



Binary Classifiers for Nucleotide Sequencing Data

DeepBind (2015, Nature biotechnology)

Presenter

Gwangho Lee

Computer Science Engineering, Pusan National University

Machine Learning & Bioinformatics Lab

Contents

1. 실습 연구 소개
2. 실습 및 예시 코드
3. 후속 연구 소개

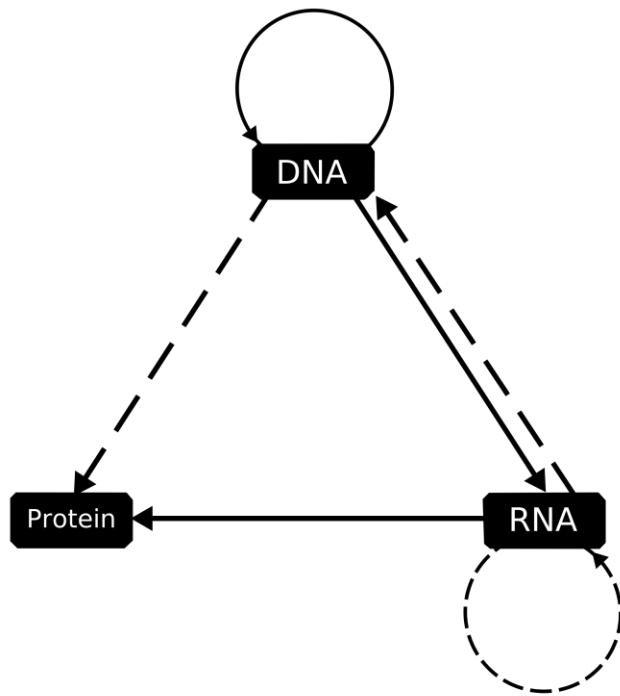
1. 실습 연구 소개 – DeepBind (2015, Nature biotechnology)

Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning

Published paper : <https://www.nature.com/articles/nbt.3300>

분자생물학의 중심 원리 (Central Dogma of Molecular Biology)

단백질로 만들어진 (유전)정보는 다른 단백질이나 핵산으로 전달될 수 없다



분자생물학의 중심 원리.
유전 정보가 전달되는 과정



일반적인 전이 과정

1. 세포내 유전정보 DNA가 스스로를 복제
2. DNA가 RNA를 생성하는 전사(transcription)
3. RNA에서 단백질(protein)을 생성하는 번역(translation)



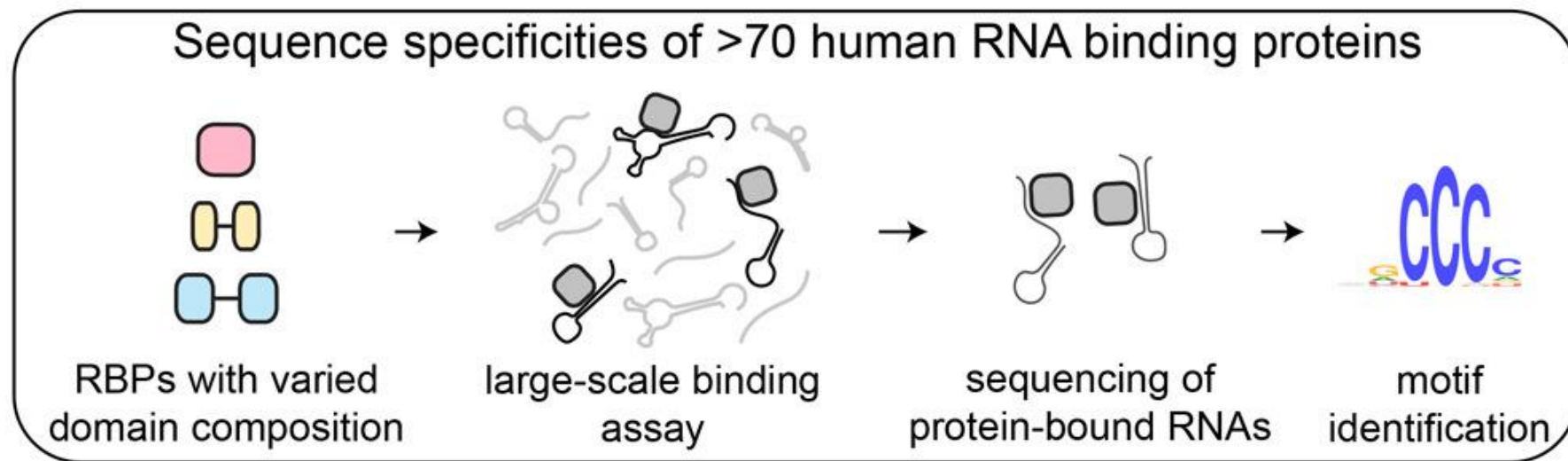
특수한 전이 과정

- 추정, 연구단계에서 논의되는 전이 현상
- 예를 들어 RNA에서 DNA를 생성하는 과정

그 외, 센트럴 도그마에 위배되는 사례

- 역전사를 통한 유전정보 전달되는 예 - [레트로바이러스](#)
- 유전정보 없이 형질 전달되는 예 - [프리온](#)
- [후성유전학](#) - [일란성 쌍둥이](#)

