SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

인공청각지능: 소리에서 의미로

Machine Listening: from sound to meaning

서울대학교 지능정보융합학과
음악오디오연구실
이교구

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Why Are We Here?

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

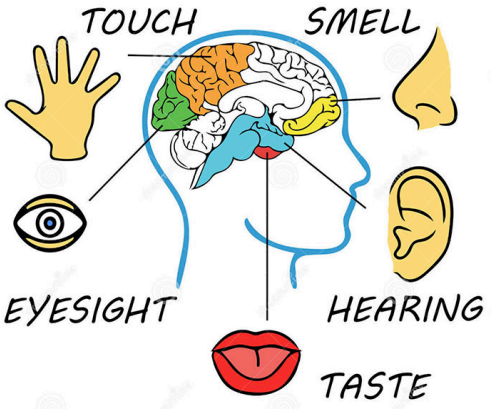
Why Are We Here?

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Why Are We Here?

- We perceive and interact with the world through our senses.
  - Sight
  - Hearing
  - Taste
  - Smell
  - Touch



Why Are We Here?

- Unfortunately, research in hearing (audition) is outnumbered by research in sight (vision) in every measure:
  - Number of researchers/scholars/students
  - Number of books/papers/articles
  - Number of academic communities
  - And more…

Why Are We Here?

SNU Convergence

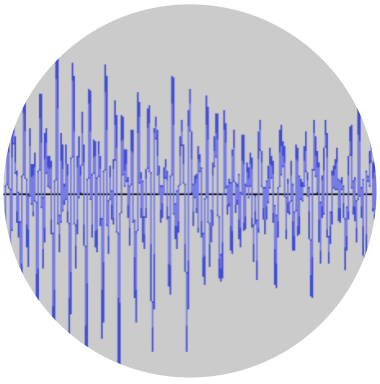
MARG MUSIC & AUDIO RESEARCH GROUP

Why Are We Here?

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

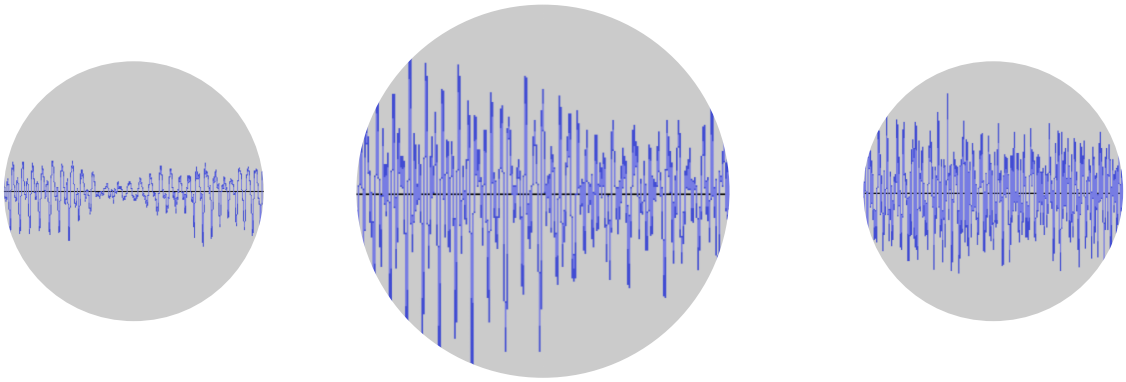
Human Listening



SNU Convergence

MUSIC & AUDIO RESEARCH GROUP

Human Listening



SNU Convergence

MUSIC & AUDIO RESEARCH GROUP

Human Listening

Arabic speech | English speech | Korean speech (Adult directed) | Korean speech (Infant directed)

Wherearethesilencesbetweenwords?

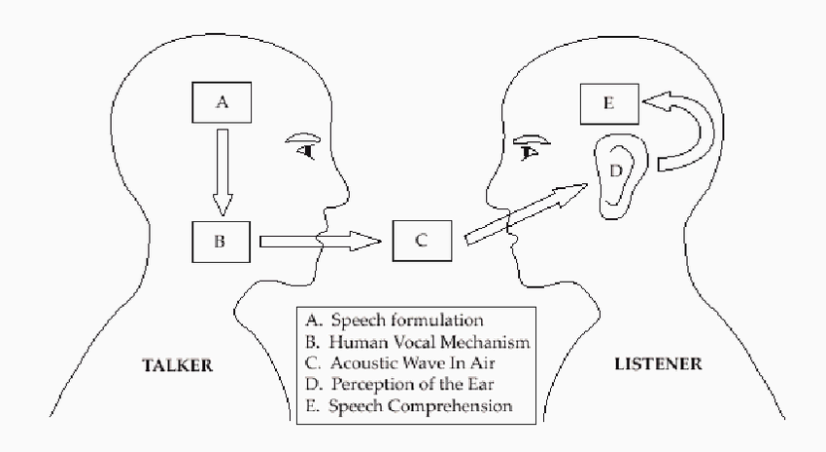
The segmentation problem

Where are the silences between words?

SNU Convergence

MUSIC & AUDIO RESEARCH GROUP

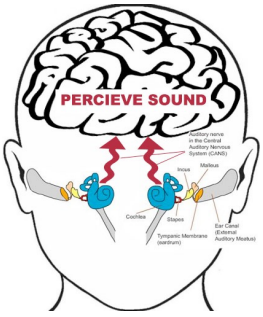
Human Listening (& Speaking)

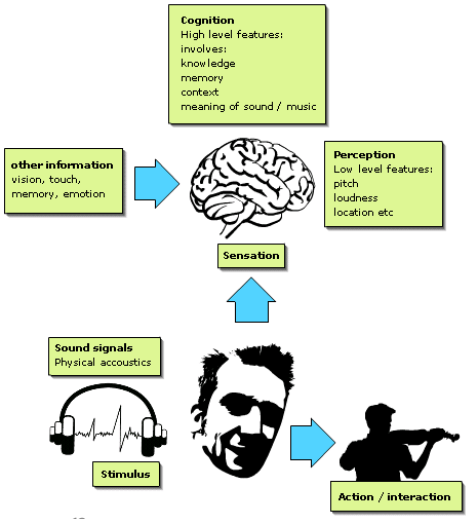


SNU Convergence

MUSIC & AUDIO RESEARCH GROUP

Human Listening (& Speaking/Playing)

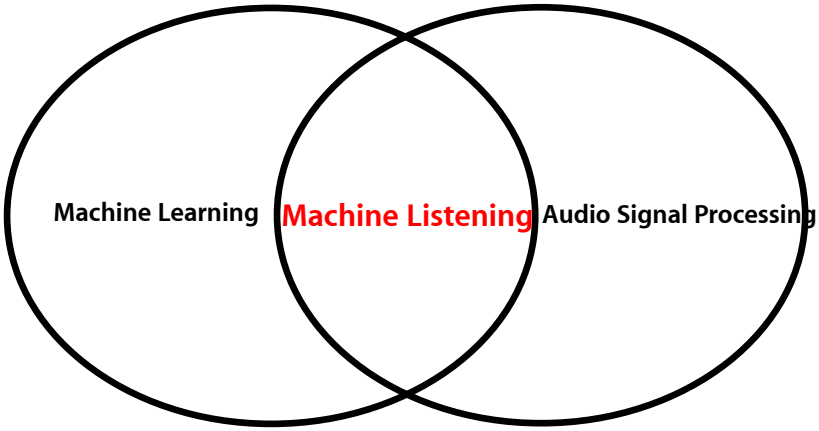




SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Machine Listening (computer audition)



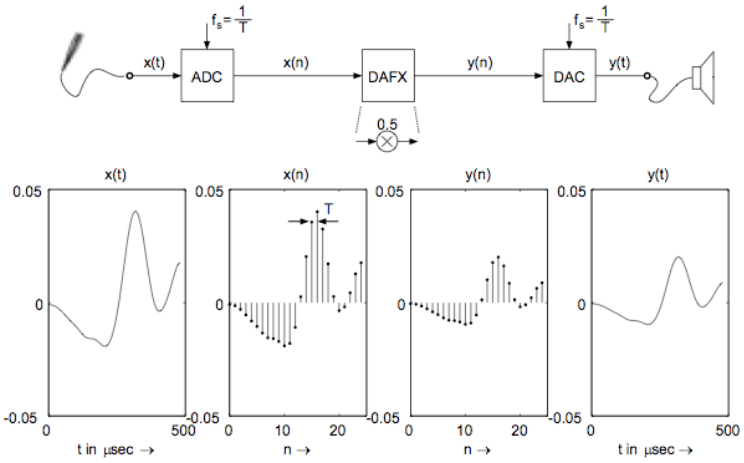
+ auditory perception/cognition, psychoacoustics…

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

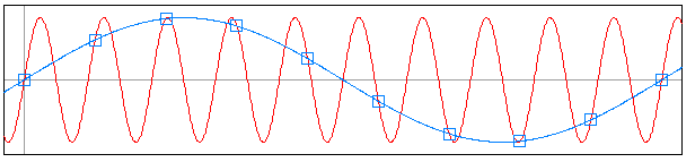
Fundamentals of Digital Audio Signal Processing

Digital Signals



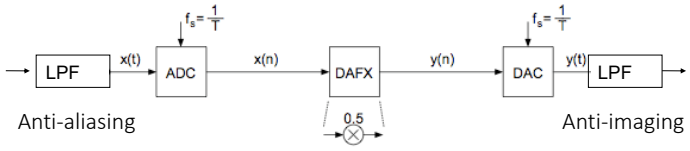
Sampling

- According to the sampling theorem: $f_s > 2f_{max}$ (Nyquist limit)
- Otherwise there is another, lower-frequency, signal that share samples with the original signal (aliasing)



- Related to the wagon-wheel effect: www.michaelbach.de/ot/mot_wagonWheel/index.html

Anti-aliasing[imaging]

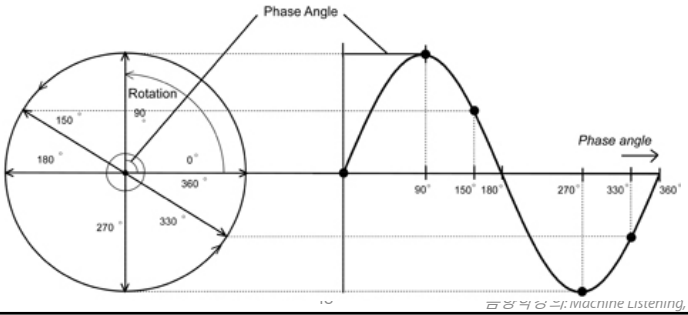


Sinusoid (continuous-time)

amplitude time (seconds) frequency (Hz)

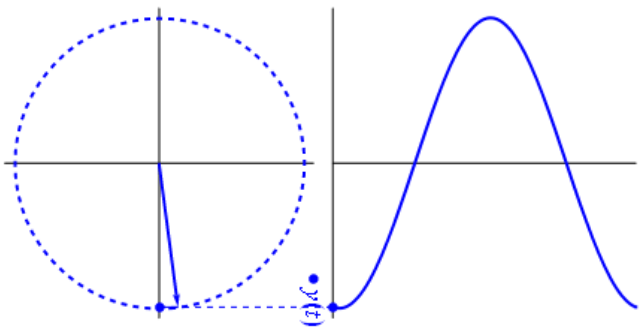
$$x(t) = A\cos(\omega t + \phi) = A\cos(2\pi f t + \phi)$$

angular frequency initial phase



Sinusoid (continuous-time) in action

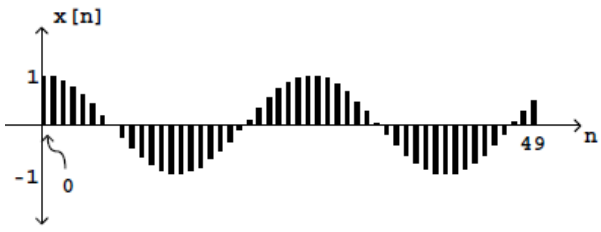
$$x(t) = A\cos(\omega t + \phi) = A\cos(2\pi f t + \phi)$$



Sinusoid (discrete-time)

- Since $t = \frac{n}{R}$, $x(n) = A\cos(\frac{2\pi f n}{R} + \phi)$,

  where R = sampling rate



SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Fourier Theorem

- Fourier theorem says: any *periodic* signal can be decomposed into a sum of sinusoids

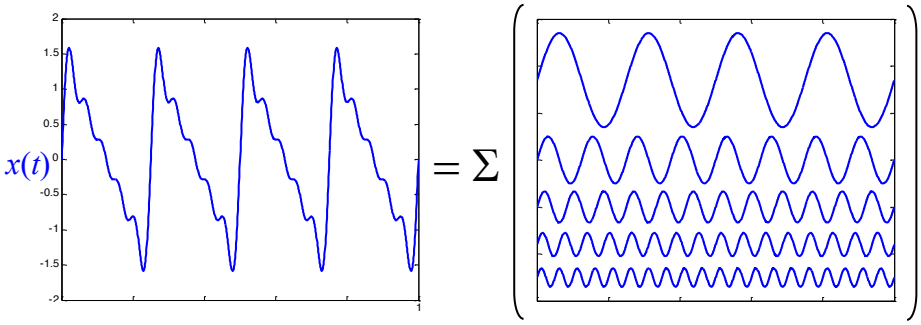
$$x(t) = a_0 + \sum_{k=1}^{\infty} a_k \cos(2\pi k f t + \phi_k)$$

- f is called *fundamental* and $2f$, $3f$, … are *harmonics* of fundamental
- Sequence of sinusoids with harmonic frequency is *harmonic series*

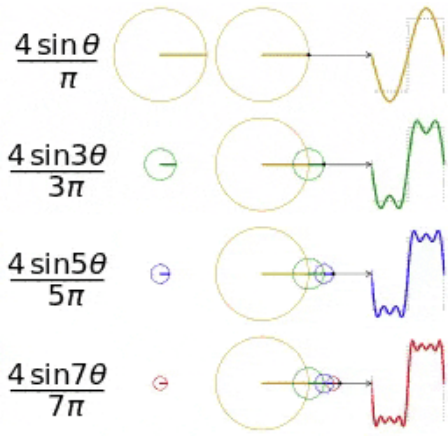
SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Fourier Theorem



Fourier Theorem



Fourier Transform (FT)

- The Fourier transform of a continuous-time signal $x(t)$ may be defined as

$$X(\omega) = FT[x(t)] = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt, \quad \omega \in (-\infty, \infty)$$

Discrete Fourier Transform (DFT)

- The Discrete Fourier Transform of a signal $x(n)$ may be defined as

$$X(k) = DFT[x(n)] = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}, \quad k = 0,1,\ldots,N-1$$

- The resulting N samples $X(k)$ are complex-valued:

$$X(k) = X_R(k) + jX_I(k)$$

$$|X(k)| = \sqrt{X_R^2(k) + X_I^2(k)} \quad \text{: magnitude}$$

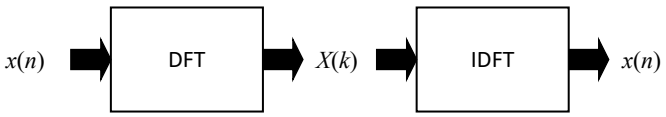
$$\varphi(k) = \arctan \frac{X_I(k)}{X_R(k)} \quad \text{: phase}$$

$$k = 0,1,\ldots,N-1$$

Inverse DFT (IDFT)

- The DFT allows perfect reconstruction of a signal $x(n)$ from its DFT $X(k)$ via inverse DFT defined as:

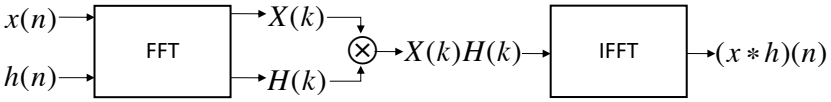
$$x(n) = IDFT[X(k)] = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi nk/N}, \quad n = 0,1,\ldots,N-1$$



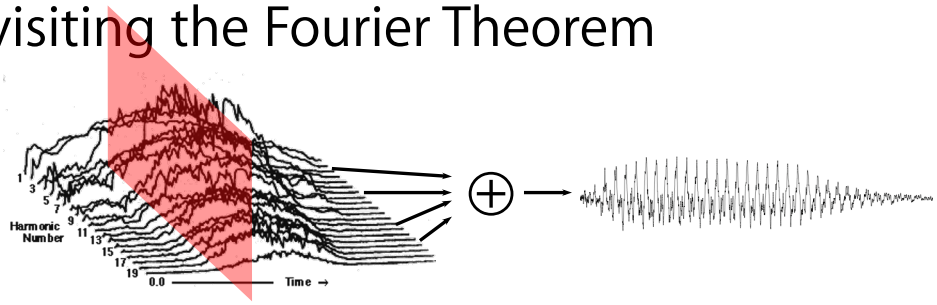
Fast Fourier Transform (FFT)

- An algorithm to efficiently compute the DFT is known as the Fast Fourier Transform (FFT) and its inverse as the IFFT
- Computational complexity
  - N-point DFT: $O(N^2)$
  - N-point FFT: $O(N \log N)$
- The FFT is so fast that even time-domain operations, like convolution, can be performed faster using FFT and IFFT instead:

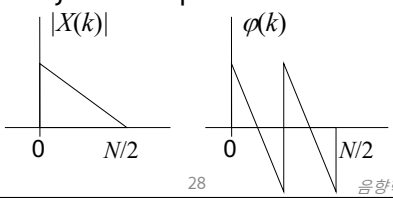
$$(x * h)(n) = \sum_{m=0}^{N-1} x(m)h(n-m)$$



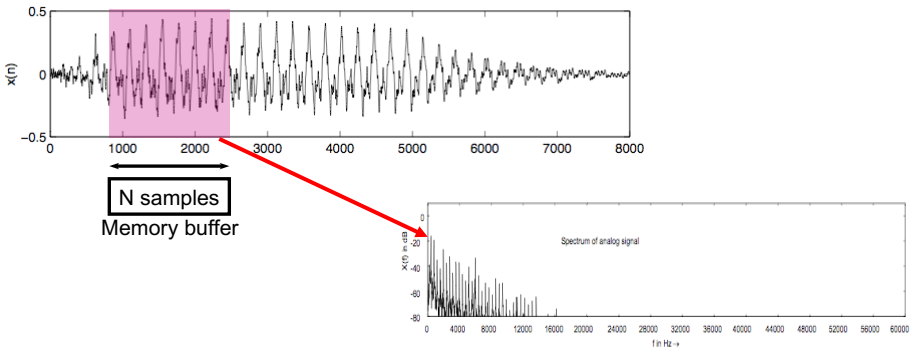
Revisiting the Fourier Theorem



- Any periodic signal can be described by a sum of a series of sinusoids with time-varying amplitudes and phases
- Thus a complex spectrum is just a snapshot of those sinusoids' parameters



Short-time Fourier Transform (STFT)



- For block processing, a short segment is sent to a buffer and processed as a block
- The DFT done in this way is called the short-time Fourier Transform or STFT

