2025 edition

Deep Learning for Music Analysis and Generation

Introduction

Intros + course logistics

Yi-Hsuan Yang Ph.D.
yhyangtw@ntu.edu.tw

Course Website

https://cool.ntu.edu.tw/courses/52013

https://affige.github.io/teaching.html

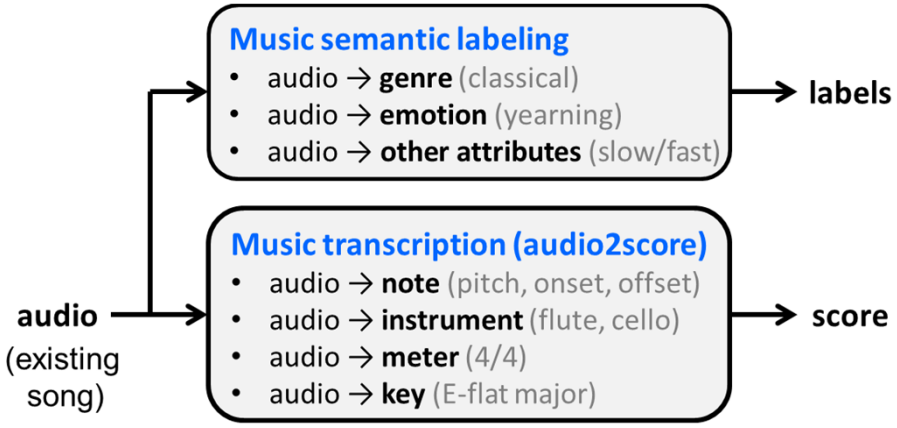
https://github.com/affige/DeepMIR/

Outline

- **Intros of Music & AI**
- Course logistics
- Fundamentals of musical audio
- Appendix: ML 101

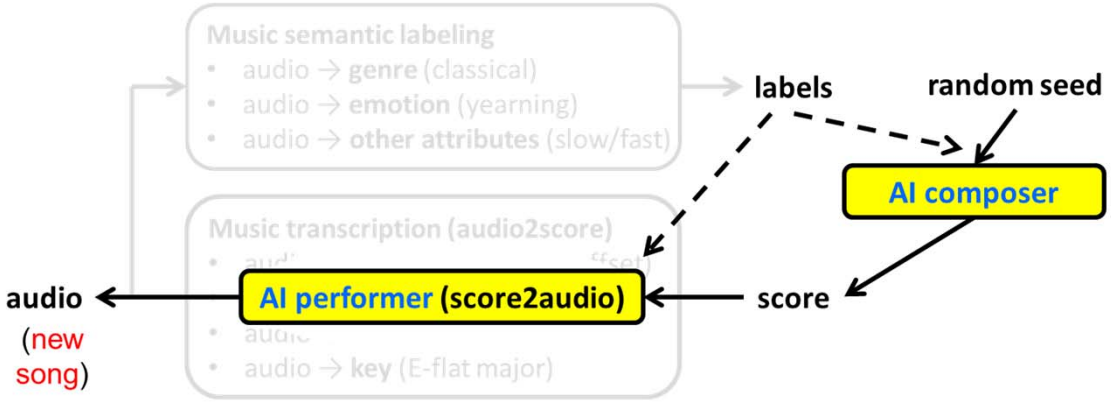
Music AI; or *Music Information Research* (MIR)

- **Music analysis**



  - music understanding
  - music search
  - music recommendation

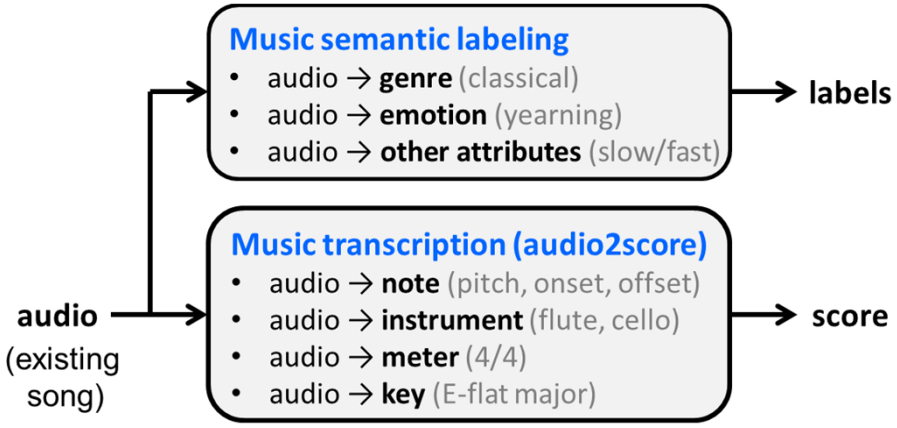
- **Music generation**



  - MIDI generation
  - audio generation
  - MIDI-to-audio generation

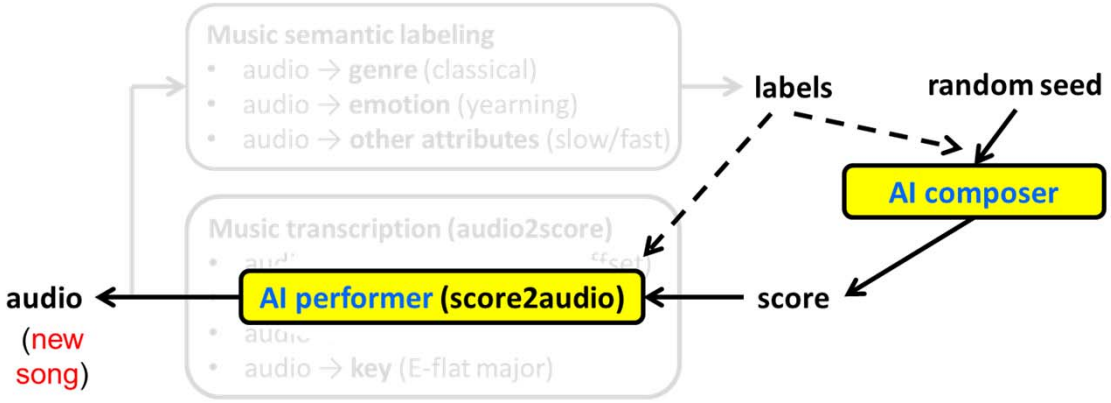
Music AI: {signal processing, machine learning} + music

- **Music analysis**



  - music understanding
  - music search
  - music recommendation

- **Music generation**

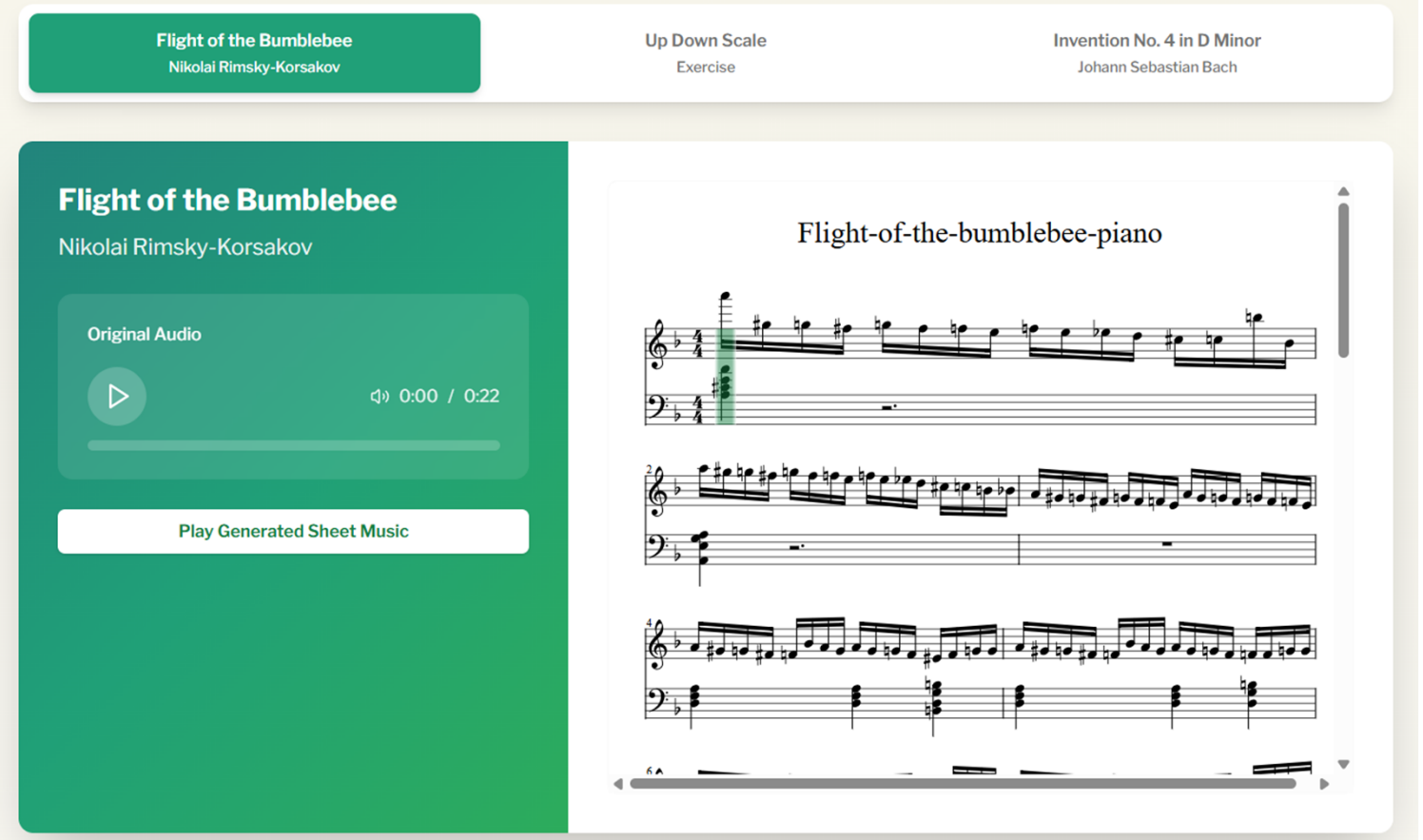


  - MIDI generation
  - audio generation
  - MIDI-to-audio generation

Music Analysis Demo 1: Music Transcription

https://songscription.ai/

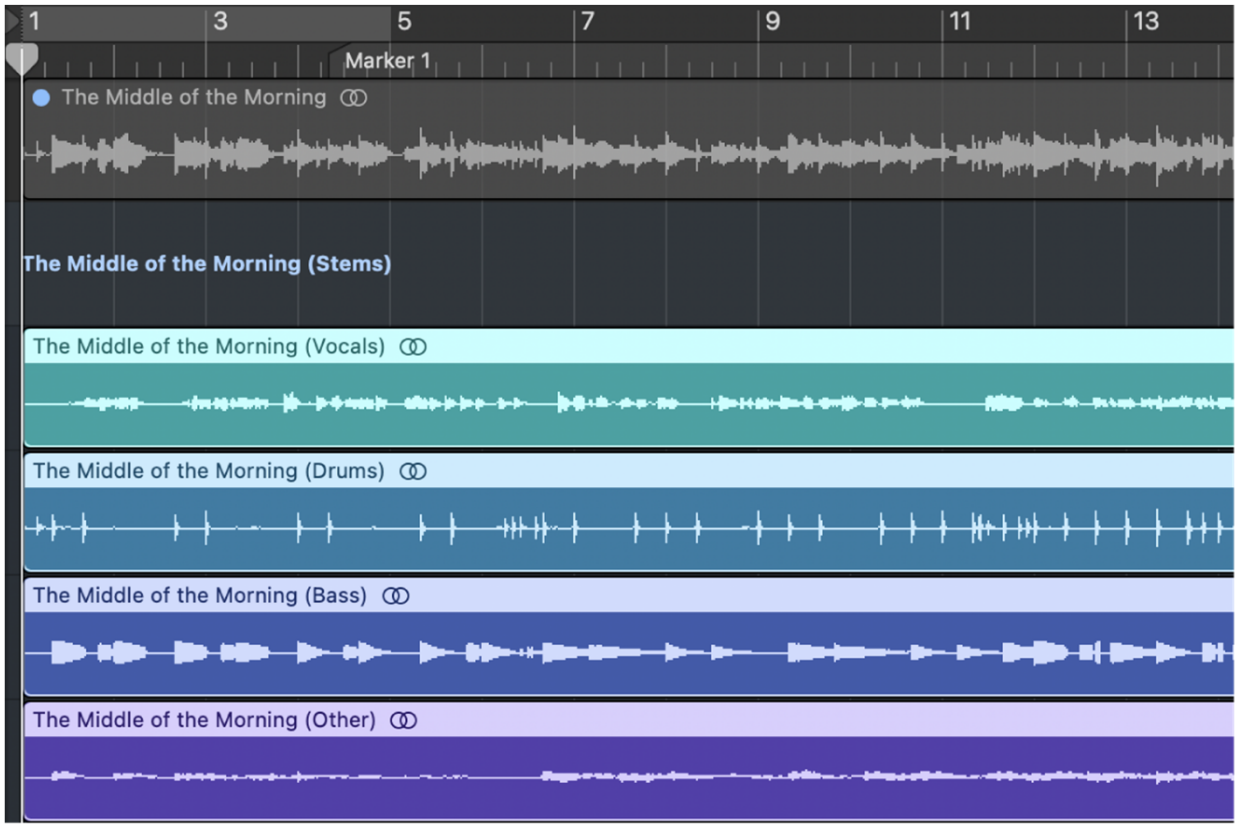
- songscription.ai
- AnthemScore
- Melody Scanner
- Klangio



