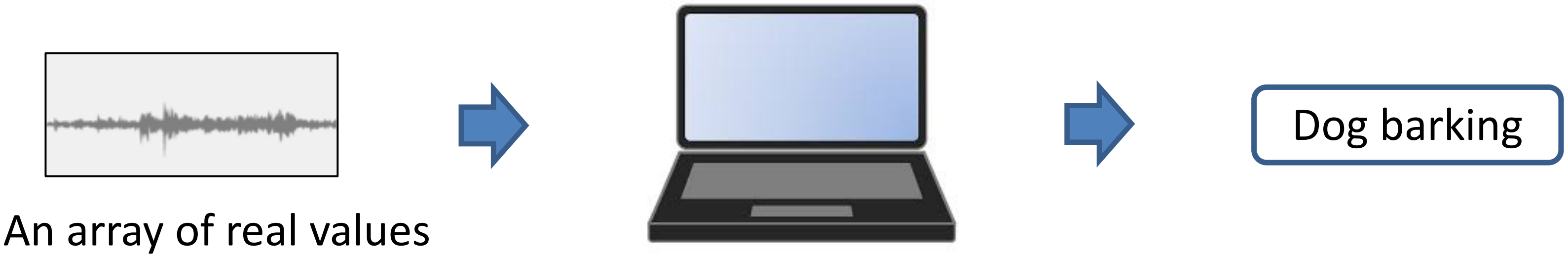
Sound object labeling



Dog barking

Goal

- Building a system that automatically labels an audio event

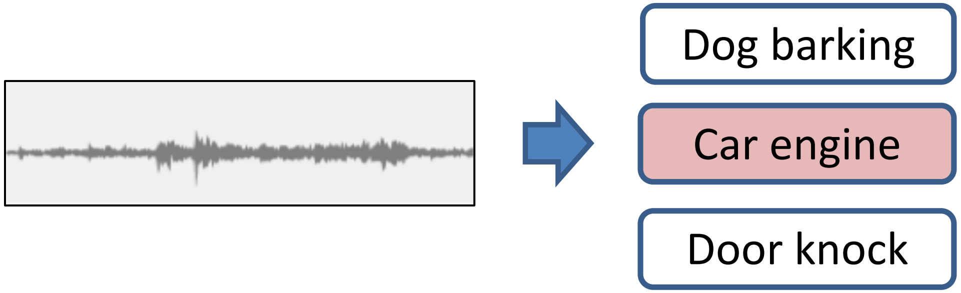


An array of real values

Dog barking

Tasks

- Audio classification

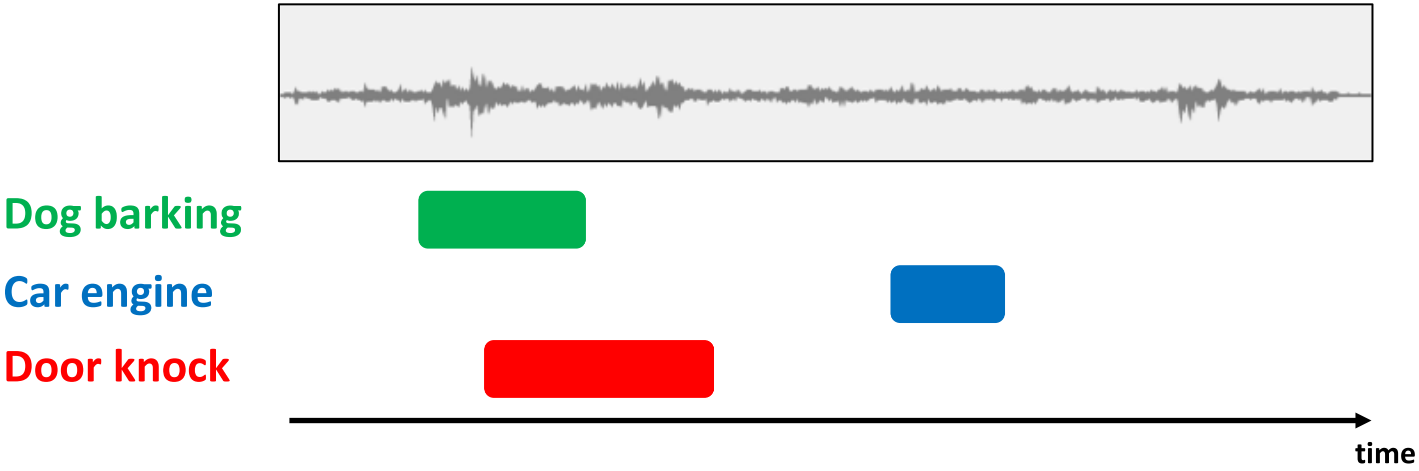


Dog barking

Car engine

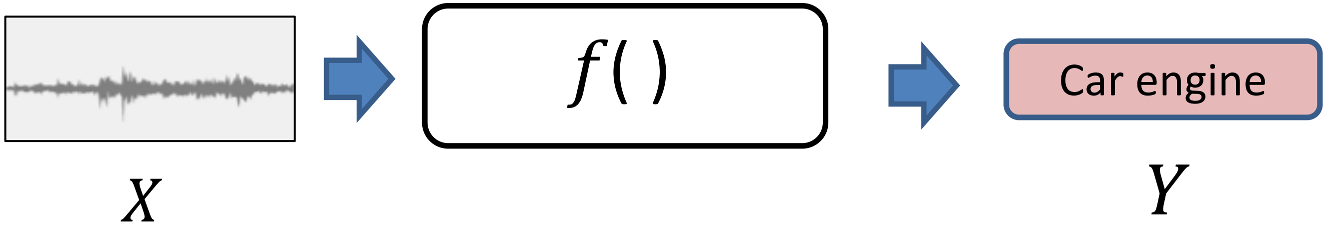
Door knock

- Sound Event Detection (SED)



MACHINE LEARNING: CLASSIFICATION

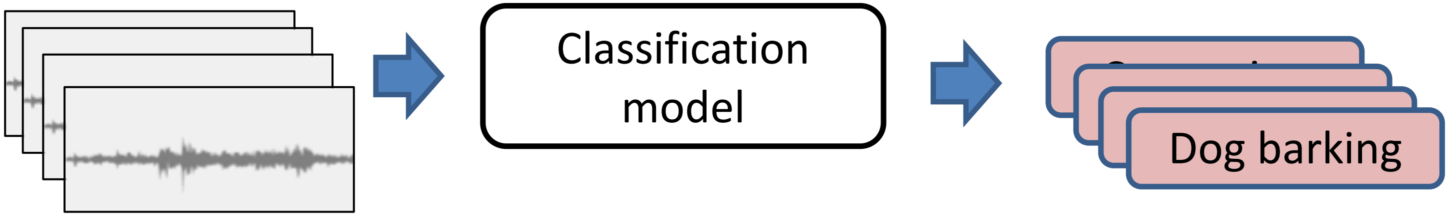
Supervised learning from data



Function we want to learn → $Y = f(X)$
(Target function)

Training input

Ground truth labels



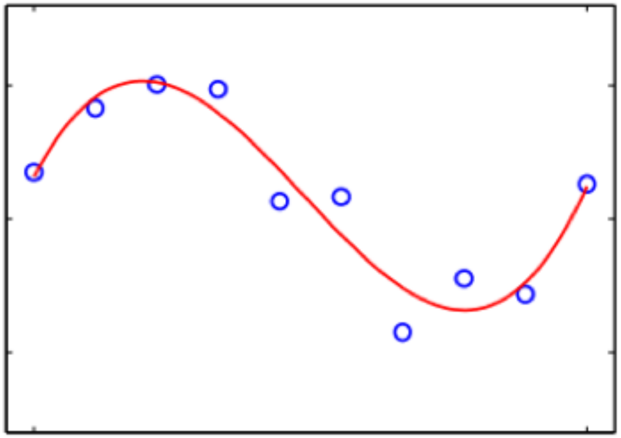
$X = \{x_{1,} x_{2,} \ldots, x_n\}$ $h(\ )$ $y = \{y_{1,} y_{2,} \ldots y_n\}$

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

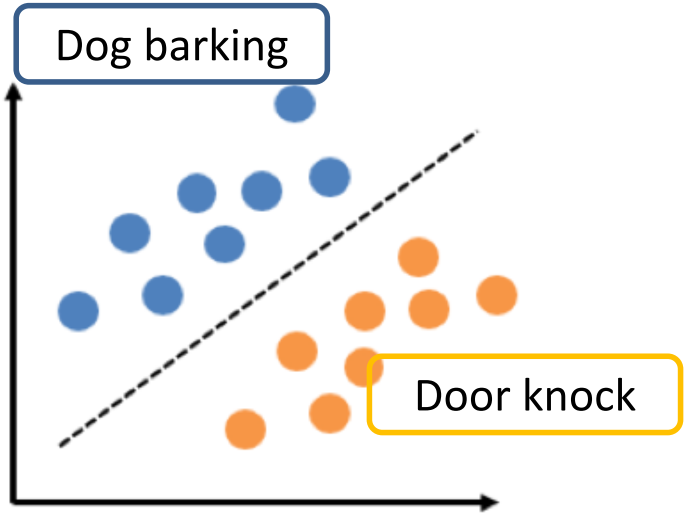
On the training data D = $\{< x_{1,} y_1 >, \ldots < x_{n,} y_n >\}$

Supervised Learning

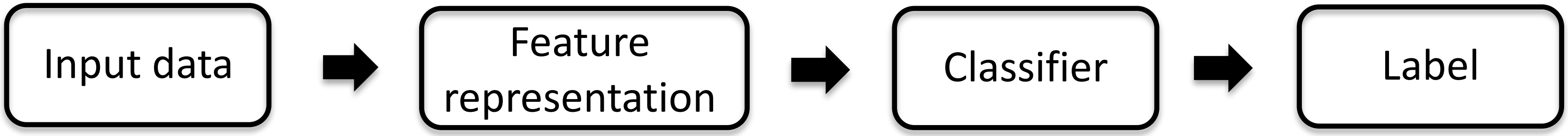
- Regression
  - A target function maps X onto **continuous real values** Y.



- 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

$\vec{x} = <a_1, a_2, \ldots, a_n>$

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

- Decision Tree

- Nearest Neighbor

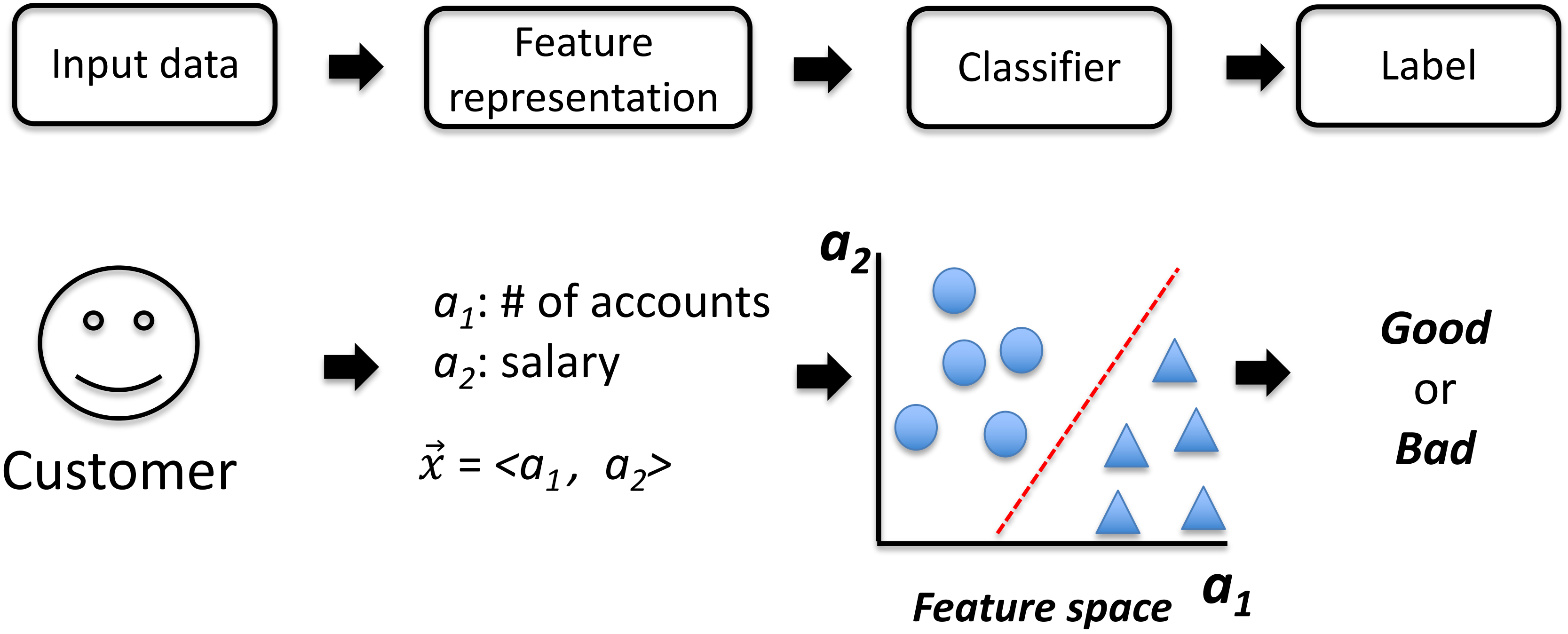
- Neural Networks

“Cat image”

“Piano sound”

Overview of general classification tasks

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



Input data → Feature representation → Classifier → Label

Customer

a_1: # of accounts

a_2: salary

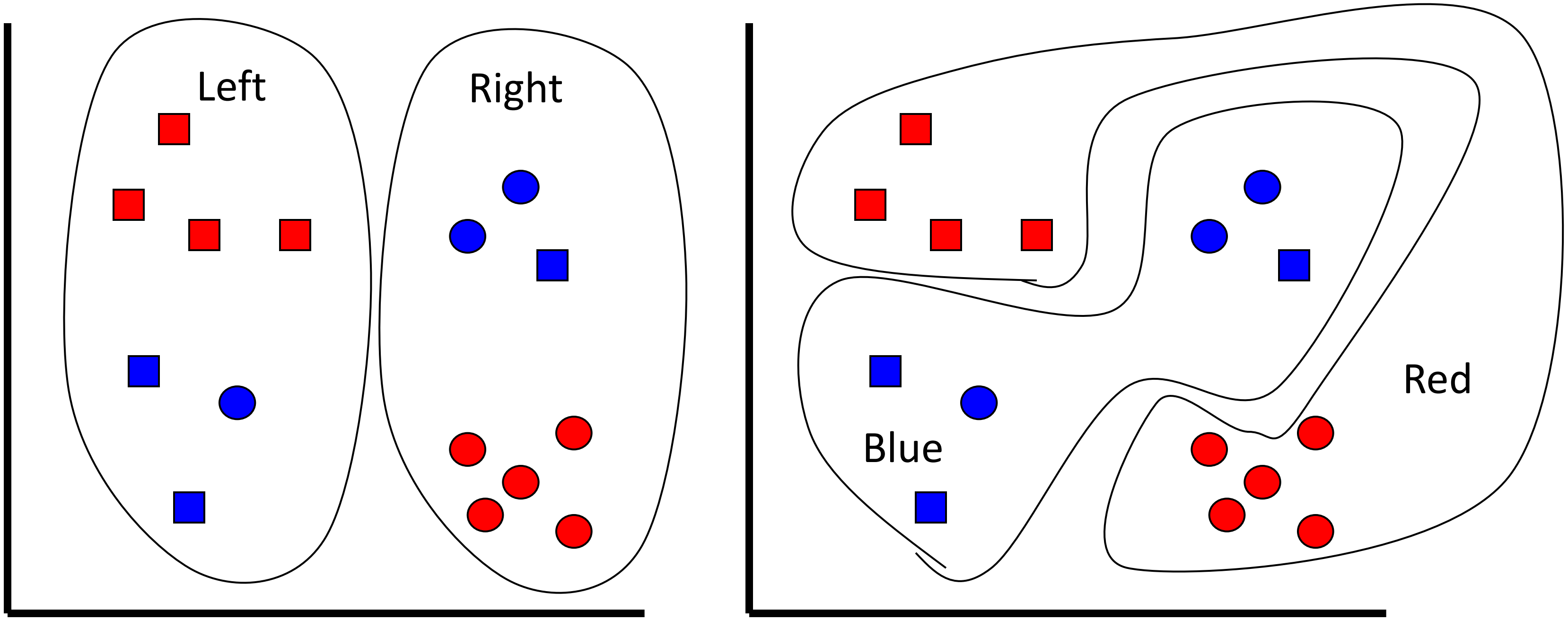
$\vec{x} = <a_1,\ a_2>$

Feature space

Good
or
Bad

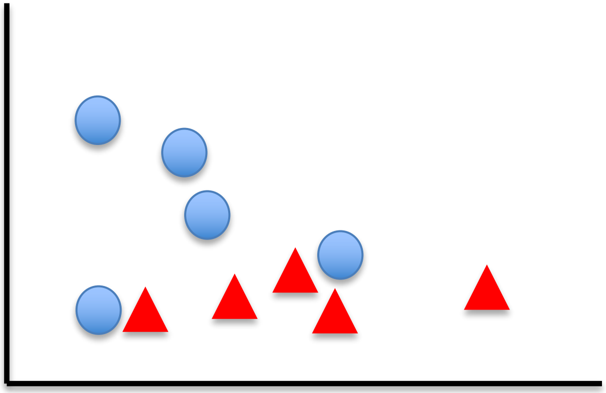
Different Classifiers

- Different classifications need different classifiers.

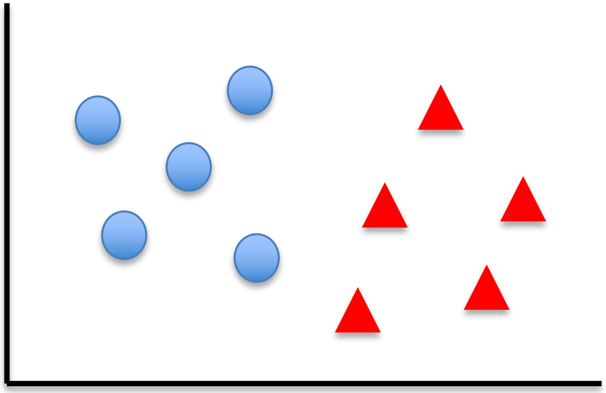


Feature selection is important

- How things cluster depend on what you are measuring.

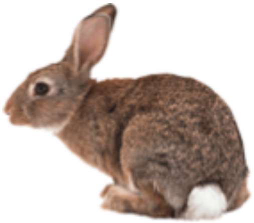


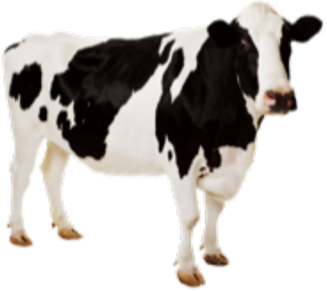
Bad feature representations

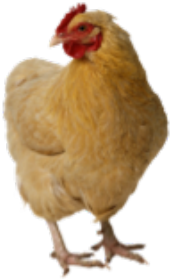


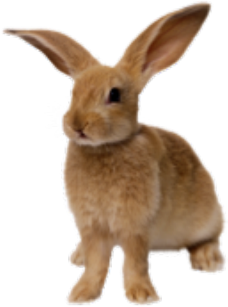
Good feature representations

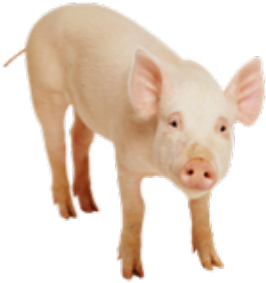
Which of these go together?