Frequency Resolution

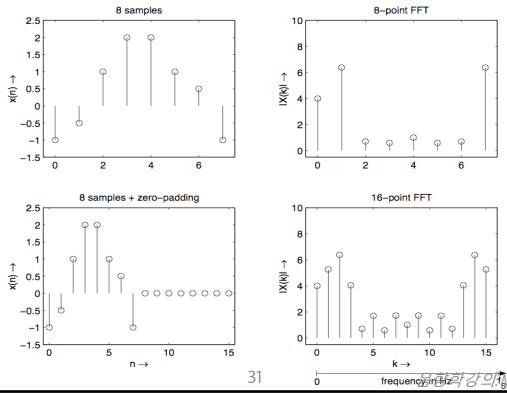
- Determined by how many sinusoids are used to describe a spectrum
- In N-point DFT or FFT, the frequency resolution is given by

$$\Delta f = f_s / N$$

- Intuitively, we can have finer frequency resolution if we increase N
- However, this results in poorer temporal resolution => tradeoff between frequency[spectral] and time[temporal] resolution

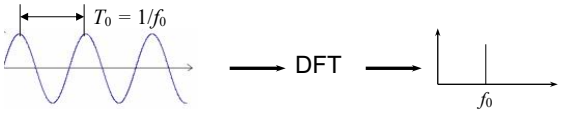
Zero-Padding

- A possible solution to increase frequency resolution without increasing N - *i.e.*, without losing time resolution
- Add zero-valued samples - thus doesn't change spectrum itself - to yield better spectral resolution



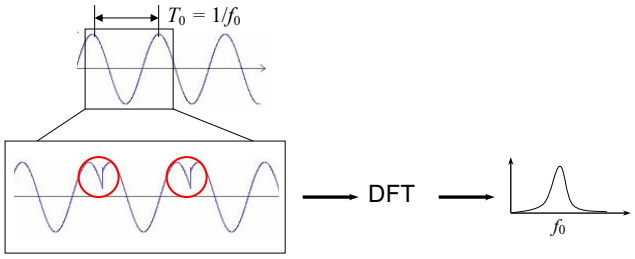
DFT of a Sinusoid

- In theory the DFT of a pure sinusoid results in a single sharp line at the frequency of the sinusoid



Spectral Leaking

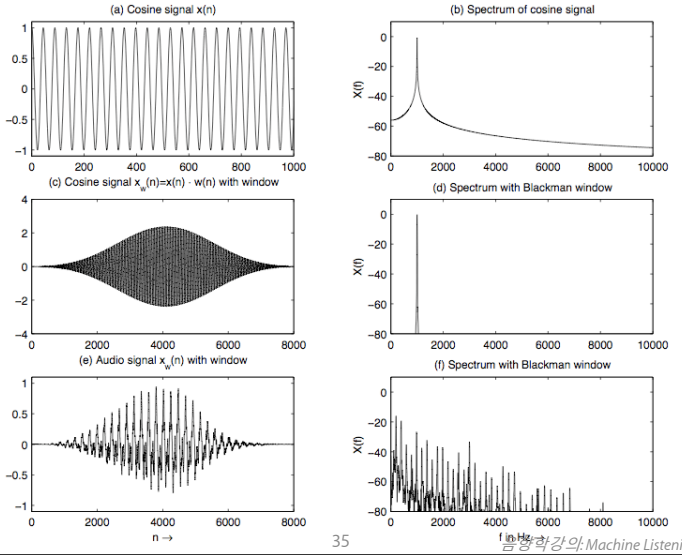
- In practice, unless we perform f_0-synchronous analysis, there are discontinuities (sharp changes) at the segment boundaries that introduce some noise. Thus the spectral line around f_0 is smeared.
- This is known as *spectral leaking*



Windowing

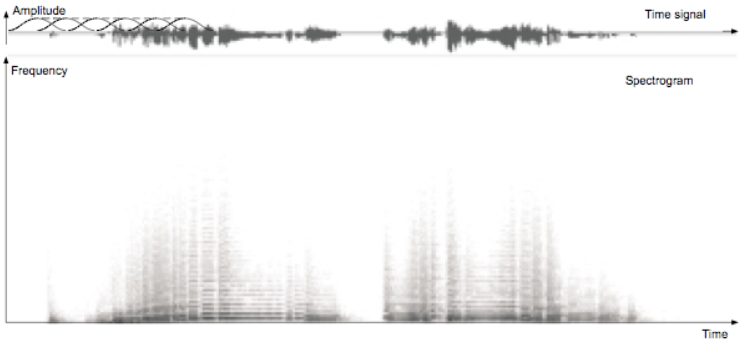
- In order to reduce spectral leaking, we need to avoid abrupt changes between the segment boundaries
- This is done by multiplying a *window* whose amplitude gradually reaches zero at both ends, thus guaranteeing the continuity of a segmented signal when repeated
- Possible windows are: rectangular, hamming, hann, Blackman, Gaussian, and so on.

Windowing Example



2-D Time-Frequency Representation

- Using the STFT, independent DFTs are calculated on windowed segments
- The segments usually overlap to compensate for the loss of temporal resolution
- Produces a 2-D *spectrogram*



Acoustic Features

Audio Feature Extraction

- Raw audio samples are:
  - Noisy
  - Redundant
  - Computationally inefficient
  - Not a good input to audio applications
- Need to convert them to more robust, compact yet meaningful representations called *audio features* or *acoustic features*

Acoustic (audio) Features

- Spectral features
  - Spectral low-level features: spectral centroid, spectral flatness measure, spectral flux, etc.
  - Spectral envelope: LPCs, MFCCs, etc.
- Temporal features
  - ZCR (zero crossing rate), tempo histogram, novelty function, etc.
- Tonal features
  - PCP (pitch class profile) or chroma, chromagram, tonal centroid, etc.

Spectral Low-level Features

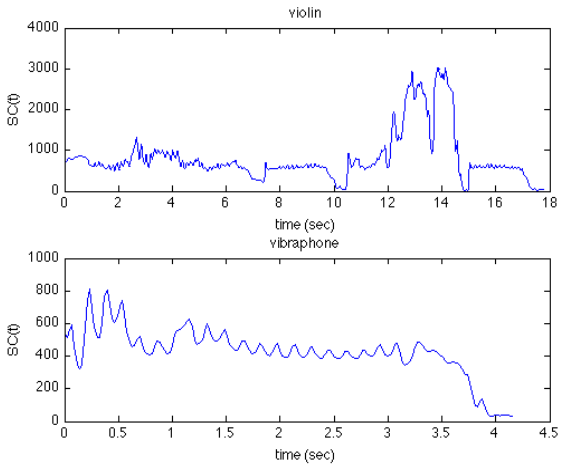
Spectral Centroid (SC)

$$SC = \frac{\sum_{k=0}^{N/2} f_k |X(k)|^2}{\sum_{k=0}^{N/2} |X(k)|^2},$$

where f_k is the center frequency of kth bin and |X(k)| is the DFT

- Characterizes the center of gravity of the (power) spectrum
- Usually associated with the "sharpness / dullness (or brightness / darkness)" of a sound

Spectral Centroid: Example



(a) violin

(b) vibraphone

Temporal evolution of the spectral centroid for two instrumental sounds
(a) violin and (b) vibraphone

Spectral Spread (SS)

$$SS = \sqrt{\frac{\sum_{k=0}^{N/2} (f_k - SC)^2 |X(k)|^2}{\sum_{k=0}^{N/2} |X(k)|^2}}$$

- Measure of the average spread of the spectrum in relation to its centroid
- Noisy, broadband signals have high SS while tonal sounds show lower SS

Spectral Flatness Measure (SFM)

$$SFM_b = \frac{\sqrt[N_b]{\prod_{k_b} |X(k_b)|^2}}{\frac{1}{N_b} \sum_{k_b} |X(k_b)|^2}, \quad k_b = k_l, k_l + 1, \ldots, k_u$$

where N_b is the number of spectral bins in a subband or
$N_b = k_u - k_l + 1$

- Reflects how "flat" a signal's power spectrum is
- Calculated as the ratio of the geometric mean and the harmonic mean
- Usually computed *per* spectral band (critical bands or bark bands, etc.), thus SFM_b
- Flatness for the whole spectrum is the average of the subband measures

Harmonic Spectral Centroid (HSC)

$$HSC = \frac{\sum_{h=1}^{N_h} f_h A_h}{\sum_{h=1}^{N_h} A_h}$$

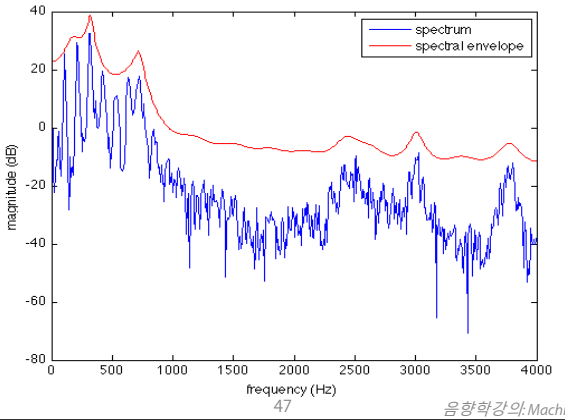
where f_h and A_h are the frequency and the amplitude of the h_{th} harmonic, respectively

- Measure of the amplitude-weighted mean of the harmonic (spectral) peaks of the spectrum
- Compared to SC, HSC focuses only on harmonic (spectral) peaks, which are more musically meaningful in general
- Harmonic spectral spread (HSS) is similarly defined

Spectral Envelope

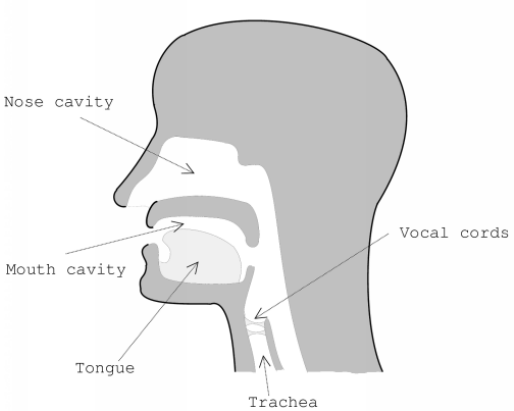
What is the Spectral Envelope?

- Spectral envelope is a smoothed version of the spectrum that preserves its overall shape while neglecting its fine spectral structure



Human Speech System

- Vocal cords act as an oscillator, which generates a spectrally rich *source* signal
- Everything else is *filter*: vocal tract, mouth/nose cavity, tongue
- Thus called "source-filter" model
- These filters define the shape of the spectral envelope

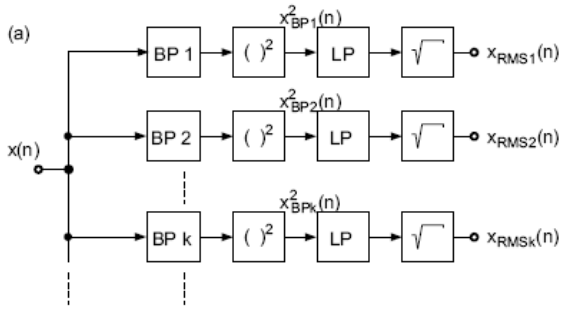


Spectral Envelope Estimation

- A few popular techniques to estimate the spectral envelope
  - Channel vocoder: estimates the amplitude of the signal within several frequency bands
  - Linear prediction: estimates the parameters (or filter coefficients) of a filter that approximates the spectrum
  - Cepstrum analysis: inverse-FFT the log-spectrum and low-pass filters it to obtain the envelope

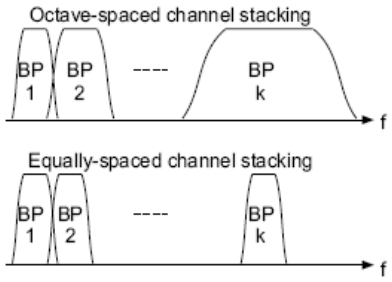
Channel Vocoder (1)

- Filters a signal with a bank of bandpass filters
- Calculates RMS of each bandpassed signal
- The more filters used, the finer spectral envelope estimated



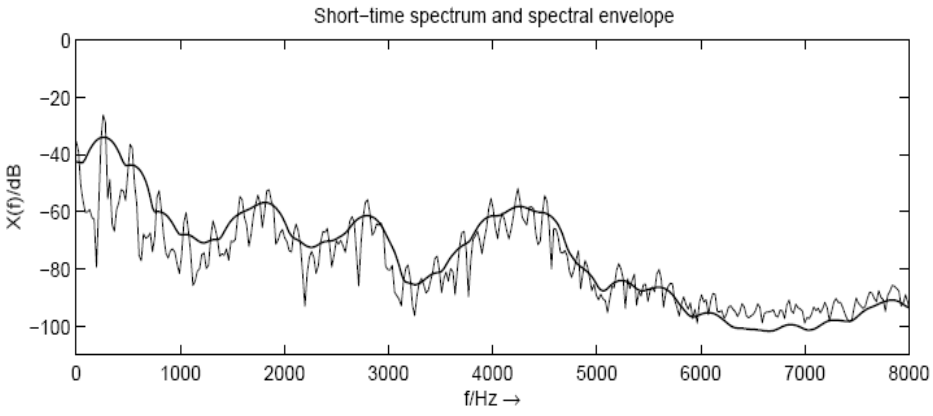
Channel Vocoder (2)

- In the frequency domain, multiply the spectrum with the filters' frequency response and square-root the sum of each filter's output



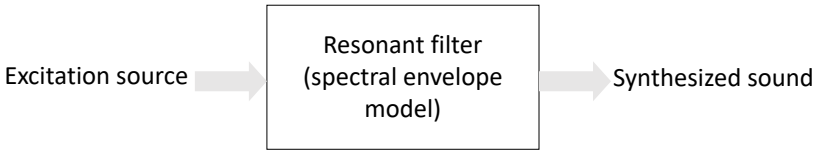
- The filterbank can be either linearly or logarithmically spaced (e.g., constant-Q or Mel-scale filter bank)

Channel Vocoder Example



Linear Predictive Coding (1)

- Linear predictive coding is a source-filter model that approximates the way a sound is generated as an excitation (a pulse train or noise) passing through an all-pole resonant filter



- Widely used in speech and music applications
- Reduces large amount of data (e.g., N samples) to a few filter coefficients while preserving the overall shape

Linear Predictive Coding (2)

- The nth sample $x(n)$ is extrapolated, i.e., predicted by a linear combination of p past samples:

$$x(n) \approx \hat{x}(n) = \sum_{k=1}^{p} a_k x(n-k)$$

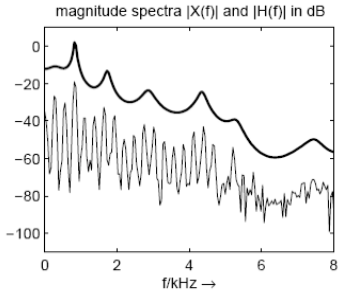
- The residual error is given by

$$e(n) = x(n) - \hat{x}(n) = x(n) - \sum_{k=1}^{p} a_k x(n-k)$$

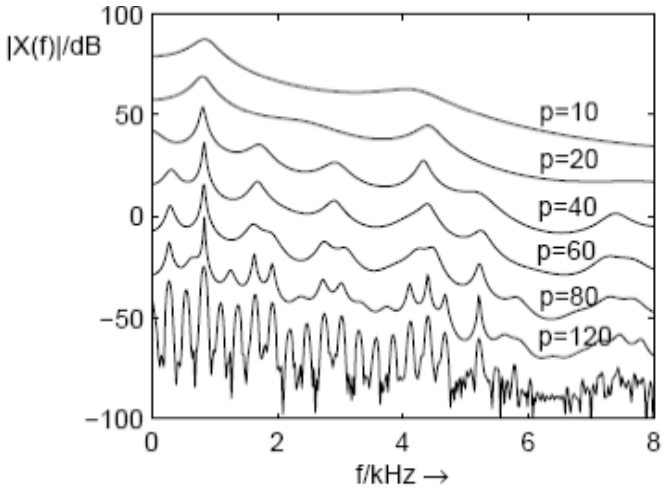
and we want to minimize this error

Linear Predictive Coding (3)

- The parameters a_k's are called linear predictive coefficients (LPCs)
- The filter represented by these coefficients is a resonant filter and its frequency response represents the spectral envelope
- The higher the filter order p, the closer the approximation is to the signal's spectrum



LPC Order and Approximation

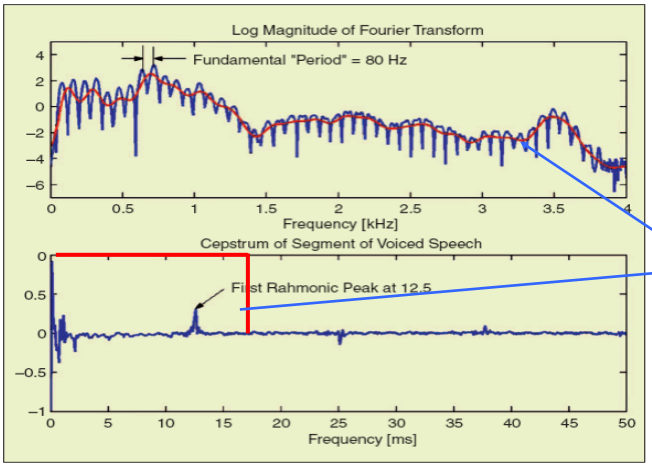


Cepstrum Analysis

- Cepstrum is the result of taking the FFT of the log-spectrum as if it were a *signal*
- Measures the rate of change in different spectral bands
- The name cepstrum was coined by Bogert *et al.* (1963) by reversing the first four letters of the spectrum (similarly for quefrency alanysis and liftering, etc.)
- For a real signal $x(n)$, the real cepstrum is calculated as follows:

$$c_R(n) = IFFT\left(\log(|X(k)|)\right)$$

Cepstrum & Spectral Envelope



By low-pass "liftering" the cepstrum we obtain the spectral envelop the signal

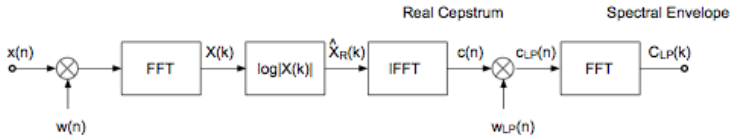
Spectral Envelope Estimation

- Using a low-pass window of the form:

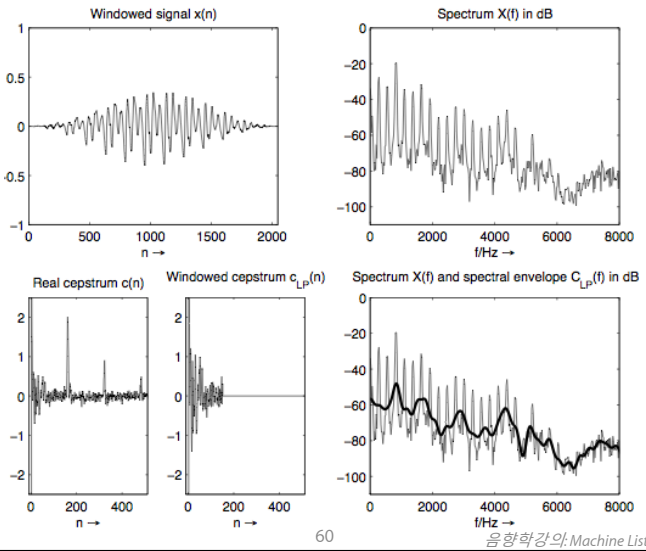
$$\omega_{LP}(n) = \begin{cases} 1 & n = 0, N_1 \\ 2 & 1 \le n \le N_1 \\ 0 & N_1 < n \le N-1 \end{cases}$$

we can low-pass the cepstrum and obtain the spectral envelope by:

$$c_{LP}(n) = c_R(n) \cdot \omega_{LP}(n)$$
$$C_{LP}(k) = FFT[c_{LP}(n)]$$



Cepstrum Example

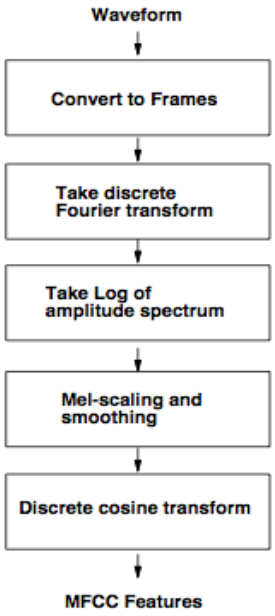


SNU Convergence

MUSIC & AUDIO RESEARCH GROUP

MFCC

- Mel-frequency Cepstral Coefficients (MFCCs) are a variation of cepstrum, motivated by human perception (Logan, 2000)
- Most extensively used in speech and music applications (e.g., speech recognition, genre classification, instrument recognition, etc.), due to its ability to compactly represent the spectral characteristics (just ~13 coefficients)



SNU Convergence

MUSIC & AUDIO RESEARCH GROUP

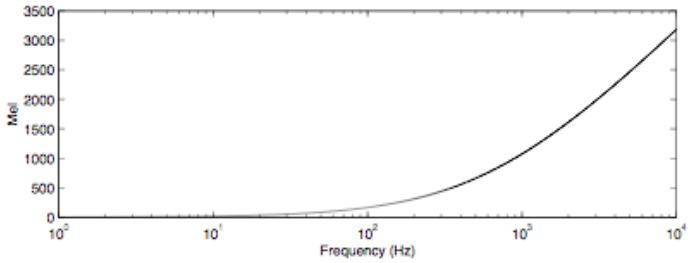
Mel Scale

- The Mel scale is a non-linear perceptual scale of pitches judged to be equidistant
- Approximately linear below 1 kHz and logarithmic above
- 1 kHz corresponds to 1000 Mel (reference point)
- With the Mel scale, a 1000-Mel tone should sound as twice as high as a 500-Mel tone (this is not true with linear frequency Hz)

Mel vs. Linear Frequency

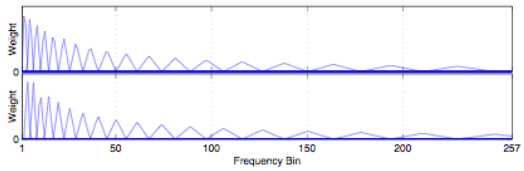
- The relation between Mel and Hz is given by

$$m = 1127.01048 \log(1 + f/700)$$
$$f = 700(e^{m/1127.01048} - 1)$$

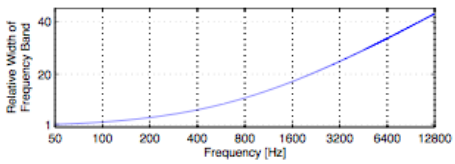


Mel-frequency Spectrum

- To convert a linear spectrum to Mel we can use a filterbank of overlapping triangular windows:



- Such that the width d of each window increases according to the Mel scale, and the height of each triangle is $2/d$

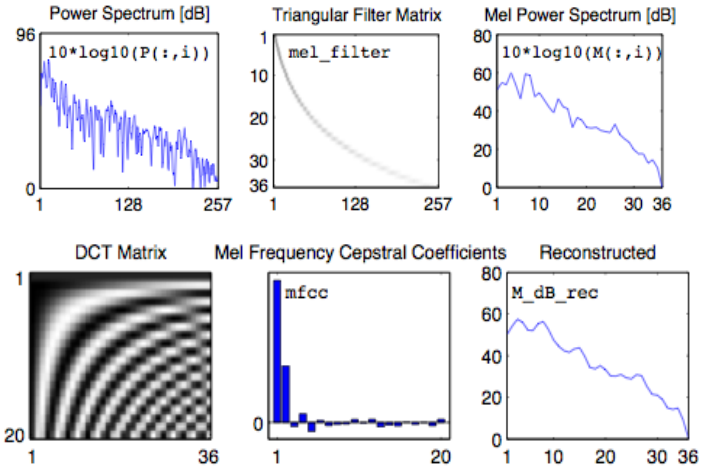


Decorrelation of Mel-scale Spectrum

- The resulting Mel-scale spectral vectors are highly correlated with each other; i.e. highly redundant
- Thus a more efficient representation of the log-spectrum can be obtained by applying a transform that decorrelates those vectors (Rabiner and Juang, 93)
- This decorrelation is commonly approximated by means of the Discrete Cosine Transform (DCT)
- The DCT is similar to a DFT but only for real numbers. It has the property that most of its energy is concentrated on a few initial coefficients (thus effectively compressing the spectral info)

$$X_{DCT}(k) = \sqrt{\frac{2}{N}} \sum_{N=0}^{N-1} x(n) \cos[\frac{\pi}{N}(n + \frac{1}{2})k]$$

Fast Computation of MFCCs



MFCCs roughly model certain characteristics of human auditory perception: the non-linear perception of loudness and frequency and spectral masking (Pampalk, 2006)

Tonal Features

Tonality

- Very important attribute in (Western) tonal music
- Explain the relationship among the tones
- Several musical attributes are closely related with tonality
  - Scale
  - Key
  - Pitch
  - Interval
  - Chord

Tonal Features vs. Spectral Features

- Spectral features
  - Good for describing certain spectral characteristics (e.g., sharpness, noisiness, etc.)
  - Good for representing sonic texture or timbre by capturing overall frequency magnitude response (e.g., LPCs, cepstral coefficients, MFCCs)
  - Not good for tonal analysis: pitch- or tone-relevant information gets lost
- Tonal features retain tonal structure in musical audio
  - Tonal relations
  - Interval relations

Constant-Q Transform

Constant-Q Transform

- In DFT, the center frequency f_k of the frequency bin is given by

$$f_k = \frac{f_s}{N} k, \quad k = 0,1,...,N-1$$

  where f_s is the sampling rate and N is the DFT size
- Therefore, all the frequency bins are linearly spaced
- However, musical scale as well as human hearing mechanism are *logarithmic*
- Brown proposed the constant-Q transform whose frequency resolution conforms to equal-tempered scale (1990)
- Well suited for pitch-related analysis

Constant-Q Transform (cont'd)

- In constant-Q transform the kth spectral frequency is defined as

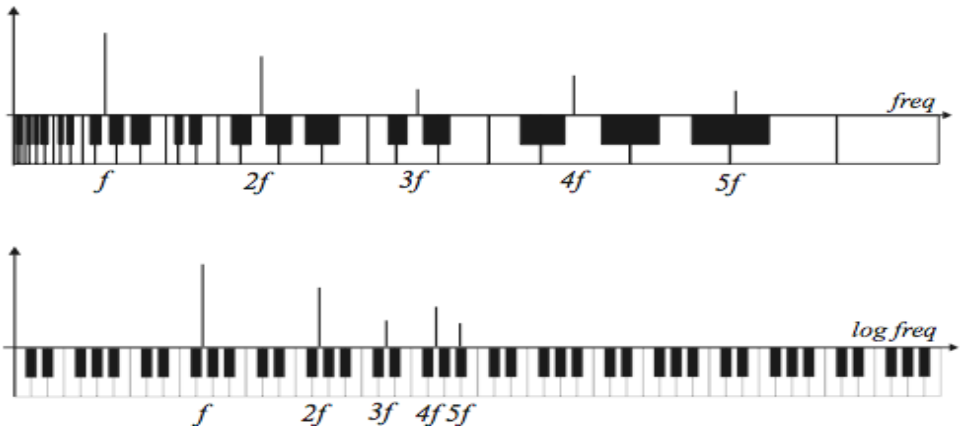
$$f_k = (2^{1/B})^k f_{\min}, \quad k = 0,1,...,N-1,$$

  where B is the number of bins in an octave and f_{min} is the minimum frequency set by user
- It is called "constant-Q" because Q or "quality factor" is constant along the frequency axis, which is defined as

$$Q = \frac{f_k}{f_w}, \quad k = 0,1,...,N-1,$$

  where f_w is the filter width

Linear vs. Logarithmic Frequency



Computation of CQ Transform

- CQ transform can be obtained from the DFT using logarithmically-spaced filterbank

$$X_{cq}(k) = \frac{1}{N(k)} \sum_{n=0}^{N(k)-1} x(n) w(n,k) e^{-j2\pi Q n / N(k)}$$

$$N(k) = f_s Q / f_k$$

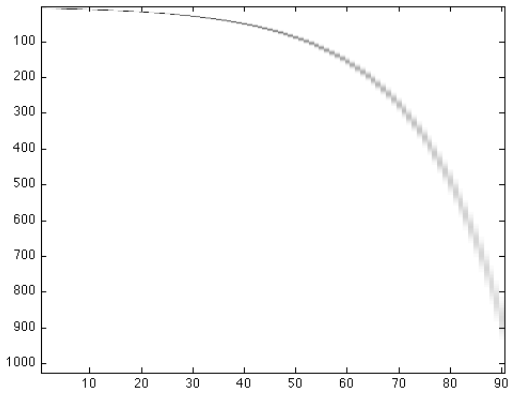
- That uses a variable window length to obtain more resolution at lower frequencies and less at higher (logarithmic distribution of bins in frequency)

Computation of CQT (cont'd)

- CQ transform can be efficiently computed using a *CQ kernel* which is a 2-d matrix that maps the DFT to the CQT

N-point DFT vector	x	NxM CQ kernel matrix	=	M-point CQT vector

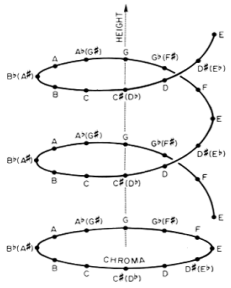
CQ Kernel Matrix

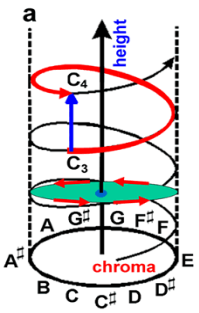


Chroma

Pitch Helix

- The pitch helix is a pitch space where linear pitch is wrapped around a cylinder, thus modeling the special relationship that exists between octave intervals





- Two dimensions
  - Height: naturally organizes *absolute* pitches from low to high
  - Chroma: represents the inherent circularity of pitch (*relative* relationship between pitch classes)

Chroma (aka Pitch Class Profile)

- Good for describing relative pitch relationship, disregarding absolute pitch height
- Very useful for harmony analysis, key and chord, in particular
- A key and/or a chord can be described as a function of its pitch classes
- Almost universal feature for key/chord estimation applications
- Chroma as audio feature first introduced by Fujishima (1999)

Computation of Chroma

- Easily computed from the constant-Q transform by collapsing it to an octave, or

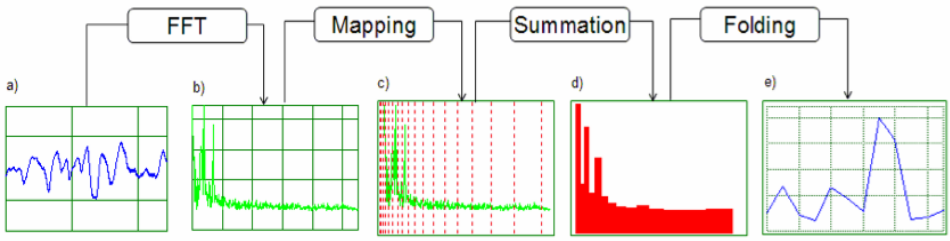
$$Chroma(b) = \sum_{m=0}^{M-1} \left| X_{CQ}(b + mB) \right|,$$

where $X_{CQ}(k)$ is the CQ transform,
M is the total number of octaves of interest,
B is the number of chroma bins in an octave,
and $b = 1,2,\ldots,B$ is the chroma bin index

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

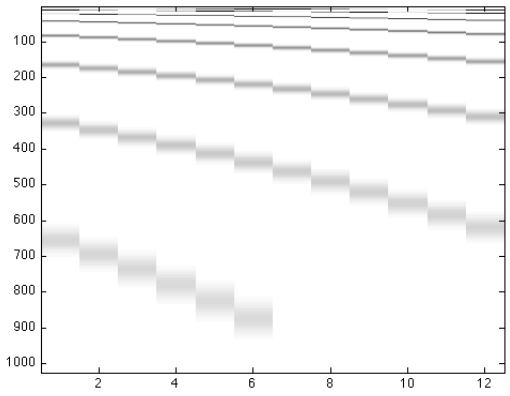
Computation of Chroma (cont'd)



SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Chroma Kernel Matrix



Machine Learning

Machine Learning definition

- Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.
- Tom Mitchell (1998) Well-posed Learning Problem: A computer program is said to *learn* from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Machine learning algorithms:
- Supervised learning
- Unsupervised learning

Others: Reinforcement learning, recommender systems.

deeplearning.ai

Introduction to Deep Learning

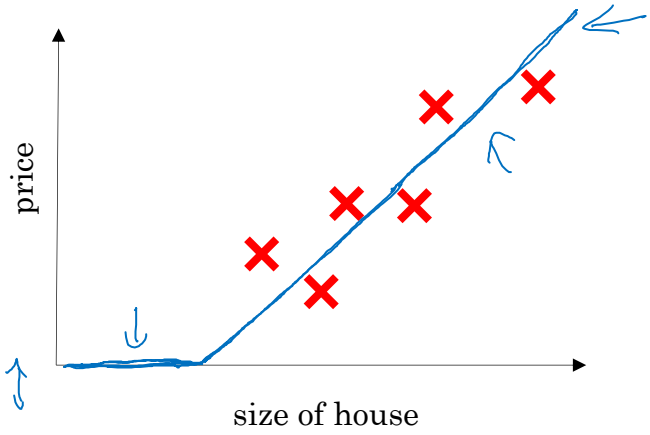
What is a Neural Network?

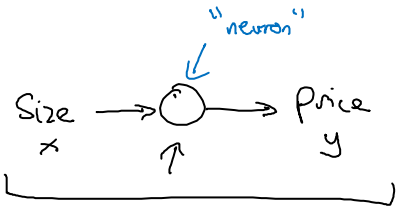
Lecture slides are from: https://www.coursera.org/learn/deep-neural-network/

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Housing Price Prediction

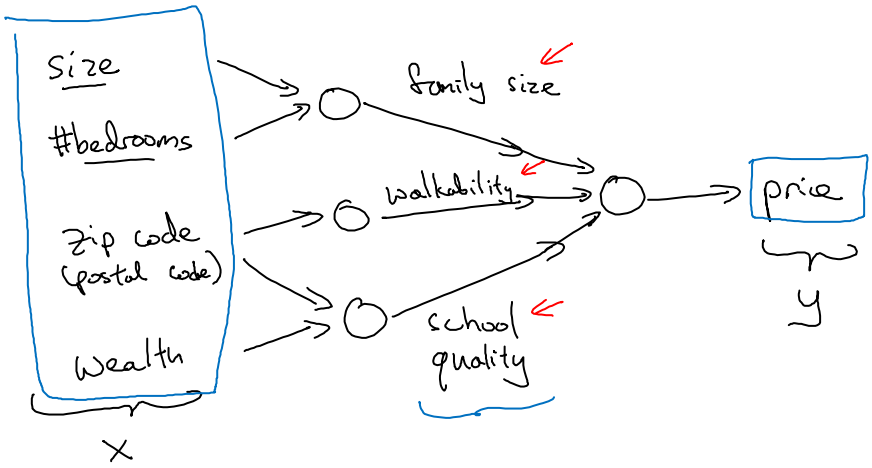




SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

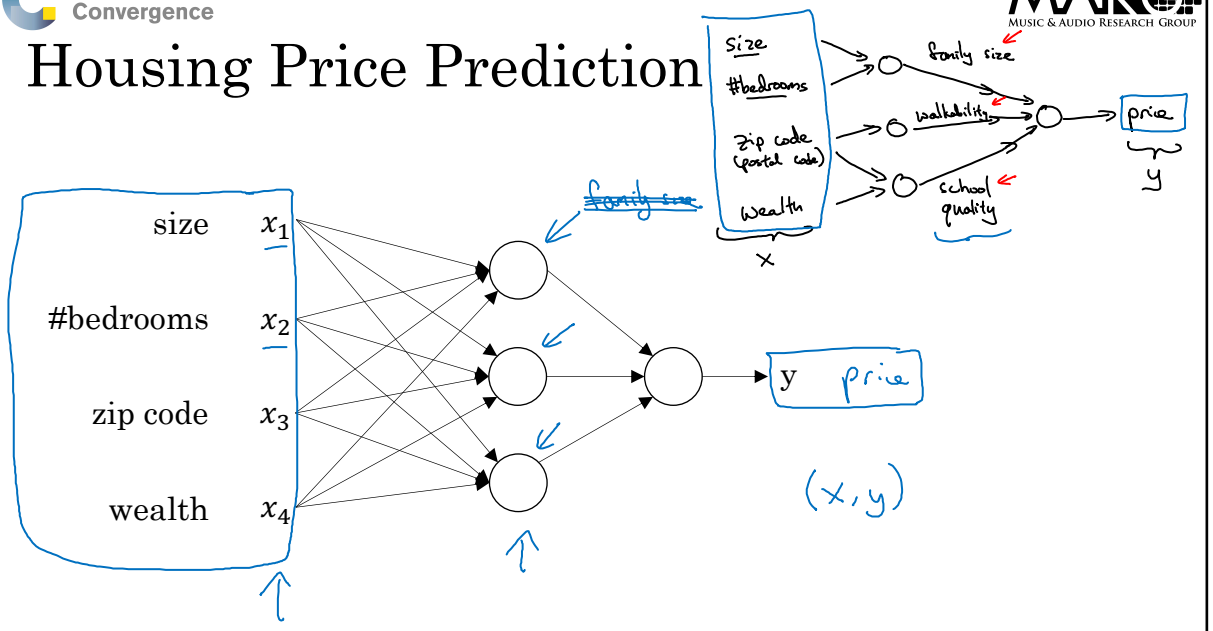
Housing Price Prediction



SNU Convergence

MUSIC & AUDIO RESEARCH GROUP

Housing Price Prediction



deeplearning.ai

Introduction to Deep Learning

Supervised Learning with Neural Networks

SNU Convergence

MUSIC & AUDIO RESEARCH GROUP

Supervised Learning

Input(x)	Output (y)	Application
Home features	Price	Real Estate
Ad, user info	Click on ad? (0/1)	Online Advertising
Image	Object (1,…,1000)	Photo tagging
Audio	Text transcript	Speech recognition
English	Chinese	Machine translation
Image, Radar info	Position of other cars	Autonomous driving