RNA Binding Proteins (RBPs)



“Knowing the sequence specificities of DNA- and RNA-binding proteins is essential for developing models of the regulatory processes in biological systems and for identifying causal disease variants.”

DNA- 또는 RNA-결합 단백질들에 대한 서열 특이성을 밝혀내는 것은 생물학적 시스템들이나 질병의 인과관계를 규명하는 것에 아주 핵심적인 역할을 한다

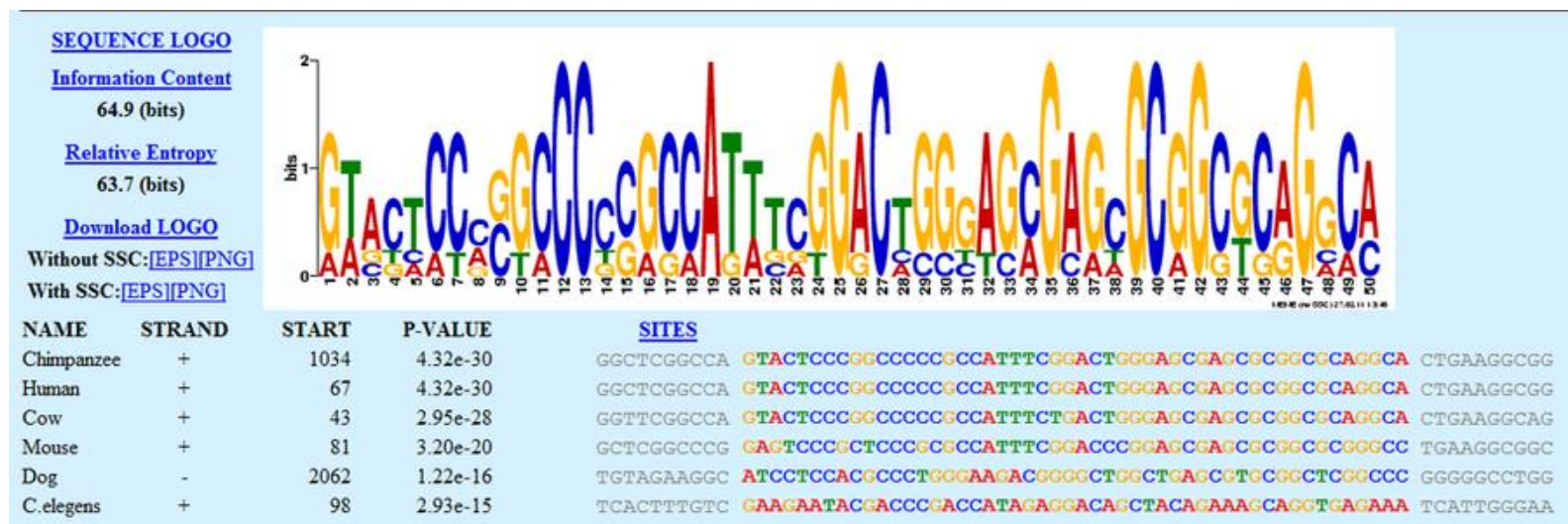
DeepBind (2015)

✓ 연구의 목표

대부분의 DNA- and RNA-binding proteins 연구와 유사하게 해당 단백질에 대한 motif를 찾는 것

✓ Motif(모티프)란 무엇인가?