Music Analysis Demo 2: Source Separation

https://www.youtube.com/watch?v=qQUtCipFRHQ

- Apple Logic Pro
- iZotope RX
- Gaudio Studio
- RipX
- moises.ai
- lalal.ai



Source: https://support.apple.com/en-md/guide/logicpro/lgcp61bae908/mac

Music Analysis Demo 3: Audio Understanding

https://research.nvidia.com/labs/adlr/AF3/

Audio Flamingo 3

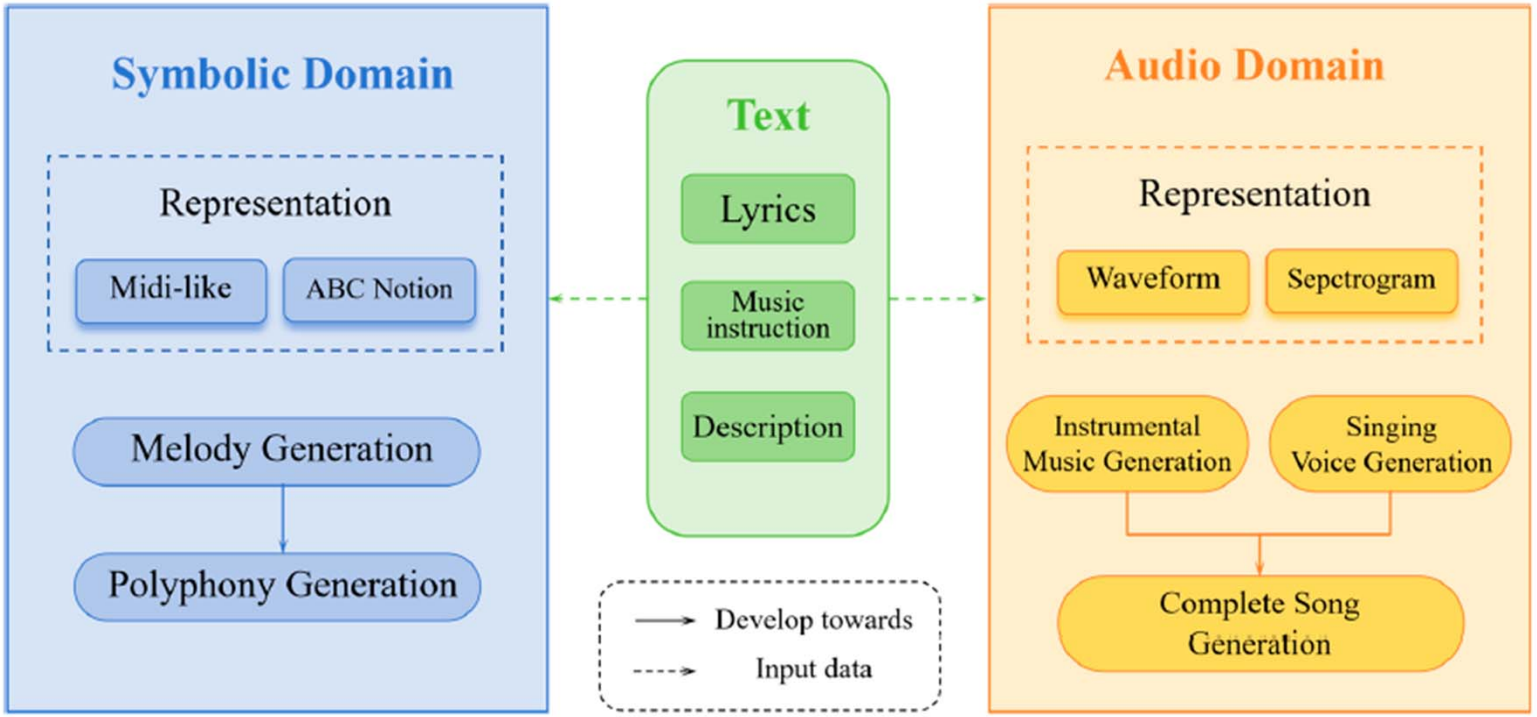
Audio	Instruction	Ground-truth	Audio Flamingo 3
0:02 / 0:09	What is surprising about the relationship between the barking and the music?	Dogs barking in sync with the music	The **barking and music are synchronized**, creating a rhythmic and harmonious interaction.
0:11 / 0:19	What makes this saxophone performance unlike typical saxophone sounds?	Aside from "Interstellar Spaces", Saxophones do not bark nor meow	The **saxophone is played in a way that resembles a dog howling**, creating a unique and unusual sound.
0:09 / 0:09	What makes this combination of acoustic and ambient sounds unlikely to appear together?	Unlikely existence of banjo and rain sounds at the same time.	The **contrast between the lively music and the sound of raindrops** creates an unusual auditory experience.

Positive Use Cases of Music Generation

- **Make music easier to play with** (i.e., for common people)
  - "democratization" of music creation
- **Make musicians' life easier** (i.e., for musicians)
  - inspire ideas
  - suggest continuations
  - suggest accompaniments
- **Create copyright free music for videos or games**
- **Music education / learning**
- **Digital archival**

Evolution of Music Generation Research

- Monophonic MIDI (melody) → polyphonic MIDI (e.g. piano)
- Single-instrument MIDI → multi-instrument MIDI
- MIDI → audio
- Solo → full song

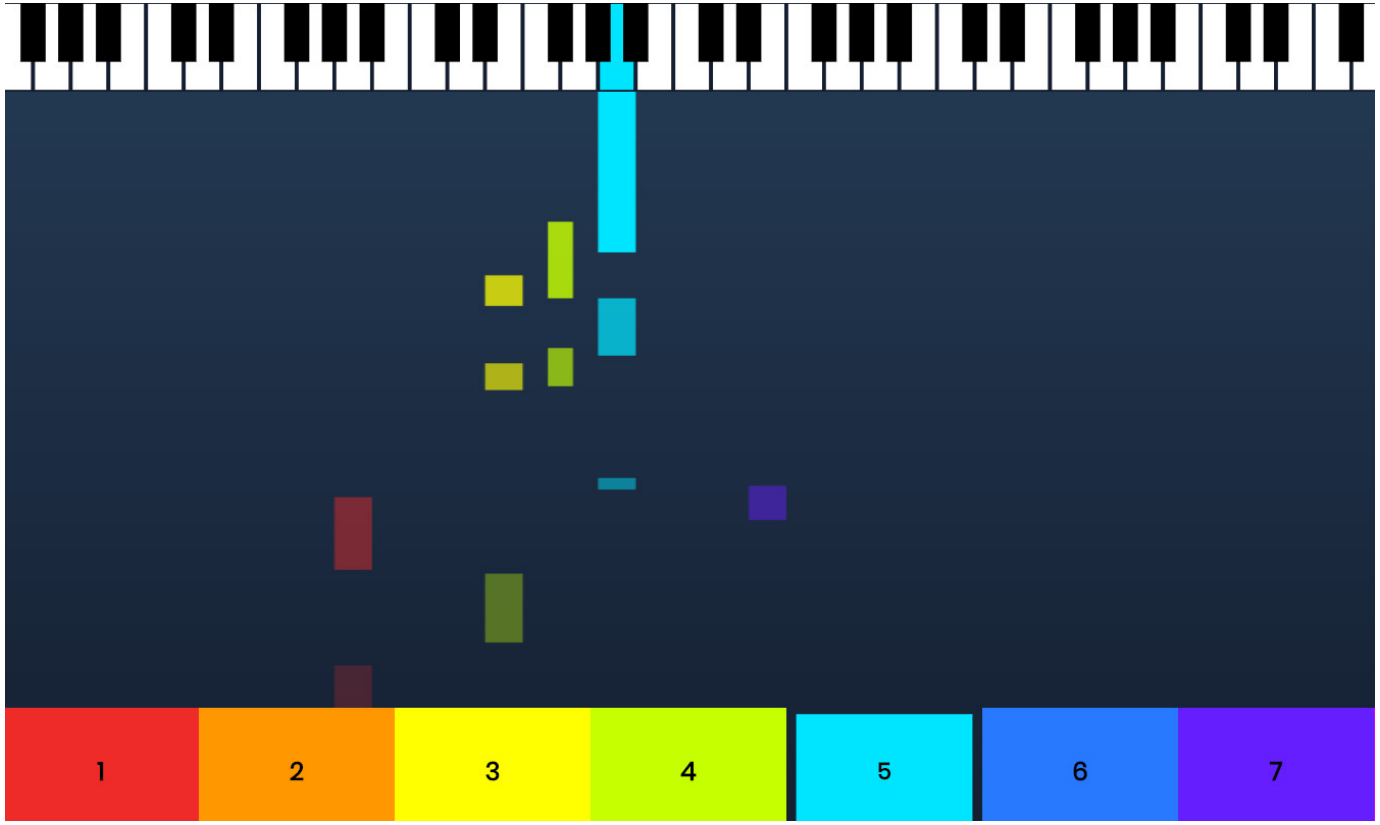


Source: https://www.mdpi.com/2079-9292/14/6/1197

Music Generation Demo 1: Piano Genie

(Make music easier to play with)

https://magenta.tensorflow.org/pianogenie



Music Generation Demo 2: Tone Transfer

(Make music easier to play with)

https://sites.research.google/tonetransfer

Music Generation Demo 3: Song Generation

(Create copyright free music; **for musicians**)

https://www.youtube.com/watch?v=MCLVhRjIvys

moises

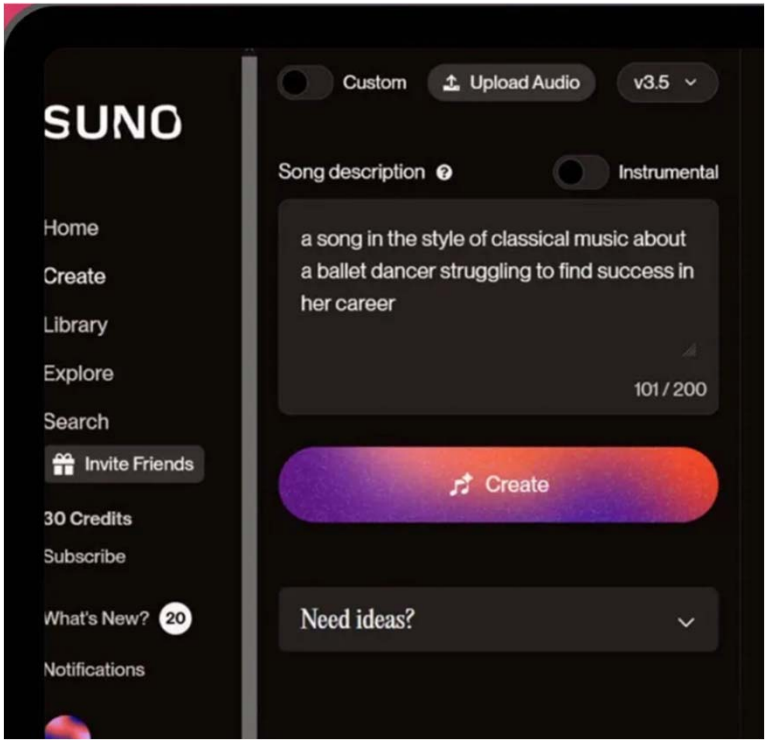
Music Generation Demo 4: Song Generation

(Create copyright free music; **for common people**)

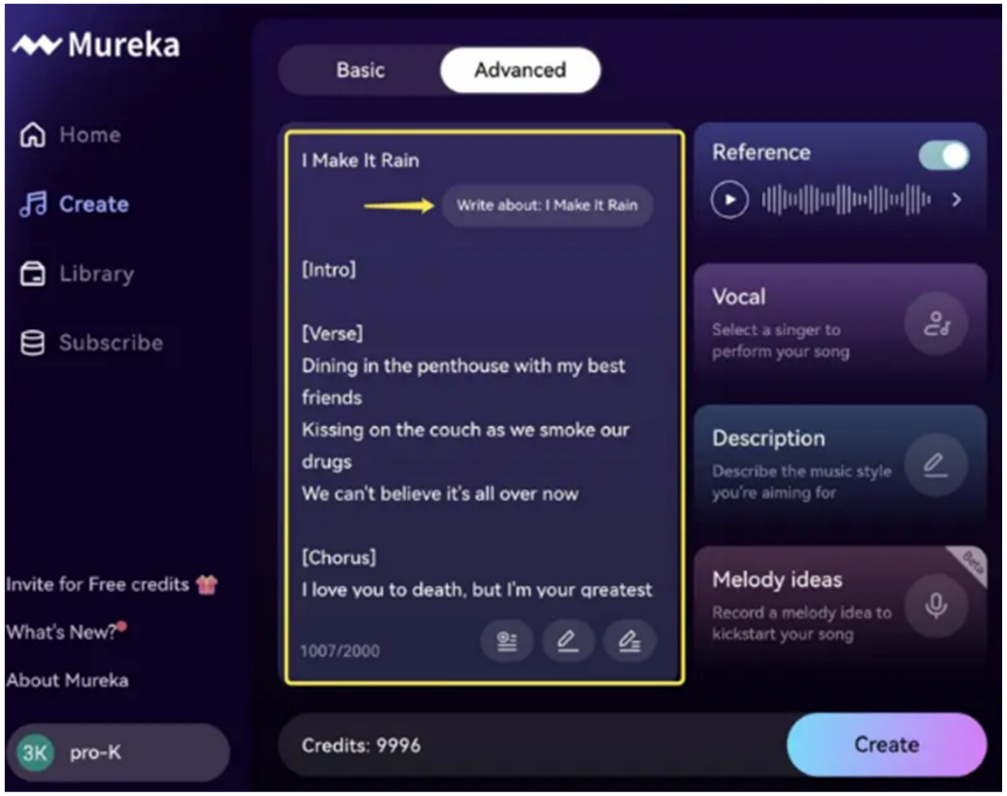
https://www.youtube.com/watch?v=zWYSIudPNfI

Suno

Mureka



https://suno.com/



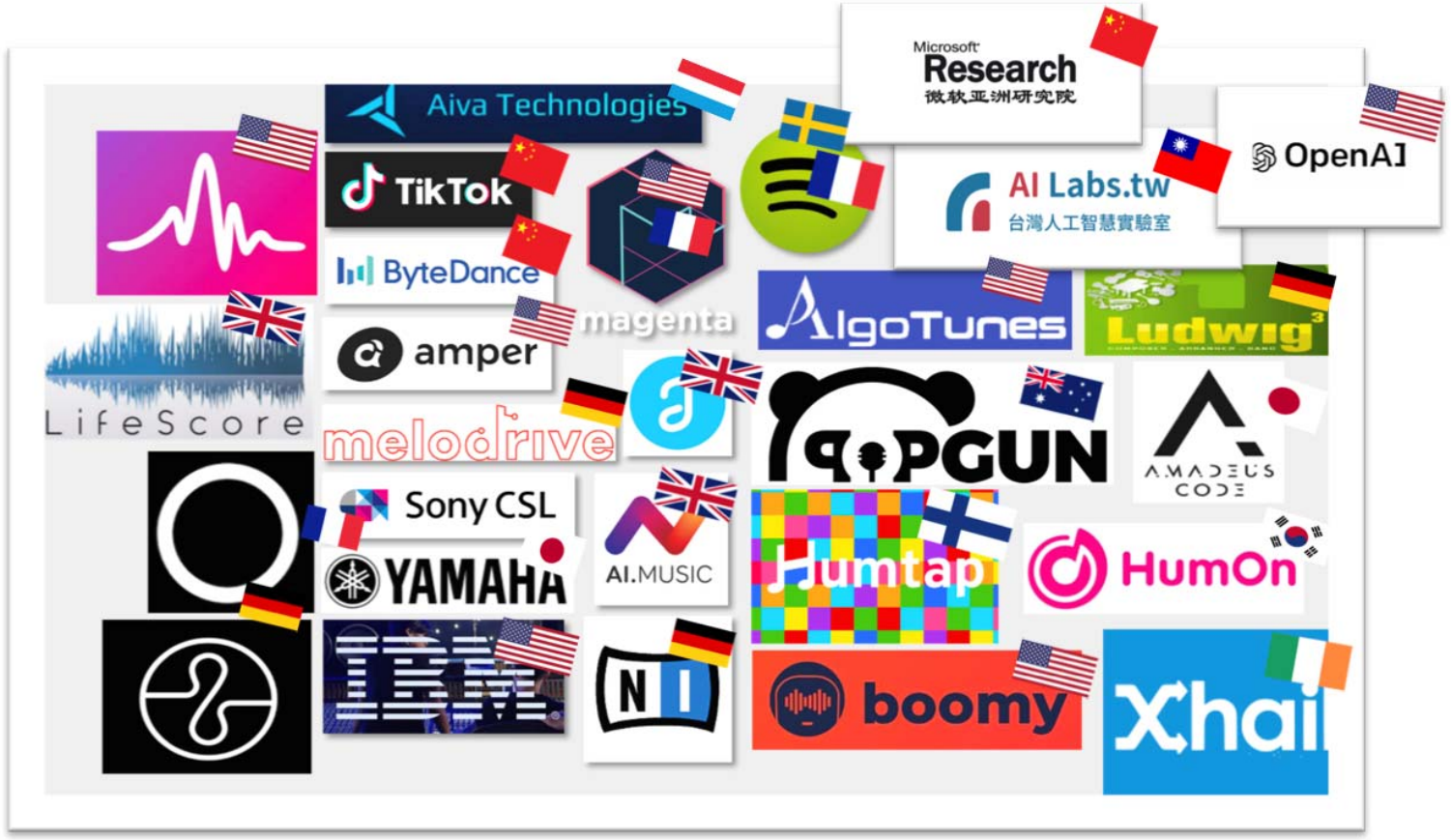
https://www.mureka.ai/

Music Generation Demo 5: Text-to-Music/Video

(Create copyright free music for videos or games)

https://www.youtube.com/watch?v=8S95f1BR1Ls

Global Interest in Music AI: Industry



(**Slide from Rujing Huang, Bob L. T. Sturm, and Andre Holzapfel**, "De-centering the West: East Asian Philosophies and the Ethics of Applying Artificial Intelligence to Music," ISMIR 2021)