Standard NN

CNN

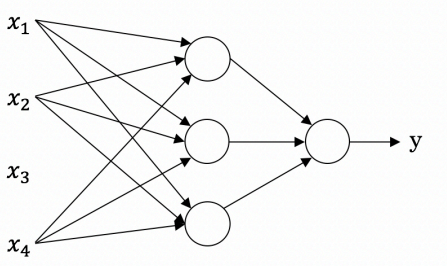
RNN

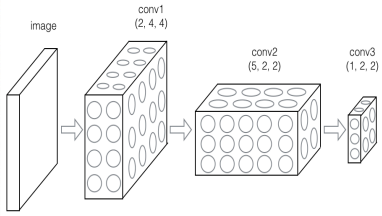
Custom/ Hybrid

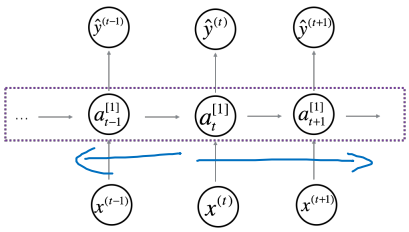
SNU Convergence

MUSIC & AUDIO RESEARCH GROUP

Neural Network examples







Standard NN

Convolutional NN

Recurrent NN

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

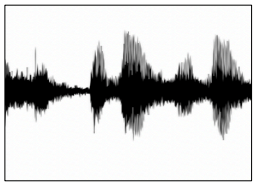
Supervised Learning

Structured Data

Size	#bedrooms	⋯	Price (1000$s)
2104	3		400
1600	3		330
2400	3		369
⋮	⋮		⋮
3000	4		540

User Age	Ad Id	⋯	Click
41	93242		1
80	93287		0
18	87312		1
⋮	⋮		⋮
27	71244		1

Unstructured Data



Audio

Image

Four scores and seven years ago…

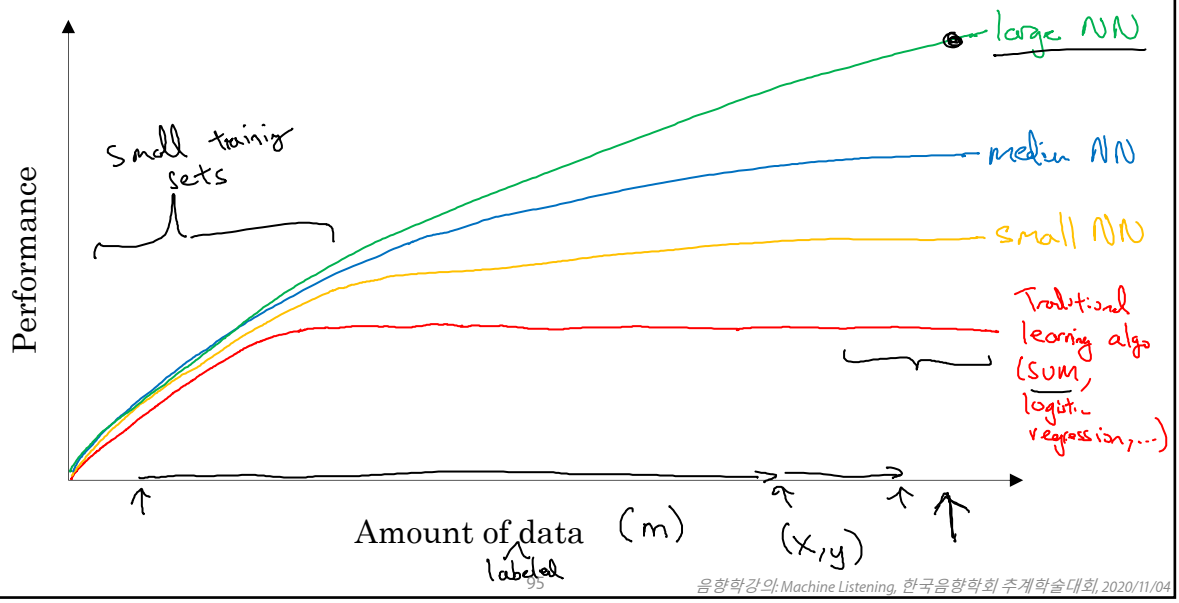
Text

deeplearning.ai

Introduction to Neural Networks

Why is Deep Learning taking off?

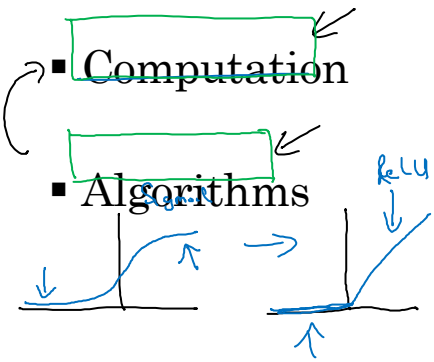
Scale drives deep learning progress

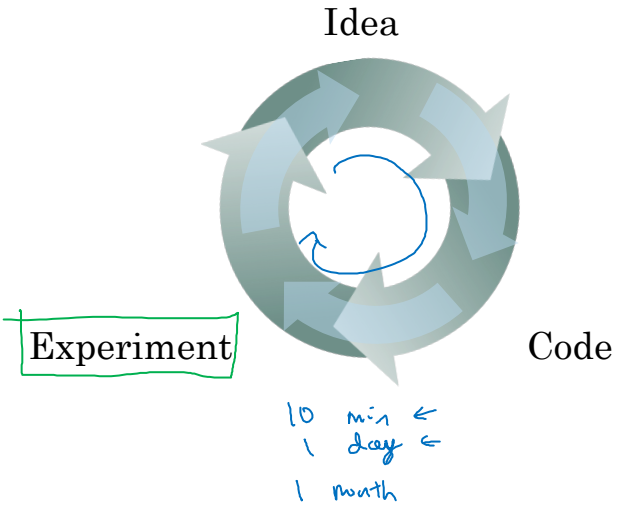


95

Scale drives deep learning progress

- Data
- Computation
- Algorithms





96

deeplearning.ai

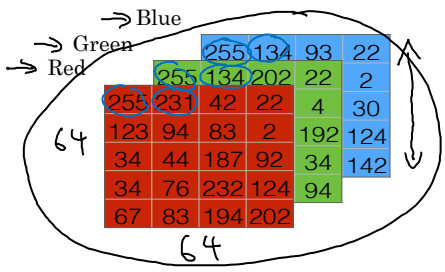
Basics of Neural Network Programming

Binary Classification

Binary Classification

1 (cat) vs 0 (non cat)

y



$$x = \begin{bmatrix} 255 \\ 231 \\ \vdots \\ 255 \\ 134 \\ \vdots \end{bmatrix}$$

$64 \times 64 \times 3 = 12288$

$n = n_x = 12288$

$x \longrightarrow y$

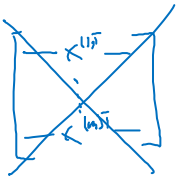
Notation

(x, y) $\quad x \in \mathbb{R}^{n_x}$, $y \in \{0, 1\}$

m training examples: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

$m = m_{train}$ $\quad m_{test}$ = #test examples.

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$ (n_x rows, m columns)



$X \in \mathbb{R}^{n_x \times m}$ $\quad X.shape = (n_x, m)$

$$Y = \begin{bmatrix} y^{(1)} & y^{(2)} & \cdots & y^{(m)} \end{bmatrix}$$

$Y \in \mathbb{R}^{1 \times m}$

$Y.shape = (1, m)$

deeplearning.ai

Basics of Neural Network Programming

Logistic Regression

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Logistic Regression

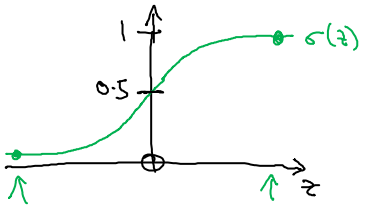
Given x, want $\hat{y} = P(y=1|x)$

$x \in \mathbb{R}^{n_x}$

$0 \le \hat{y} \le 1$

Parameters: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$.

Output $\hat{y} = \sigma(w^T x + b)$, z



$x_0 = 1$, $x \in \mathbb{R}^{n_x+1}$

$\hat{y} = \sigma(\theta^T x)$

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix}$ θ_0 ← b, $\theta_1 \dots \theta_{n_x}$ ← w

$\sigma(z) = \frac{1}{1+e^{-z}}$

If z large $\sigma(z) \approx \frac{1}{1+0} = 1$

If z large negative number

$\sigma(z) = \frac{1}{1+e^{-z}} \approx \frac{1}{1+\text{Bignum}} \approx 0$

deeplearning.ai

Basics of Neural Network Programming

Logistic Regression cost function

Logistic Regression cost function

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, where $\sigma(z^{(i)}) = \frac{1}{1+e^{-z}}$ (i)

$z^{(i)} = w^T x^{(i)} + b$

$x^{(i)}$, $y^{(i)}$, $z^{(i)}$: i-th example.

Given $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$.

Loss (error) function: $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$

$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y) \log(1-\hat{y}))$

If $y=1$: $\mathcal{L}(\hat{y}, y) = -\log \hat{y}$ ← want $\log \hat{y}$ large, want $\hat{y}$ large.

If $y=0$: $\mathcal{L}(\hat{y}, y) = -\log(1-\hat{y})$ ← want $\log 1-\hat{y}$ large .... want $\hat{y}$ small

Cost function: $J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)}) \right]$

deeplearning.ai

Basics of Neural Network Programming

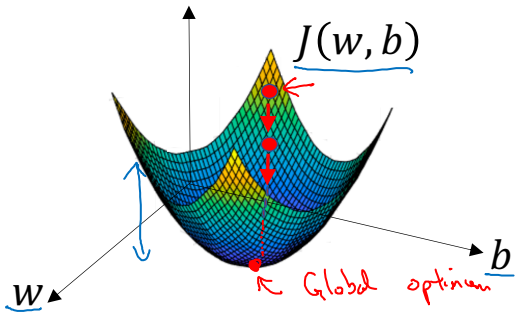
Gradient Descent

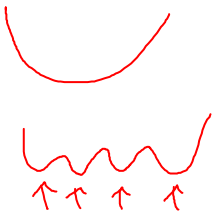
Gradient Descent

Recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$ ←

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

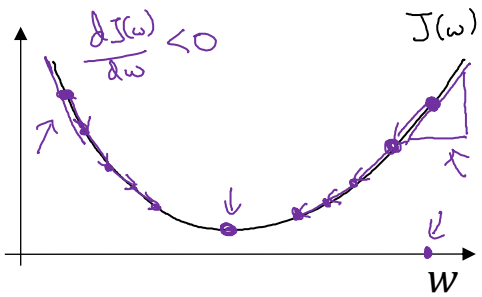
Want to find w, b that minimize $J(w,b)$





105

Gradient Descent



Repeat {

$w := w - \alpha \frac{dJ(w)}{dw}$ ← learning rate

"dw"

}

$w := w - \alpha dw$

$\frac{dJ(w)}{dw} = ?$

$J(w,b)$

$w := w - \alpha \frac{\partial J(w,b)}{\partial w}$ → $\frac{\partial J(w,b)}{\partial w}$ → dw

$b := b - \alpha \frac{\partial J(w,b)}{\partial b}$ → $\frac{\partial J(w,b)}{\partial b}$ → db

"partial derivative" J

106

deeplearning.ai

Basics of Neural Network Programming

Logistic Regression Gradient descent

SNU Convergence

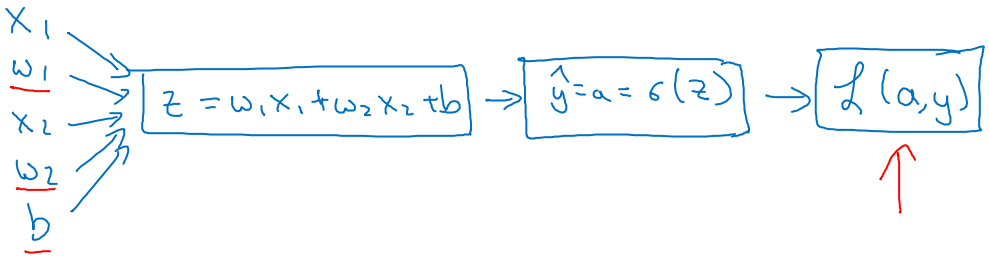
MARG MUSIC & AUDIO RESEARCH GROUP

Logistic regression recap

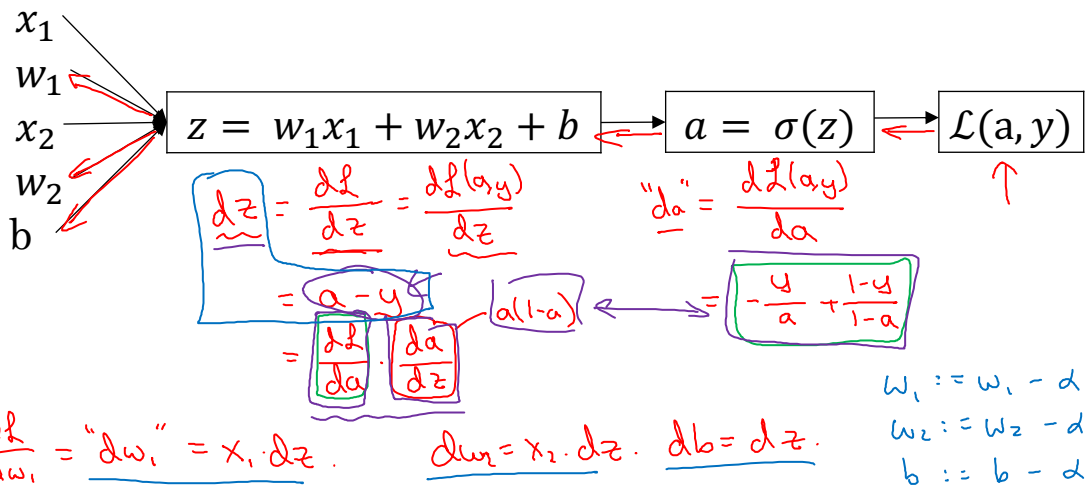
$z = w^T x + b$

$\hat{y} = a = \sigma(z)$

$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$



Logistic regression derivatives



deeplearning.ai

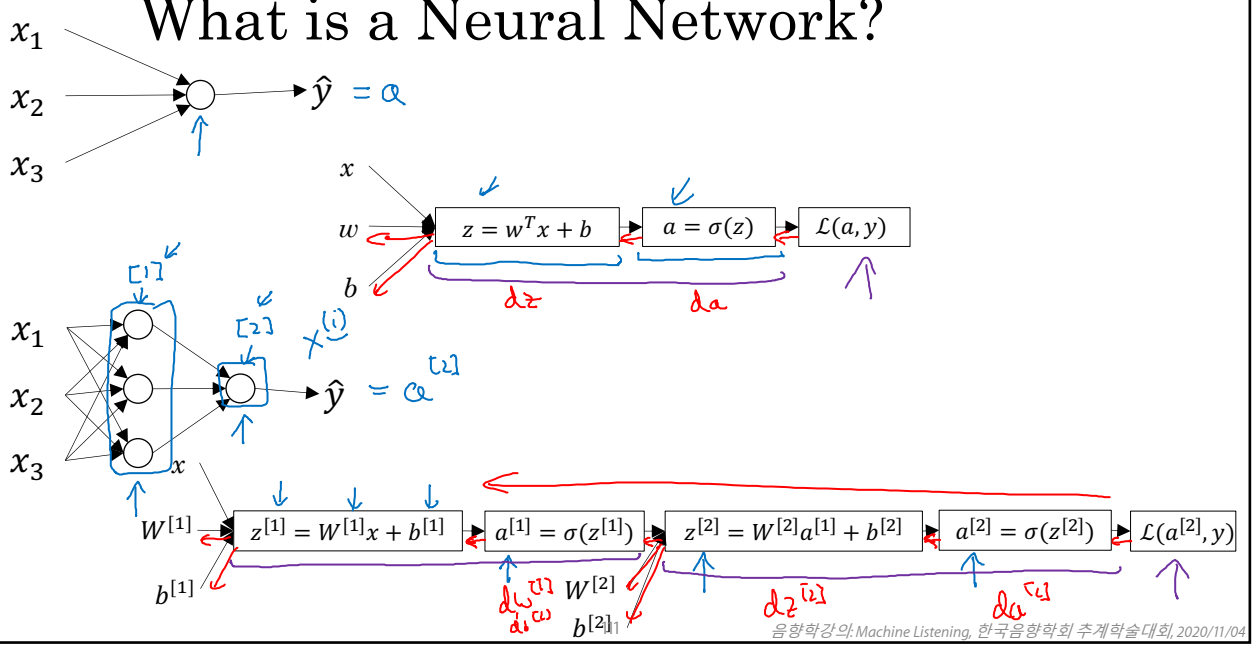
One hidden layer Neural Network

Neural Networks Overview

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

What is a Neural Network?