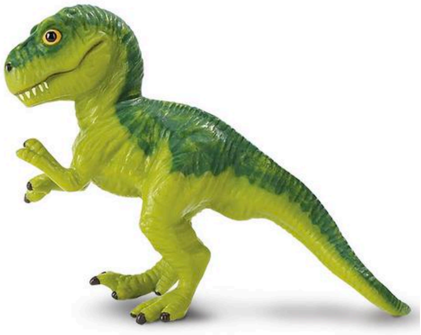


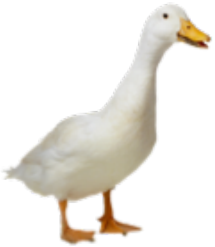


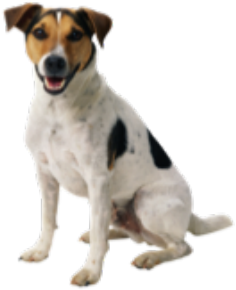




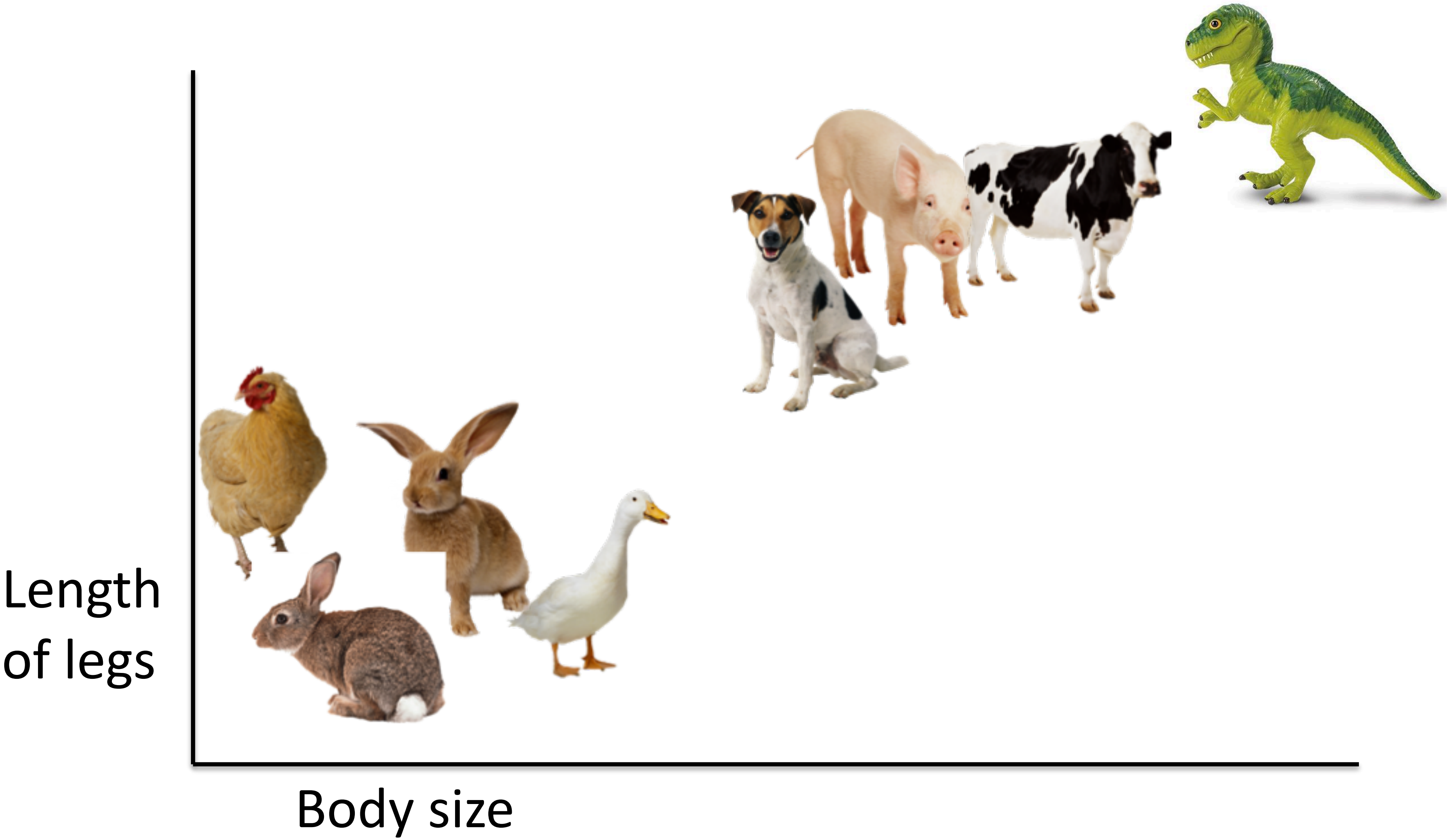




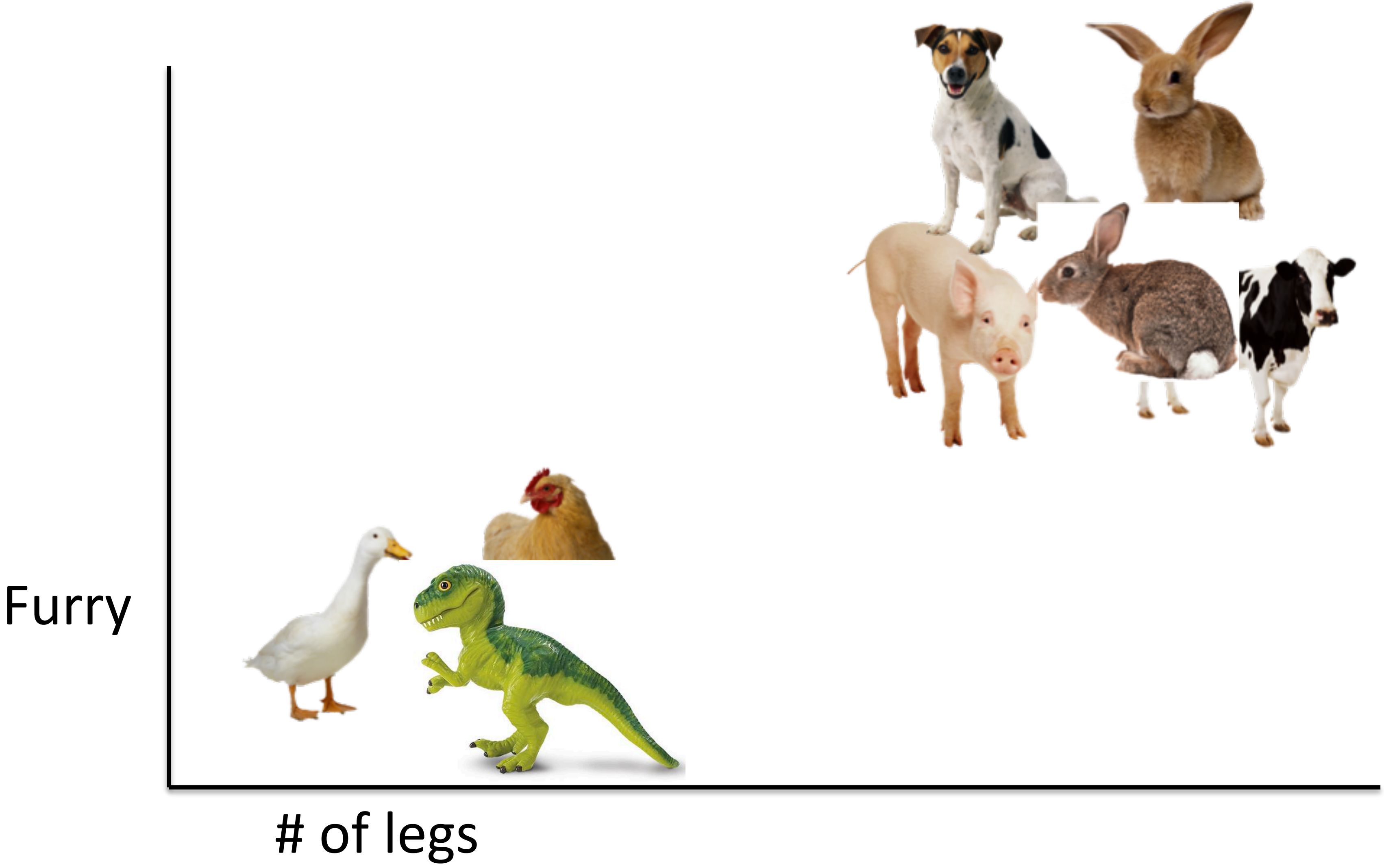




Which of these go together?



Which of these go together?

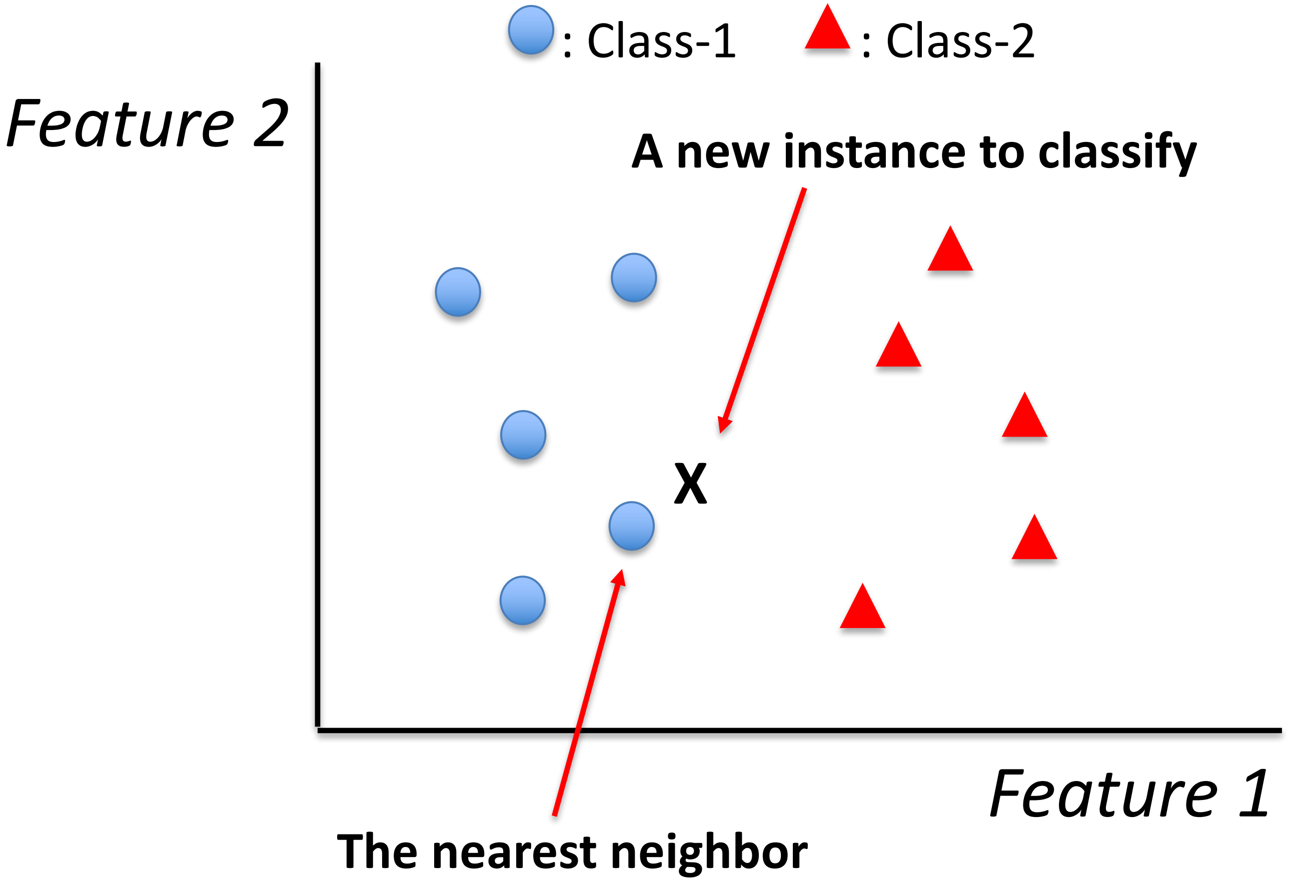


Nearest Neighbor (NN) Classifier

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

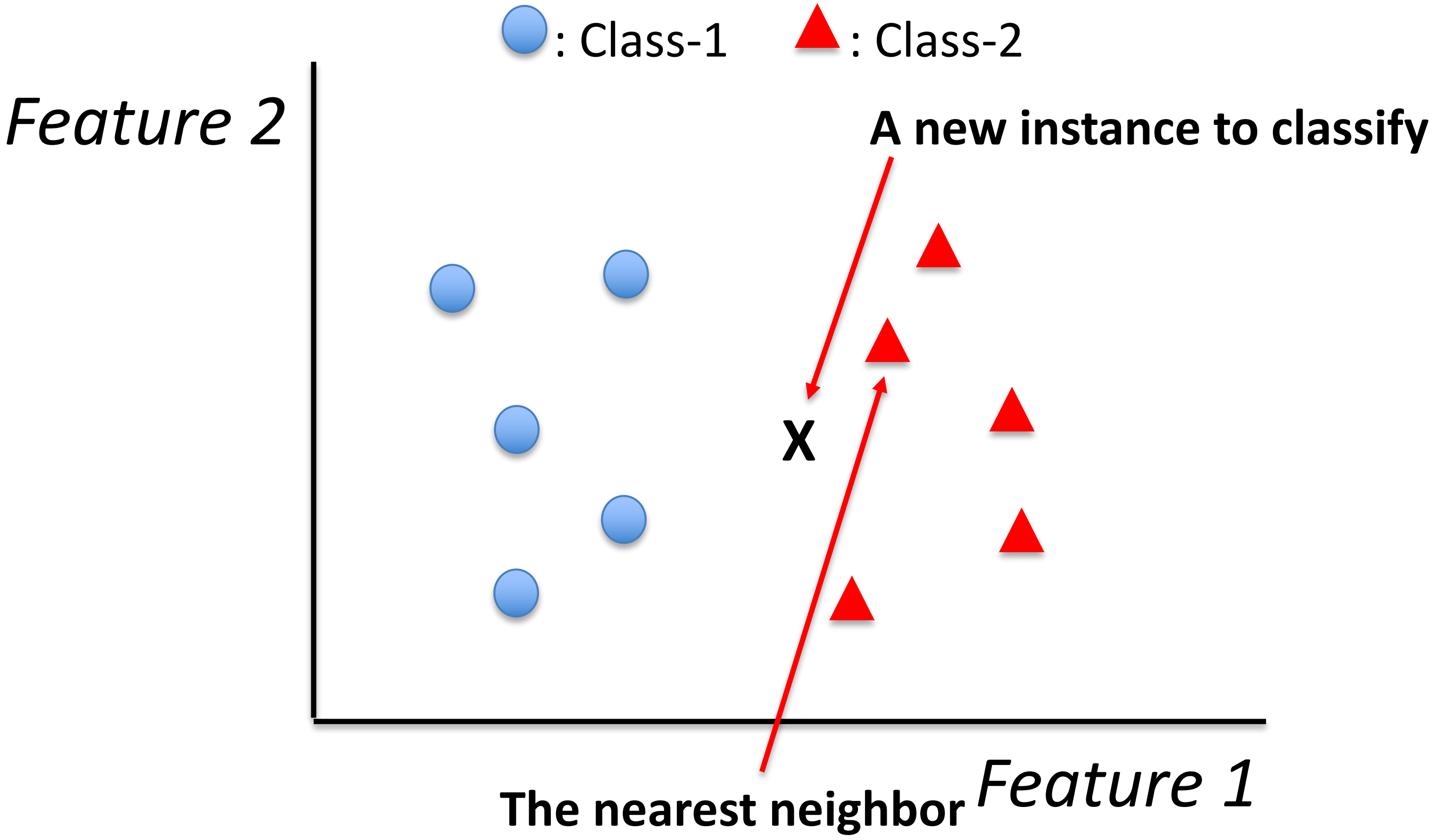
- How do you tell what things are similar?
  1. Extract proper features.
  2. Measure distance / similarity in the feature space.