Global Interest in Music AI: Academia

- MIT EECS | Anna Huang (https://czhuang.github.io/)
- CMU CS | Chris Donahue (https://chrisdonahue.com/)
- U Michigan SMTD | Hao-Wen Dong (https://hermandong.com/)
- Stanford CCRMA | Ge Wang (https://ccrma.stanford.edu/~ge/)
- NYU MARL | Brian McFee (https://brianmcfee.net/)
- U Rochester | Zhiyao Duan (https://hajim.rochester.edu/ece/sites/zduan/)
- Georgia Tech | Alexander Lerch (https://www.alexanderlerch.com/)
- Centre for Digital Music, Queen Mary University of London (https://www.c4dm.eecs.qmul.ac.uk/)
- Academia Sinica | Li Su (https://homepage.iis.sinica.edu.tw/pages/lisu/)

Music Generation: One More Example

Outline

- Intros of Music & AI
- **Course logistics**
- Fundamentals of musical audio
- Appendix: ML 101

This Course: Prerequisites

- **Graduate level** (CommE5070) @ NTU GICE
  - It's ***NOT*** a music course
  - It's an EE/CS graduate-level course working on music data/problems
- Suitable for people who have
  - Great interest in music
  - Good background in machine learning & deep learning
  - Good coding experience in python and a deep learning framework such as PyTorch
- Not a class designed for deep learning beginners

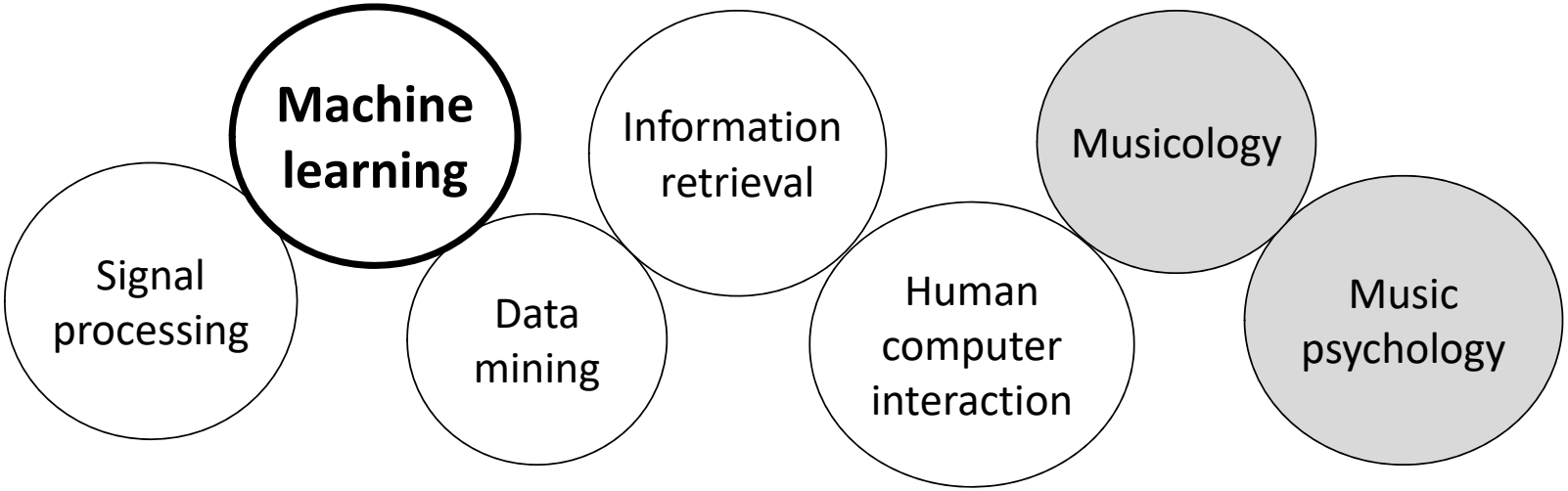
This Course: Wills and Won’ts

- Will talk about
  - Domain knowledge in music data representation
  - Domain knowledge in music analysis: timbre, rhythm, pitch
  - Deep learning-based music analysis
  - Deep learning-based audio generation
  - Deep learning-based MIDI sequence generation

- **Won’t** talk about (too much on)
  - Basics in machine learning and deep learning
  - Applications in other domains

This Course: Objective

- Get you ready to do **ML** research on music AI
- Lots of hands-on



Lecturer

- Lecturer
  - Yi-Hsuan Yang (楊奕軒)
    - https://affige.github.io/
    - yhyangtw@ntu.edu.tw

- Office hour
  - Thursday 9:30-11:30, or by appointment
  - Office: EE2-337 (電二)

Teaching Assistants

- TAs
  - Wei-Jaw Lee (李維釗)
    - weijaw2000@gmail.com
  - Fang-duo Tsai (蔡芳鐸)
    - fundwotsai2001@gmail.com

- Office hour
  - Monday 13:30-15:00, or by appointment
  - Office: BL-505 (博理館)

Location & Time

- Location:學新118
- Time: Thursday 6,7,8
  - **6**: 13:20-14:10
  - **7**: 14:20-15:10
  - **8**: 15:20-16:10 (i.e., 10 mins earlier)

Online Option

- Live through Google Meet: http://meet.google.com/fem-ouef-fby

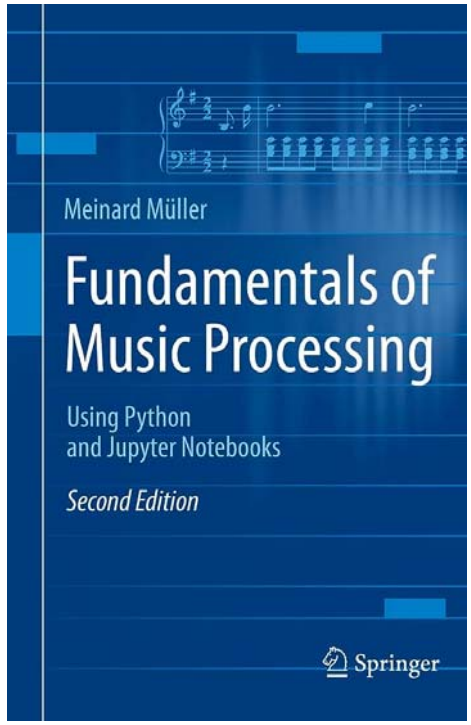
Textbook

(for the music analysis part)

- Reference textbook

Meinard Müller
Fundamentals of Music Processing
Using Python and Jupyter Notebooks

ISBN: 978-3-030-69808-9
Springer, April 2021
https://www.audiolabs-erlangen.de/fau/professor/mueller/bookFMP
https://www.audiolabs-erlangen.de/resources/MIR/FMP/C0/C0.html



- Related book
  - *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*, Wiley
    https://github.com/alexanderlerch/pyACA
    https://github.com/alexanderlerch/ACA-Slides

FMP Notebook

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C0/C0.html

Part	Title	Notions, Techniques & Algorithms	HTML	IPYNB
B	Basics	Basic information on Python, Jupyter notebooks, Anaconda package management system, Python environments, visualizations, and other topics	[html]	[ipynb]
0	Overview	Overview of the notebooks (https://www.audiolabs-erlangen.de/FMP)	[html]	[ipynb]
1	Music Representations	Music notation, MIDI, audio signal, waveform, pitch, loudness, timbre	[html]	[ipynb]
2	Fourier Analysis of Signals	Discrete/analog signal, sinusoid, exponential, Fourier transform, Fourier representation, DFT, FFT, STFT	[html]	[ipynb]
3	Music Synchronization	Chroma feature, dynamic programming, dynamic time warping (DTW), alignment, user interface	[html]	[ipynb]

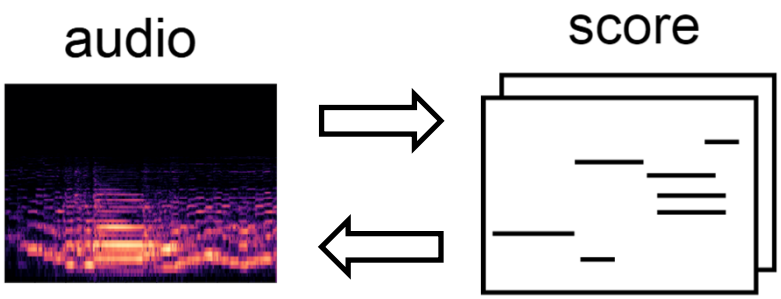
Part	Title	Notions, Techniques & Algorithms	HTML	IPYNB
4	Music Structure Analysis	Similarity matrix, repetition, thumbnail, homogeneity, novelty, evaluation, precision, recall, F-measure, visualization, scape plot	[html]	[ipynb]
5	Chord Recognition	Harmony, music theory, chords, scales, templates, hidden Markov model (HMM), evaluation	[html]	[ipynb]
6	Tempo and Beat Tracking	Onset, novelty, tempo, tempogram, beat, periodicity, Fourier analysis, autocorrelation	[html]	[ipynb]
7	Content-Based Audio Retrieval	Identification, fingerprint, indexing, inverted list, matching, version, cover song	[html]	[ipynb]
8	Musically Informed Audio Decomposition	Harmonic/percussive separation, signal reconstruction, instantaneous frequency, fundamental frequency (F0), trajectory, nonnegative matrix factorization (NMF)	[html]	[ipynb]

Grading Policy

- Grading policy
  - **Assignments** (60%), 3 times, each 20%
  - **Final Project** (40%): for teams of 2 or **3** (recommended)

- Work reasonably hard to get a high score

Course Material



Topics

- **audio → audio**: signal processing
- **audio → score**: transcription
- **score → score**: composition
- **score → audio**: synthesis
- **audio → knowledge**: audio analysis
- **score → knowledge**: symbolic-domain analysis

Syllabus

- **W1**. Intros + fundamentals of musical audio
- **W2**. Music classification + music foundation models
- **W3**. Audio codec models + audio language models + audio captioning
  - **HW1**: Singer classification
- **W4**. Audio effect modeling
- **W5**. Transformer-based music generation
- **W6**. Diffusion-based music generation
  - **HW2**: Controllable text-to-music generation
- **W7**. Singing voice generation & song generation
- **W8**. DDSP & automatic mixing
- **W9**. Fundamentals of symbolic music & symbolic MIDI generation
  - **HW3**: Piano MIDI generation
- **W10**. Advanced symbolic MIDI generation
- **W11**. Cover generation & MIDI-to-audio generation
- **W12. Project pitch** (proposals of final projects)
- **W13**. Miscellaneous Topics
- **W14**. Miscellaneous Topics
- (W15. Break)
- **W16. Oral presentation** of final projects

Guest Lectures

- **W4**. Audio effect modeling
  - by Barry & Ray from **Positive Grid**
  - https://www.positivegrid.com/

- **W8**. DDSP & automatic mixing
  - by my PhD student Yen-Tung Yeh
  - https://ytsrt66589.github.io/

- For I will be attending conferences these two weeks

Assignments

- **Programming** (in python) + **report** (in English)
  - We assume that you are comfortable with programming in python and deep learning frameworks such as PyTorch
  - One new assignment **every three weeks** (W3, W6, W9); each due in 3 weeks
  - Submit **code** + **model** + **report**
  - Will select around 3 people to share their work
  - NO cheating: Will run *plagiarism detector*
- Topics
  - HW1: singer classification
  - HW2: diffusion-based controllable text-to-music generation
  - HW3: Transformer-based pop piano MIDI generation

Assignments

		HW1 Classification	HW2 Text-to-music	HW3 MIDI generation	Final project
W1	Intros + course logistics, fundamentals of musical audio				
W2	Music classification & music foundation models				
W3	Audio codec models & audio language models & audio captioning	announced			
W4	Audio effect modeling				
W5	Transformer-based music generation				
W6	Diffusion-based music generation	due	announced		
W7	Singing voice generation & song generation				
W8	DDSP & automatic mixing				
W9	Fundamentals of symbolic music & symbolic MIDI generation		due	announced	
W10	Advanced symbolic MIDI generation				team up
W11	Cover generation & MIDI-to-audio generation				
W12	Project pitch			due	proposal
W13	Miscellaneous topics				
W14	Miscellaneous topics				
W15	break				
W16	Final presentation				oral report
W17					
W18					paper report

NO Cheating

- Once caught: **failure** of the course

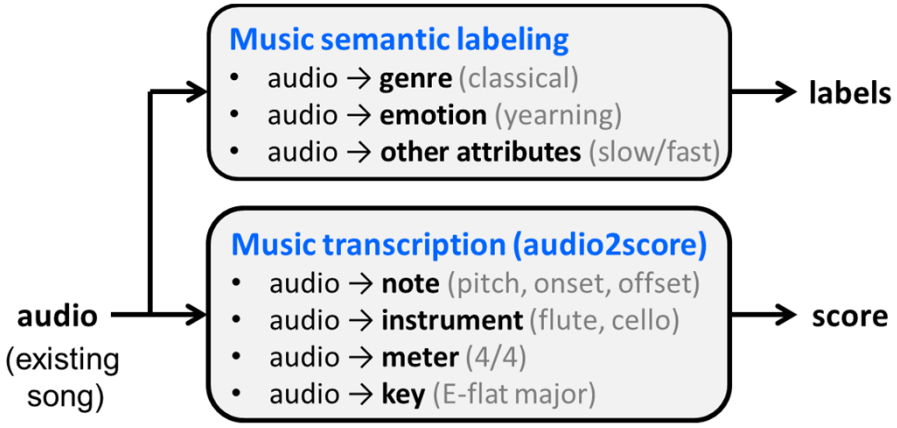
Final Project

- For teams of 2 or **3** (recommended)
- Start earlier & form teams

- Deadline for **team-up**: W10
- **Project pitch**: W12
- **Final presentation**: W16
- Deadline for **final report**: W16+2