111

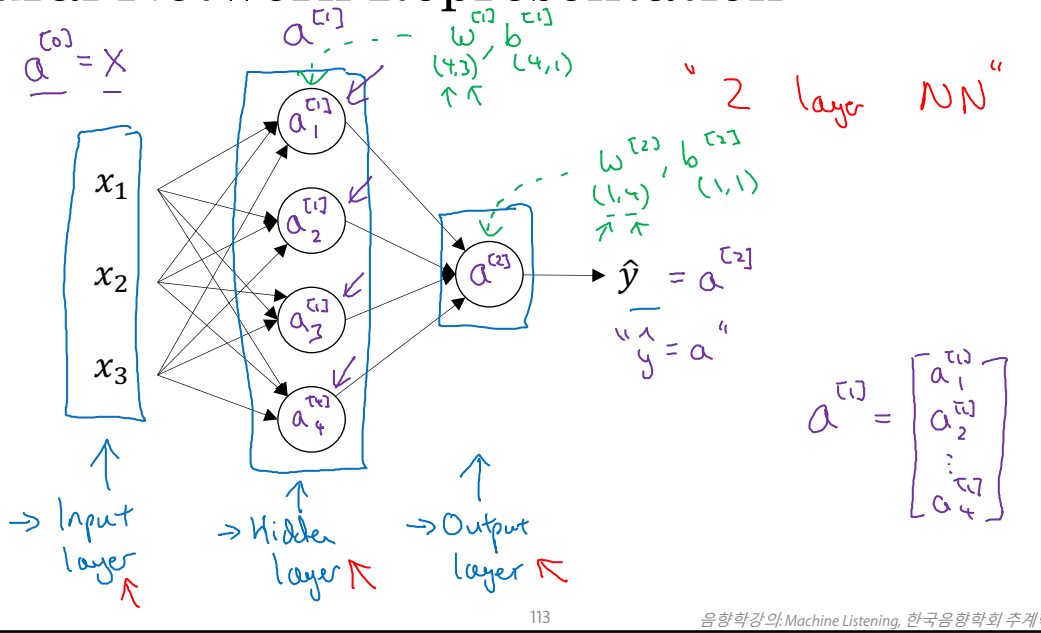
deeplearning.ai

One hidden layer Neural Network

Neural Network Representation

112

Neural Network Representation

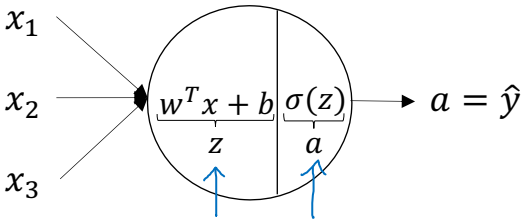


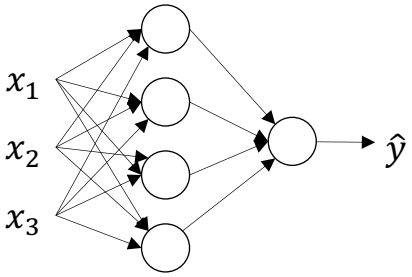
deeplearning.ai

One hidden layer Neural Network

Computing a Neural Network's Output

Neural Network Representation

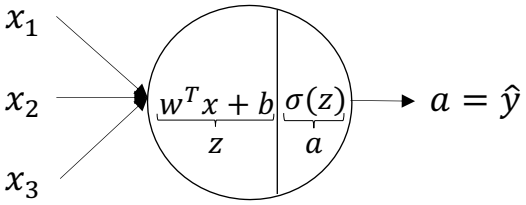


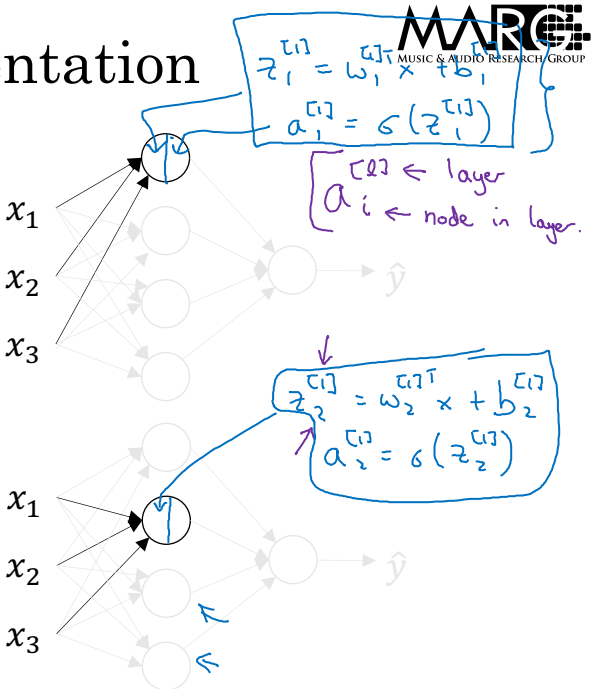


$$z = w^T x + b$$

$$a = \sigma(z)$$

Neural Network Representation

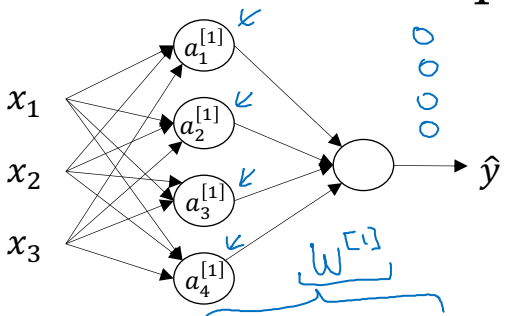




$$z = w^T x + b$$

$$a = \sigma(z)$$

Neural Network Representation

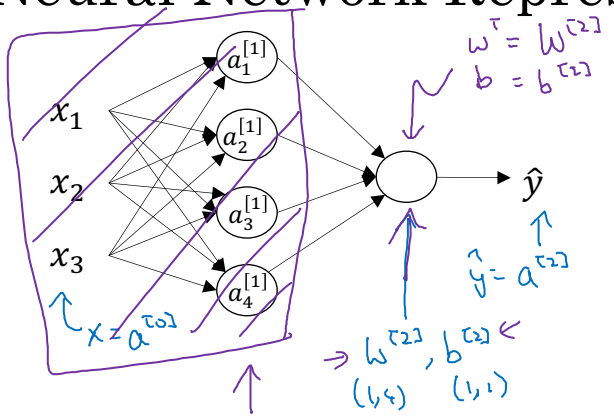


$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$$
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$$
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$
$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

$$z^{[1]} = W^{[1]} x + b^{[1]}, \quad a^{[1]} = \sigma(z^{[1]})$$

117

Neural Network Representation learning



Given input x:

$$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$$
$$a^{[1]} = \sigma(z^{[1]})$$
$$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$$
$$a^{[2]} = \sigma(z^{[2]})$$

118

deeplearning.ai

One hidden layer Neural Network

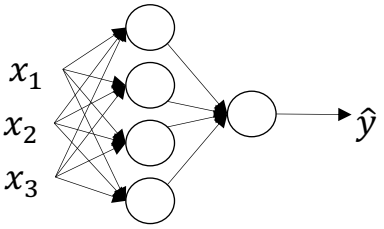
Vectorizing across multiple examples

119

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Vectorizing across multiple examples



$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = \sigma(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = \sigma(z^{[2]})$$

$x \rightarrow a^{[2]} = \hat{y}$

$x^{(1)} \rightarrow a^{[2](1)} = \hat{y}^{(1)}$

$x^{(2)} \rightarrow a^{[2](2)} = \hat{y}^{(2)}$

$x^{(m)} \rightarrow a^{[2](m)} = \hat{y}^{(m)}$

$a^{[2](i)}$ ← example i, ← layer 2

for $i = 1$ to m,

$z^{[1](i)} = W^{[1]} x^{(i)} + b^{[1]}$

$a^{[1](i)} = \sigma(z^{[1](i)})$

$z^{[2](i)} = W^{[2]} a^{[1](i)} + b^{[2]}$

$a^{[2](i)} = \sigma(z^{[2](i)})$

Vectorizing across multiple examples

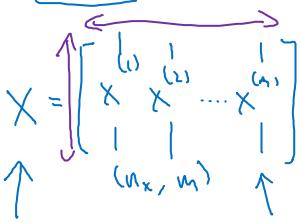
```
for i = 1 to m:
```

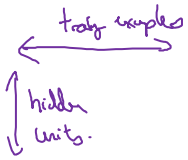
$$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$$

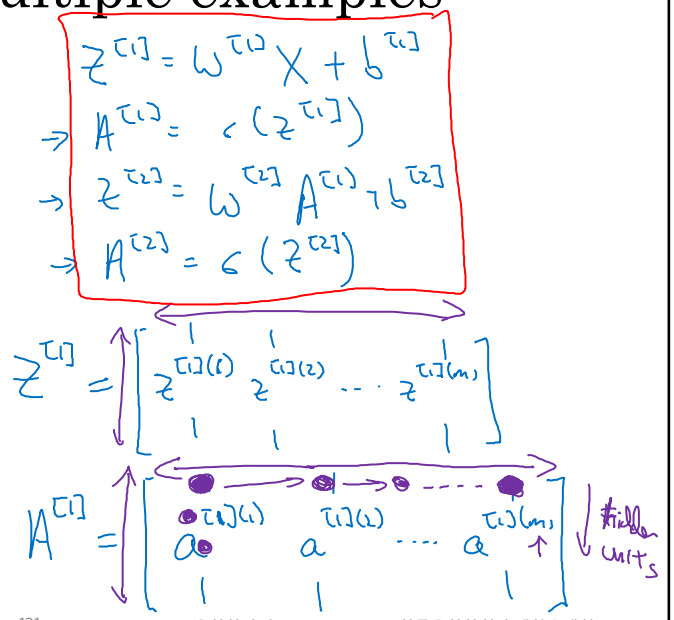
$$a^{[1](i)} = \sigma(z^{[1](i)})$$

$$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$$

$$a^{[2](i)} = \sigma(z^{[2](i)})$$





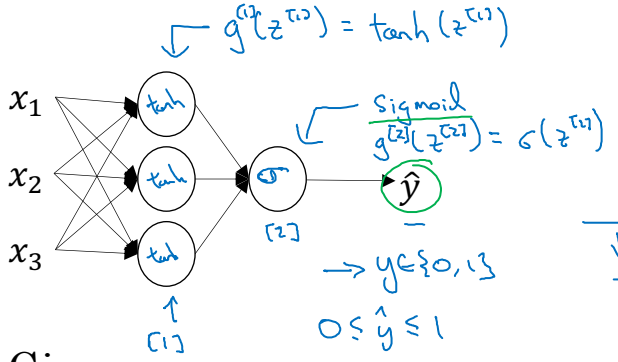


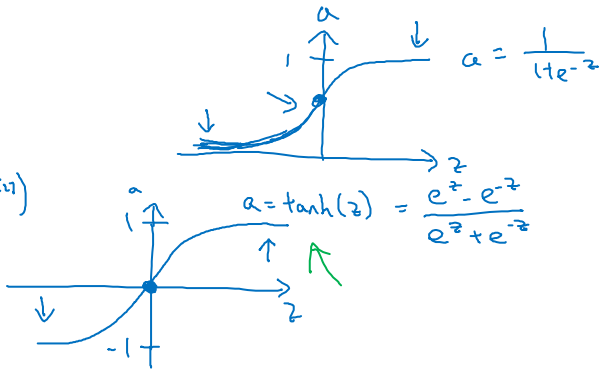
deeplearning.ai

One hidden layer Neural Network

Activation functions

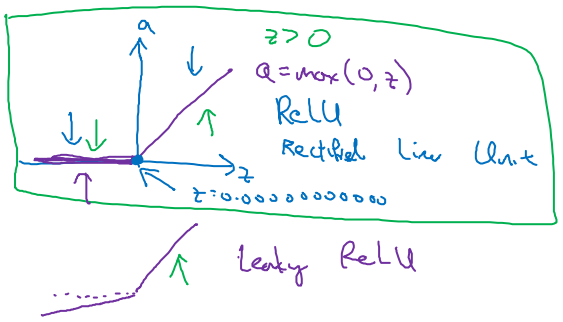
Activation functions





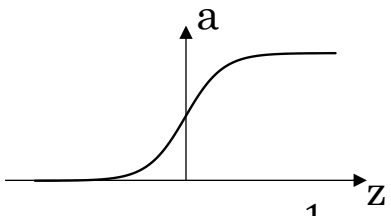
Given x:

$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$\rightarrow a^{[1]} = \sigma(z^{[1]}) \quad g^{[1]}(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$\rightarrow a^{[2]} = \sigma(z^{[2]}) \quad g^{[2]}(z^{[2]})$$

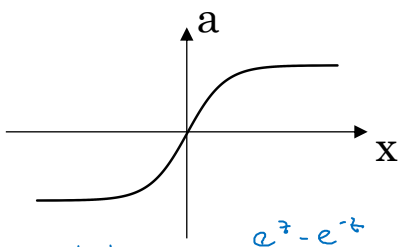


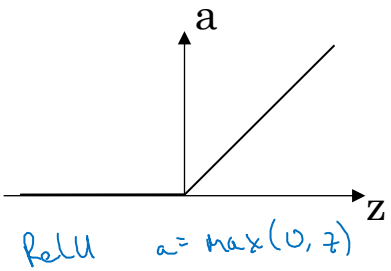
123

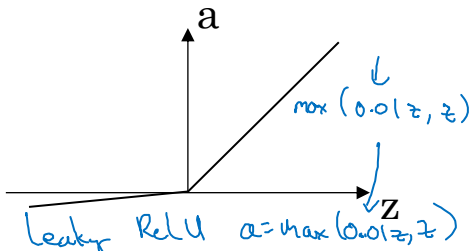
Pros and cons of activation functions



sigmoid: $a = \frac{1}{1+e^{-z}}$







deeplearning.ai

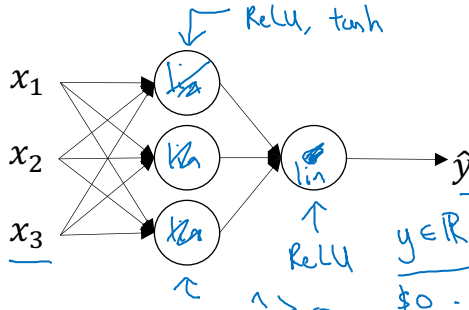
One hidden layer Neural Network

Why do you need non-linear activation functions?

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Activation function



$y \in \mathbb{R}$

\$0 ..... \$1,000,000s

$\hat{y} \geq 0$.

Given x:

$$z^{[1]} = W^{[1]}x + b^{[1]}$$

$$a^{[1]} = g^{[1]}(z^{[1]}) \quad z^{[1]}$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = g^{[2]}(z^{[2]}) \quad z^{[2]}$$

$g(z) = z$ "linear activation function"

$$a^{[1]} = z^{[1]} = W^{[1]}x + b^{[1]}$$

$$a^{[2]} = z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = W^{[2]}\left(W^{[1]}x + b^{[1]}\right) + b^{[2]}$$

$$= \underbrace{\left(W^{[2]}W^{[1]}\right)}_{W'}x + \underbrace{\left(W^{[2]}b^{[1]} + b^{[2]}\right)}_{b'}$$

$$= W'x + b'$$

$g(z) = z$

deeplearning.ai

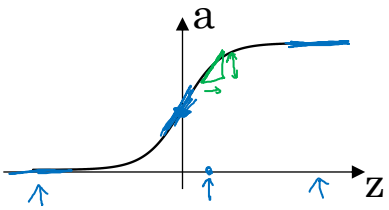
One hidden layer Neural Network

Derivatives of activation functions

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Sigmoid activation function



$$g(z) = \frac{1}{1+e^{-z}}$$

$a = g(z) = \frac{1}{1+e^{-z}}$

$g'(z) = \frac{d}{dz} g(z)$ = slope of $g(z)$ at z

$= \frac{1}{1+e^{-z}} \left(1 - \frac{1}{1+e^{-z}}\right)$

$= g(z)\left(1 - g(z)\right)$ ←

$= a(1-a)$

$g'(z) = a(1-a)$

$z = 10$. $g(z) \approx 1$

$\frac{d}{dz} g(z) \approx 1\,(1-1) \approx 0$

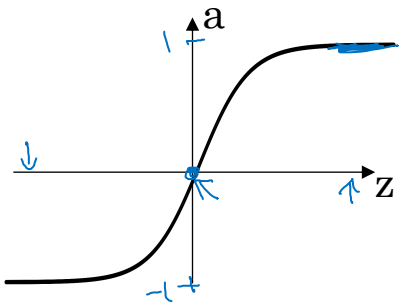
$z = -10$ $g(z) \approx 0$

$\frac{d}{dz} g(z) \approx 0 \cdot (1-0) \approx 0$

$z = 0$ $g(z) = \frac{1}{2}$

$\frac{d}{dz} g(z) = \frac{1}{2}\left(1 - \frac{1}{2}\right) = \frac{1}{4}$

Tanh activation function



$$g(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = \frac{d}{dz} g(z) = \text{slope of } g(z) \text{ at } z = 1 - (\tanh(z))^2$$

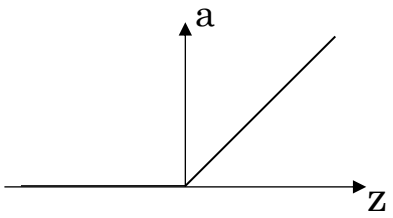
$a = g(z)$, $g'(z) = 1 - a^2$

$z = 10$ $\tanh(z) \approx 1$, $g'(z) \approx 0$

$z = -10$ $\tanh(z) \approx -1$, $g'(z) \approx 0$

$z = 0$ $\tanh(z) = 0$, $g'(z) = 1$

ReLU and Leaky ReLU



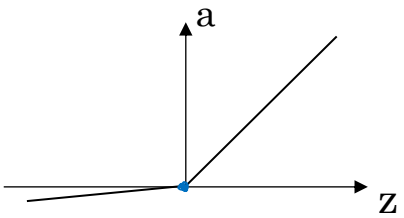
ReLU

$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

~~undef if $z = 0$~~

$z = 0.0000\ldots0$



Leaky ReLU

$$g(z) = \max(0.01z, z)$$

$$g'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

deeplearning.ai

One hidden layer Neural Network

Gradient descent for neural networks

131

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Gradient descent for neural networks

Parameters: $W^{[1]}$ $(n^{[1]}, n^{[0]})$, $b^{[1]}$ $(n^{[1]}, 1)$, $W^{[2]}$ $(n^{[2]}, n^{[1]})$, $b^{[2]}$ $(n^{[2]}, 1)$

$n_x = n^{[0]}$, $n^{[1]}$, $n^{[2]} = 1$

Cost function: $J(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}, y)$ ($\hat{y} = a^{[2]}$)

Gradient descent:

→ Repeat {

→ Compute predicts $(\hat{y}^{(i)}, i = 1, \dots, m)$

$dW^{[1]} = \frac{\partial J}{\partial W^{[1]}}$, $db^{[1]} = \frac{\partial J}{\partial b^{[1]}}$, ...