Nearest Neighbor (NN) Classifier



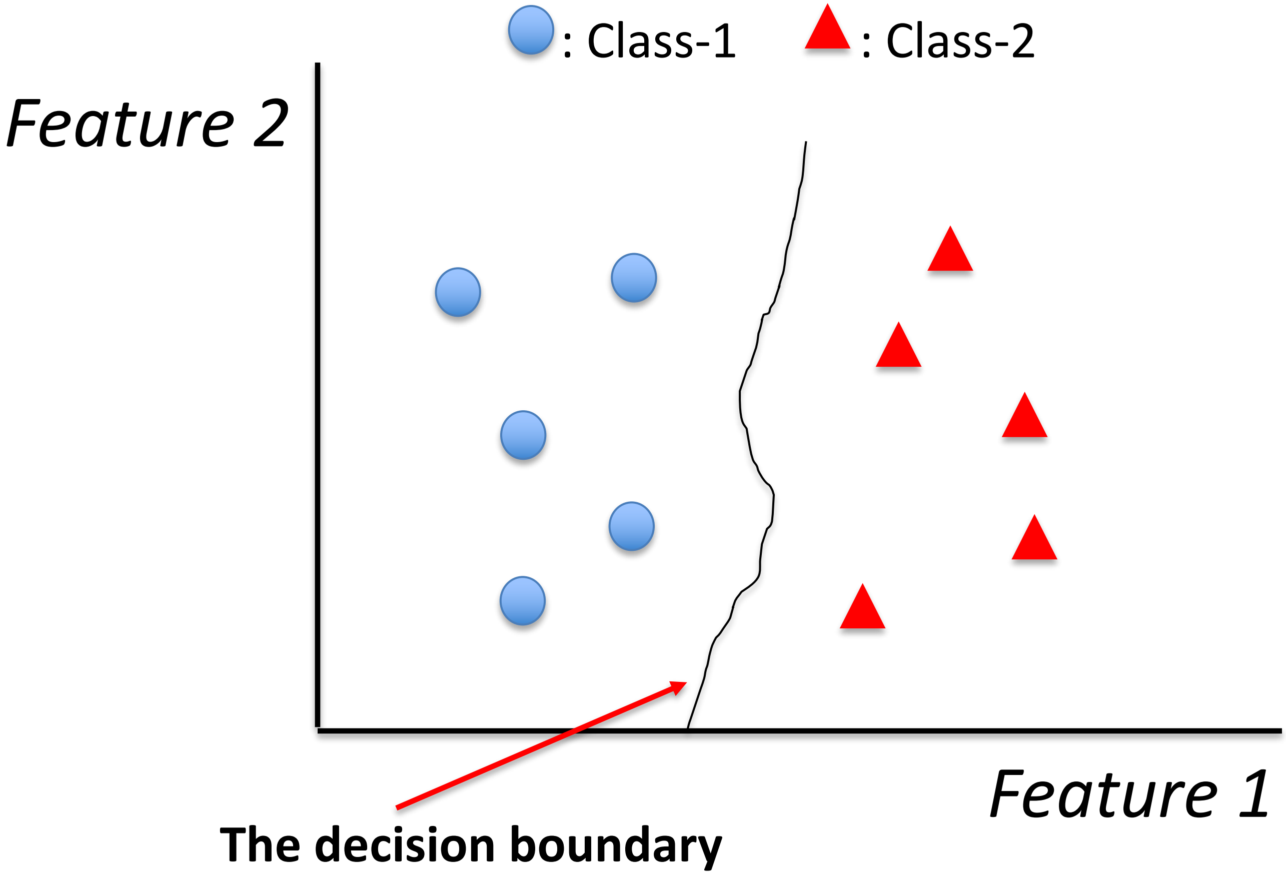
X is classified into class-1

Nearest Neighbor (NN) Classifier



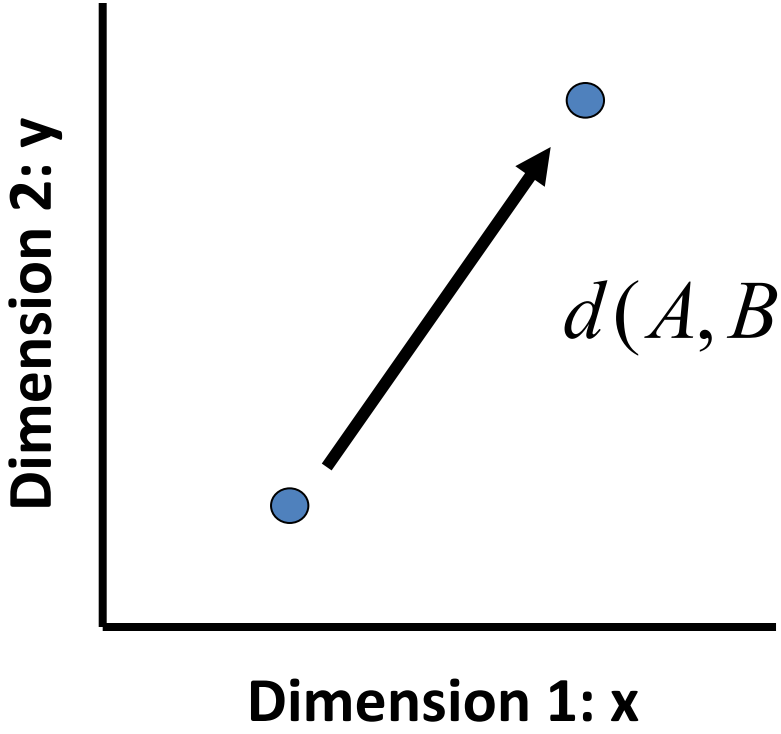
X is classified into class-2

Nearest Neighbor (NN) Classifier



How do we measure distance?

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



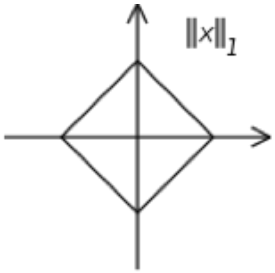
$$d(A,B) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$$

L^p norms

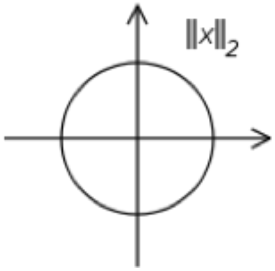
- L^p norms are all special cases of this function:

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

p changes the norm



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:*

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

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

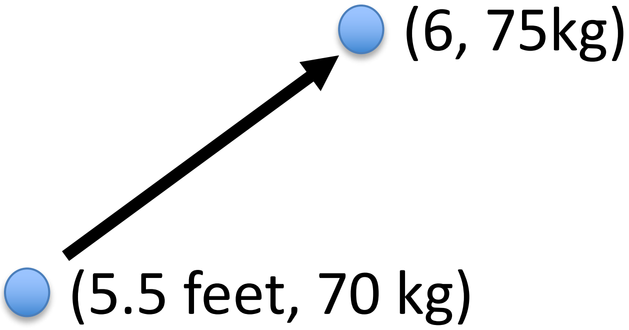
Feature Scaling

- 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)

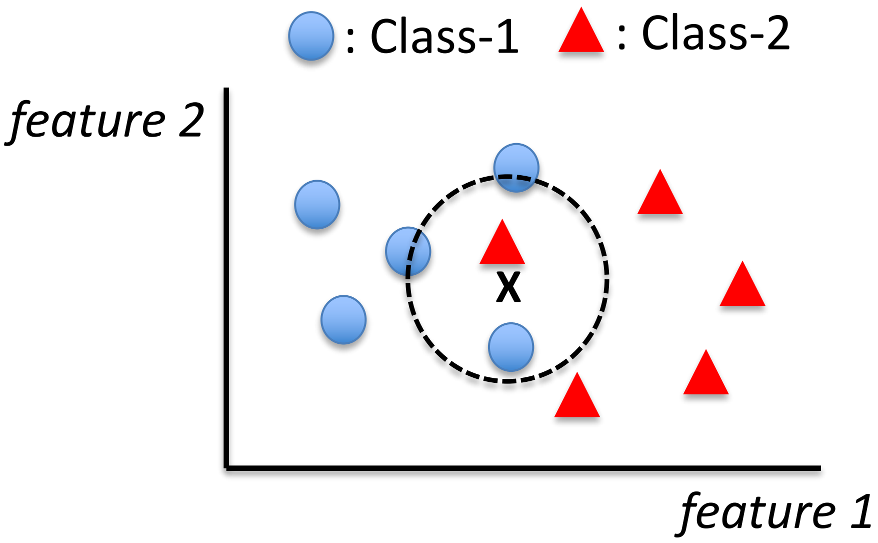


In this Euclidean space, the second feature dominates the distance, which might lead to mis-clustering.

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)



Considering 4 nearest neighbors (k=4), most popular class is Class-1

Choosing K

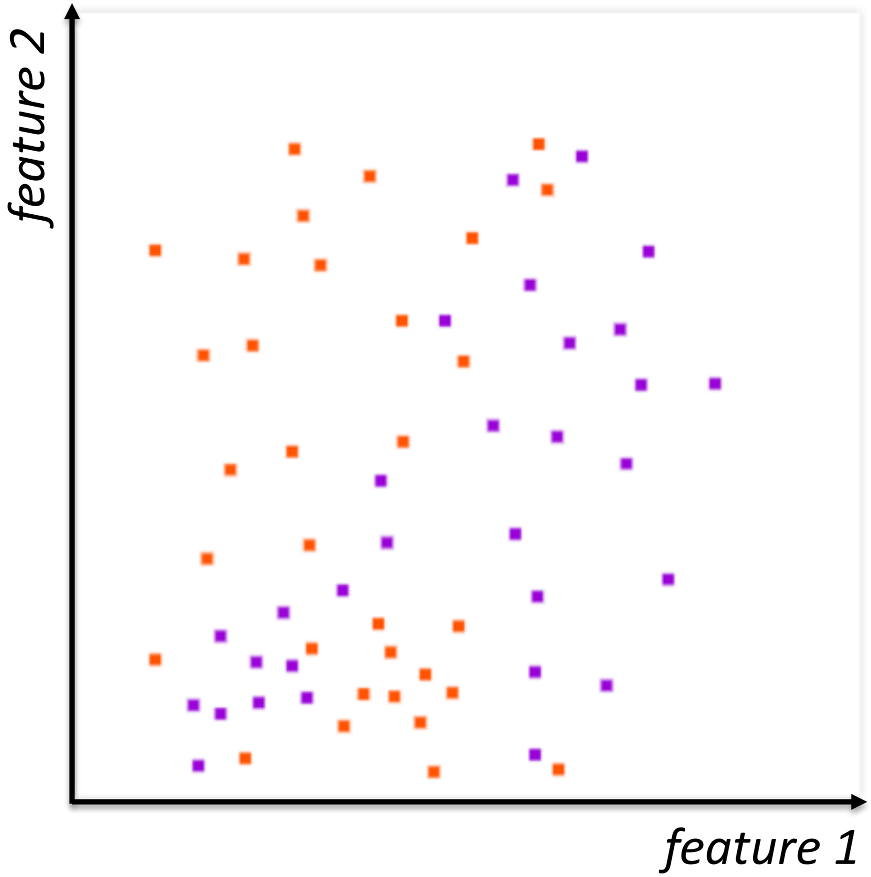
- Making K too small fits the output to the noise in the dataset (overfitting)

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

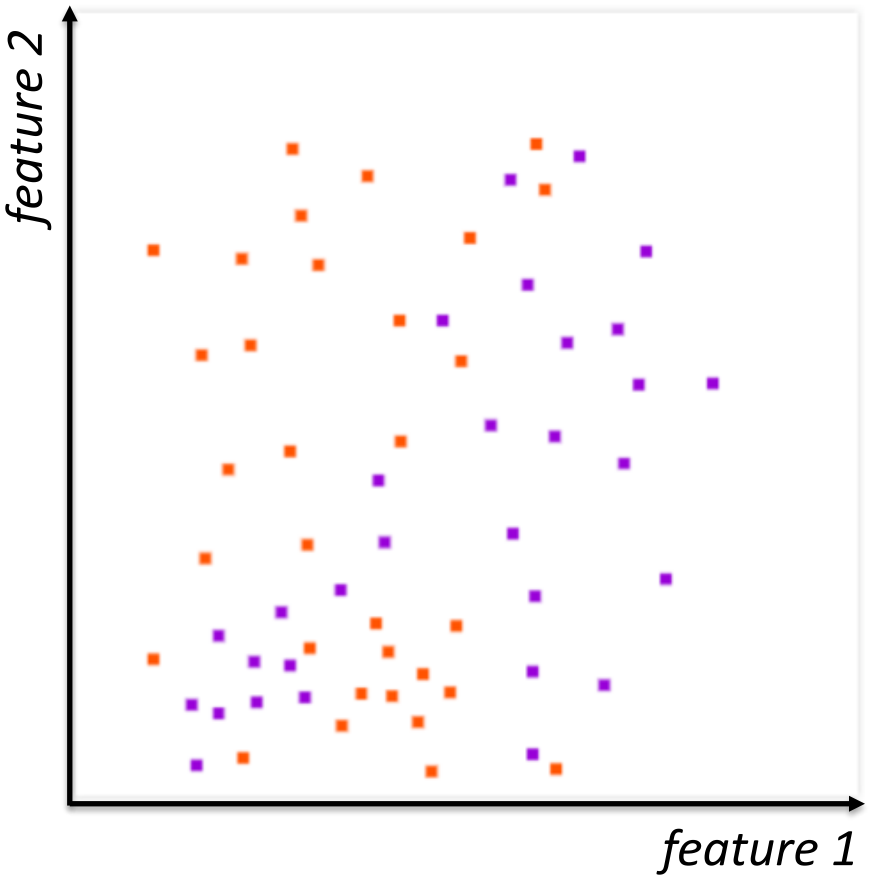
- Choose K empirically using cross-validation