유전학에서 뉴클레오타이드 또는 아미노산 서열 패턴을 의미, 이를 sequence motif라고 함



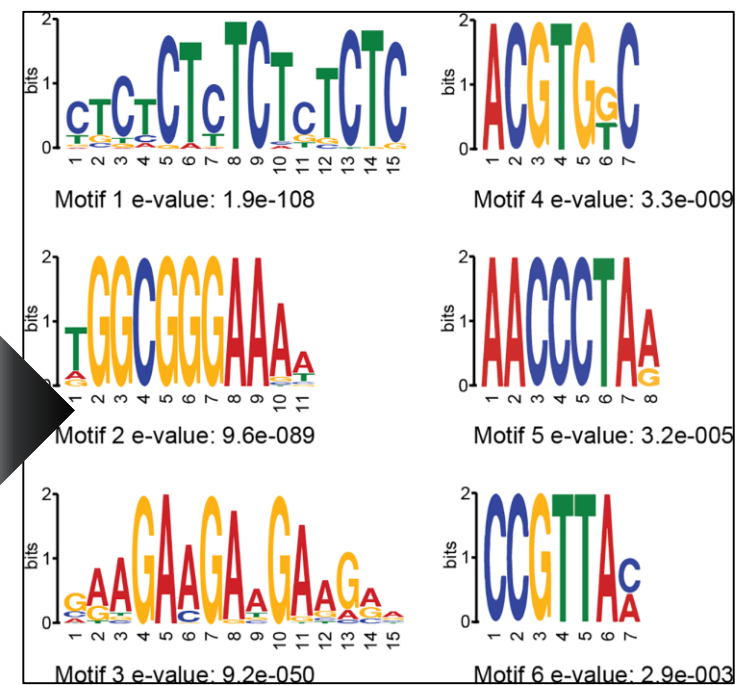
DeepBind (2015)