$W^{[1]} := W^{[1]} - \alpha\, dW^{[1]}$

$b^{[1]} := b^{[1]} - \alpha\, db^{[1]}$

$W^{[2]} := \dots$ $b^{[2]} := \dots$

}

132

Formulas for computing derivatives

Forward propagation:

$z^{[1]} = W^{[1]} X + b^{[1]}$

$A^{[1]} = g^{[1]}(z^{[1]})$

$z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$

$A^{[2]} = g^{[2]}(z^{[2]}) = \sigma(z^{[2]})$

Back propagation:

$dz^{[2]} = A^{[2]} - Y$

$Y = [y^{(1)}\ y^{(2)} \cdots y^{(m)}]$

$dW^{[2]} = \frac{1}{m} dz^{[2]} A^{[1]T}$

$(n^{[2]}, m)$ ← $(n^{[2]}, 1)$ ←

$db^{[2]} = \frac{1}{m}$ np.sum$(dz^{[2]}$, axis=1, keepdims=True$)$

$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(z^{[1]})$

$(n^{[1]}, m)$ $(n^{[1]}, m)$

↑ element-wise product

$dW^{[1]} = \frac{1}{m} dz^{[1]} X^T$

$db^{[1]} = \frac{1}{m}$ np.sum$(dz^{[1]}$, axis=1, keepdims=True$)$

$(n^{[1]}, 1)$ $(n^{[1]},)$ reshape ↑

deeplearning.ai

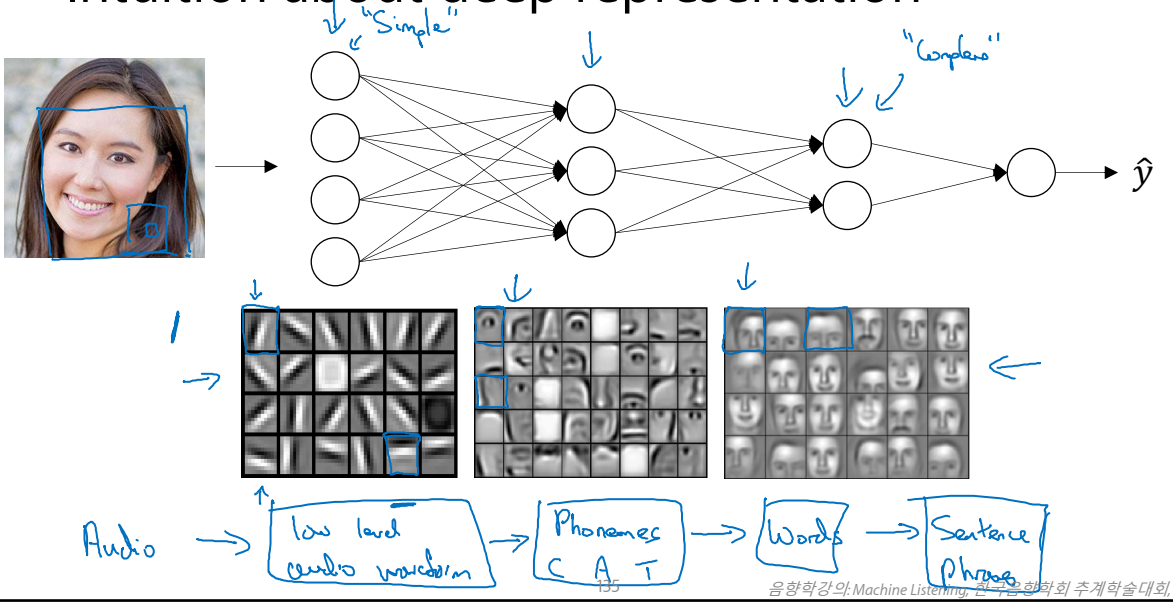
Deep Neural Networks

Why deep representations?

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Intuition about deep representation

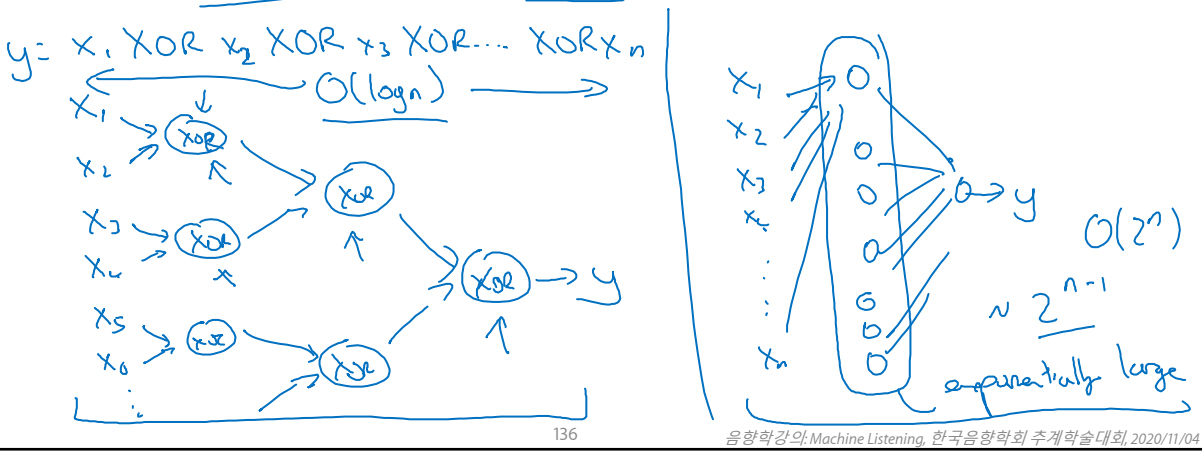


SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Circuit theory and deep learning

Informally: There are functions you can compute with a "small" L-layer deep neural network that shallower networks require exponentially more hidden units to compute.



deeplearning.ai

Deep Neural Networks

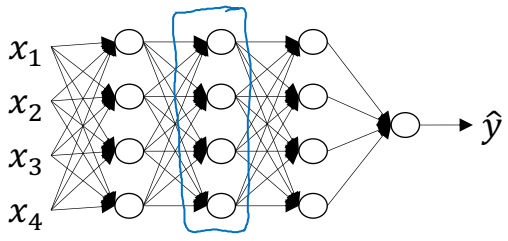
Building blocks of deep neural networks

137

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Forward and backward functions



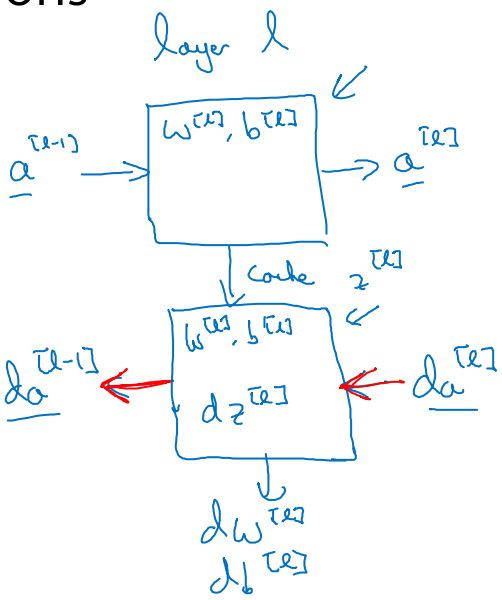
Layer l: $W^{[l]}, b^{[l]}$

→ Forward: Input $a^{[l-1]}$, output $a^{[l]}$

$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$ cache $z^{[l]}$

$a^{[l]} = g^{[l]}(z^{[l]})$

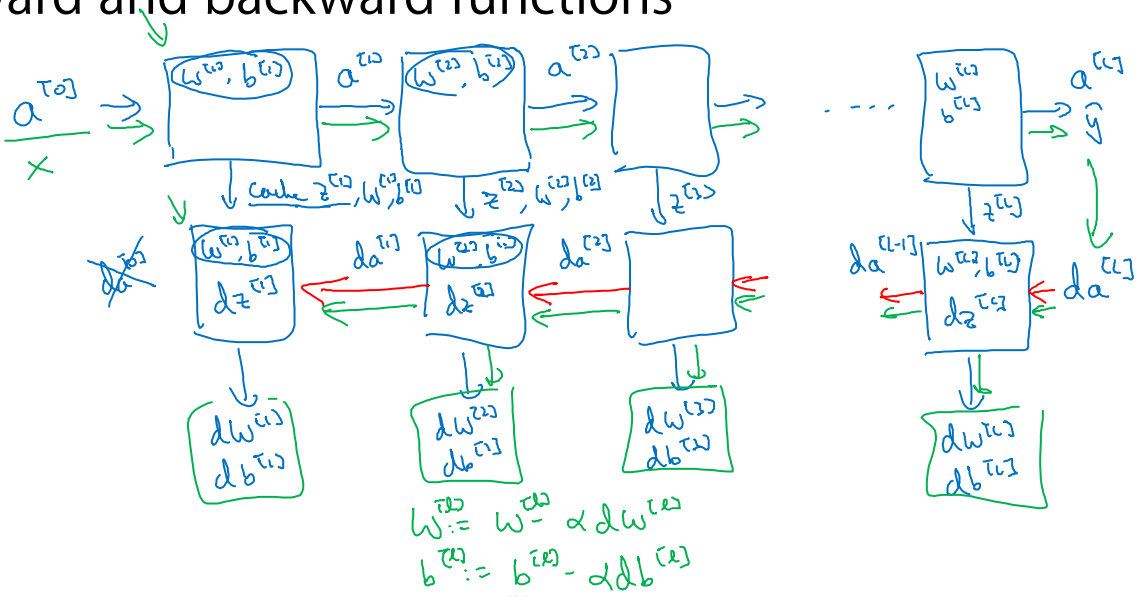
→ Backward: Input $da^{[l]}$, cache($z^{[l]}$), output $da^{[l-1]}$, $dW^{[l]}$, $db^{[l]}$



SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Forward and backward functions



deeplearning.ai

Deep Neural Networks

Parameters vs Hyperparameters

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

What are hyperparameters?

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}$ ...

Hyperparameters: learning rate α

#iterations

hidden layers L

hidden units $n^{[1]}, n^{[2]}, \ldots$

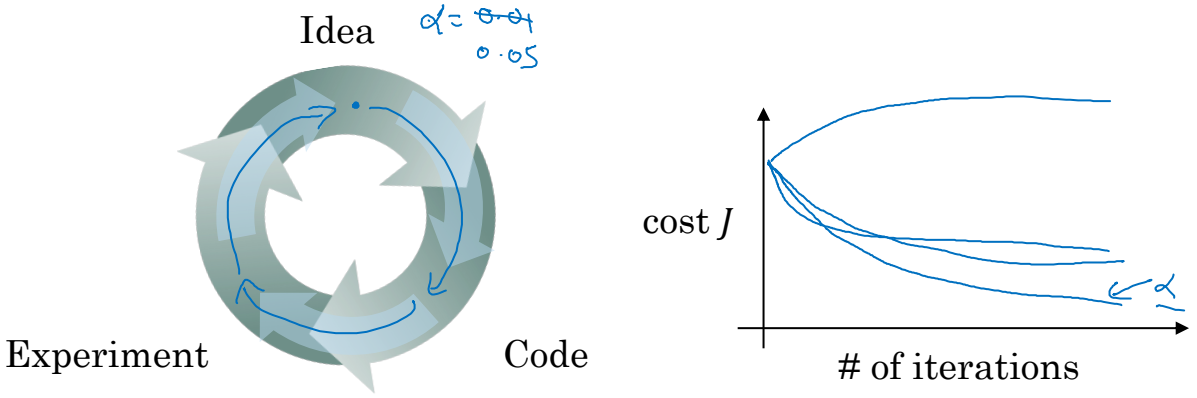
Choice of activation function

Later: Momentum, mini-batch size, regularizations, ...

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Applied deep learning is a very empirical process



$\alpha = 0.01$ 0.05

Vision, Speech, NLP, Ads, Search, Recommendation.

deeplearning.ai

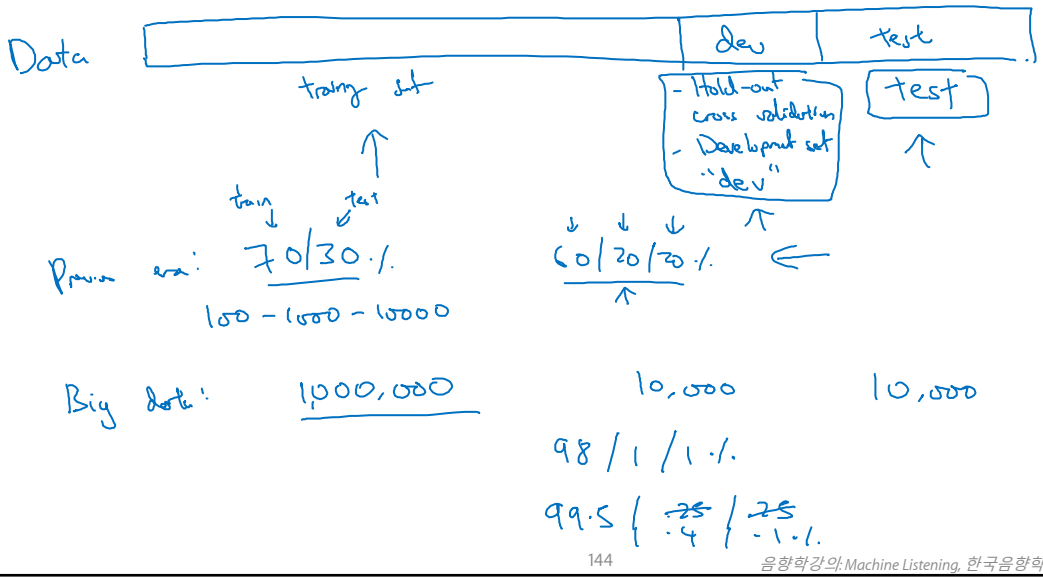
Setting up your ML application

Train/dev/test sets

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Train/dev/test sets

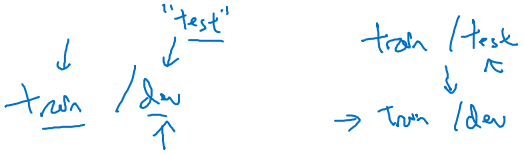


Mismatched train/test distribution

Training set:
Cat pictures from webpages

Dev/test sets:
Cat pictures from users using your app

→ Make sure dev and test come from same distribution.



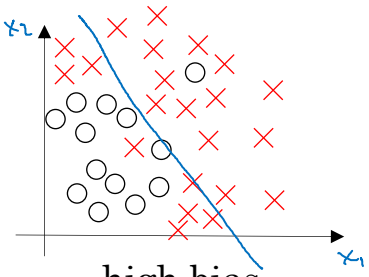
Not having a test set might be okay. (Only dev set.)

deeplearning.ai

Setting up your ML application

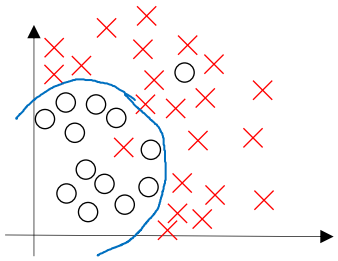
Bias/Variance

Bias and Variance

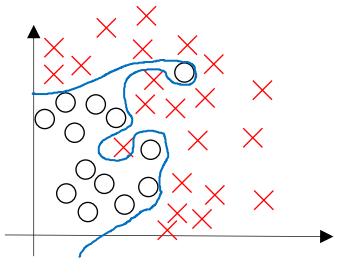


high bias

Underfitting



→ "just right"



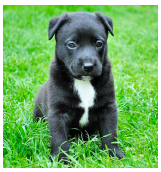
high variance

Overfitting

Bias and Variance
Cat classification

y=1

y=0



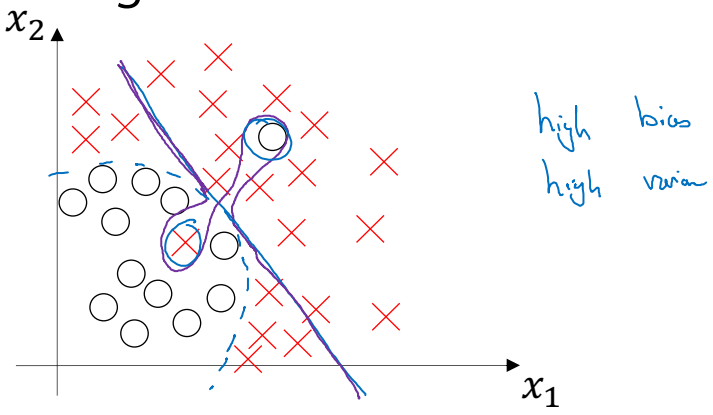
Train set error:	1%	15%	15%	0.5%
Dev set error:	11%	16%	30%	1%
	high variance	high bias	high bias & high variance	low bias low variance

Human: ~0%

Optimal (Bayes) error: ~~0%~~ 15%

Blurry images

High bias and high variance



deeplearning.ai

Regularizing your neural network

Regularization

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Logistic regression

$\min_{w,b} J(w,b)$

$w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

λ = regularization parameter

lambda lambd

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\|w\|_2^2$$

$+\frac{\lambda}{2m} b^2$ omit

L_2 regularization $\|w\|_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w$

L_1 regularization $\frac{\lambda}{2m}\sum_{j=1}^{n_x}|w_j| = \frac{\lambda}{2m}\|w\|_1$

w will be sparse

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Neural network

$$J(w^{[1]}, b^{[1]}, \dots, w^{[L]}, b^{[L]}) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L}\|w^{[l]}\|_F^2$$

$$\|w^{[l]}\|_F^2 = \sum_{i=1}^{n^{[l]}}\sum_{j=1}^{n^{[l-1]}} (w_{ij}^{[l]})^2$$

$w^{[l]}$: $(n^{[l]}, n^{[l-1]})$

"Frobenius norm" $\|\cdot\|_2^2$ $\|\cdot\|_F^2$

$$dw^{[l]} = (\text{from backprop}) + \frac{\lambda}{m} w^{[l]}$$

$$\frac{\partial J}{\partial w^{[l]}} = dw^{[l]}$$

$$w^{[l]} := w^{[l]} - \alpha\, dw^{[l]}$$

"Weight decay"

$$w^{[l]} := w^{[l]} - \alpha\left[(\text{from backprop}) + \frac{\lambda}{m} w^{[l]}\right]$$

$$= w^{[l]} - \frac{\alpha\lambda}{m} w^{[l]} - \alpha\,(\text{from backprop})$$

$$= \left(1 - \frac{\alpha\lambda}{m}\right) w^{[l]} - \alpha\,(\text{from backprop})$$

$\left(1 - \frac{\alpha\lambda}{m}\right) < 1$

음향학강의: Machine Listening, 한국음향학회 추계학술대회, 2020/11/04

deeplearning.ai

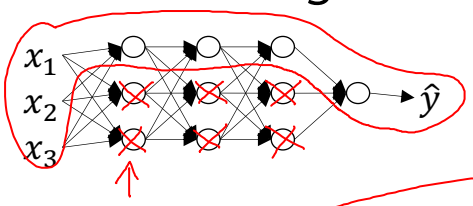
Regularizing your neural network

Why regularization reduces overfitting

SNU Convergence

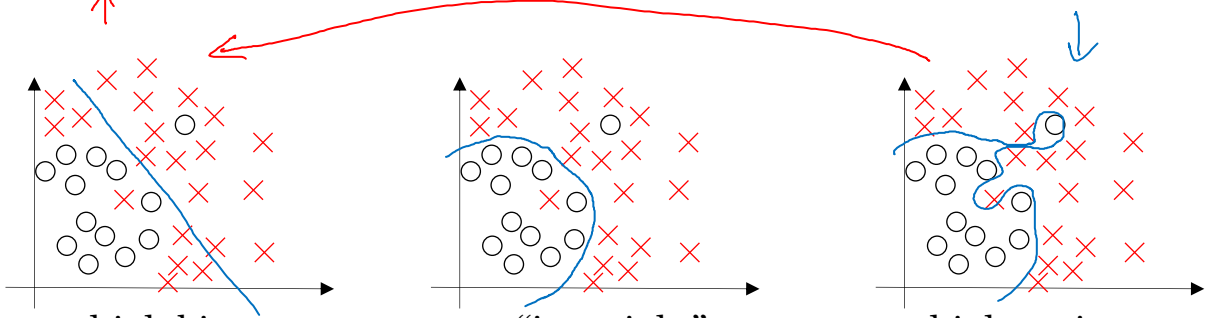
MARG MUSIC & AUDIO RESEARCH GROUP

How does regularization prevent overfitting?



$$J(w^{[l]}, b^{[l]}) = \frac{1}{m}\sum_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} \|w^{[l]}\|_F^2$$

$w^{[l]} \approx 0$

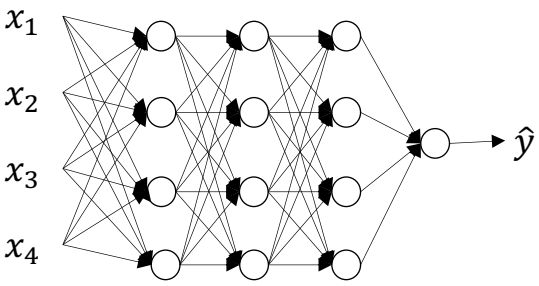


deeplearning.ai

Regularizing your neural network

Dropout regularization

Dropout regularization



0.5 0.5 0.5

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Implementing dropout ("Inverted dropout")

Illustrate with layer $l=3$. keep-prob $= 0.8$ (0.2)

$d3 = np.random.rand(a3.shape[0], a3.shape[1]) < keep\text{-}prob$

$a3 = np.multiply(a3, d3)$ # $a3 *= d3$.