Choosing K

K=1

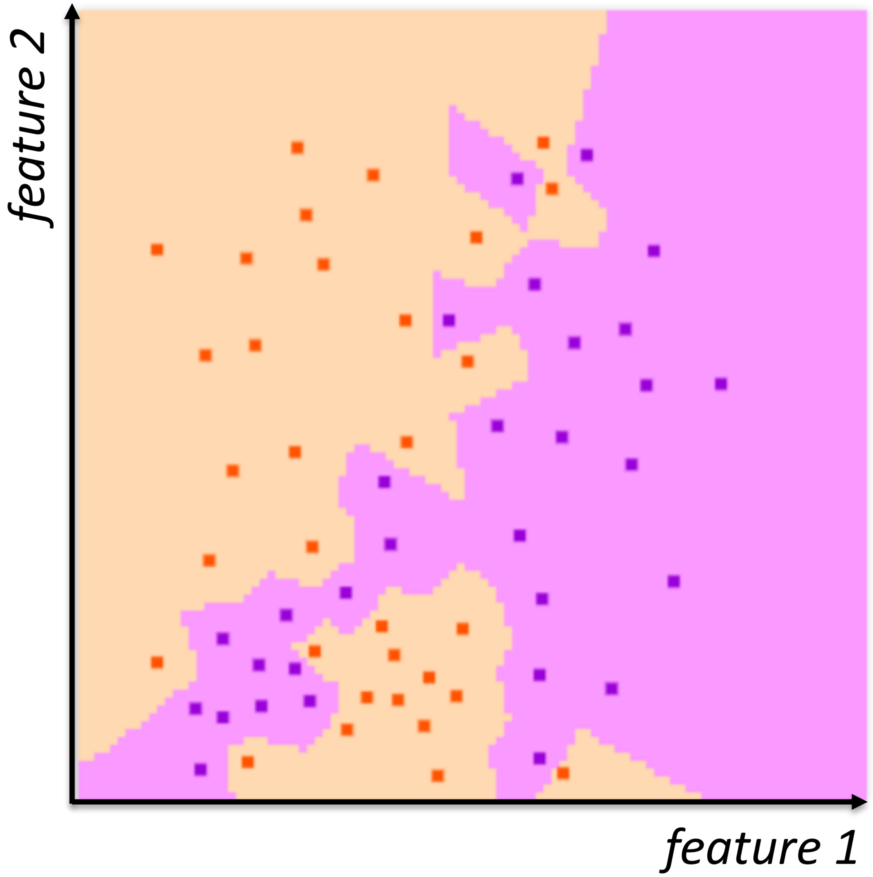


K=20



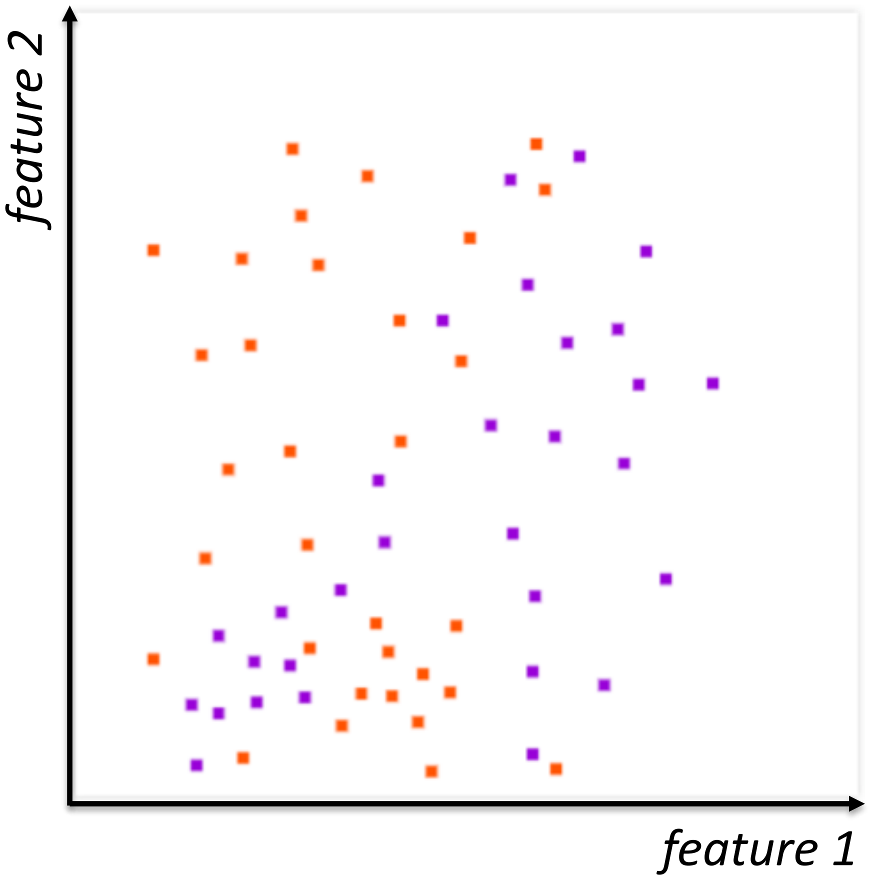
Choosing K

K=1



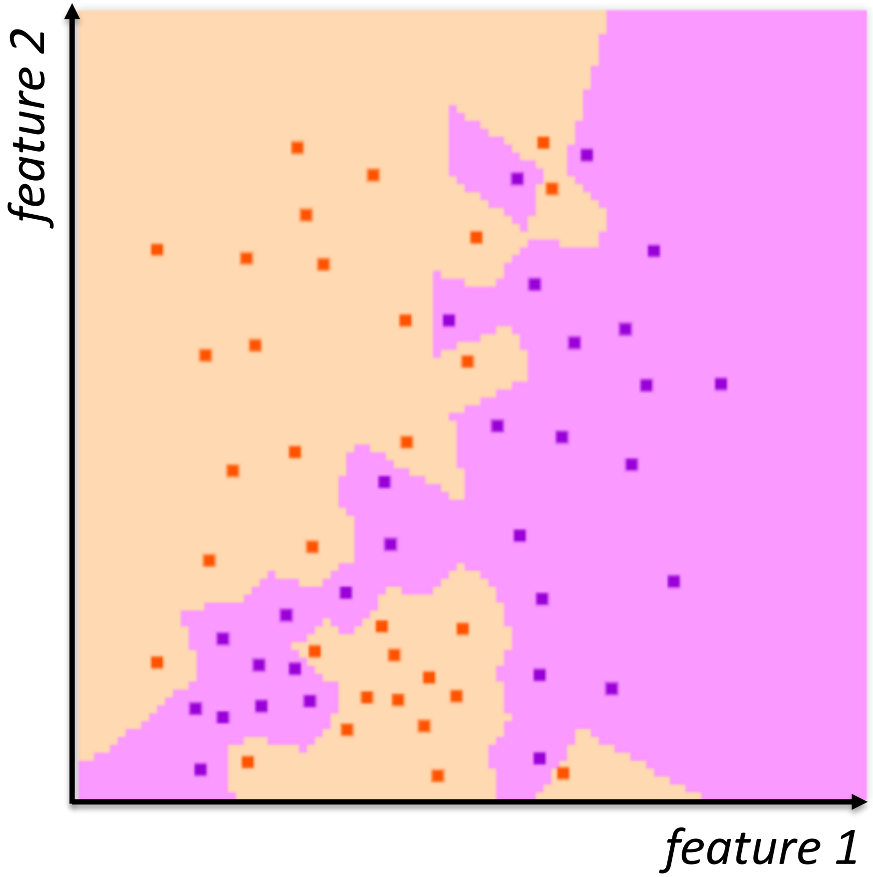
Overfitting

K=20



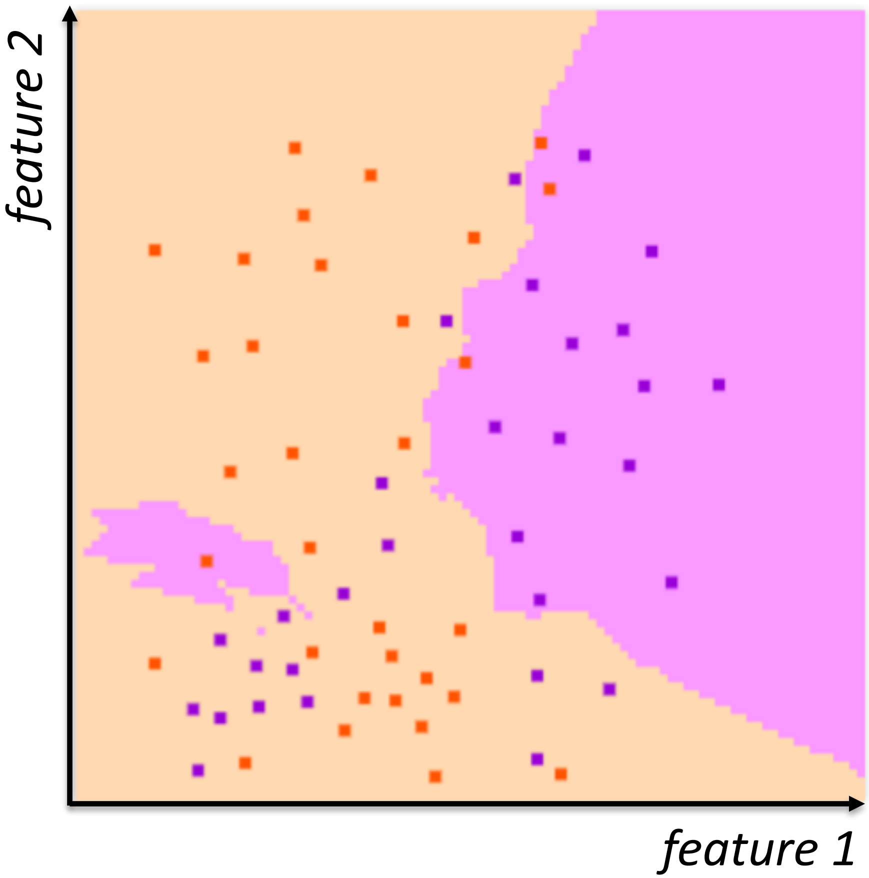
Choosing K

K=1



Overfitting

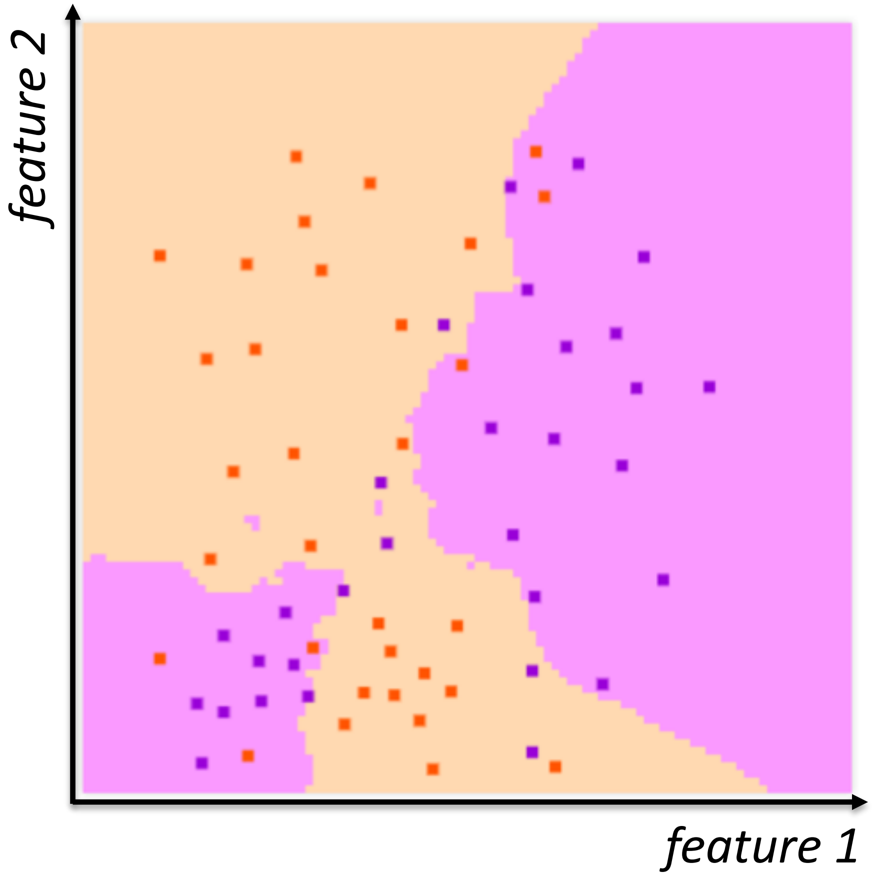
K=20



Underfitting

Choosing K

K=10



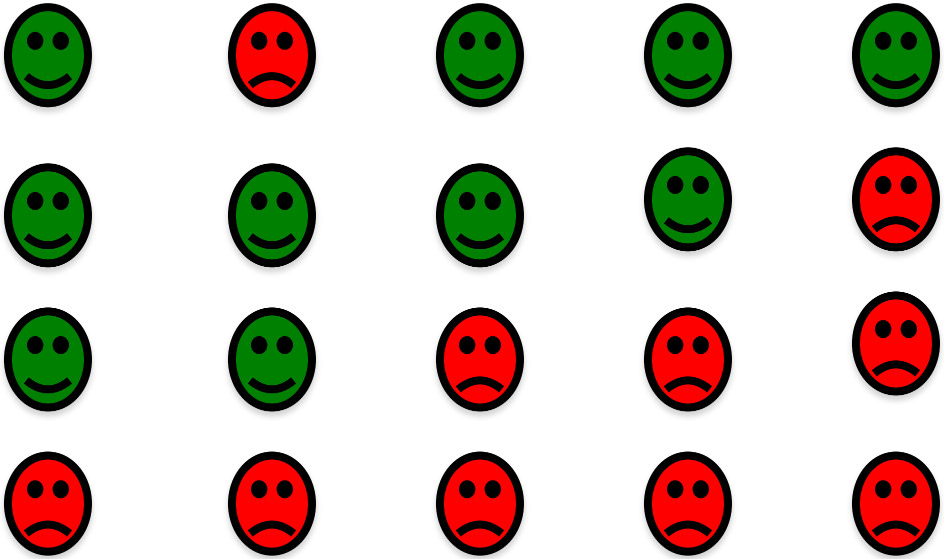
N-fold cross validation

1) Split data into N groups

2) Train on N-1 groups

3) Validate on the Nth

4) Rotate, repeat



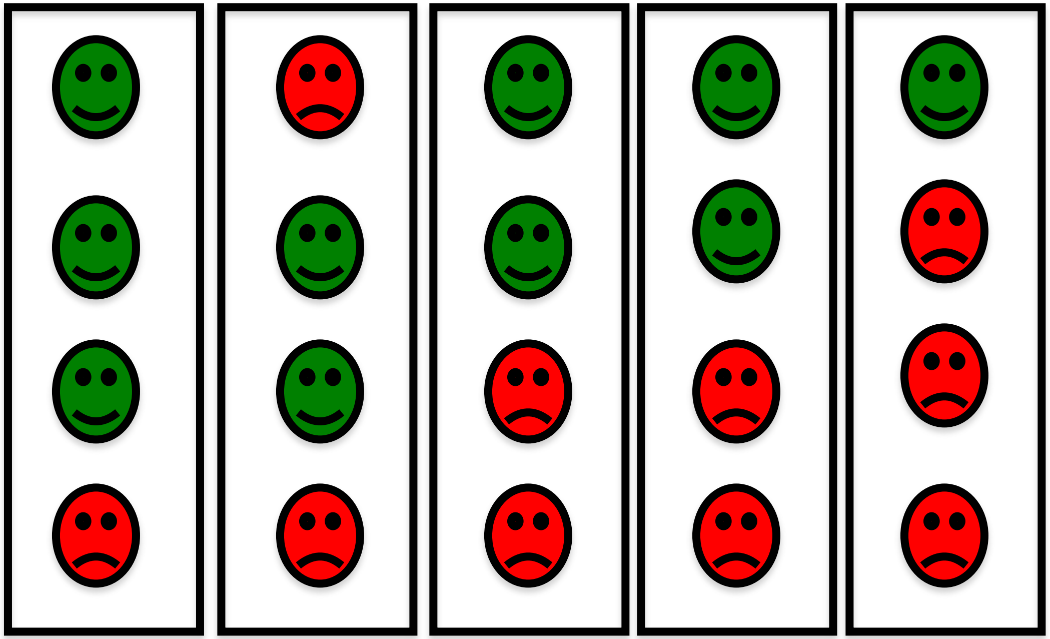
N-fold cross validation

1) Split data into N groups

2) Train on N-1 groups

3) Validate on the Nth

4) Rotate, repeat



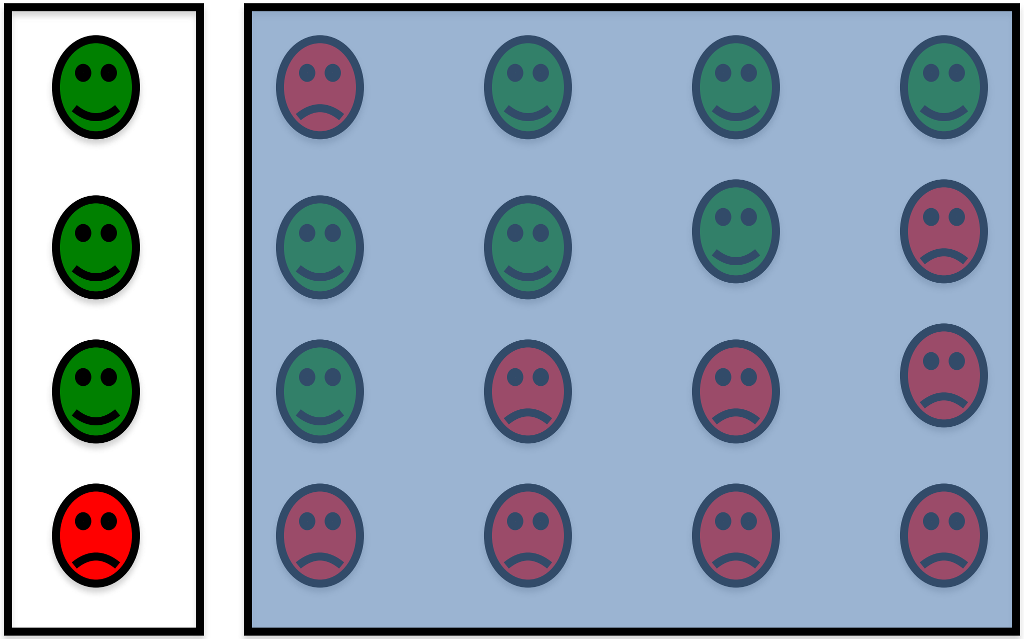
N-fold cross validation

1) Split data into N groups

2) Train on N-1 groups

3) Validate on the Nth

4) Rotate, repeat



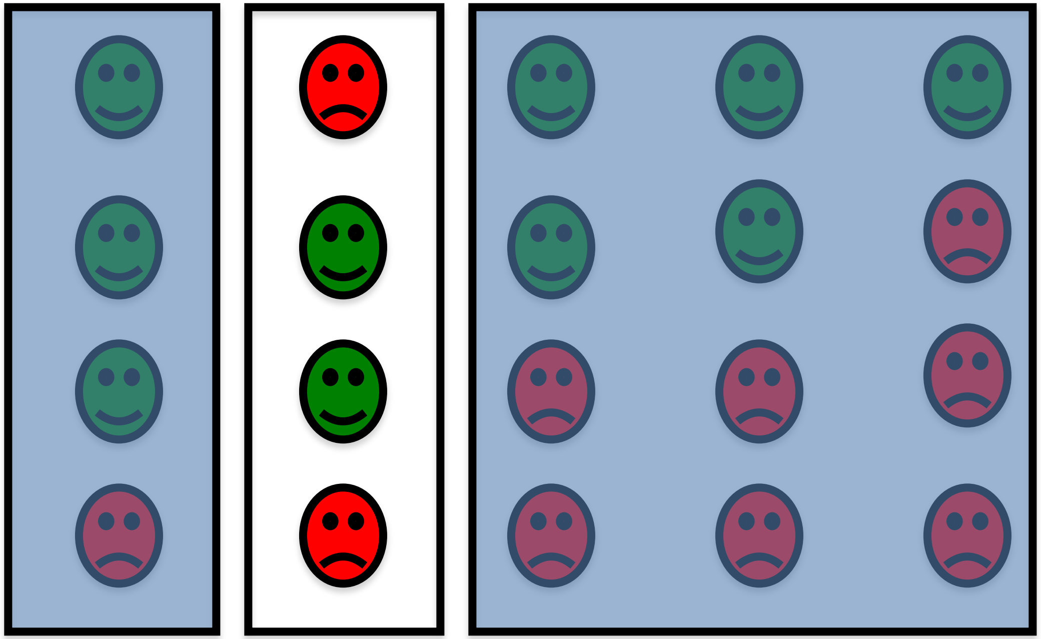
N-fold cross validation

1) Split data into N groups

2) Train on N-1 groups

3) Validate on the Nth

4) Rotate, repeat



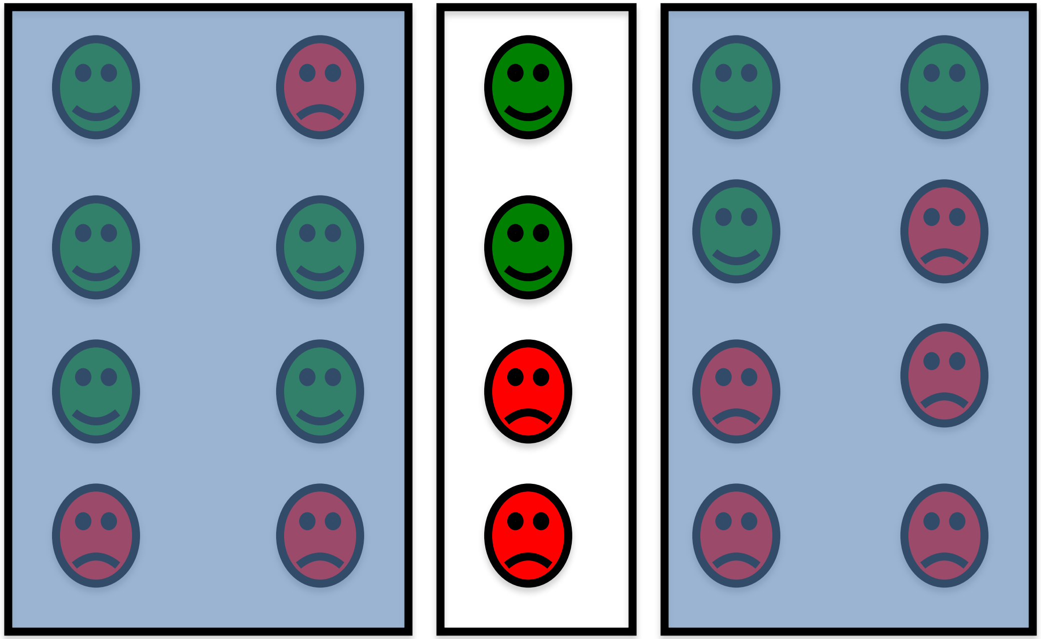
N-fold cross validation

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
  - Classifier2: 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.

Predicted label

True label

	Piano	**violin**	**Guitar**
Piano	**19**	**0**	**1**
violin	**0**	**15**	**5**
Guitar	**1**	**3**	**16**

Evaluation: Confusion matrix

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

Predicted label

True label

	Piano	**violin**	**Guitar**
Piano	**19**	**0**	**1**
violin	**0**	**15**	**5**
Guitar	**1**	**3**	**16**

Classification accuracy:
50/60 = 83%

Predicted label

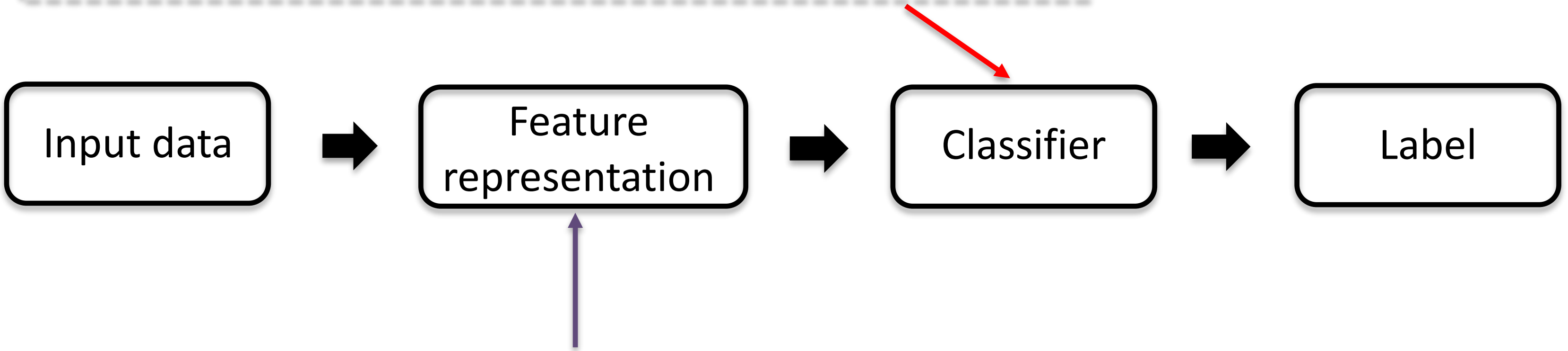
True label

	Piano	**violin**	**Guitar**
Piano	**20**	**0**	**0**
violin	**7**	**11**	**2**
Guitar	**1**	**0**	**19**

Classification accuracy:
50/60 = 83%

Now that we know..

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

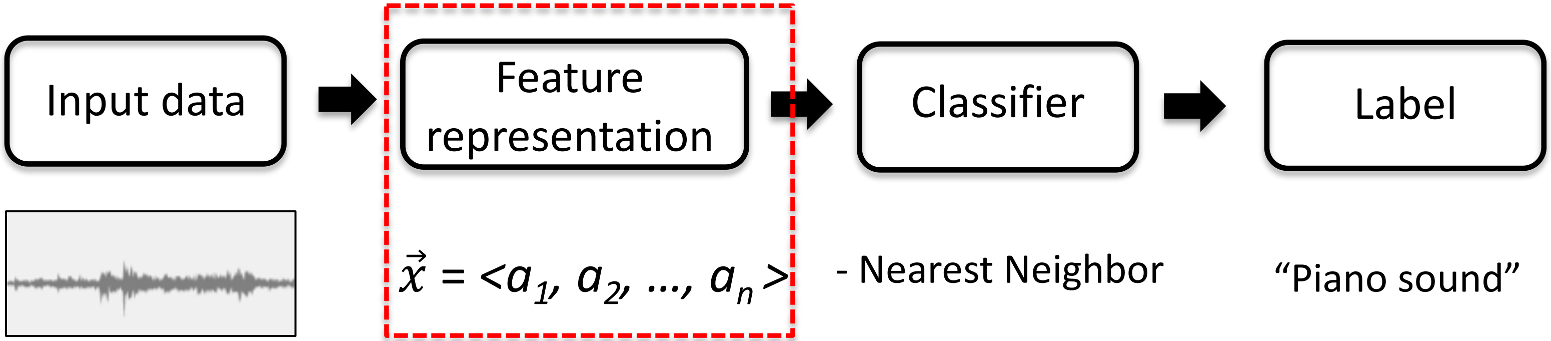


Input data → Feature representation → Classifier → Label

- We need to learn how to extract feature representations from audio input to build audio classification model.