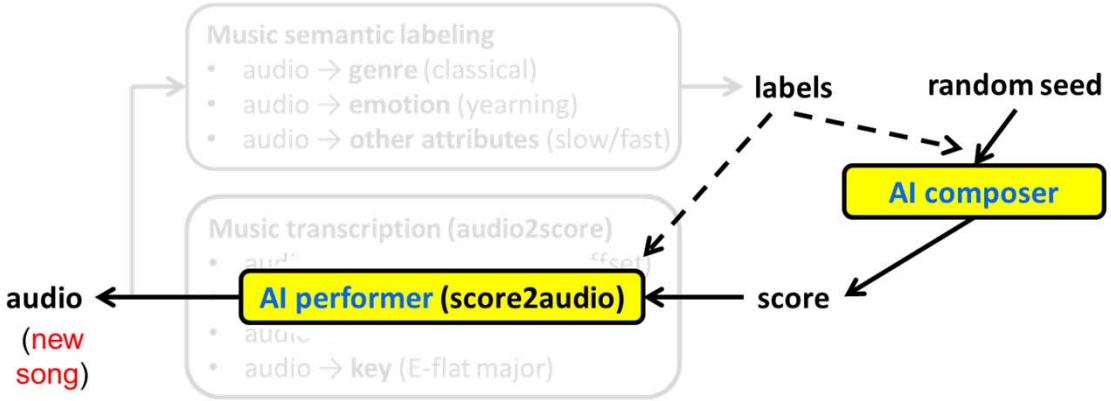
Final Project

- **Music analysis**



  - music semantic labeling
  - music transcription
  - source separation

- **Music generation**



  - MIDI generation
  - audio generation
  - MIDI-to-audio generation

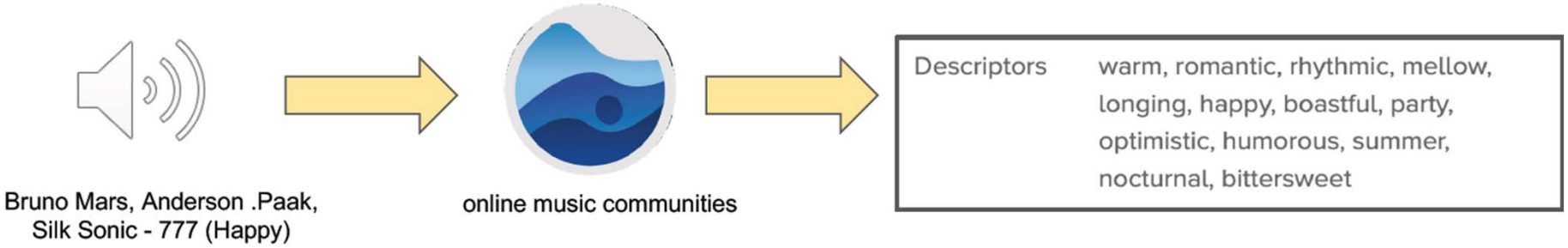
Final Project Showcase

https://affige.github.io/teaching_deepmir23.html

- **List of final projects** presented by the class students
  - *Positive Grid guitar effect removal*
    - extensions of this project has been published at DAFx'24
  - *Personalization on music generation*
    - extensions of this project has been published at ISMIR'24
  - *Music2fail: transfer music to fail recorder style*
    - extensions of this project has been submitted to APSIPA ASC'24
  - *Music generation from image and text*
    - extensions still ongoing
  - *Neural screaming voice conversion*
    - extensions still ongoing
  - *Acapella accompaniment generation*
    - extensions still ongoing
  - *Toward cross-lingual singing voice conversion*
  - *Personalized singing voice beautifier*
  - *Multimodal music emotion recognition*
  - *Music emotion recognition using contrastive language aud*
  - *Evaluation toolkit for controllable text-to-music models*
  - *Muse2bach*
  - *GuitarPedal simulation*
  - *OSU! Taiko auto mapper*
  - *AI Nice Chord progression*
  - *Timbre transfer by diffusion models*

Final Project Showcase

- Multimodal music emotion recognition: audio + lyrics + MIDI



Song	Model	Prediction
Bruno Mars, Anderson .Paak, Silk Sonic - 777 (Happy)	CRNN	Happy
	MERT	Happy
	Whisper + BERT	Angry
	Late fusion	Angry
	LSTM-Attn + Remi	Angry

1

Final Project Showcase

- Guitar effect removal

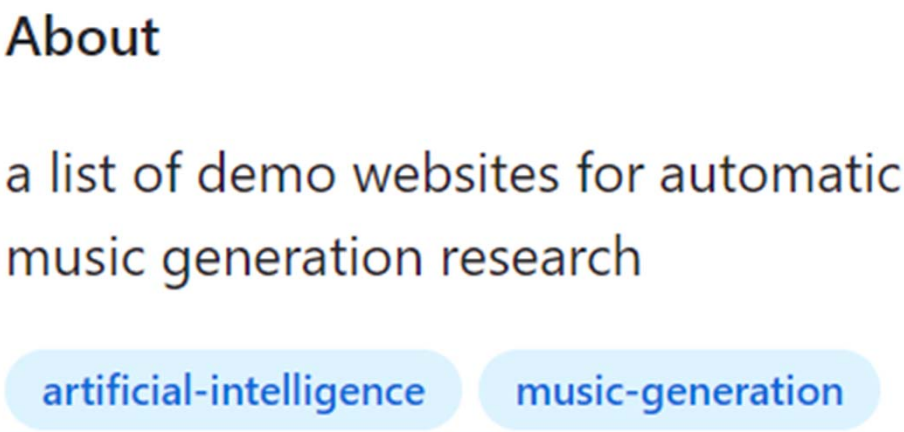
  https://y10ab1.github.io/guitar_effect_removal/

Final Project Showcase

- Transfer music to failed recorder style
  https://navi0105.github.io/demo/music2fail/

genmusic_demo_list

https://github.com/affige/genmusic_demo_list



Resources

- ML/DL
  - http://speech.ee.ntu.edu.tw/~tlkagk/courses.html
  - https://www.csie.ntu.edu.tw/~htlin/course/
  - https://www.csie.ntu.edu.tw/~yvchen/teaching
  - https://courses.cs.washington.edu/courses/cse599i/20au/ (generative models)
- Music information research
  - https://www.audiolabs-erlangen.de/fau/professor/mueller/teaching/2023w_mpa
  - https://musicinformationretrieval.com/
  - https://mac.kaist.ac.kr/~juhan/gct634/index.html
  - http://www.jordipons.me/apps/teaching-materials/
  - https://www.upf.edu/web/smc/audio-signal-processing-for-music-applications

Resources

- Conference proceedings
  - **Int'l Soc. Music Information Retrieval Conf.** (ISMIR)
  - Int'l Conf. Acoustic, Speech, and Signal Processing (ICASSP)
  - ACM MM, ACM ICMR, ACM SIGIR, IEEE ICME
- Transactions
  - IEEE Trans. Audio, Speech and Language Processing (TASLP)
  - IEEE Trans. Multimedia (TMM)
  - IEEE Trans. Signal Processing (TSP)

Additional Enrollment

https://forms.gle/p6ro7tE9ibMb5S9VA

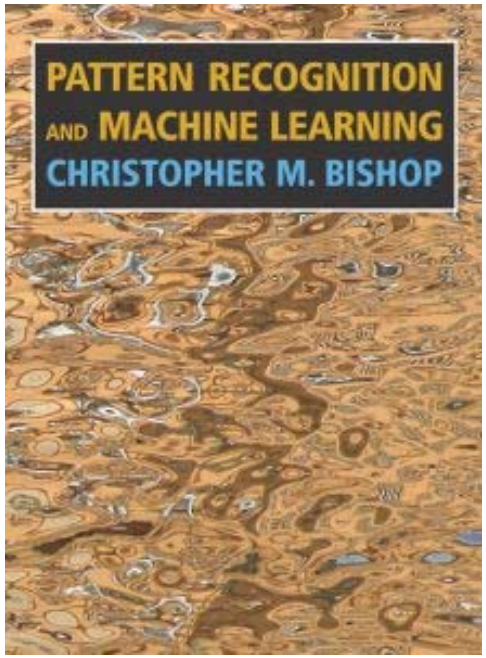
- **Sign up** at NTU Cool (選課意願登記), if you are an NTU student
- **AND, Fill the form** before 23:59, **September 5** (Friday)
  - DL background
  - Music background
  - Ideas for final project
- Will announce the result before next **Monday**
  - Not first-come-first-serve; will read your submission
  - Will only send a mail to those selected

Outline

- Intros of Music & AI
- Course logistics
- Fundamentals of musical audio
- **Appendix: ML 101**
  - Will talk about DL 101 in the next lecture

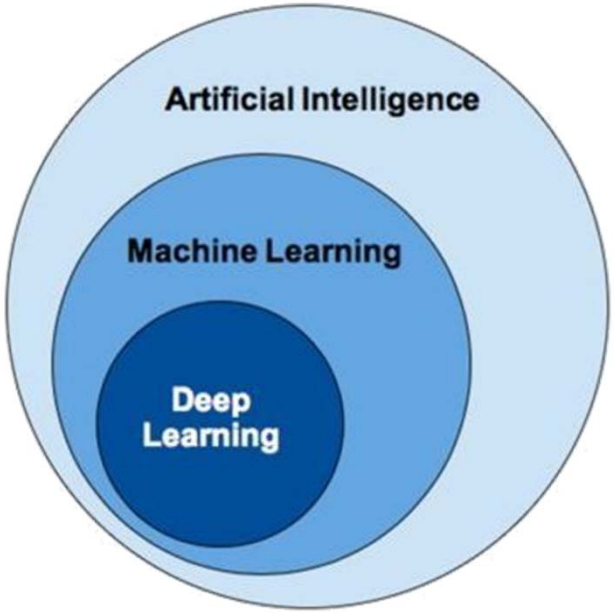
Machine Learning

https://www.microsoft.com/en-us/research/people/cmbishop/prml-book/

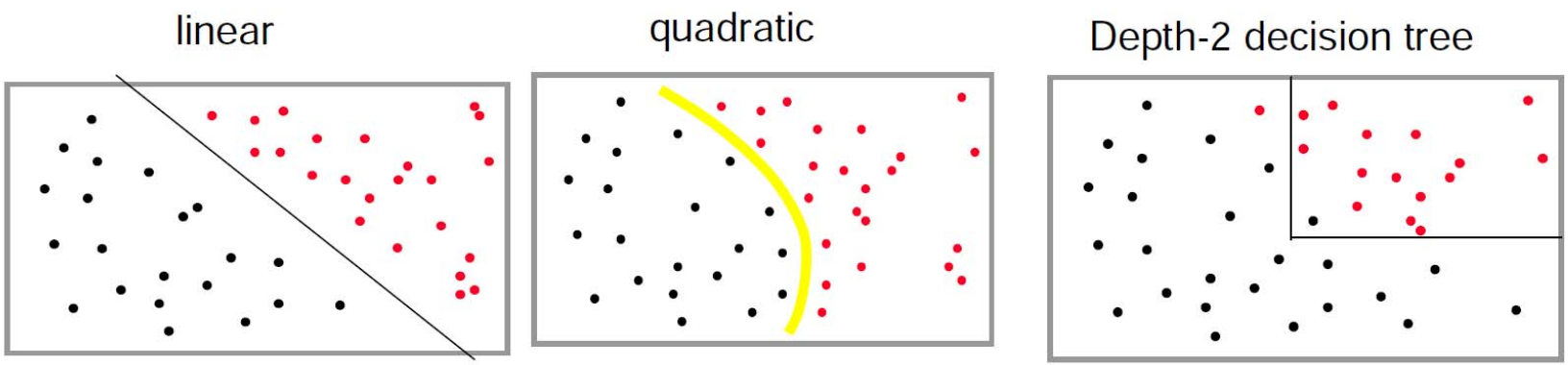


Freely available

1. Introduction
2. Probability Distributions
3. Linear Models for Regression
4. Linear Models for Classification
5. Neural Networks
6. …



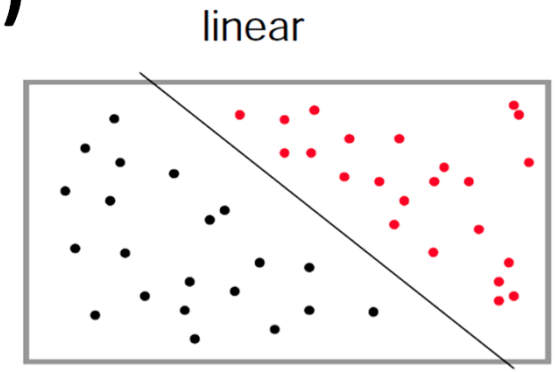
Classification: Overview (1/2)

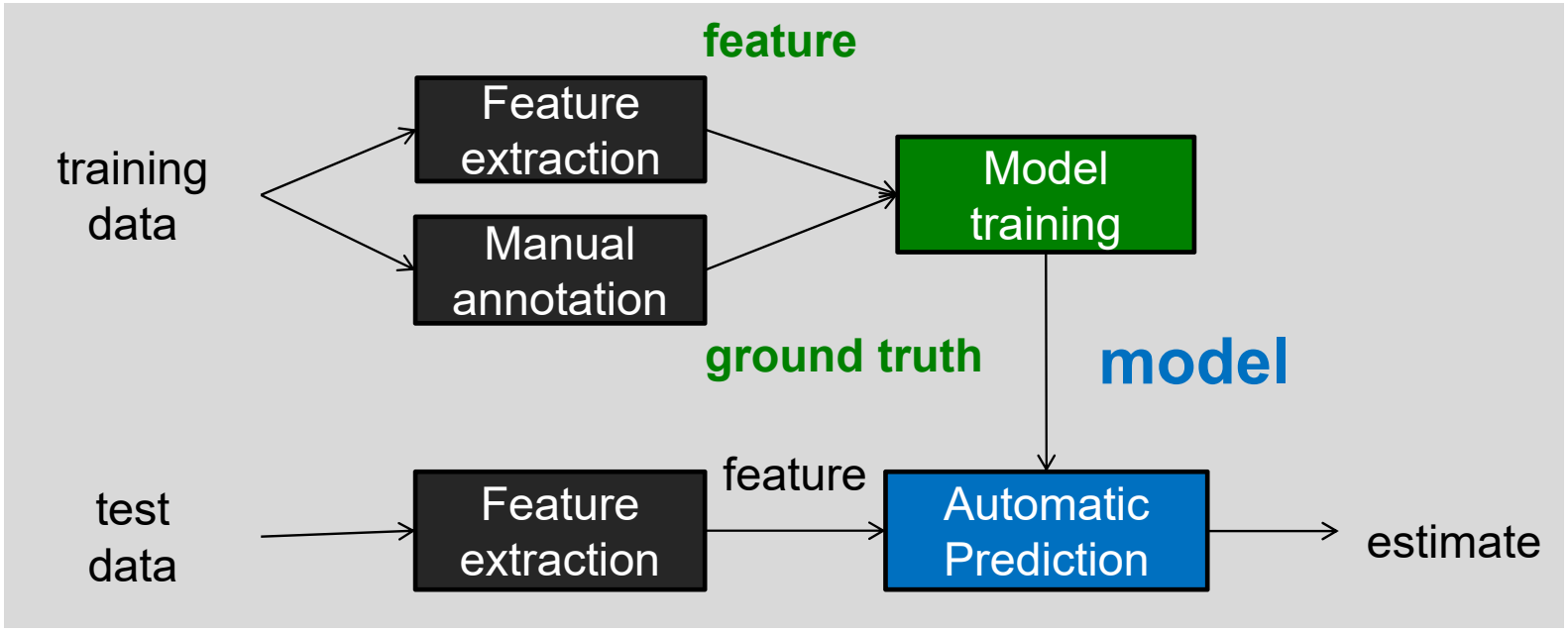


- Given: $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$
  - $\boldsymbol{x}_i \in \mathbb{R}^M$: feature representation of a data instance
  - $y_i \in \{-1, +1\}$: class label of that instance
- Goal
  - learn a separating hyperplane (decision boundary)

Classification: Overview (2/2)

- Given: $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$
  - $\boldsymbol{x}_i \in \mathbb{R}^M$: feature representation
  - $y_i \in \{-1, +1\}$: class label





Model Training (1/3)