$a3 /= keep\text{-}prob$

50 units. ⇝ 10 units shut off

$z^{[4]} = w^{[4]} \cdot a^{[3]} + b^{[4]}$

$a^{[3]}$ reduced by 20%. Test

$/= 0.8$

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Making predictions at test time

$a^{[0]} = X$

No drop out.

$z^{[1]} = w^{[1]} a^{[0]} + b^{[1]}$

$a^{[1]} = g^{[1]}(z^{[1]})$

$z^{[2]} = w^{[2]} a^{[1]} + b^{[2]}$

$a^{[2]} = \ldots$

$\downarrow$

$\hat{y}$

$/= keep\text{-}prob$

deeplearning.ai

Regularizing your neural network

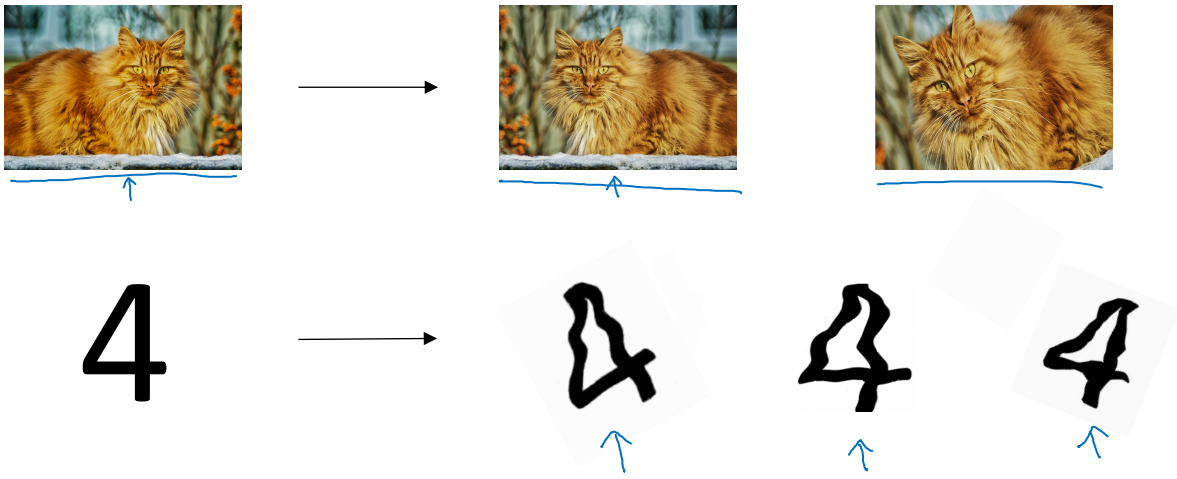
Other regularization methods

159

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Data augmentation

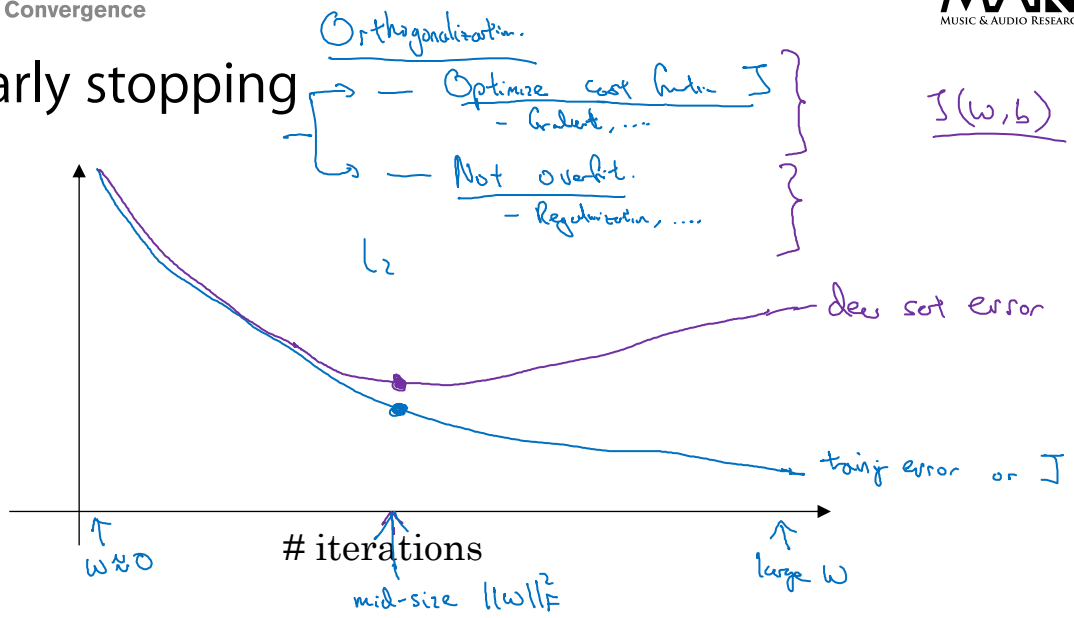


160

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Early stopping



deeplearning.ai

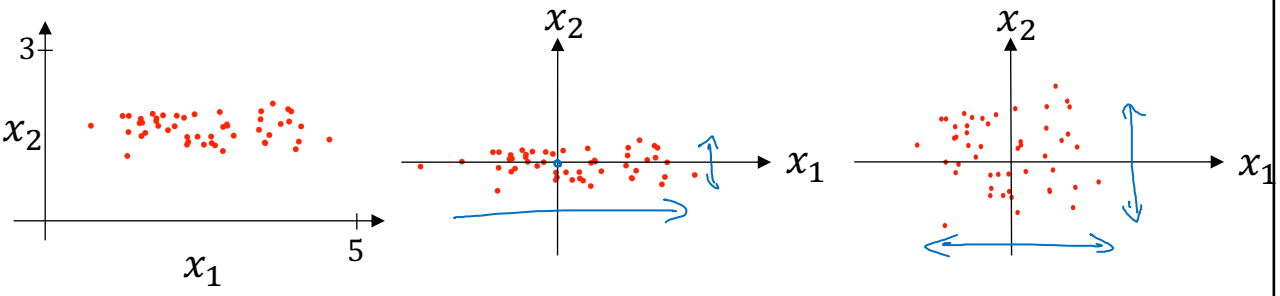
Setting up your optimization problem

Normalizing inputs

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Normalizing training sets



Subtract mean:

$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$

$x := x - \mu$

Normalize variance

$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} **2$ ← element-wise

$x /= \sigma^2$

Use same μ σ^2 to normalize test set.

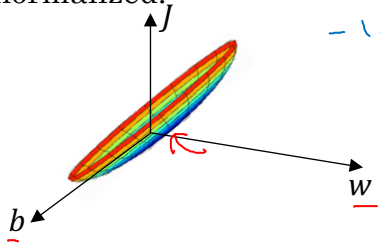
SNU Convergence

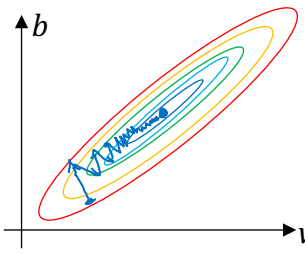
MARG MUSIC & AUDIO RESEARCH GROUP

Why normalize inputs?

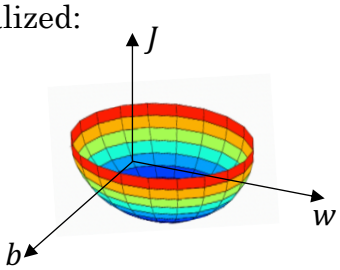
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$$

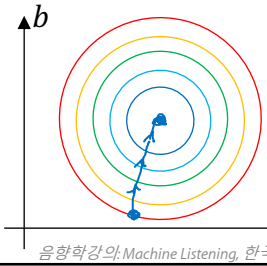
Unnormalized:





Normalized:





deeplearning.ai

Optimization Algorithms

Mini-batch gradient descent

Batch vs. mini-batch gradient descent

Vectorization allows you to efficiently compute on m examples.

$X = [x^{(1)}\ x^{(2)}\ x^{(3)} \ldots x^{(1000)} \mid x^{(1001)} \ldots x^{(2000)} \mid \ldots \mid \ldots x^{(m)}]$ (n_x, m)

$X^{\{1\}}$ $(n_x, 1000)$ $X^{\{2\}}$ $(n_x, 1000)$ $\ldots$ $X^{\{5{,}000\}}$ $(n_x, 1000)$

$Y = [y^{(1)}\ y^{(2)}\ y^{(3)} \ldots y^{(1000)} \mid y^{(1001)} \ldots y^{(2000)} \mid \ldots \mid \ldots y^{(m)}]$ $(1, m)$

$Y^{\{1\}}$ $(1, 1000)$ $Y^{\{2\}}$ $(1, 1000)$ $\ldots$ $Y^{\{5{,}000\}}$ $(1, 1000)$

What if $m = 5{,}000{,}000$?

5,000 mini-batches of 1,000 each

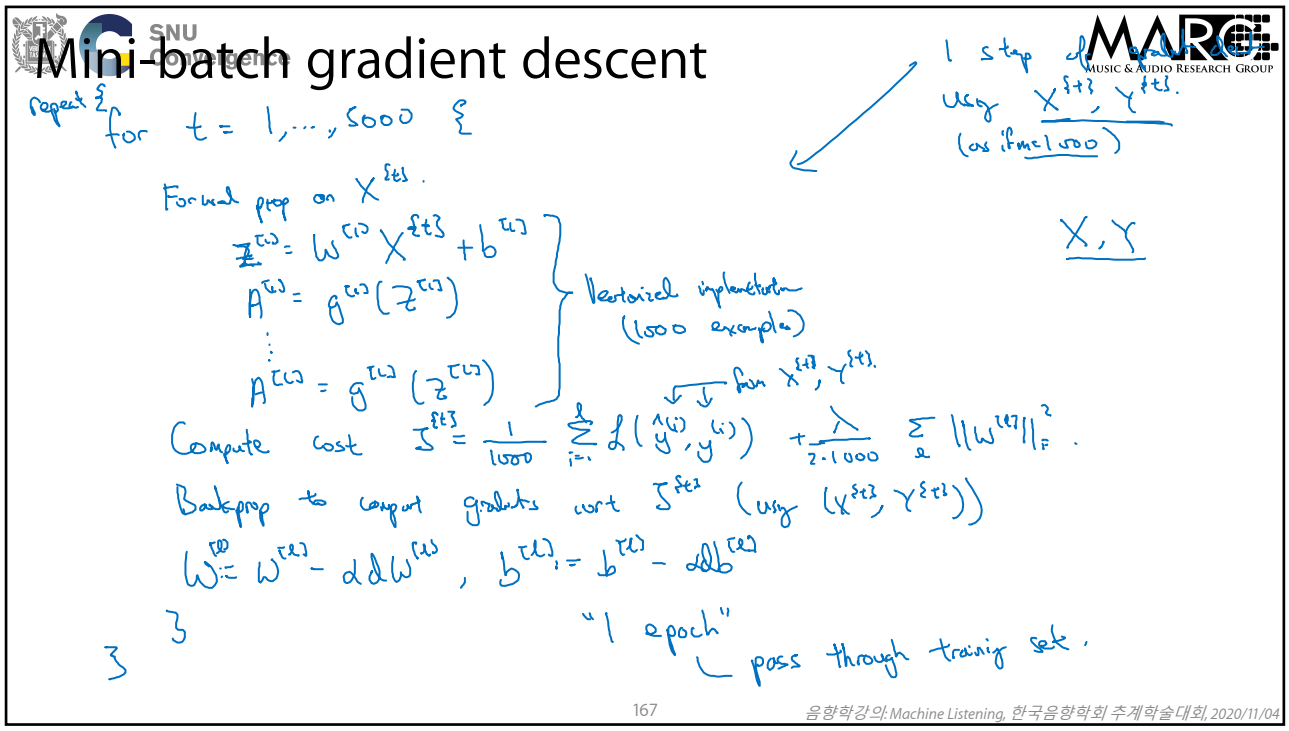
Mini-batch t: $X^{\{t\}}, Y^{\{t\}}$

$x^{(i)}$

$z^{[l]}$

$X^{\{t\}}, Y^{\{t\}}$

Mini-batch gradient descent

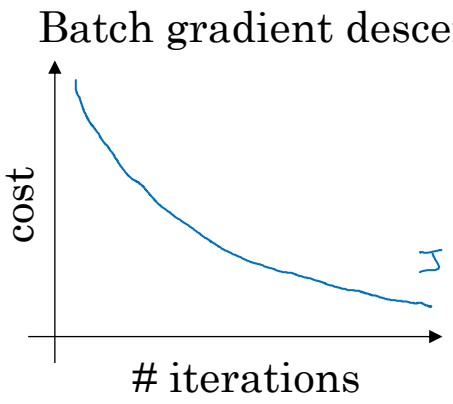


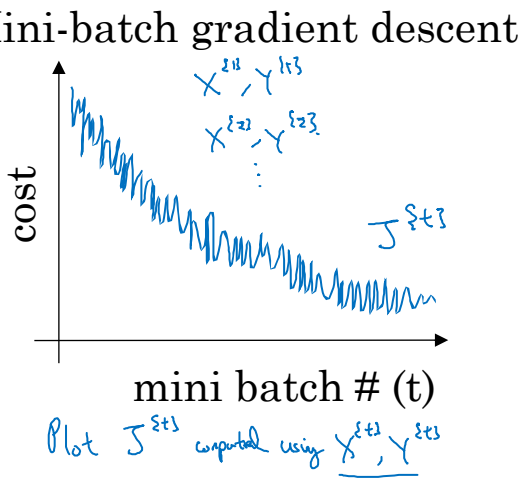
deeplearning.ai

Optimization Algorithms

Understanding mini-batch gradient descent

Training with mini batch gradient descent



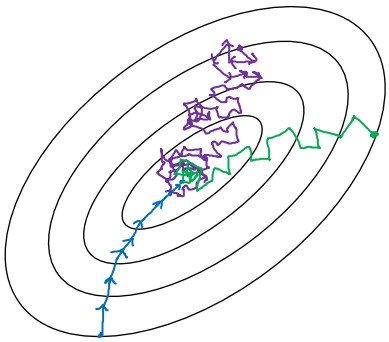


Choosing your mini-batch size

→ If mini-batch size = m : Batch gradient descent.

→ If mini-batch size = 1 : Stochastic gradient descent. Every example is it own mini-batch.

In practice: Somewhere in-between 1 and m



Stochastic gradient descent
↓
Lose speedup from vectorization

In-between (mini-batch size not too big/small)
↓
Fastest learning.
- Vectorization.
- Make progress without processing entire training set.

Batch gradient descent (mini-batch size = m)
↓
Too long per iteration

Choosing your mini-batch size

If small train set: Use batch gradient descent.
($m \leq 2000$)

Typical mini-batch sizes:

→ 64, 128, 256, 512 (1024)
2^6, 2^7, 2^8, 2^9 (2^{10})

Make sure mini-batch fit in CPU/GPU memory.
$X^{\{t\}}, Y^{\{t\}}$

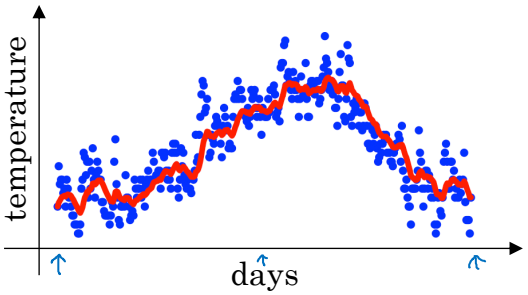
deeplearning.ai

Optimization Algorithms

Exponentially weighted averages

Temperature in London

$\theta_1 = 40°\text{F}$ 4°C ←
$\theta_2 = 49°\text{F}$ 9°C
$\theta_3 = 45°\text{F}$ ⋮
⋮
$\theta_{180} = 60°\text{F}$ 15°C
$\theta_{181} = 56°\text{F}$ ⋮
⋮



$V_0 = 0$
$V_1 = 0.9\, V_0 + 0.1\, \theta_1$
$V_2 = 0.9\, V_1 + 0.1\, \theta_2$
$V_3 = 0.9\, V_2 + 0.1\, \theta_3$
⋮
$V_t = 0.9\, V_{t-1} + 0.1\, \theta_t$

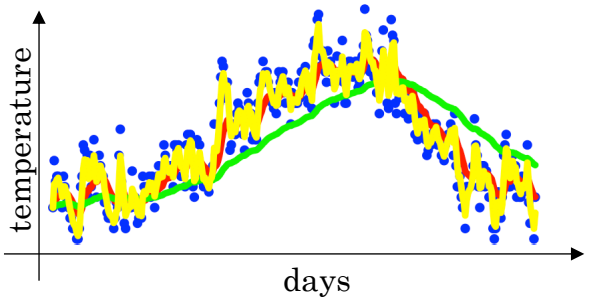
Exponentially weighted averages

Moving

$V_t = \beta\, V_{t-1} + (1-\beta)\, \theta_t$ ←
$\beta = 0.9$: ≈ 10 days' temperature.
$\beta = 0.98$: ≈ 50 days
$\beta = 0.5$: ≈ 2 days

V_t is approximately average over
→ ≈ $\frac{1}{1-\beta}$ days' temperature.