AUDIO EVENT CLASSIFICATION

Audio event classification



Input data → Feature representation → Classifier → Label

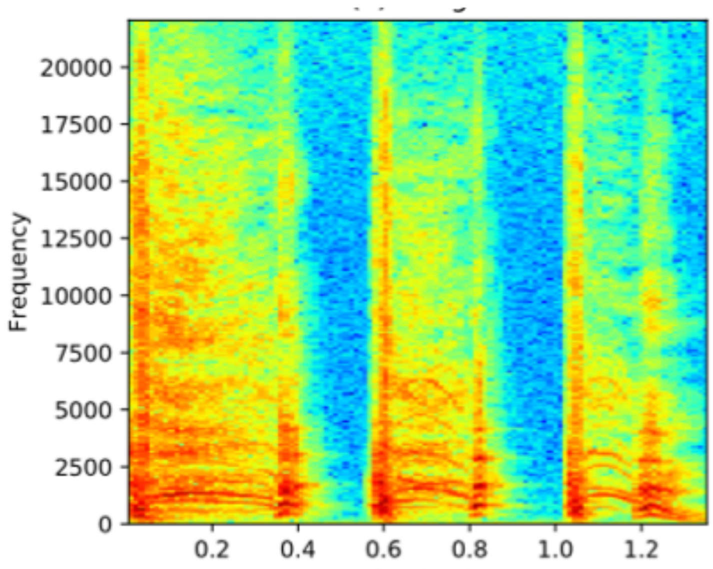
$\vec{x} = <a_1, a_2, \ldots, a_n>$

- Nearest Neighbor

“Piano sound”

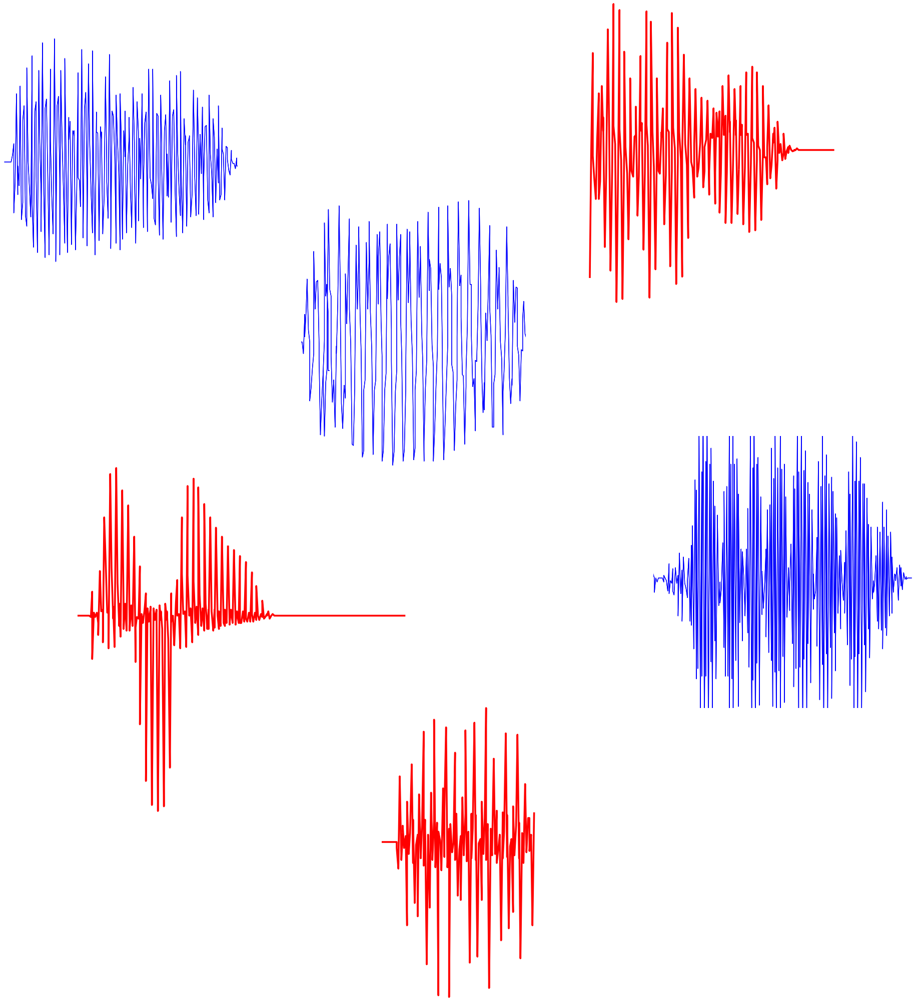
We need to convert waveform to feature representations to feed in a classifier.

- We have already learned one of feature representations: Spectrogram



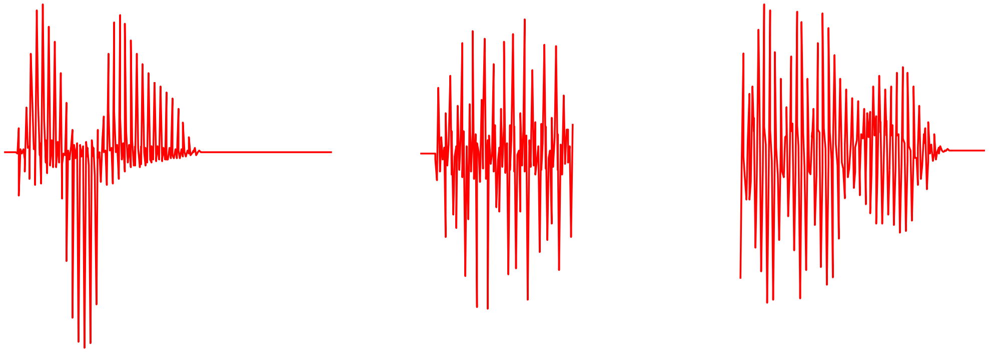
Why not use the waveform as a feature?

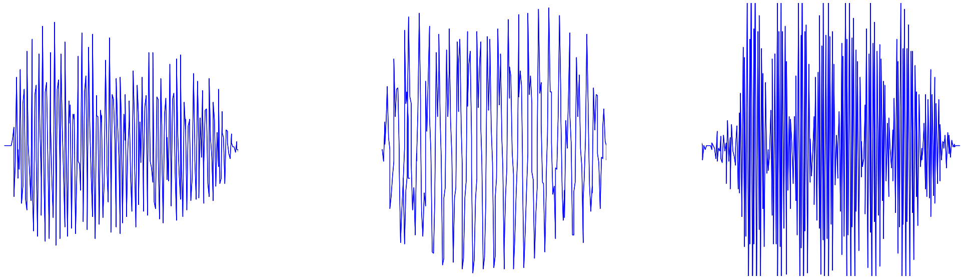
- It is hard to find meaningful patterns



Why not use the waveform as a feature?

- It is hard to find meaningful patterns





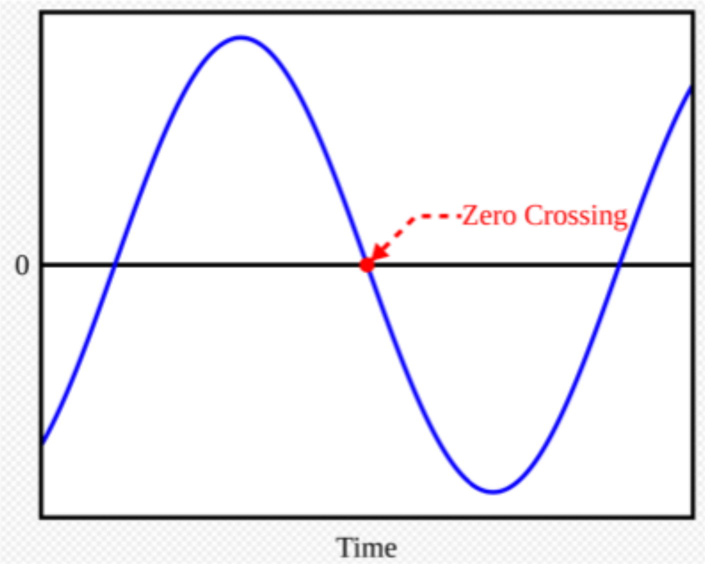
Why not use the waveform as a feature?

- 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

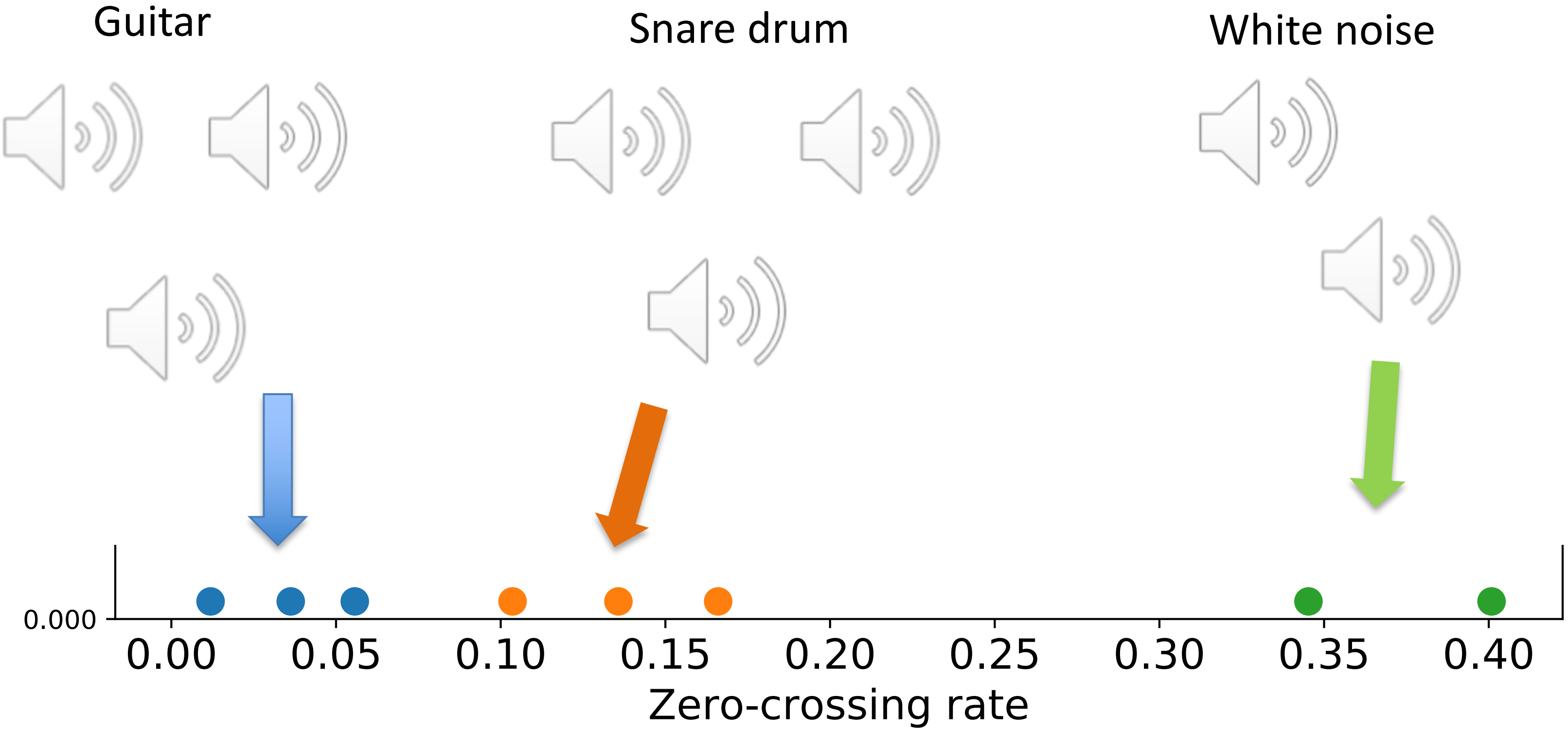
- Zero-crossing rate
  - Time-domain feature
  - Rate of sign changes in a signal
  - Low for harmonic sounds, high for noisy sounds



* Figure: https://en.wikipedia.org/wiki/Zero_crossing

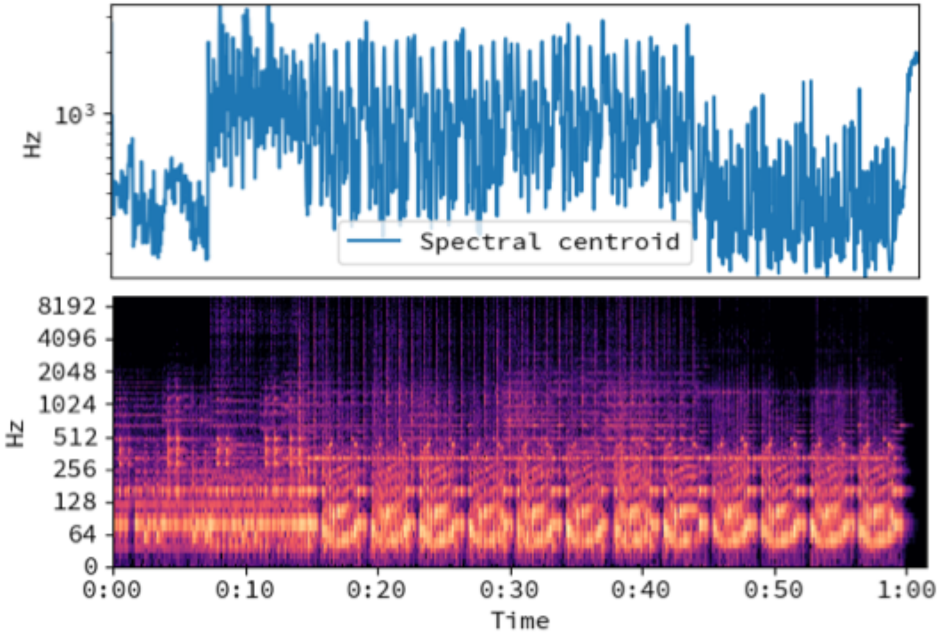
Commonly used audio features

- Zero-crossing rate



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



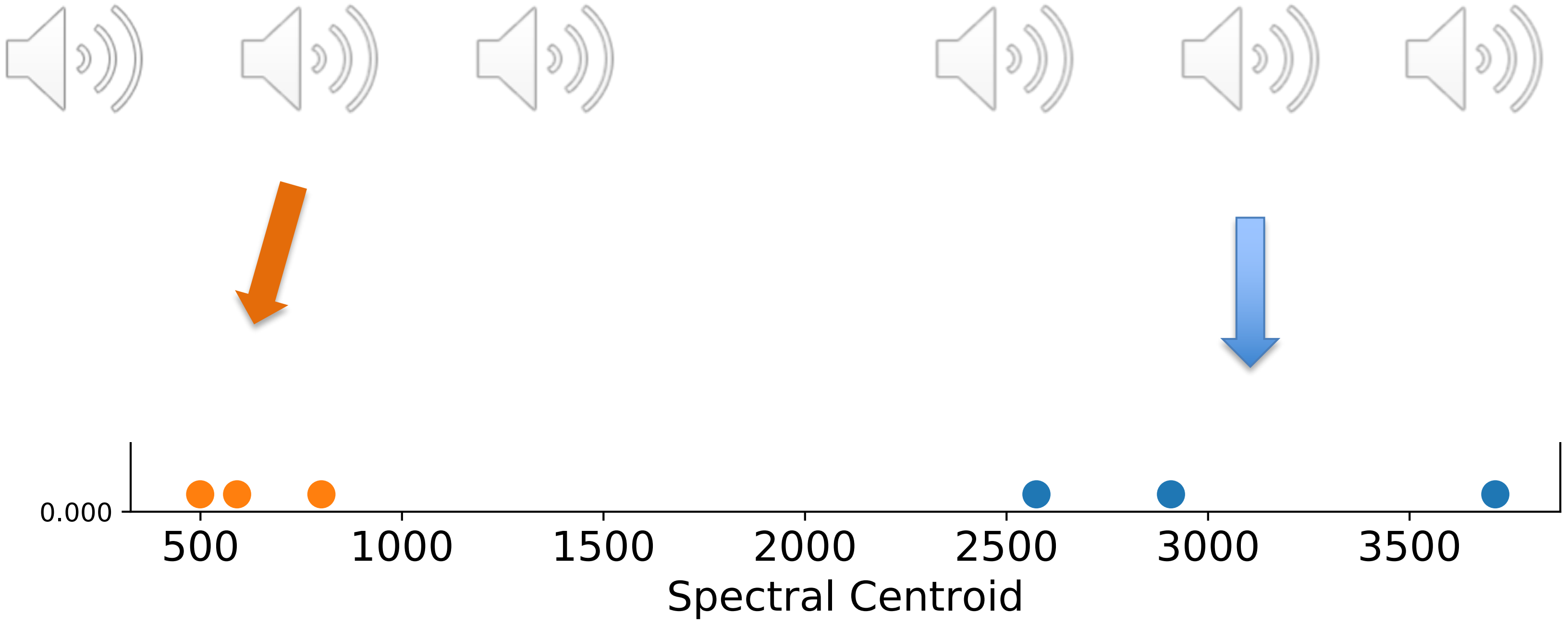
* figure: https://librosa.github.io/librosa/generated/librosa.feature.spectral_centroid.html