1. Collect data (y) and extract features ($\boldsymbol{x}$)
2. Build model: choose hypothesis class $\mathcal{H}$ and loss function l
   - hypothesis class
     - **linear**: $\hat{y} = f(\boldsymbol{x}, \boldsymbol{w}, b) = \text{sgn}(\boldsymbol{w}^T\boldsymbol{x} + b)$
     - **quadratic**: $\hat{y} = f(\boldsymbol{x}, \boldsymbol{A}, \boldsymbol{w}, b) = \text{sgn}(\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{w}^T\boldsymbol{x} + b)$

$$\text{sgn}(x) = \begin{cases} +1, & x \geq 0; \\ -1, & x < 0. \end{cases}$$

   - “parameters” to be *learned*: $\theta = \{\boldsymbol{A}, \boldsymbol{w}, b\}$

Model Training (2/3)

1. Collect data (y) and extract features ($\boldsymbol{x}$)
2. Build model: choose hypothesis class $\mathcal{H}$ and loss function l
   - loss function
     - **indicator function**: $l(y, \hat{y}) = 0$ if $y = \hat{y}$;
       $l(y, \hat{y}) = 1$ otherwise
     - **square loss**: $l(y, \hat{y}) = (1 - y\hat{y})^2$
     - **hinge loss**: $l(y, \hat{y}) = \max(0, 1 - y\hat{y})$

     - while $y \in \{-1, +1\}$, $\hat{y}$ may not
     - hinge loss is used in support vector machine (SVM)

Model Training (3/3)

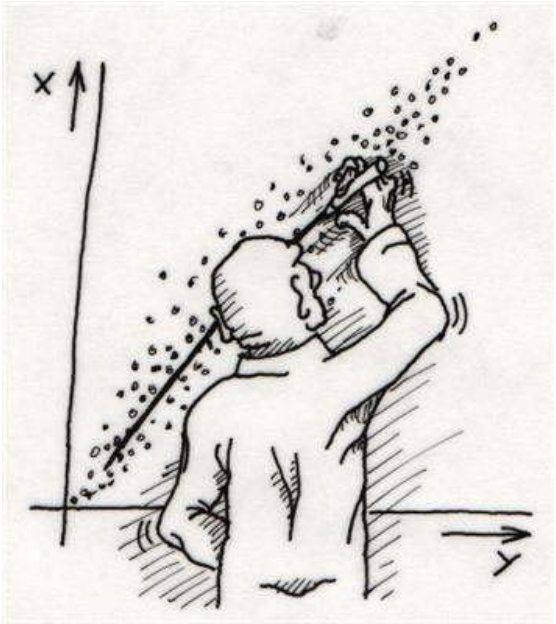
1. Collect data (y) and extract features ($\boldsymbol{x}$)
2. Build model: choose hypothesis class $\mathcal{H}$ and loss function l
3. Optimization: minimize the empirical loss
   - empirical loss: $\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(y_i, f(\boldsymbol{x}_i, \theta))$
   - optimization: $\min_{\theta} \mathcal{L}(\theta) \Rightarrow$ by computing $\frac{\partial \mathcal{L}(\theta)}{\partial \theta}$

More on Loss Functions (1/4)

- Different loss function leads to different machine learning problems
  - **classification**: $l(y, \hat{y}) = (1 - y\hat{y})^2$
  - **regression**: $l(y, \hat{y}) = (y - \hat{y})^2$

Given N inputs $(\boldsymbol{x}_i, y_i)$, $1 \leq i \leq N$, where $\boldsymbol{x}_i$ is feature and y_i is the numerical value to be predicted, train a regression model $f(.)$ such that the **mean squared error** (**MSE**) is minimized

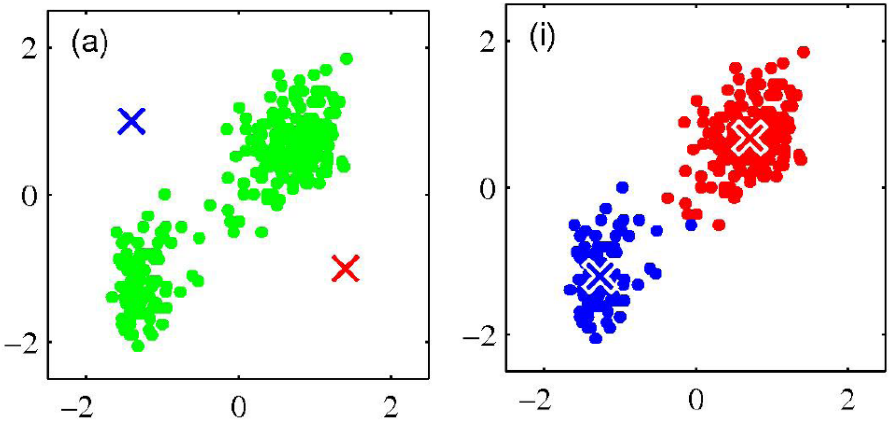
$$\min_f \frac{1}{N} \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i))^2$$



More on Loss Functions (2/4)

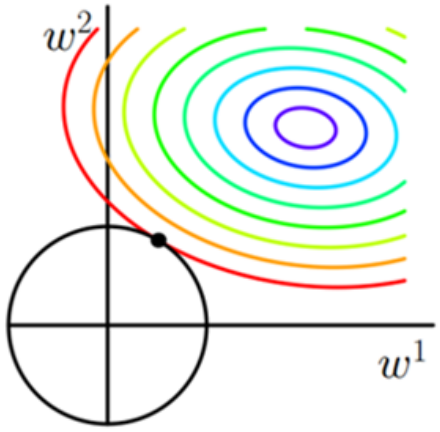
- Different loss function leads to different machine learning problems
  - **classification**: $l(y, \hat{y}) = (1 - y\hat{y})^2$
  - **regression**: $l(y, \hat{y}) = (y - \hat{y})^2$
  - **clustering** (an unsupervised task)

$$\mathcal{L}(\Delta) = \sum_{k=1}^{K} \sum_{i \in C_k} ||x_i - \mu_k||^2$$



More on Loss Functions (3/4)

- Adding regularizers
  - original: $\mathcal{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2$
  - modified: $\mathcal{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2 + \lambda\|\boldsymbol{w}\|_2^2$



(b) ℓ_2-ball meets quadratic function. ℓ_2-ball has no corner. It is very unlikely that the meet-point is on any of axes."

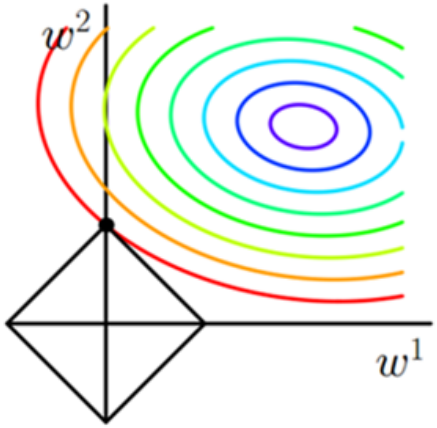
$$\|\boldsymbol{w}\|_2^2 = \boldsymbol{w}^T\boldsymbol{w} = \sum_{j=1}^{M} w_j^2$$

prefer $\boldsymbol{w}$ = [1.5, -0.5, 0.8, -1.0]^T
over $\boldsymbol{w}$ = [9.5, -8.0, 2.5, -3.0]^T,
despite that the latter has smaller MSE

- $\lambda \geq 0$ is a parameter to be tuned (i.e. empirically determined)

More on Loss Functions (4/4)

- Adding regularizers
  - modified: $\mathcal{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2 + \lambda\|\boldsymbol{w}\|_2^2$
  - modified: $\mathcal{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2 + \lambda\|\boldsymbol{w}\|_1$



(a) ℓ_1-ball meets quadratic function. ℓ_1-ball has corners. It's very likely that the meet-point is at one of the corners.

$$\|\boldsymbol{w}\|_1 = \sum_j |w_j|$$

prefer $\boldsymbol{w}$ = [1.5, 0, 0, -2.0]^T
over $\boldsymbol{w}$ = [1.5, -0.5, 0.8, -1.0]^T,
despite that the latter has smaller MSE

- L1 is the only norm that is both convex and sparsity promoting

Example: Ridge Regression (for Regression)

- Loss function

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2 + \lambda\|\boldsymbol{w}\|_2^2$$

or, $\mathcal{L}(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + \lambda\,\boldsymbol{w}^T\boldsymbol{w}$

$N \times 1 \quad N \times M \quad M \times 1$

- Optimization

$$\frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial \boldsymbol{w}} = 2\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) + 2\lambda\,\boldsymbol{w} \overset{\text{let}}{=} \boldsymbol{0}$$

$M \times N \quad N \times 1 \quad M \times 1$

$$\therefore \boldsymbol{X}^T\boldsymbol{y} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})\boldsymbol{w}$$

$$\therefore \boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

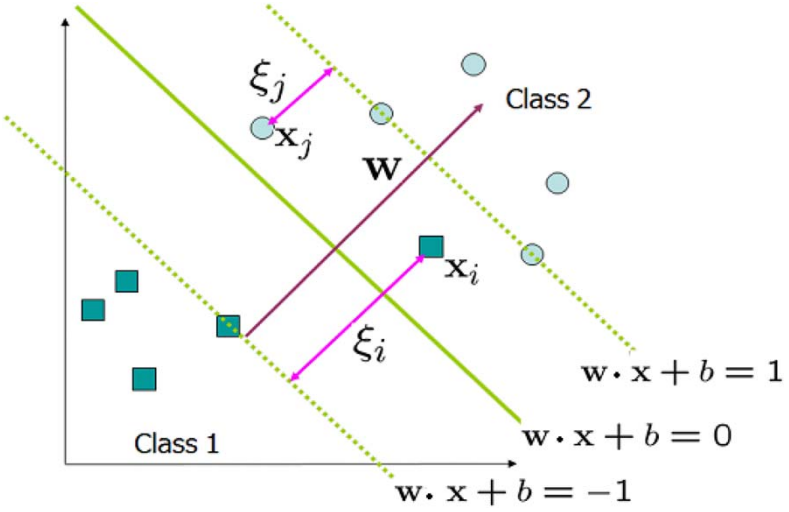
Example: Support Vector Machine (1/3)

- Loss function

$$\min_{\boldsymbol{w},\xi_i} \|\boldsymbol{w}\|_2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to} \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i \text{ for } i = 1 \ldots N$$

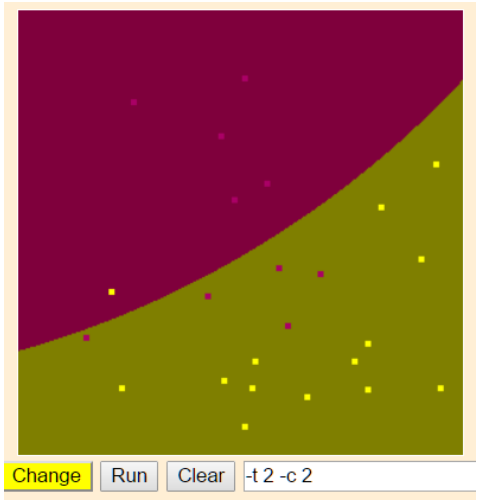
- the idea of hinge loss $\max(0, 1 - y\hat{y})$ is also to have $y\hat{y} \geq 1$
- the slack variables $\xi_i \geq 0$ allow for errors

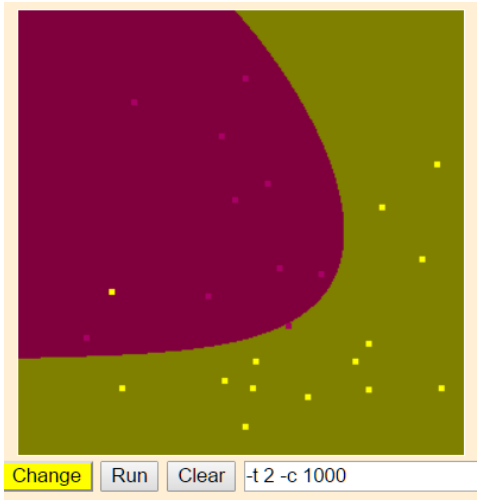


Example: Support Vector Machine (2/3)

- Loss function: $\min \|\boldsymbol{w}\|_2 + C \sum_{i=1}^{N} \xi_i$
  - smaller C: reduced model complexity
  - larger C : less misclassification
- Demo: https://www.csie.ntu.edu.tw/~cjlin/libsvm/

smaller C





larger C

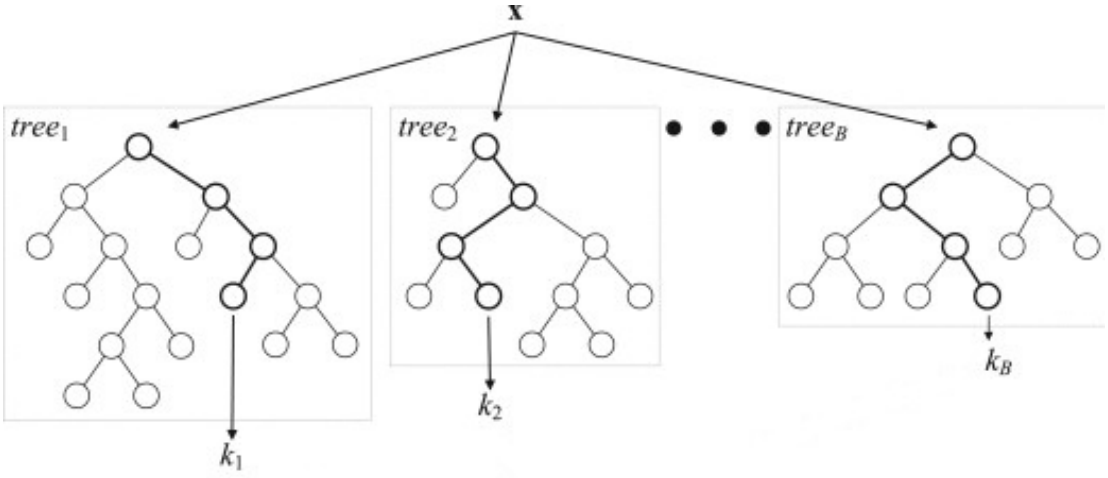
Example: Support Vector Machine (3/3)

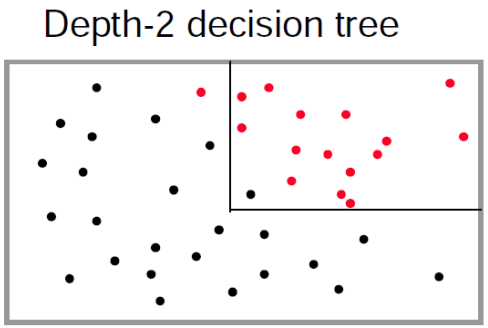
- Different kernel functions of SVM
  - the **RBF** kernel (-t 2) is often used for nonlinear SVM; important parameters: *C* and *gamma*