$\frac{1}{1-0.98} = 50$



deeplearning.ai

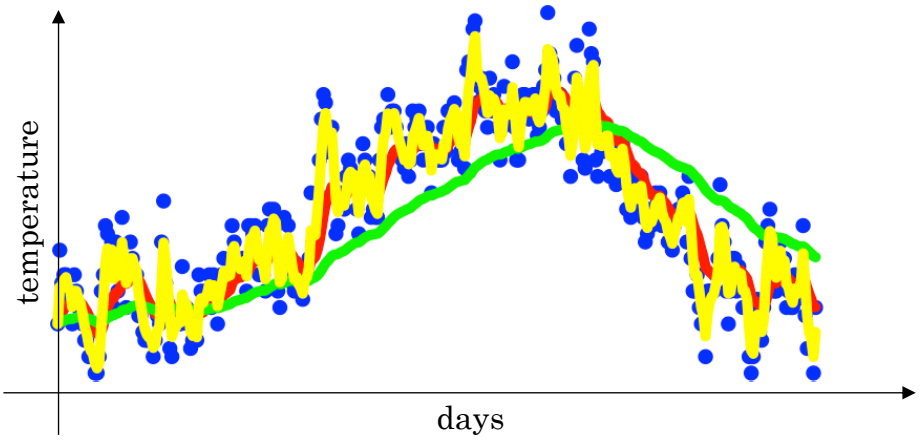
Optimization Algorithms

Understanding exponentially weighted averages

Exponentially weighted averages

$$v_t = \beta v_{t-1} + (1-\beta)\theta_t$$

$\beta = 0.9 \quad 0.98 \quad 0.5$



Exponentially weighted averages

$$v_t = \beta v_{t-1} + (1-\beta)\theta_t$$

$$v_{100} = 0.9v_{99} + 0.1\theta_{100}$$
$$v_{99} = 0.9v_{98} + 0.1\theta_{99}$$
$$v_{98} = 0.9v_{97} + 0.1\theta_{98}$$
$$\cdots$$

$$V_{100} = 0.1\theta_{100} + 0.9(0.1\theta_{99} + 0.9(\cdots))$$
$$= 0.1\theta_{100} + 0.1\times 0.9\,\theta_{99} + 0.1(0.9)^2\theta_{98} + 0.1(0.9)^3\theta_{97} + 0.1(0.9)^4\theta_{96} + \cdots$$

$0.9^{10} \approx 0.35 \approx \frac{1}{e}$

$(1-\varepsilon)^{1/\varepsilon} = \frac{1}{e}$

$\varepsilon = 0.02 \rightarrow 0.98^{50} \approx \frac{1}{e}$

$\approx \frac{1}{1-\beta}$

$\varepsilon = 1-\beta$

Implementing exponentially weighted averages

$$v_0 = 0$$
$$v_1 = \beta v_0 + (1-\beta)\,\theta_1$$
$$v_2 = \beta v_1 + (1-\beta)\,\theta_2$$
$$v_3 = \beta v_2 + (1-\beta)\,\theta_3$$
$$\cdots$$

$V_\theta := 0$

$V_\theta := \beta V + (1-\beta)\theta_1$

$V_\theta := \beta V + (1-\beta)\theta_2$

$\vdots$

$V_\theta = 0$

Repeat {

Get next θ_t

$V_\theta := \beta V_\theta + (1-\beta)\theta_t$

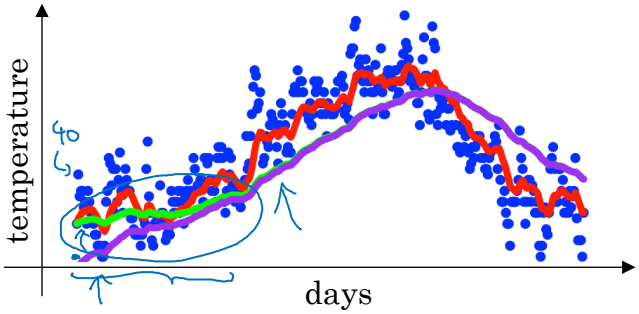
}

deeplearning.ai

Optimization Algorithms

Bias correction in exponentially weighted average

Bias correction



$\beta = 0.98$

$v_t = \beta v_{t-1} + (1-\beta)\theta_t$

$v_0 = 0$

$v_1 = 0.98 v_0 + 0.02\theta_1$

$v_2 = 0.98 v_1 + 0.02\theta_2$

$= 0.98 \times 0.02 \times \theta_1 + 0.02\theta_2$

$= 0.0196\theta_1 + 0.02\theta_2$

$\frac{v_t}{1-\beta^t}$

$t=2: \quad 1-\beta^t = 1-(0.98)^2 = 0.0396$

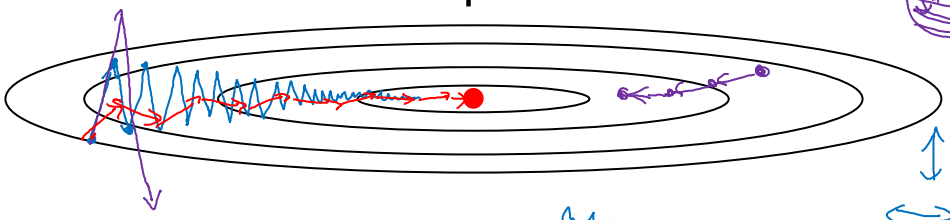
$\frac{v_2}{0.0396} = \frac{0.0196\theta_1 + 0.02\theta_2}{0.0396}$

deeplearning.ai

Optimization Algorithms

Gradient descent with momentum

Gradient descent example



slower learning

faster learning.

Momentum:

On iteration t:

Compute dW, db on current mini-batch.

$v_{dW} = \beta v_{dW} + (1-\beta) dW$

$v_{db} = \beta v_{db} + (1-\beta) db$

"$v_\theta = \beta v_{\theta-1} + (1-\beta)\theta_t$"

friction, velocity, acceleration

$W := W - \alpha v_{dW}$, $b := b - \alpha v_{db}$

Implementation details

$V_{dW} = 0$, $V_{db} = 0$

On iteration t:

Compute dW, db on the current mini-batch

$v_{dW} = \beta v_{dW} + (1-\beta) dW$

$v_{db} = \beta v_{db} + (1-\beta) db$

$W = W - \alpha v_{dW}, \quad b = b - \alpha v_{db}$

$V_{dW} = \beta V_{dW} + dW$

$\frac{V_{dW}}{1-\beta^t}$

Hyperparameters: α, β

$\beta = 0.9$

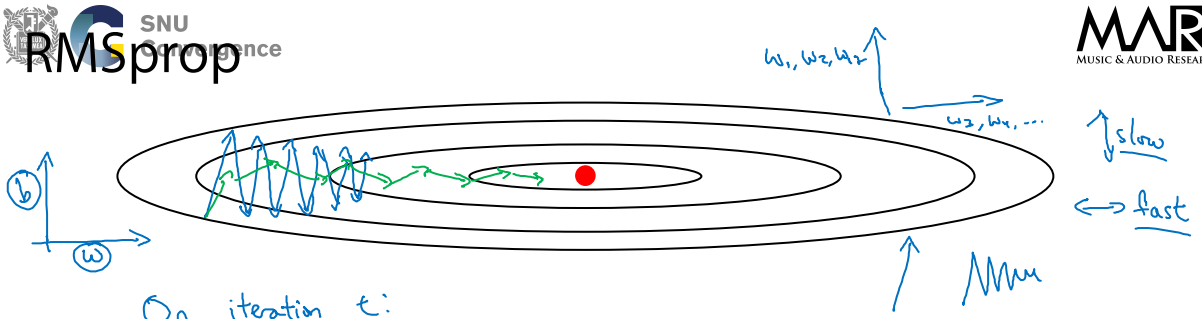
average over last ≈10 gradients

deeplearning.ai

Optimization Algorithms

RMSprop

RMSprop



On iteration t:

Compute dW, db on current mini-batch

element-wise

$S_{dW} = \beta_2 S_{dW} + (1-\beta_2) dW^2$ ← small

$S_{db} = \beta_2 S_{db} + (1-\beta_2) db^2$ ← large

$W := W - \alpha \frac{dW}{\sqrt{S_{dW}+\varepsilon}}$ ← $\qquad b := b - \alpha \frac{db}{\sqrt{S_{db}+\varepsilon}}$ ←

$\varepsilon = 10^{-8}$

deeplearning.ai

Optimization Algorithms

Adam optimization algorithm

Adam optimization algorithm

$V_{dw} = 0, S_{dw} = 0, V_{db} = 0, S_{db} = 0$

On iteration t:

Compute dw, db using current mini-batch

$V_{dw} = \beta_1 V_{dw} + (1-\beta_1) dw$, $V_{db} = \beta_1 V_{db} + (1-\beta_1) db$ ← "momentum" β_1

$S_{dw} = \beta_2 S_{dw} + (1-\beta_2) dw^2$, $S_{db} = \beta_2 S_{db} + (1-\beta_2) db$ ← "RMSprop" β_2

$V_{dw}^{corrected} = V_{dw} / (1-\beta_1^t)$, $V_{db}^{corrected} = V_{db} / (1-\beta_1^t)$

$S_{dw}^{corrected} = S_{dw} / (1-\beta_2^t)$, $S_{db}^{corrected} = S_{db} / (1-\beta_2^t)$

$$W := W - \alpha \frac{V_{dw}^{corrected}}{\sqrt{S_{dw}^{corrected}} + \varepsilon} \qquad b := b - \alpha \frac{V_{db}^{corrected}}{\sqrt{S_{db}^{corrected}} + \varepsilon}$$

Hyperparameters choice:

→ α : needs to be tune

→ β_1 : 0.9 → (dw)

→ β_2 : 0.999 → (dw^2)

→ ε : 10^{-8}

Adam : Adaptive moment estimation

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Applications of Machine Listening

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Machine Listening Applications

- Speech recognition (speech-to-text or STT)
- Speaker identification/verification
- Emotion recognition
- Speech enhancement
- Source separation
- Automatic music transcription
- Genre/Artist identification
- Music identification
- Lyric-audio alignment
- Speech synthesis (text-to-speech or TTS)
- Singing voice synthesis
- And many more…

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Speech Recognition or STT

Siri

Google ASSISTANT

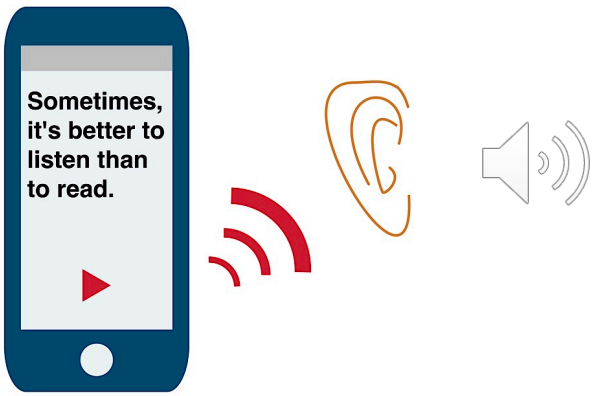
Bixby

Smartly.ai

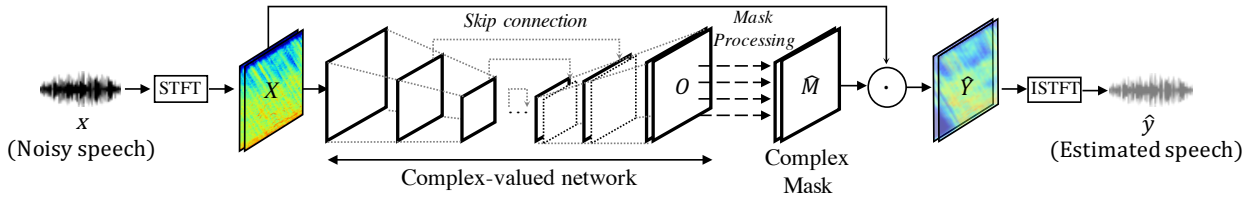
SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Speech Synthesis or TTS



Speech Enhancement



Speech Enhancement

영화

예능

드라마

스포츠

뉴스

SNU Convergence

MUSIC & AUDIO RESEARCH GROUP

Speech Enhancement

SNU Convergence

MUSIC & AUDIO RESEARCH GROUP

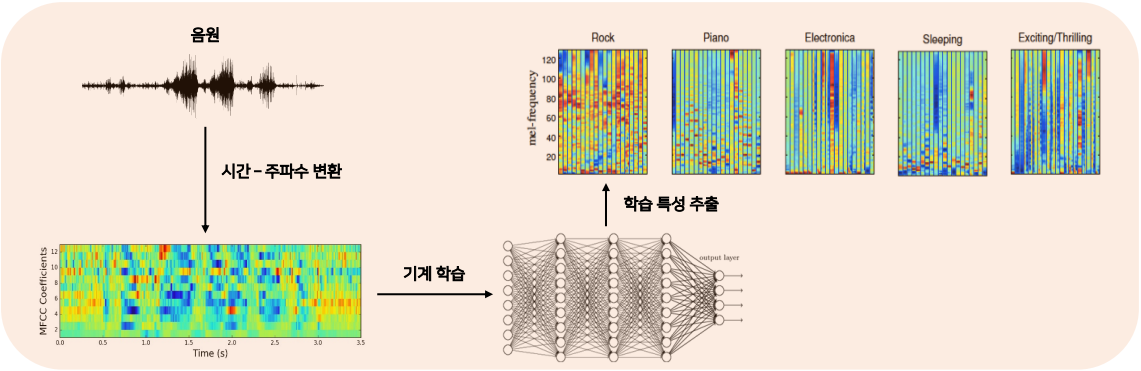
Music Identification

Music Search & Discovery

Music Source Separation

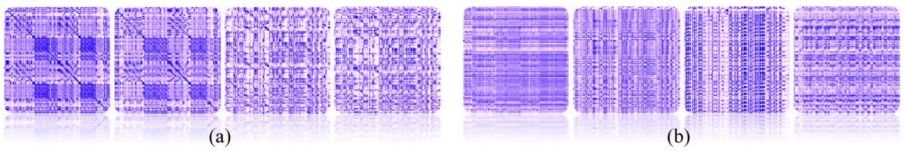
• 오리지널 음악 신호	• 오리지널 음악 신호
• 추출된 가창음성 신호	• 추출된 가창음성 신호

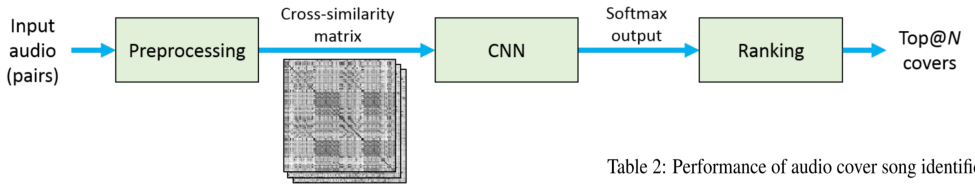
Music Recommendation



Query	Return
늪 By : 김범수	발걸음 By : 에메랄드 캐슬
As Tears Go By By : The Rolling Stones	Now And Forever By : Richard Marx

Audio Coversong Identification





붉은노을 by 이문세	커버곡 1 by 유리상자	커버곡 2 by 빅뱅

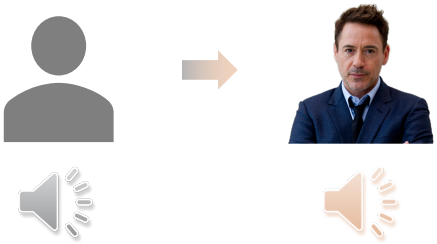
Table 2: Performance of audio cover song identification

Model	MNIT10	MAP	MR1
SimPLe Silva et al. [2016]	6.8	0.66	5.6
SimPLe + Metric Learning (Heo et al. [2017])	7.9	0.81	15.1
CNN (proposed)	8.04	0.84	2.50

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Voice Cloning



Original actor

Robert Downey Jr.

Deep Fake with voice

SNU Convergence

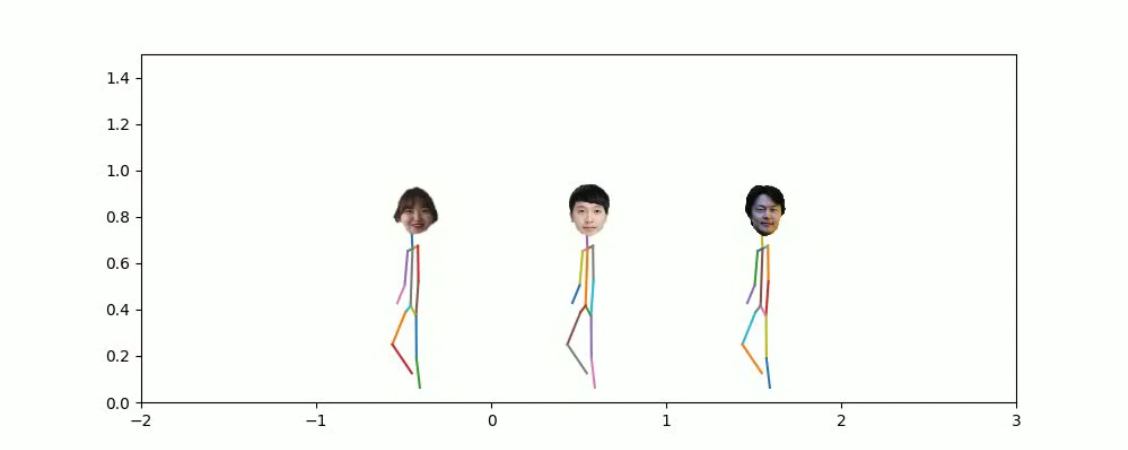
MARG MUSIC & AUDIO RESEARCH GROUP

Singing Voice Synthesis

SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

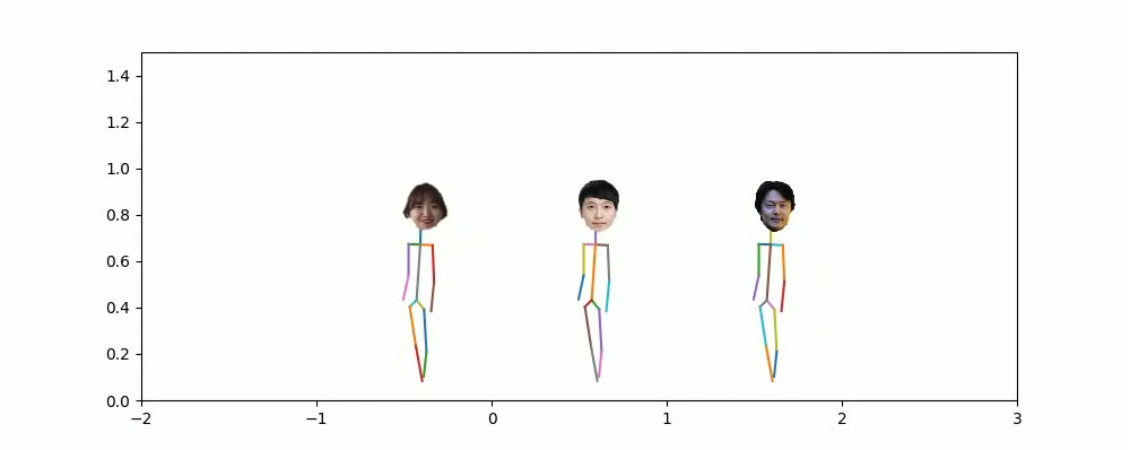
Listen to Dance



SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Listen to Dance



SNU Convergence

MARG MUSIC & AUDIO RESEARCH GROUP

Questions?

서울대학교 지능정보융합학과
음악오디오연구실
이교구
kglee@snu.ac.kr