Commonly used audio features

- Spectral centroid

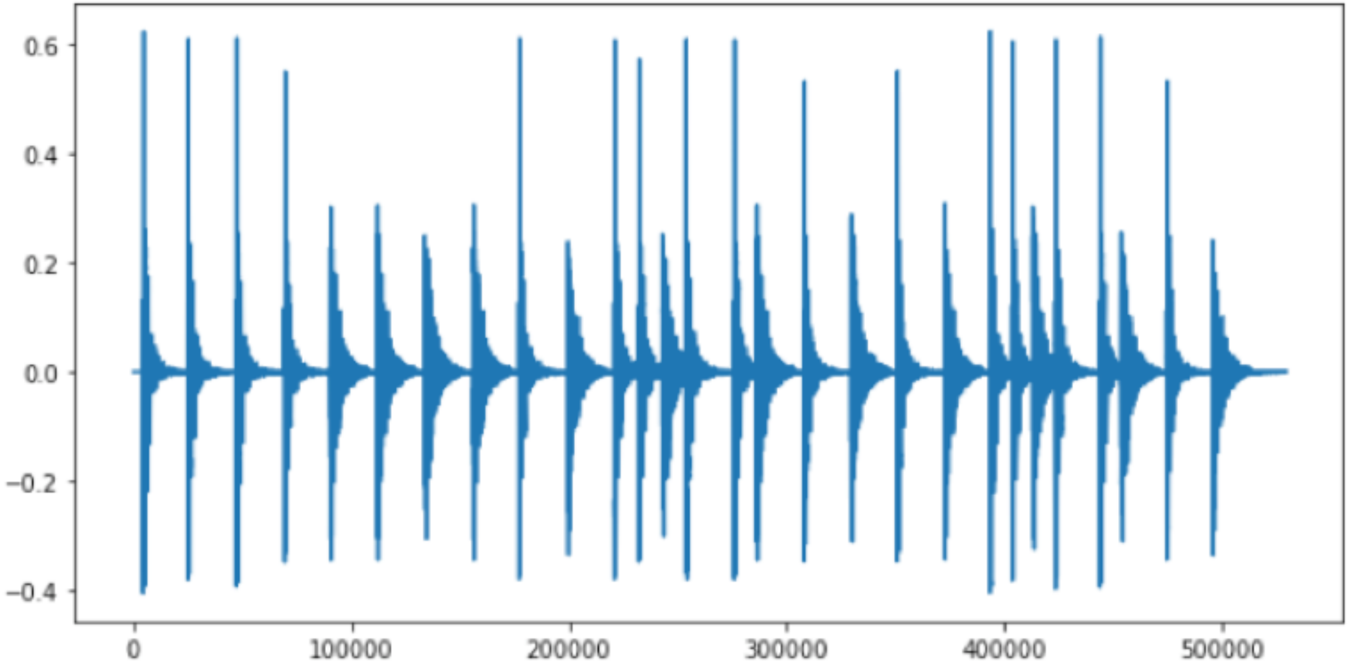
Kick drum

Snare drum



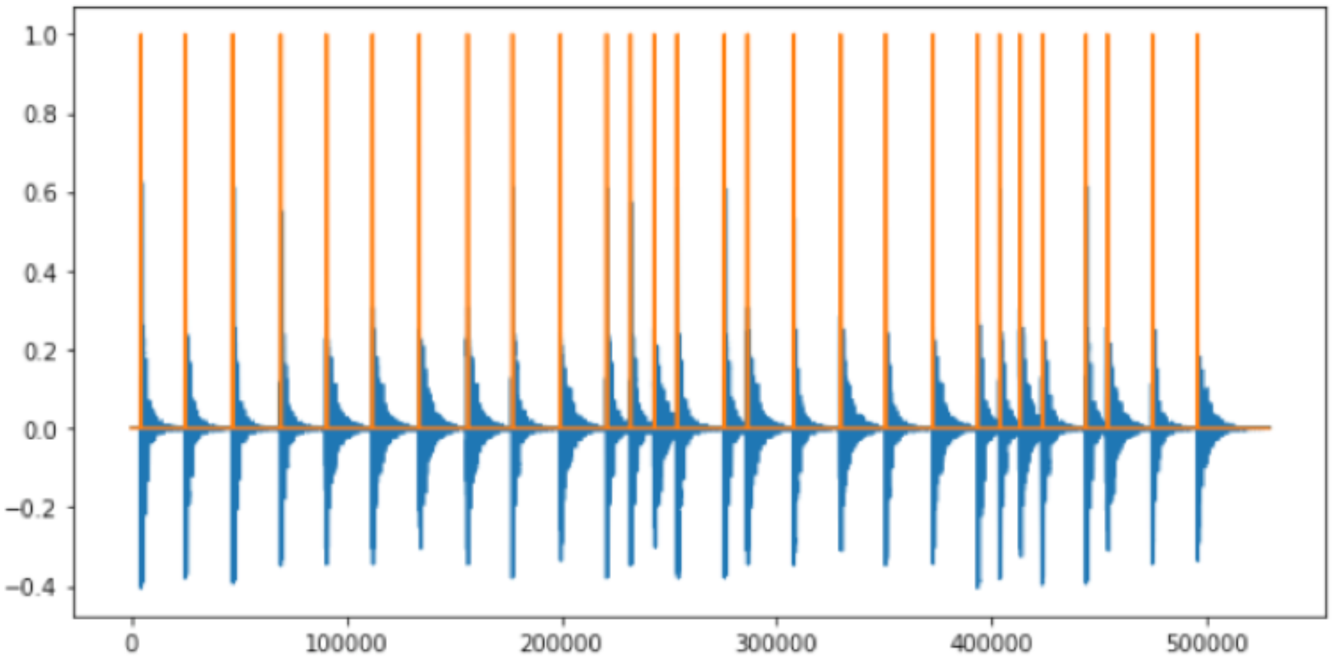
Automatic drum transcription

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



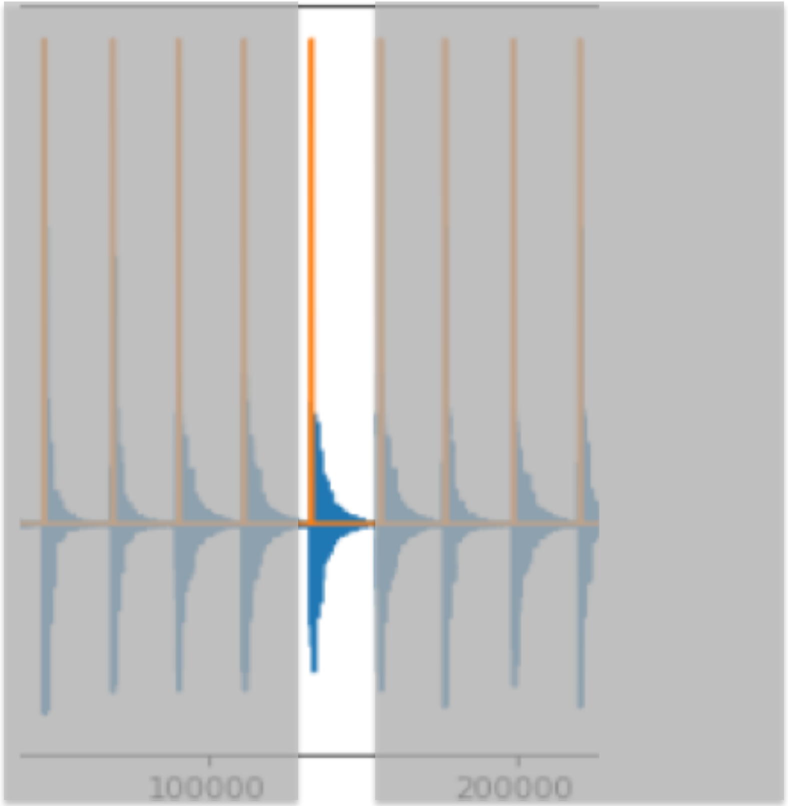
Automatic drum transcription

- Onset detection
  - **librosa.onset.onset_detect**



Automatic drum transcription

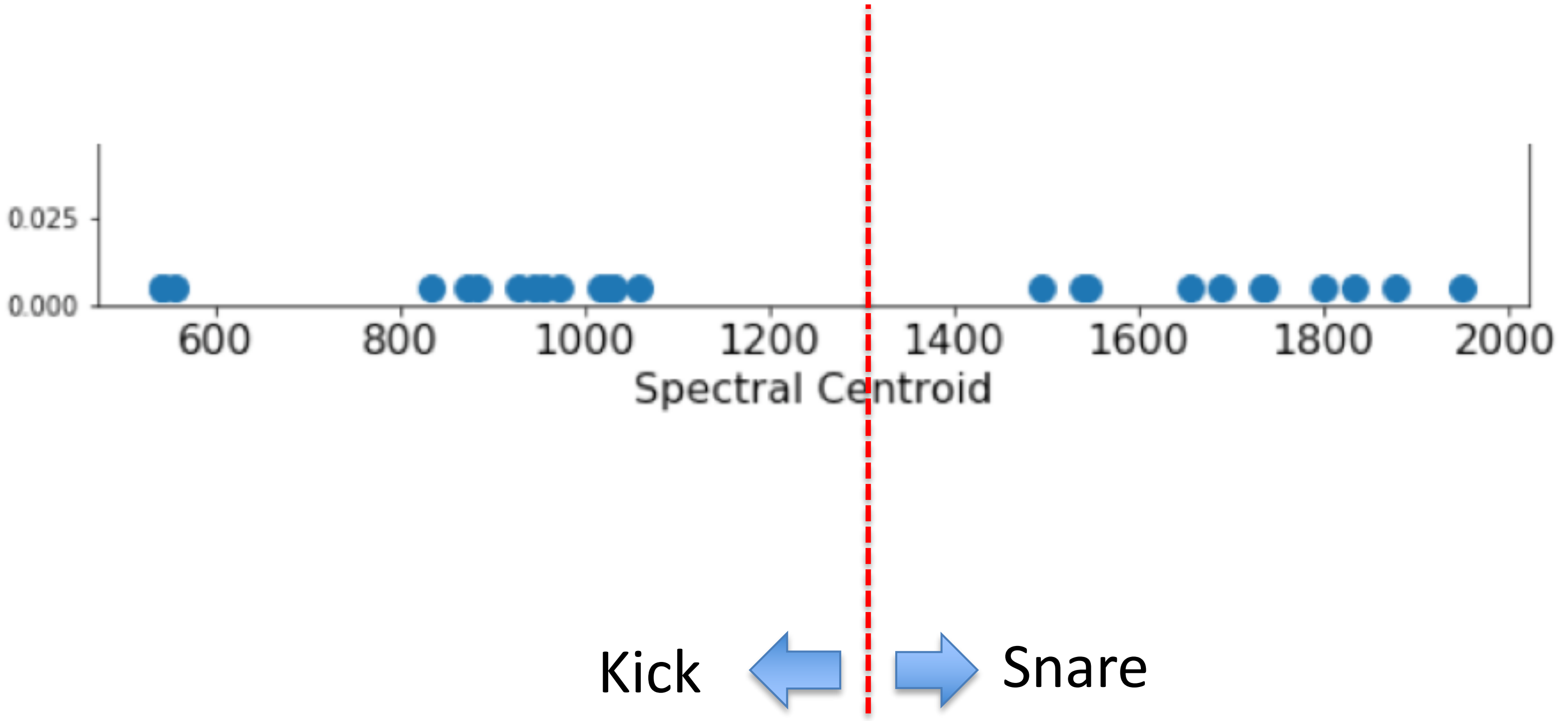
- Segmentation
  - Cutting the recording every *<onset−2048 samples>*



(Onset[t − 1] − 2048 , Onset[t] − 2048)

Automatic drum transcription

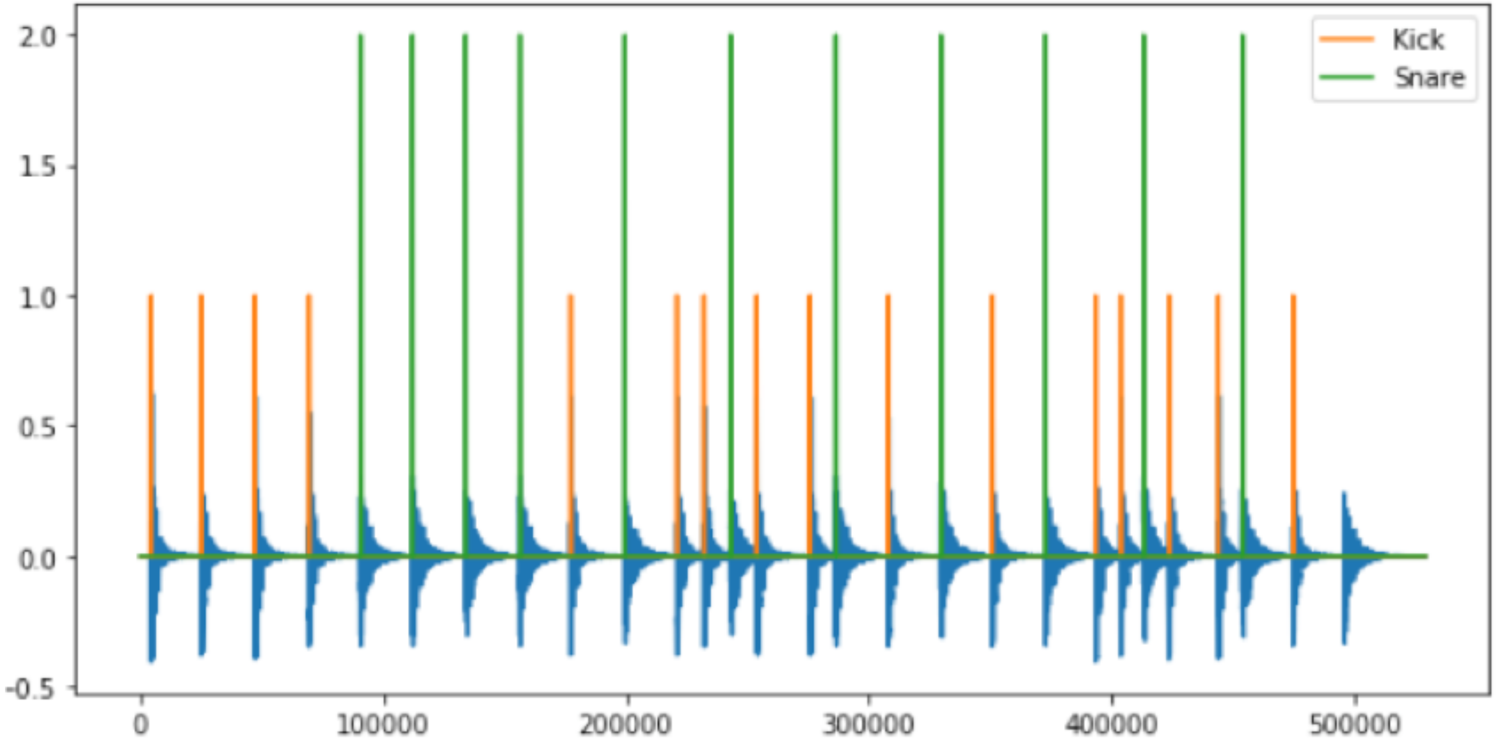
- Extracting spectral centroid from each segment



Kick

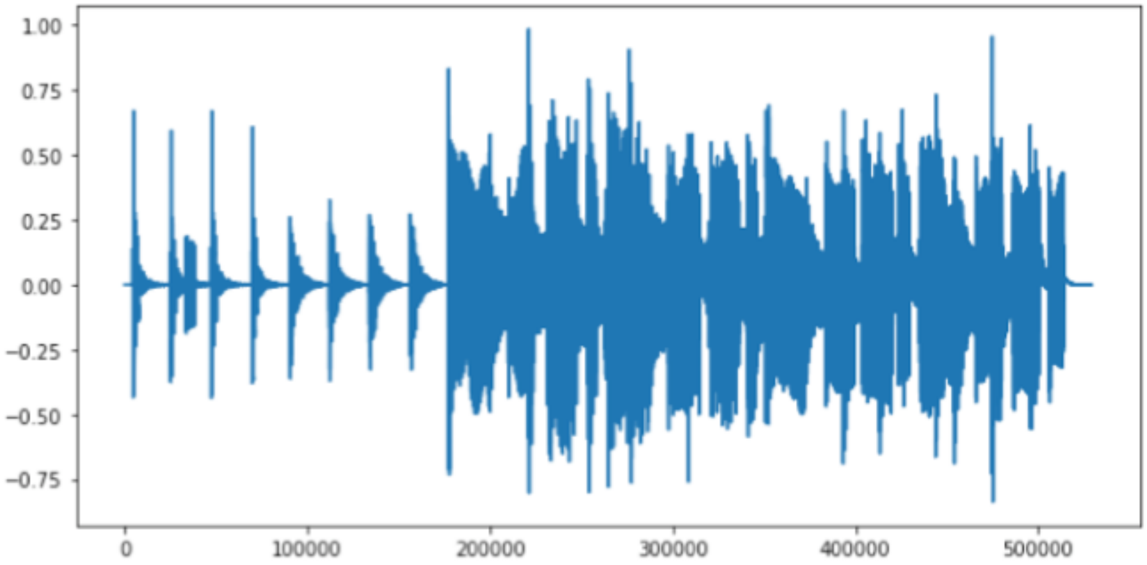
Snare

Automatic drum transcription



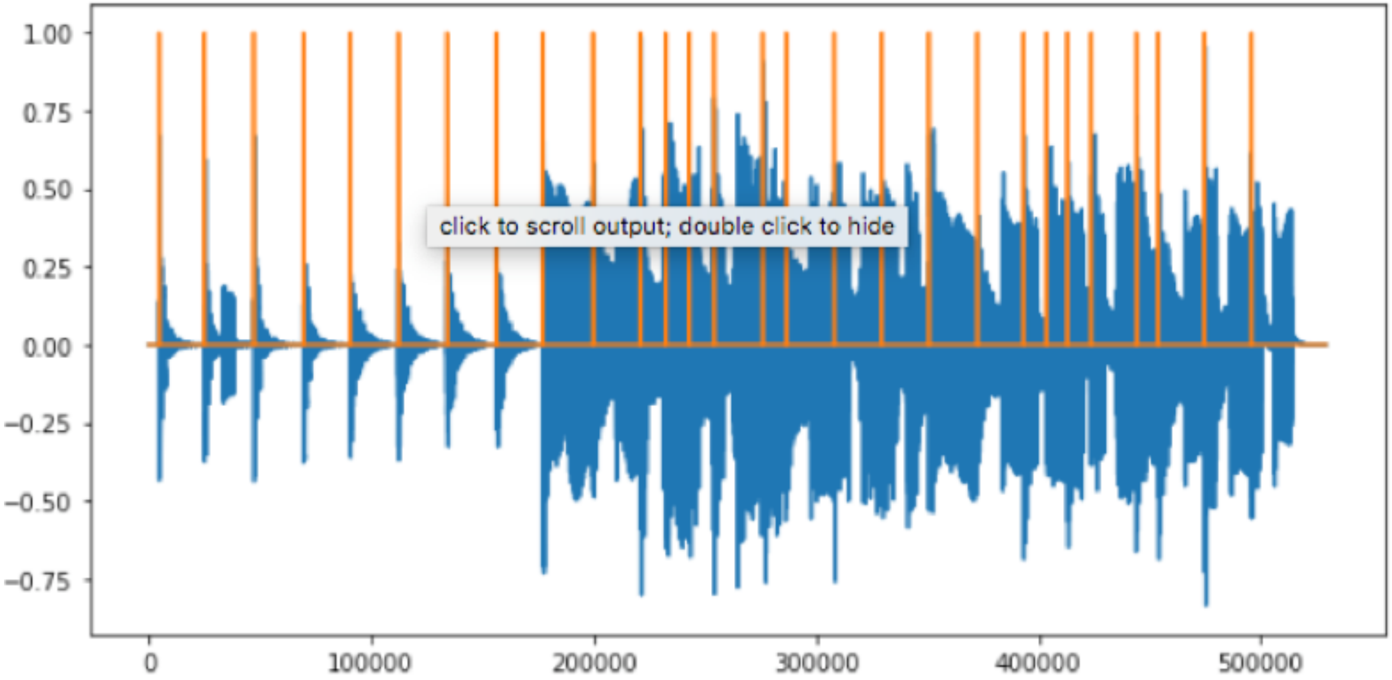
Automatic drum transcription-2

- More challenging example



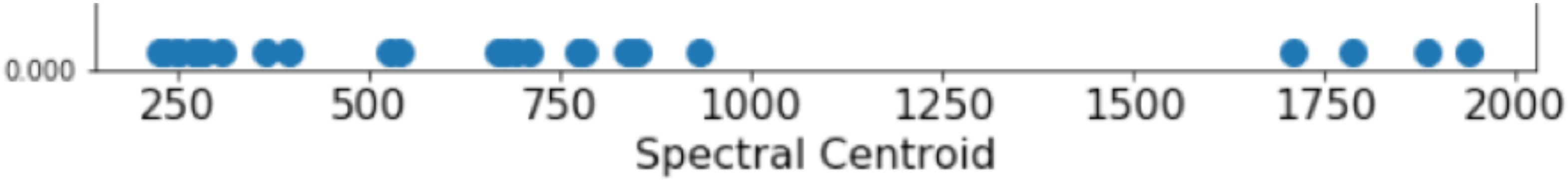
Automatic drum transcription-2

- Onset detection might not work that well on this example, but let's assume we have perfect onset info