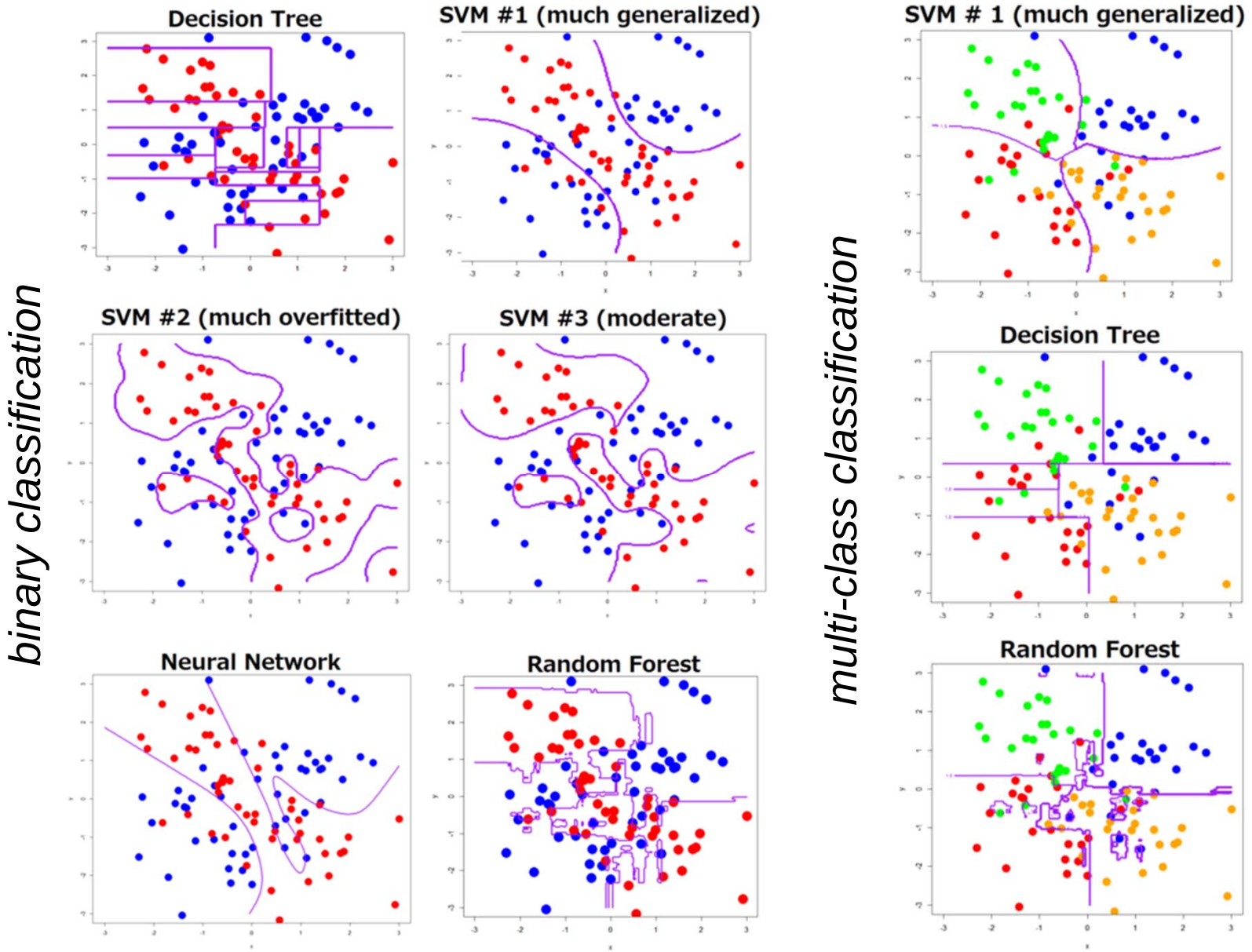
```
options:
-s svm_type : set type of SVM (default 0)
        0 -- C-SVC
        1 -- nu-SVC
        2 -- one-class SVM
        3 -- epsilon-SVR
        4 -- nu-SVR
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/num_features)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
```

Example: Random Forest

- Ensemble of decision trees
  - important parameters: number of trees, tree depth
  - strength: interpretability, non-linear

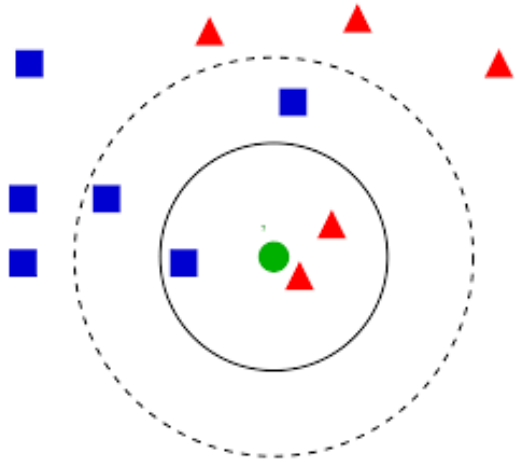


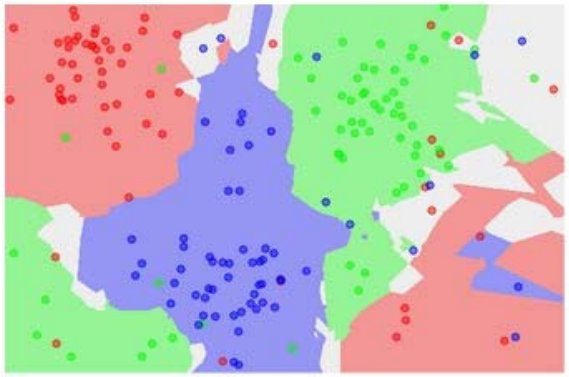



binary classification
Decision Tree
SVM #1 (much generalized)
SVM #2 (much overfitted)
SVM #3 (moderate)
Neural Network
Random Forest
multi-class classification
SVM # 1 (much generalized)
Decision Tree
Random Forest

Example: k-Nearest Neighbors

- Classify an instance by a **majority vote** of the class membership of its k nearest neighbors
  - k is a positive integer
  - performance depends on the value of k and the features
  - lazy learning
  - nonlinear classifier





Evaluation (1/3)

- Model training
  1. Collect data and extract features
  2. Build model: choose hypothesis class and loss function
  3. Optimization: minimize the empirical loss
- **Evaluation**
  4. Evaluate on unseen, testing data

Evaluation (2/3)

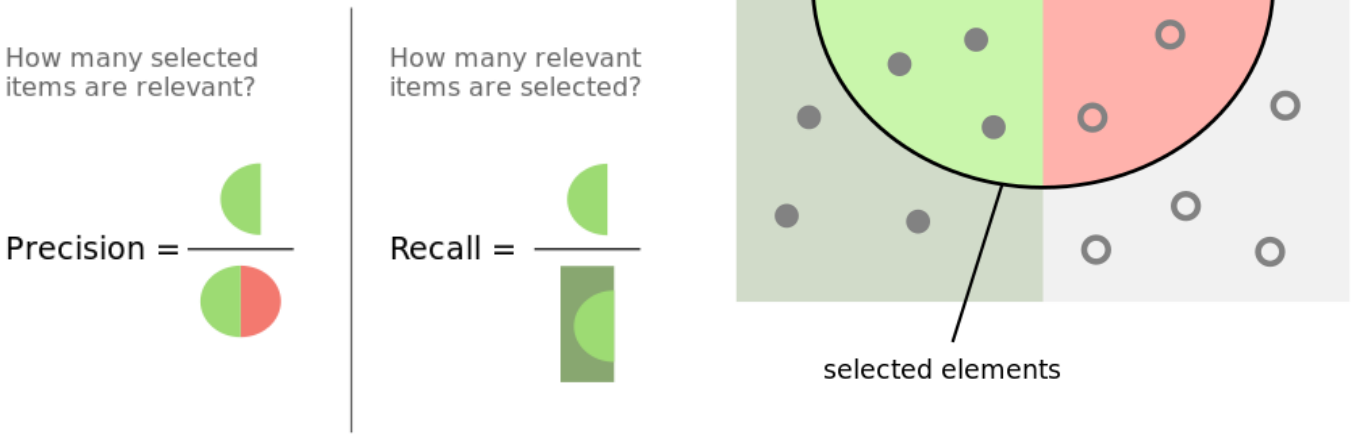
- Performance measures
  - classification accuracy (CA)
  - per-class precision (P), recall (R), F-score (F)
  - confusion table

		Predicted		
		Cat	Dog	Rabbit
Actual class	Cat	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

CA = (5+3+11)/(5+3+0+2+3+1+0+2+11)
P(cat) = 5/(5+2)
R(cat) = 5/(5+3)

Evaluation (3/3)

- Performance measures
  - precision (**P**): TP/(TP+FP)
  - recall (**R**): TP/(TP+FN)
  - F-score (**F**): 2PR/(P+R)



Train, Validation, Test (1/2)

- Test set is unseen, how to evaluate our classifier?
  - use part of the training set as the "**validation set**"
  - e.g. 80% for training, 20% for validation
  - use the validation set to: 1) tune the parameters, 2) select classification algorithms, and 3) select features
- ***K*-fold Cross validation**
  - divide the data into *K* folds, use one fold for validation and the rest of training, repeat *K* times and report the average result

Train, Validation, Test (2/2)

- **Model overfitting**
  - training accuracy ≠ validation accuracy
  - that's why we need a balance between model complexity and training error
- **Cross-dataset generalizability**
  - validation accuracy ≠ test accuracy
- **Cheating issue**
  - make sure there is **no overlaps** between the training and test sets
  - **song-level split** (preferred) vs. chunk-level split

Data Labeling (1/2)

- Collect data and extract features
- How to get the labels?
  - **existing datasets**
    - due to copyright restrictions, it's not easy to get the audio files
  - **crawl the web**
    - unsure of the quality; do not how the labeling was done
  - **manually annotate**
    - labor intensive and time consuming but can be worthwhile

Data Labeling (2/2)

- Issues to be taken care of while building a new dataset
  1. **be aware of “confounds”**
     - Example 1: Jazz from the 60’s vs. contemporary Rock
     - Example 2: Happy songs from artist 1 vs. sad songs from artist 2
  2. **try to make the annotation process easier**
     - binary decision -> multiple choices -> rating
     - gamification
  3. **clear instructions**
  4. **check for reliability/consistency**

Feature Extraction and Processing

- Collect data and extract features
- The usual pipeline
  - frame-level **feature extraction**
  - **pooling** (from frame-level to clip-level)
  - **feature normalization**
- Classification tasks in MIR
  - **clip-level prediction**: genre classification
  - **frame-level prediction**: onset detection, pitch detection
  - for clip-level prediction, pooling is often used

Feature Pooling

- Rationale
  - thousands of frames per clip → thousands of feature vectors per clip
  - we can either train a frame-level classifier, but eventually still have to integrate the result to the clip-level
  - or, we can integrate the feature vectors before training the classifier
- Approach
  - take statistics such as **mean** and **standard deviation**

Feature Normalization/Scaling

- For each individual feature dimension
  - **linear normalization**: $x' = \frac{x - \min(x)}{\max(x) - \min(x)}$
  - **z-score normalization**: $x' = \frac{x - \text{mean}(x)}{\text{std}(x)}$
  - the values mean(x) and std(x) are computed from all the *training* instances (i.e. *not* including the validation and the test sets)
- For each individual instance
  - **scaling to unit length**: $x' = \frac{x}{|x|}$

Early Fusion or Late Fusion

- You can extract features of different types to characterize different aspects of the signal
  - timbre, pitch, rhythm, lyrics, etc
- **Early fusion** (feature-level fusion)
  - concatenate them into a single feature vector then train a classifier
- **Late fusion** (decision-level fusion)
  - train a classifier for each feature type, and then aggregate the estimate of these classifiers (e.g., majority vote or average) to make the final decision

Feature Selection

- Not all features would be relevant
- Feature selection according to
  - domain knowledge
  - empirical observation at the feature matrix (e.g., remove a feature if there is NaN or outlier [or replace the NaNs by the average])
  - statistical methods

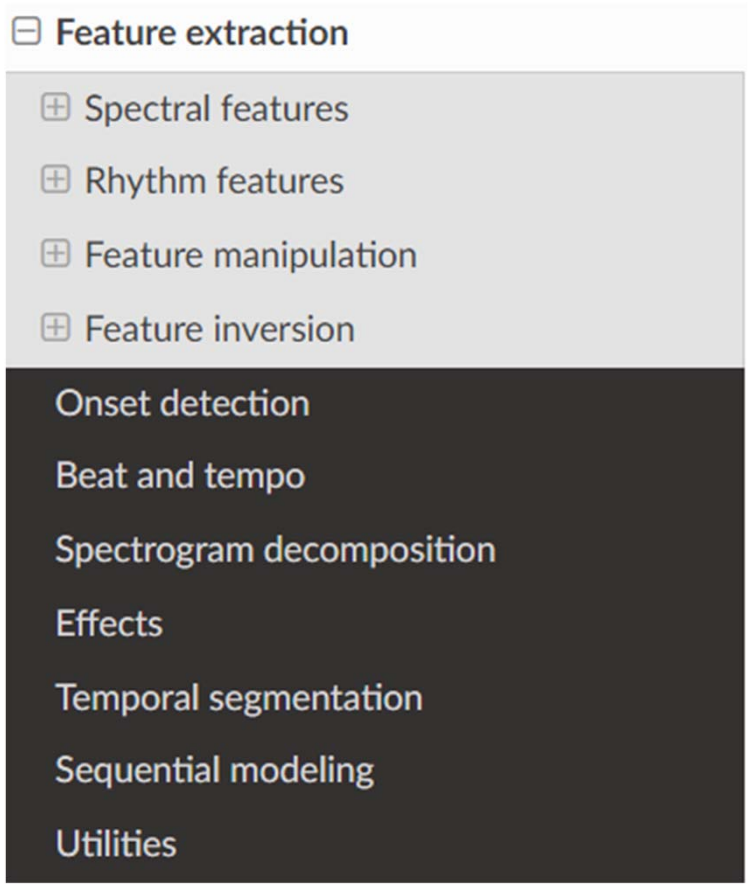
Library: Torchaudio

https://pytorch.org/audio/0.11.0/tutorials/audio_feature_extractions_tutorial.html

```
waveform, sample_rate = get_speech_sample()

n_fft = 1024
win_length = None
hop_length = 512

# define transformation
spectrogram = T.Spectrogram(
    n_fft=n_fft,
    win_length=win_length,
    hop_length=hop_length,
    center=True,
    pad_mode="reflect",
    power=2.0,
)
# Perform transformation
spec = spectrogram(waveform)

print_stats(spec)
plot_spectrogram(spec[0], title="torchaudio")
```