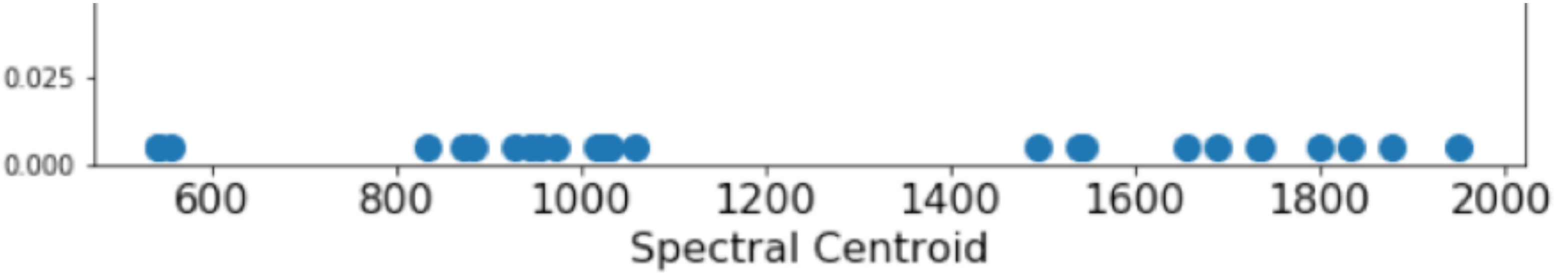


Automatic drum transcription-2

- Segmentation and feature extraction

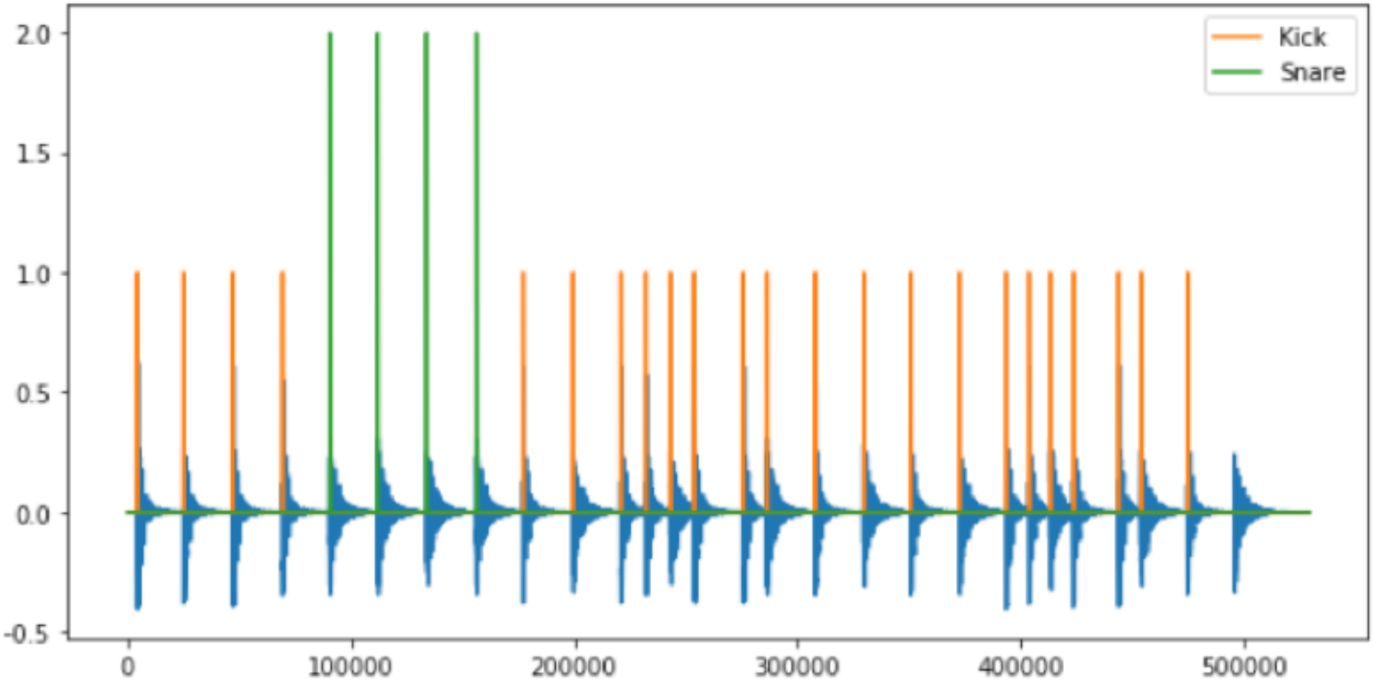


- The previous example



Automatic drum transcription-2

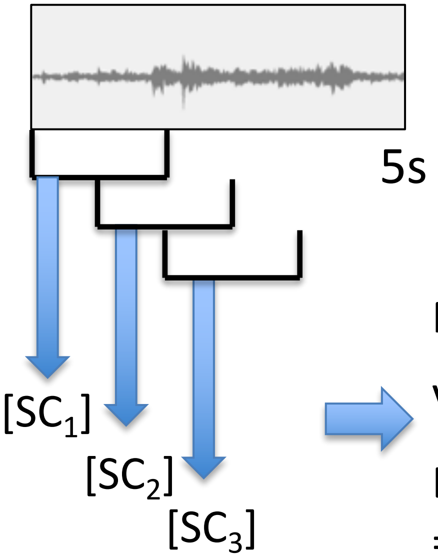
- More challenging example



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



$\vec{x}$ = <a_1, a_2, …, a_n>

Mean([SC_1, SC_2, SC_3, …, SC_t]) = a_1

Variance([SC_1, SC_2, SC_3, …, SC_t]) = a_2

Delta-mean([SC_1, SC_2, SC_3, …, SC_t])
= mean([$SC_2 - SC_1$, $SC_3 - SC_2$,…, $SC_t - SC_{t-1}$]) = a_3

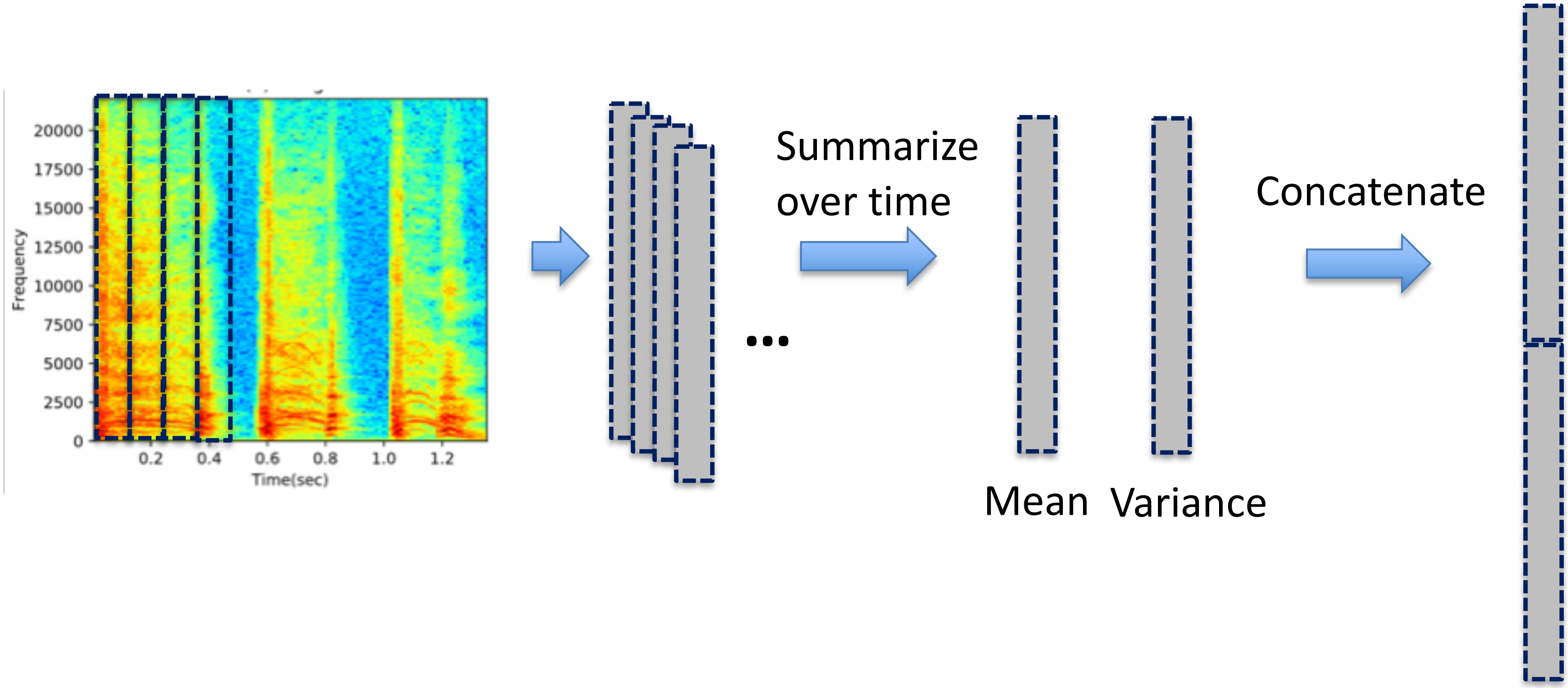
Delta-var([SC_1, SC_2, SC_3, …, SC_t])
= var([$SC_2 - SC_1$, $SC_3 - SC_2$,…, $SC_t - SC_{t-1}$]) = a_4

$\vec{x}$ = <a_1, a_2, a_3, a_4>

*SC: Spectral Centroid

Feature summarization

- Example for multi dimensional features



Example using a TINY spectrogram

Spectrogram

frequency

1	0	3	1	5
0	.4	0	.4	.2
0	29	1	20	10
10	10	10	10	10
0	0	0	50	0

Time

Mean
1
.2
12
10
10

Variance
3.2
0.03
124.4
0
400

Example using a TINY spectrogram

Spectrogram

frequency

1	0	3	1	5
0	.4	0	.4	.2
0	29	1	20	10
10	10	10	10	10
0	0	0	50	0

Time

Delta

-1	3	-2	4
.4	-.4	.4	-.2
29	-28	19	-10
0	0	0	0
0	0	50	-50

frame at (*t+1*) – frame at *t*

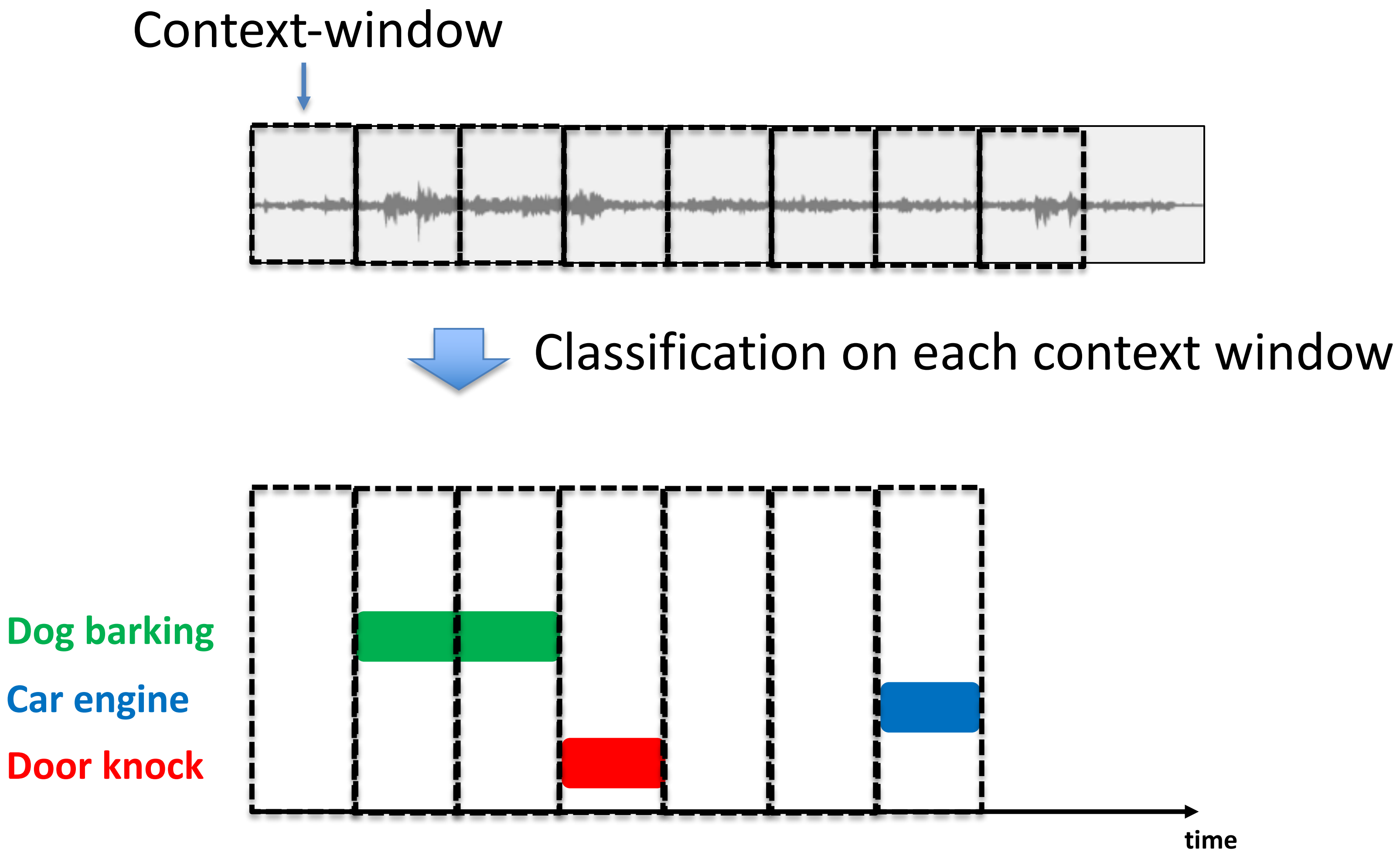
Delta -Mean	Delta -Variance
1	6.5
.05	.13
2.5	515.3
0	0
0	1250

Example using a TINY spectrogram

Mean	Variance	Delta -Mean	Delta -Variance
1	3.2	1	6.5
.2	0.03	.05	.13
12	124.4	2.5	515.3
10	0	0	0
10	400	0	1250

➔ 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

time

Challenges

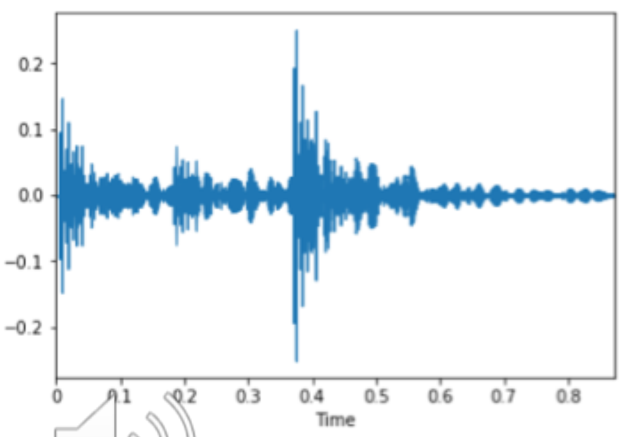
- Polyphonic environment, background noise
- Noisy labels
- Using a hierarchical relationship between audio labels
- Weakly labeled training dataset
- A small amount of **labeled** training dataset
- A large amount of **unlabeled** training dataset

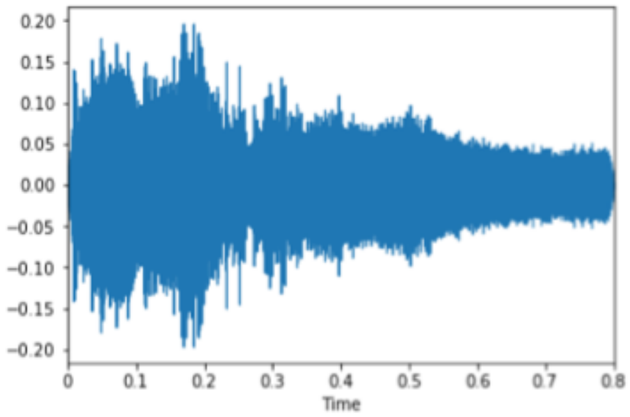
Datasets for sound object labeling

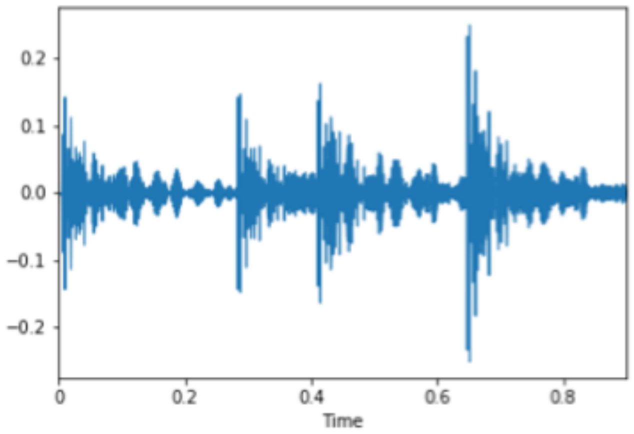
- Urban sound dataset: https://urbansounddataset.weebly.com/
- AudioSet: https://research.google.com/audioset/
- ESC: https://github.com/karoldvl/ESC-50
- DCASE: http://dcase.community/challenge2018/index
- IRMAS: https://www.upf.edu/web/mtg/irmas
- Vocal Imitation Set: https://zenodo.org/record/1340763#.XEtAJs9KiRs

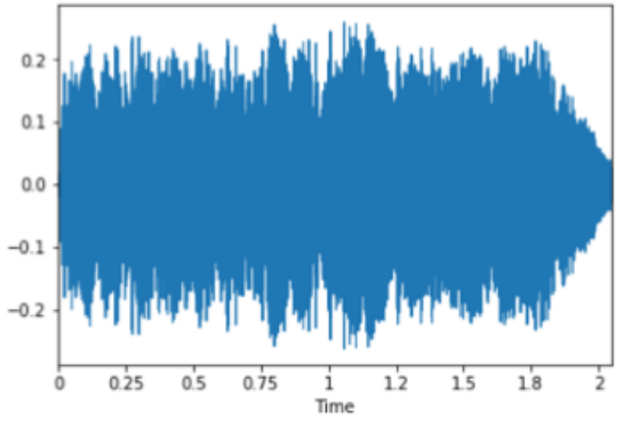
EXAMPLE: DOOR KNOCKING / PHONE RINGING CLASSIFICATION

Training data





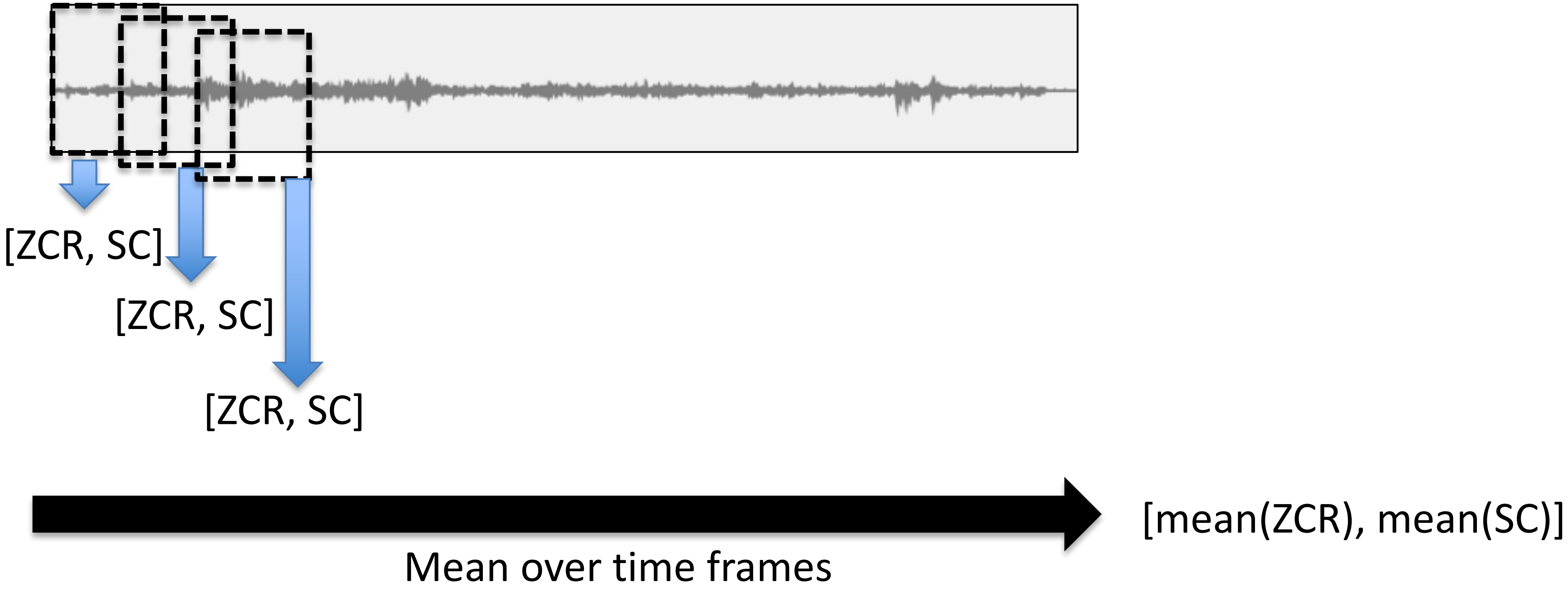




Feature extraction and summarization

- 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



[ZCR, SC]

[ZCR, SC]

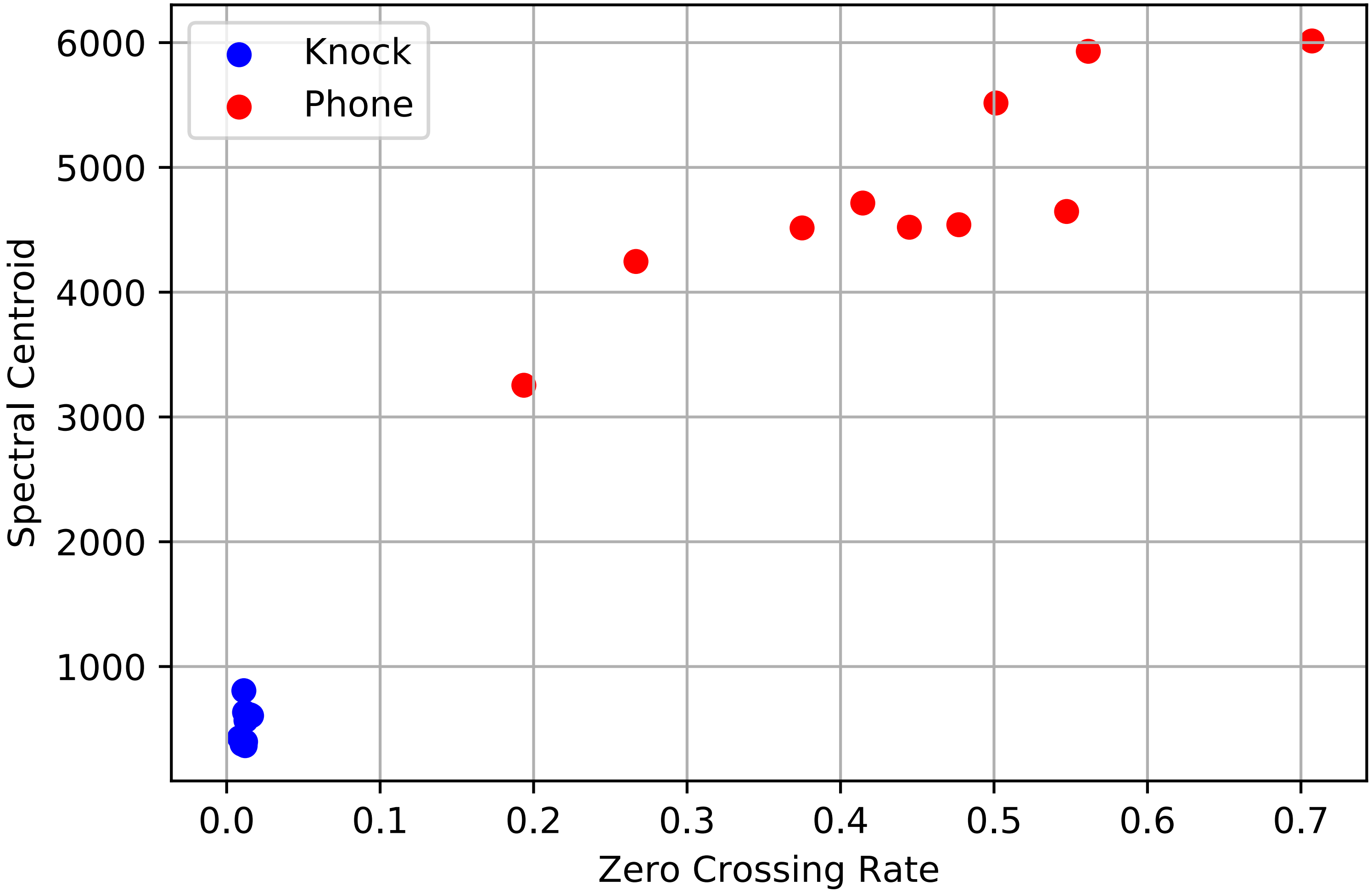
[ZCR, SC]

[mean(ZCR), mean(SC)]

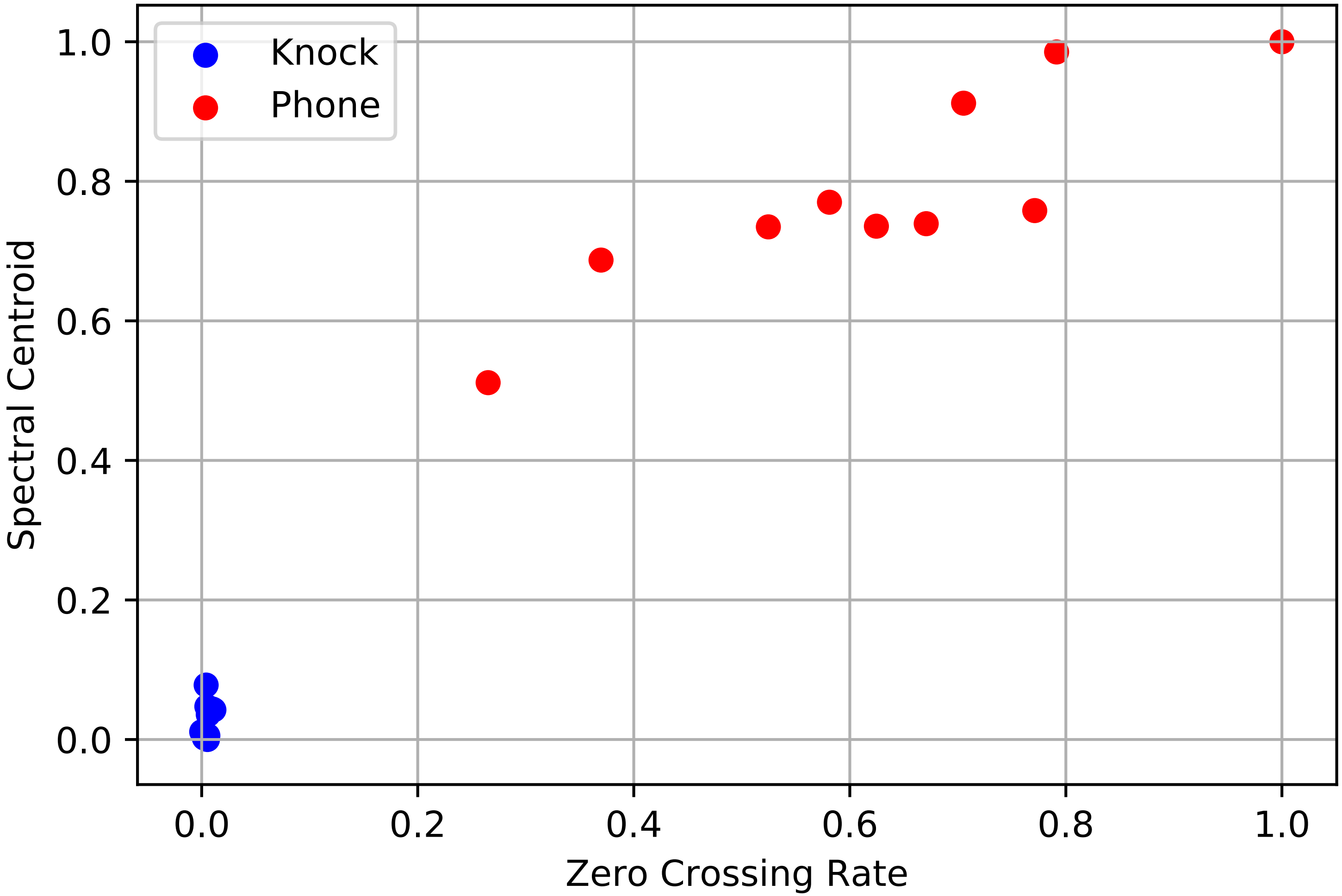
Mean over time frames

Now we can map all the signals into 2-dimensional feature space

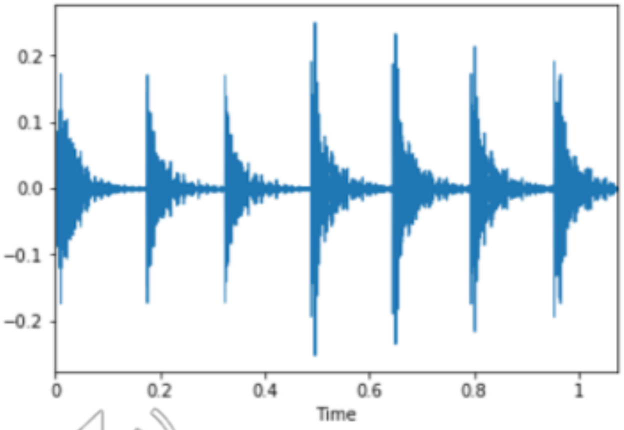
Plotting them in the feature space

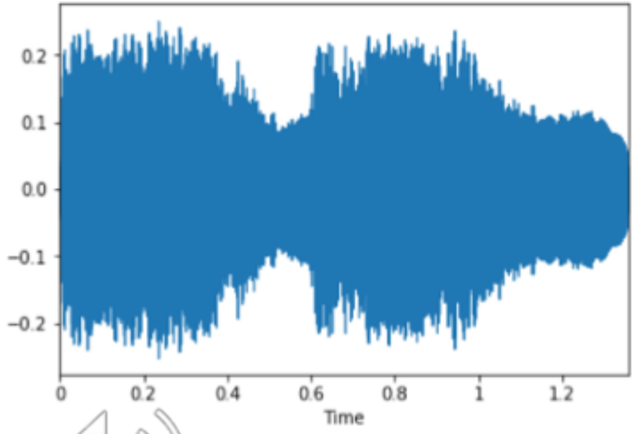


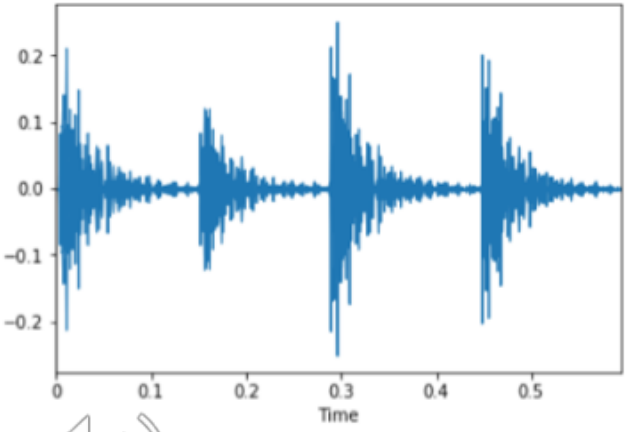
Feature scaling

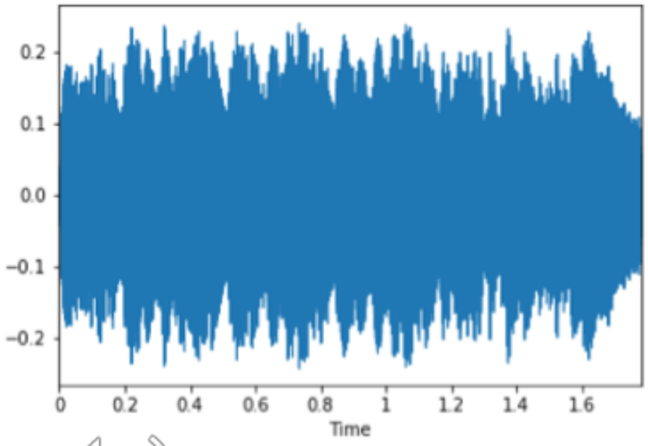


Testing examples



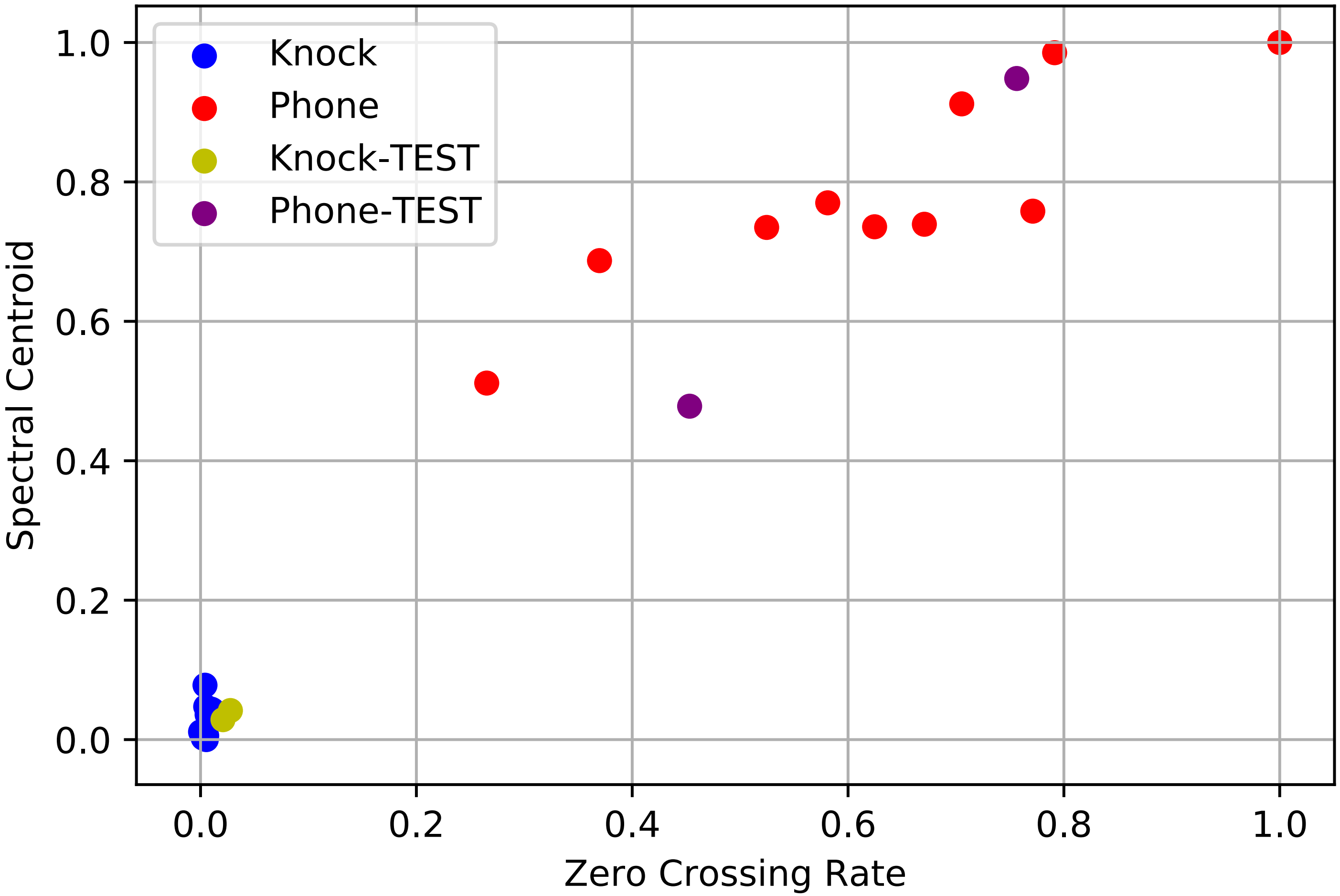






Plotting test examples

Nearest Neighbor classifier would perfectly work in this testing case


Knock
Phone
Knock-TEST
Phone-TEST
Spectral Centroid
Zero Crossing Rate