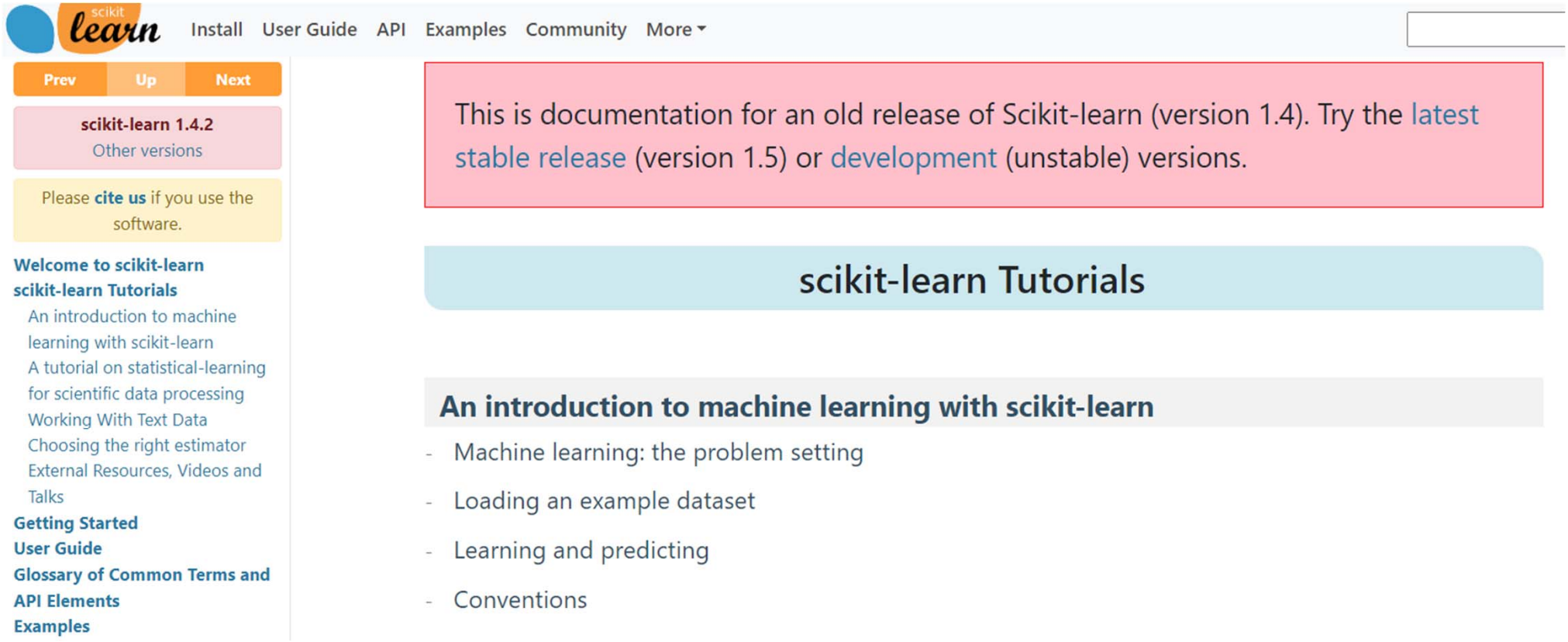
Python: Librosa

https://librosa.org/doc/main/feature.html



Python: scikit-learn

https://scikit-learn.org/1.4/tutorial/index.html



Practical Issues 1: Multi-label Classification

- A data instance can be labeled with more than one classes
- *Multi-class* classification (one-hot; one out of many) vs. *multi-label* classification (multi-hot)

- Harder for ML (may need multiple binary classifiers) but simple for DL: it's just a matter of changing the loss function
  - Multi-class classification: **categorical cross entropy** (CE)
  - Multi-label classification: **binary cross entropy** (BCE)

Practical Issues 2: Data Imbalance

- Simply predicting everything as the majority class would already give you good accuracy
- A naïve workaround: **subsampling the majority class**

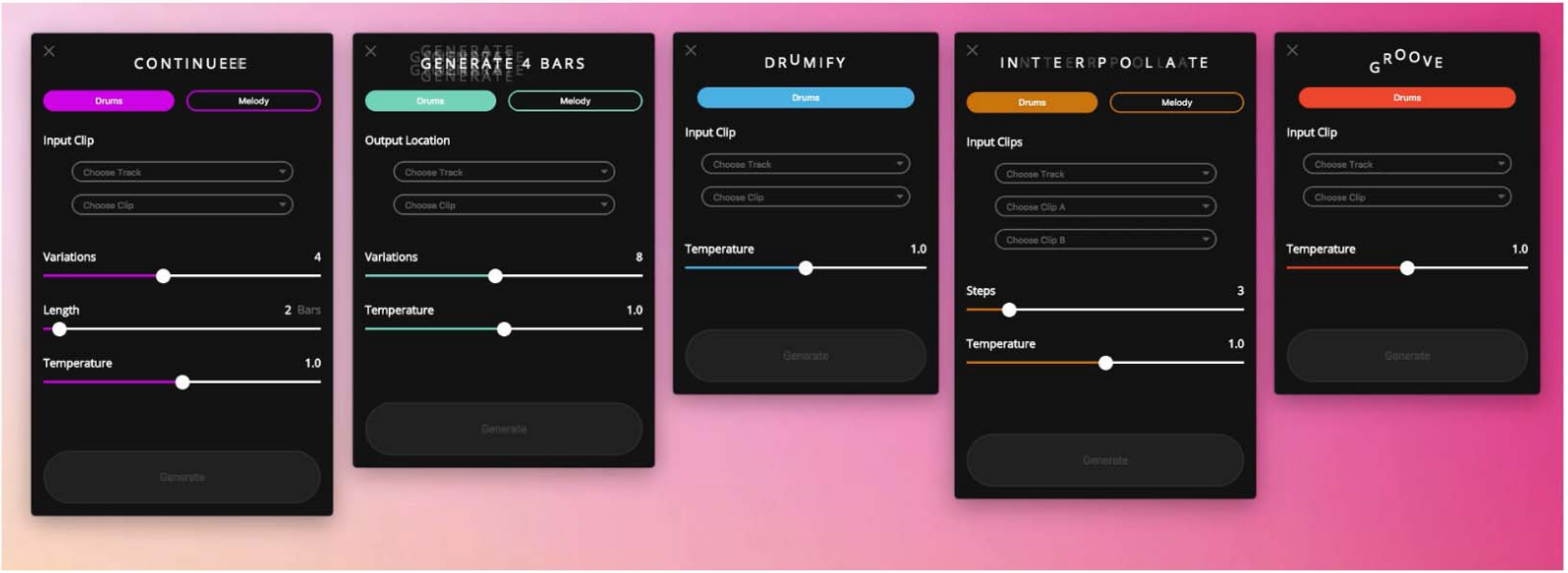
Practical ML Hints (not for DL)

- Look into the features
- Remember to normalize the features
- Use validation set to properly tune the parameters
- Nonlinear classifier usually performs better, but also requires longer training time
- Use a subset of data to fast prototype your ideas first, before using the full set for more complete experiments
- Study the classification result (e.g., confusion table)

Demo 3: Magenta Studio

(Make musicians' life easier)

https://magenta.tensorflow.org/studio/

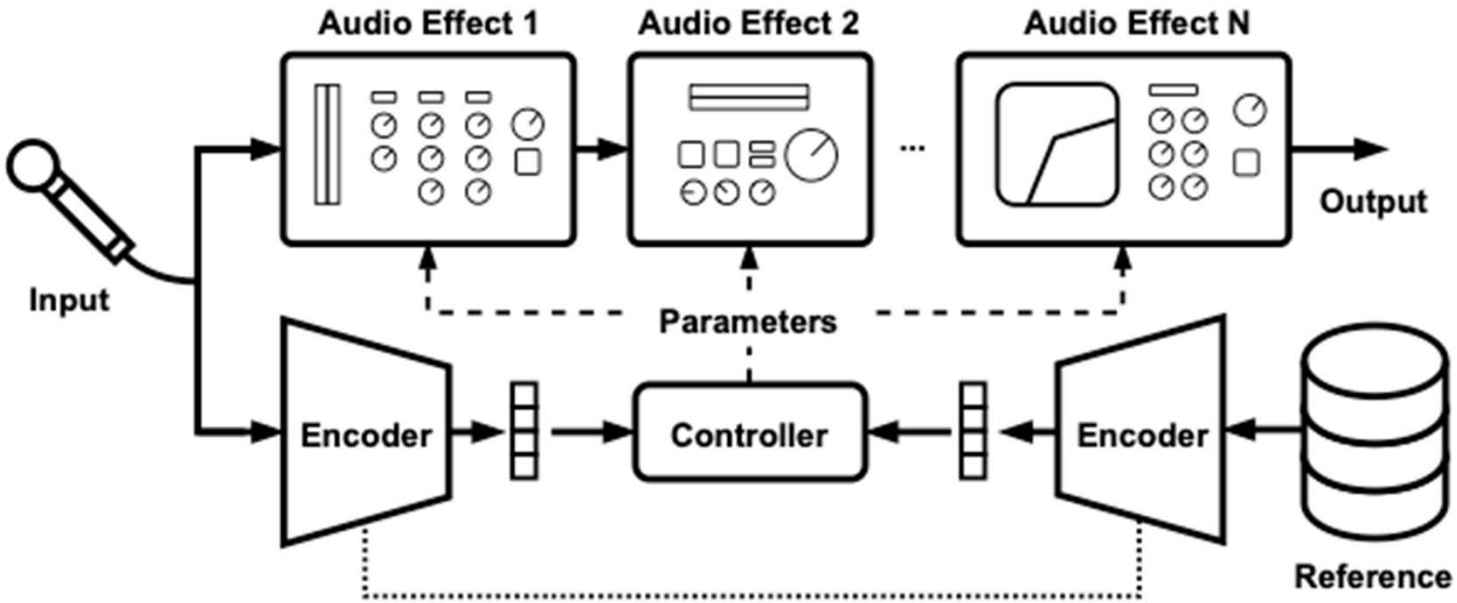


Continue," "Generate 4 bars," "Drumify," "Interpolate," "Groove"

Demo 4: Mixing Style Transfer

(Make musicians' life easier)

https://csteinmetz1.github.io/DeepAFx-ST/

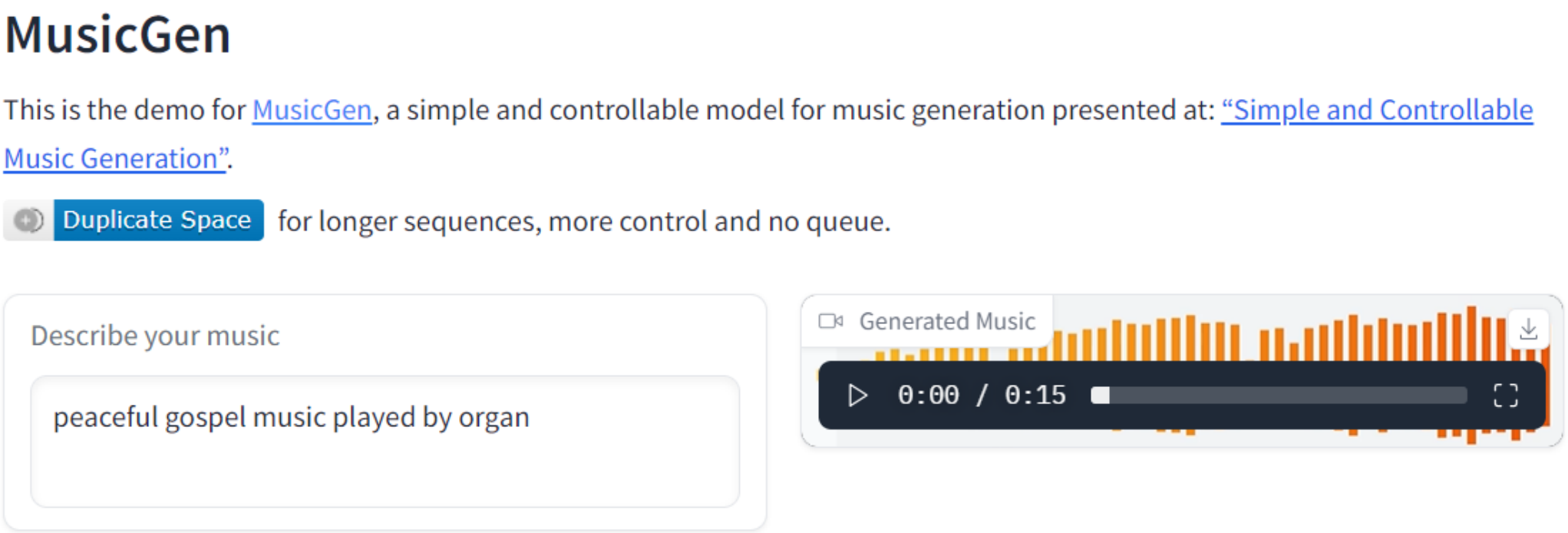


Demo 5: Text-to-Music

(Create copyright free music for videos or games)

https://ai.honu.io/papers/musicgen/

https://huggingface.co/spaces/facebook/MusicGen



Demo 6: Image-to-Sound by Adobe Firefly

https://www.youtube.com/watch?v=30xueN12guw
https://www.adobe.com/tw/sensei/generative-ai/firefly.html

Future Vision: Adobe Firefly for @AdobeVideo

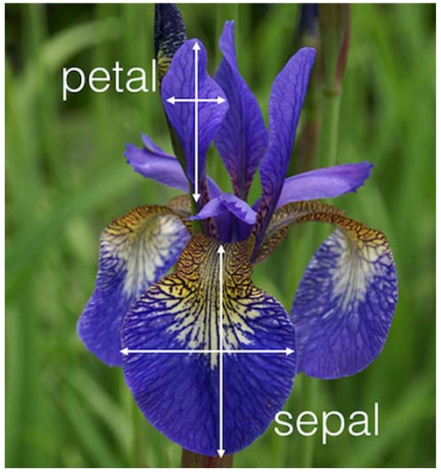
Demo 7: AI MV

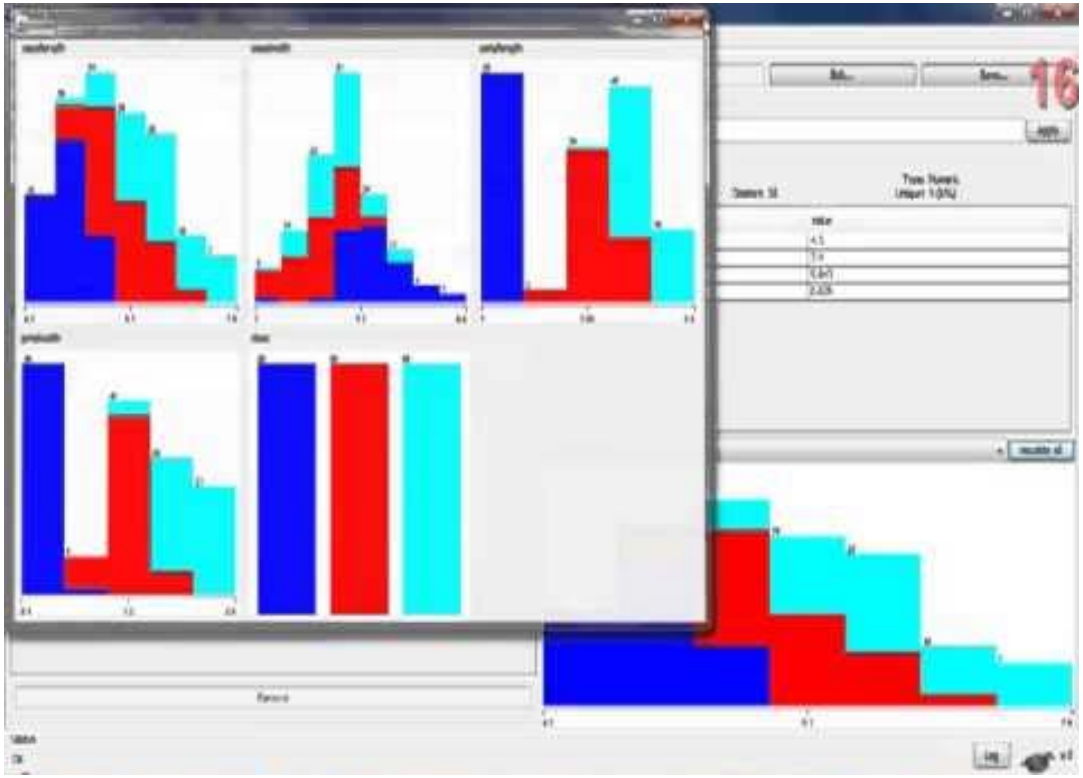
https://www.ziaxaza.com/

WEKA

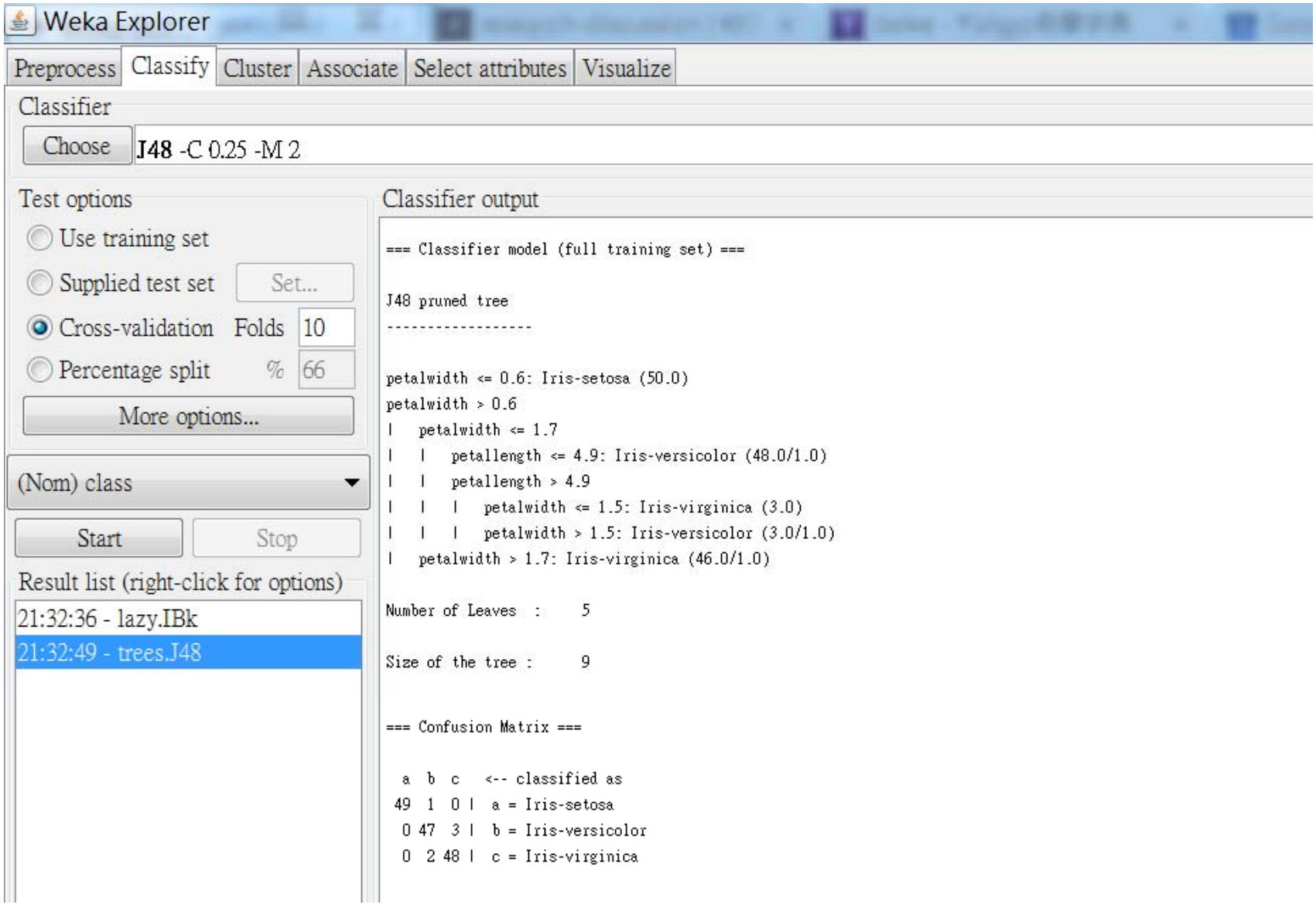
- http://www.cs.waikato.ac.nz/ml/weka/

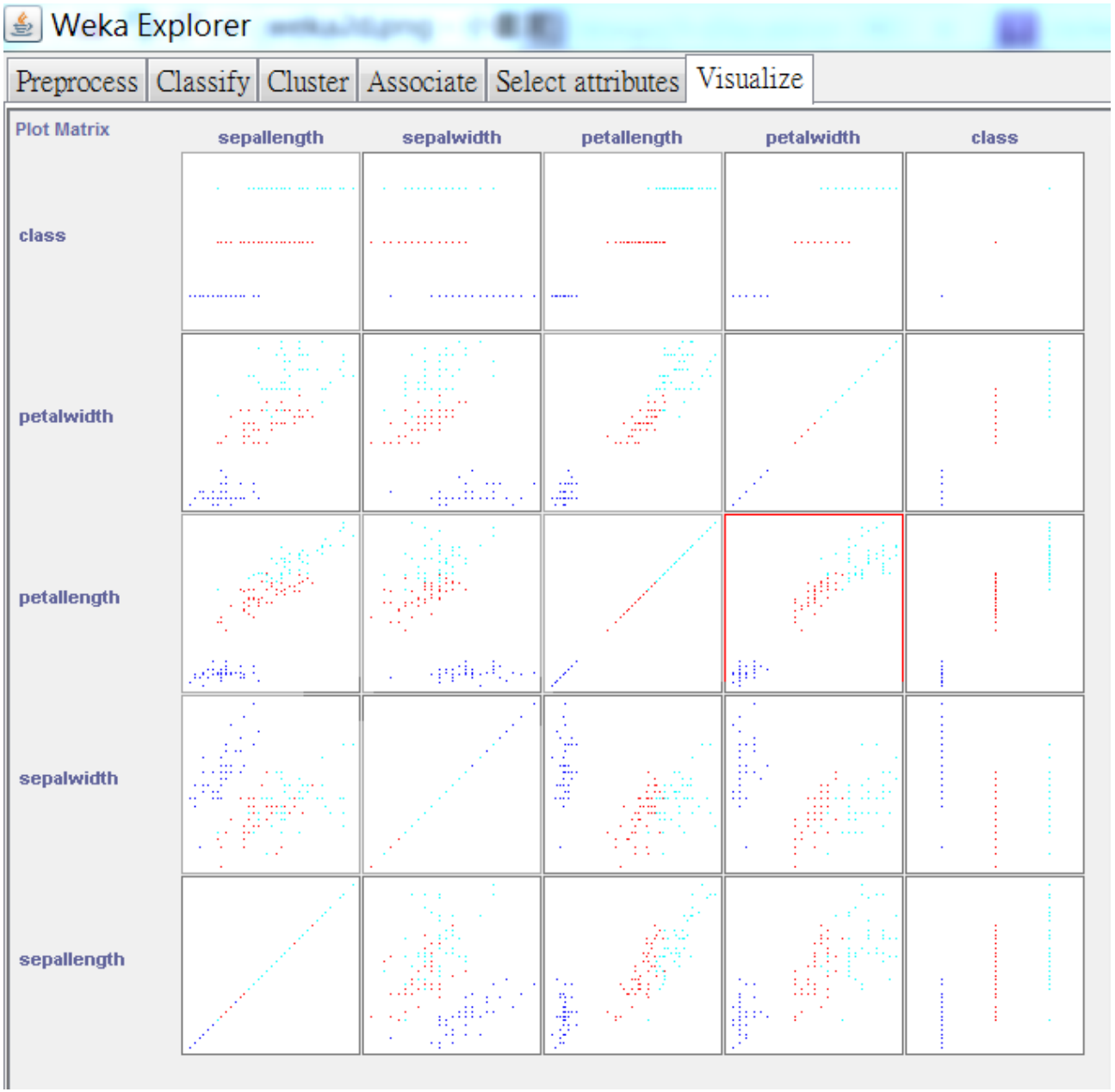
The IRIS Dataset





http://mirlab.org/jang/books/dcpr/dataSetIris.asp?title=2-2%20Iris%20Dataset
http://storm.cis.fordham.edu/~gweiss/data-mining/datasets.html


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
21:32:36 - lazy.IBk
21:32:49 - trees.J48
Classifier output
=== Classifier model (full training set) ===
J48 pruned tree
petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
| petalwidth <= 1.7
| | petallength <= 4.9: Iris-versicolor (48.0/1.0)
| | petallength > 4.9
| | | petalwidth <= 1.5: Iris-virginica (3.0)
| | | petalwidth > 1.5: Iris-versicolor (3.0/1.0)
| petalwidth > 1.7: Iris-virginica (46.0/1.0)
Number of Leaves : 5
Size of the tree : 9
=== Confusion Matrix ===
a b c <-- classified as
49 1 0 | a = Iris-setosa
0 47 3 | b = Iris-versicolor
0 2 48 | c = Iris-virginica


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Plot Matrix
sepallength
sepalwidth
petallength
petalwidth
class
class
petalwidth
petallength
sepalwidth
sepallength

Matlab: MIRtoolbox

- http://www.mathworks.com/matlabcentral/fileexchange/24583-mirtoolbox/content/MIRtoolbox1.3.4_matlabcentral/MIRToolbox/**mirfeatures**.m

- f = mirframe(a,.05,.5);
- r.spectral = mirstruct;
- r.spectral.tmp.s = mirspectrum(f);
- r.spectral.centroid = mircentroid(r.spectral.tmp.s);
- r.spectral.brightness = mirbrightness(r.spectral.tmp.s);
- r.spectral.spread = mirspread(r.spectral.tmp.s);

Python: ESSENTIA (1/2)

- http://essentia.upf.edu/documentation/python_tutorial.html
  - **Time-domain descriptors**: duration, loudness, LARM, Leq, Vickers' loudness, zero-crossing-rate, log attack time and other signal envelope descriptors
  - **Spectral descriptors**: Bark/Mel/ERB bands, MFCC, GFCC, LPC, spectral peaks, complexity, rolloff, contrast, HFC, inharmonicity and dissonance
  - **Tonal descriptors**: Pitch salience function, predominant melody and pitch, HPCP (chroma), chords, key and scale, tuning frequency
  - **Rhythm descriptors**: beat detection, BPM, onset detection, rhythm transform, beat loudness
  - **Other high-level descriptors**: danceability, dynamic complexity, audio segmentation, SVM classifier
  - **Statistical descriptors**: median, mean, variance, power means, raw and central moments, spread, kurtosis, skewness, flatness

Or http://www.ifs.tuwien.ac.at/~schindler/lectures/MIR_Feature_Extraction.html

Python: ESSENTIA (2/2)

- # load audio
  loader = essentia.standard.**MonoLoader**(filename = 'audio.mp3')
  audio = loader()
- # extract frame-level MFCCs
  w = **Windowing**(type = 'hann')
  spectrum = **Spectrum**()
  mfcc = **MFCC**()
  X= **Pool**()
  for frame in FrameGenerator(audio, frameSize = 1024, hopSize = 512):
  mfcc_bands, mfcc_coeffs = mfcc(spectrum(w(frame)))
  X.add('lowlevel.mfcc', mfcc_coeffs)
- # pooling
  aggrPool = **PoolAggregator**(defaultStats = ['mean', 'var', 'min', 'max'])(pool)
- # output to json file
  YamlOutput(filename = 'mfcc.sig‘, , format="json")(pool)

Matlab & Python: LIBSVM

- https://www.csie.ntu.edu.tw/~cjlin/libsvm/
- You will need to a C compiler for Matalb (mex –setup) http://www.mathworks.com/support/compilers/R2013a/index.html?sec=win64

Interfaces and extensions to LIBSVM

Language	Description	Maintainers and Their Affiliation	Supported LIBSVM version	Link
Java	Java code close to LIBSVM C code.	LIBSVM authors at National Taiwan University.	The latest	Included in LIBSVM package
Java	Refactored Java code for faster training/testing.	David Soergel at University of California, Berkeley.	2.88	jlibsvm
MATLAB and OCTAVE	A simple MATLAB and OCTAVE interface	LIBSVM authors at National Taiwan University.	The latest	Included in LIBSVM package
MATLAB	An old version (no longer available)	Junshui Ma and Stanley Ahalt at Ohio State University	2.33	Dead Link
R	Please install by typing *install.packages('e1071')* at R command line prompt. (document and examples).	David Meyer at the Wirtschaftsuniversität Wien (Vienna University of Economics and Business Administration)	3.17	WWW
Python	A python interface has been included in LIBSVM since version 2.33.	Initiated by Carl Staelin at HP Labs. Updated/maintained by LIBSVM authors.	The latest	Included in LIBSVM package

Python: scikit-learn (2/3)

- # load the iris dataset
  from sklearn.datasets import load_iris
  iris = load_iris()
  X, y = iris.data, iris.target
- # train a classifier using SVM
  from sklearn.svm import SVC
  clf = SVC()
  clf.**fit**(X, y)
- # testing
  X_new = [[5.0, 3.6, 1.3, 0.25]]
  clf.**predict**(X_new)
- # or, train a classifier using Random Forest
  from sklearn.ensemble import RandomForestClassifier
  clf = RandomForestClassifier()
  clf.**fit**(X, y)

Python: scikit-learn (3/3)

- # split data into train and validation
  from sklearn.cross_validation import train_test_split
  X_train, X_test, y_train, y_test = **train_test_split**(X, y, test_size=0.2)
- # tune the parameters C and gamma
  from sklearn.grid_search import GridSearchCV
  tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4], 'C': [1, 10, 100, 1000]}, {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]

  clf = **GridSearchCV**(SVC(C=1), tuned_parameters, cv=5, scoring='f1_weighted')
  clf.fit(X_train, y_train)
  print(clf.best_params_)
- # evaluate on test set
  from sklearn.metrics import classification_report
  y_true, y_pred = y_test, clf.predict(X_test)
  classification_report(y_true, y_pred)

Some Pointers

- Tutorial video for ESSENTIA
  - https://www.youtube.com/watch?v=cHZONQ2-x7I
- Documentations
  - http://essentia.upf.edu/documentation/algorithms_overview.html
  - http://scikit-learn.org/stable/user_guide.html
  - http://scikit-learn.org/stable/modules/classes.html
- Papers
  - http://ismir2007.ismir.net/proceedings/ISMIR2007_p127_lartillot.pdf
  - http://mtg.upf.edu/node/2793
  - http://jmlr.org/papers/v12/pedregosa11a.html
  - http://arxiv.org/abs/1309.0238

Music Generation Demo 3: KaraSinger

https://jerrygood0703.github.io/KaraSinger/

Lyrics:
In this paper we propose
a novel neural network model
called Karaoke singer for a less studied
singing voice synthesis task
named score-free SVS
in which the prosody and melody are
spontaneously decided by machine.

Music Generation Demo 4: AI Sandee

https://www.youtube.com/watch?v=nWTuZIRU80A

「音樂製作人的工作是無法被取代的」。AI vocal 要怎麼唱，能唱得多好，終究需要專業音樂製作人，以人類的美學和經驗去引導 AI，要如何將 AI 昇華到情感面，終究還是需要製作人的能力，以及對音樂的想像力。

作為一個仍在線上的歌手與製作人，由我親自處理自己的AI vocal，讓這首歌傳達出「創作者、歌者不怕 AI 的挑戰」、「我們擁有自己聲音的控制權」等訊息，同時也是「人類的思考和意志，才是人之所以為人」的巨大宣示。

透過聆聽《教我如何做你的愛人》，試著探討：「若 AI 已經能模擬原唱的一切，那麼原唱歌手的價值會是什麼？」

當 AI 真正學會唱歌之後，就是創作人與歌手，重新理解自身價值的時候了。....by公主