✓ 연구의 동기

Q. 부분 서열 탐색 알고리즘으로 패턴 찾으면 될 일이 아닌가? 🤔

A. 전통적인 연구 방향성이 맞음, 하지만 최근의 머신러닝 & 딥 러닝 기술의 발전을 통해 새로운 접근을 시도하는 중

	FoldID	EventID	seq	Bound	
1	A	seq_00001_peak	GCTGGGAAGCCTTCGGGTGTCCAGAACTCACTGGCAAGGATCCCTCTGAAACACAGCATTTTGGGAACCCCTTCAGTAGGAGTCCCTATTGTTTTCAGC	→	1
2	A	seq_00003_peak	GAGTTCCTTTGATGTGCAACAAGCTGAGAGCTCCAGCTGAAGGATTTCCCACTGATTACATTCAAAAGGGTTTTCTCTGTGTAGTCTCTGATGAGCA	→	1
3	A	seq_00005_peak	ACCCTACCAATGTCCGAGTGTGGGAAGAGATTCAAGCAGAGCTCTACCTTTTCAGCATCAGACATGACATCAGGTGAAAAACCATATGAATTGTCTG	→	1
4	A	seq_00007_peak	TTTGCCACATTCTCACATTGTAGGGTTTTGCCCCAGTATGAATCTCTTATGCTGAATAAGGTATGAGAACCAGGTAAGGCTTTGCCACATTCCTCAC	→	1
5	A	seq_00009_peak	GCCTGAGTCAGCATCAGCTGATCCACATGGAGAGAAGCCCTATAAATGCAACAAAGTGTACAAAAGCCCTTTGGTTGTAGTTCAGCGCTATTCTCGCCATGCA	→	1
6	A	seq_00011_peak	ACTCAGCTCATCAGGCATGCAAAATTCACATGGGGAAGGACCTGTAACAAATGCAATCAGTGCAATTAAGGCTTTGCAAGGAGCTCTCACTTGTGATGCA	→	1
7	A	seq_00013_peak	CAGTGACGTGGTAAAAAGCCTTCATCGATCAGTCATCCCTTAAGAAACACACAGCCTCTCAGCTGGAGAGAAGCCCTTTAGTGTTGAACGATGTGGAAAGT	→	1
8	A	seq_00015_peak	GTGATCGGTGAGTCTTATCCAGCACCAGAGAAGCTCATACTGGGGAGCGGCCATATGAGTGTAAAGTGTGGGAAAGCCTTTGGCTACTGCTCAGCCCTG	→	1
9	A	seq_00017_peak	CAGTGACGCCCTCTCTCTTCAACAGCAGCAGGCAAGCTCACACGGGCGAGAAGCCCTTGAATGTAAAGCAGTGTGGGAGAGCCCTTCAGCAGAGGCTTCTCC	→	1
10	A	seq_00019_peak	TGAGACAAGTGAAGCTTTGGCCACATTCATGCACACATAAGGTTTCTCCAGGTGTGAGTCTCTTATCGTGAATGAAGGCAATTGGAATACCGAAAGCT	→	1
11	A	seq_00021_peak	GCCAGCGATATTCCTGAAAAGTTTCTGACATCGGTTGACGCACTAGGGCTTCTCGCCAGTATGAATCCTCTGATGCCGTGATGAGGTACGCACTCTCGATGA	→	1
12	A	seq_00023_peak	TAATGACTGTGGGGAAGACTTTAGTCACATTACAGACTTTACTGACCATCAGAGGATCCATACGTCAGAGAAGCCCTATGATGTGAGCAGGCTTTTAGCT	→	1
13	A	seq_00025_peak	CCCTATGCGTGTGCTGAATGTGATAAGGCTTCAGCCGGAGCTTTCCCTCATCTACATCAGAGAAGCTCATACTGGAGAGAAACCTATGATATGAAGT	→	1
14	A	seq_00027_peak	CTACACATCAGAGAAGGCTCATACTGGAGAGAGACCTCAAAATGTGAAGATGTGGCAAAGCCTCAACTCTAGGTCATACCTCTACATACATCGAGAAGA	→	1
15	A	seq_00029_peak	ATGCCGAACAAAATTTGCAGGTGGGTAAAAAGCCTTCCACATTCTCTACAAACATAGGGCTTCTCCTCAGTGTGAATCCTCATGTGTGAGTGAAGGAAG	→	1
16	A	seq_00031_peak	AGACAGGAGACTGTCCCTGAAAGGCTTTGCCACACGATTACAGCTCTGTAGGGCTTCTCACCATATGAGTCTCTGTGGTCGACATGAGGCCCGGATGTTCTCT	→	1
17	A	seq_00033_peak	TAAGTTCTGAGCCAGCATCAAAAGGCTCCCCACTTCTACATCTTACATGAGTTTCTCCAGTATGAGTCTTCAGATGTTTCAGTAAGGTTGAGGCTGTAGCCGTA	→	1
18	A	seq_00035_peak	AAACCCATAGAATTGTGAAGAATGTGGGAAGGCCCTTCAAGTCTGTCCACAAACCTTGTCTCAACATGGAAGAGTTCTACATGGTGAGAAATCCATGTTGTGCA	→	1
19	A	seq_00037_peak	TAGGGCTTCTCTCCAGTATGAGTCTCTGGTGTTCAGTAAGATGTGCATCCGATTGAAGGCCTTTCCACATGCATTACACTCATAGGGTTTTTCTCCAGT	→	1
20	A	seq_00039_peak	CCAGAGAATTCATACAGGAGAGGACCTTATGAGTGTAAACGAATGTGCAAAAACCTCTTTAAGAAAGTCAAAACCTTATACATCATCAGAGAAGTTACACAG	→	1
21	A	seq_00041_peak	AACATCAGAGAATTCATACGGAGAGAAACCCCTATAAATGTGGCGAATGTGGAAAGCCCTTAAGGCAAGTTTCATGCTTACCCGGCATCAGAGAATTCAC	→	1
22	A	seq_00043_peak	AGACCCTACGAATGTAAGGAATGTAGGAAAGCCTTCAGCCAGTATGCACAGCTTGCTCAACATCAGAGAAGTTCATACTGGAGAAAAACCTTATGAATGTAA	→	1
23	A	seq_00045_peak	CAACATCTGAGGATTCATACTGGAGAAAAAGCCCTATAAATGTGAATCAATGTGGTAAAGCTTTTAGCCGCGATCACATCCCTTACTGAGACATCATAGACTTCG	→	1
24	A	seq_00047_peak	GTGGCTCAGCACTTACTAATCATCAGAGAATTCACATGGTGAGAAACCCCTATGATTGAAGGAATGTGGAAAGGCTTTTACTCAGAGCTCACAGCTTCG	→	1
25	A	seq_00049_peak	GAGAAACACACACAGGGGAGAAATCGTATGAATGTCTGCAATGTAGGAATGCCCTCAGATTGAAGTCACACCTTCTGTCATCAGAGAATCACACGGGAG	→	1
26	A	seq_00051_peak	GCATTCACTGAATCTTCTGTGCTTAAACGACATGAGAGAATTCACATGGAGAGAAACCATATGAGTGCCATGTATGTGGGAAAGCCTTCACTGAATCTTC	→	1
27	A	seq_00053_peak	TTGTCCATTCAGTGCAATCATGAATGTTTCAGTCTGTATGAGTCTCTGATGGTCCATTAGTCTGGAATTTTCTGGAGAAAGCCTTCCGCAACATCTTGA	→	1
28	A	seq_00055_peak	TGTGGCAAGGCTTCAACCAAGGCTTACATCTCACTGGACACAGAGAACTCATCACTGGGAAGGAAACATACACATGTGAAGATGTGGCAAAACCTTTAA	→	1
29	A	seq_00057_peak	GTACTGAGTGGCGCAAAACCTTCAGCCAGAGGTCAACTCTTAGATTACATTCGCAATCCACACAGGAGAGAAACCATATGAGTGTTCGAATGTGGGAAG	→	1



DeepBind (2015)

DeepBind (2015) 모델 연구에 대한 개요 (그림1)

1. High-throughput experiments - 유전체, 분자생물학 실험 데이터 수집 및 가공 단계
2. Massively parallel deep learning - 실험 데이터를 사용한 머신러닝, 딥러닝 모델링 단계 (학습)
3. Community needs - 학습된 모델을 바탕으로 유의미한 분석을 수행하는 단계

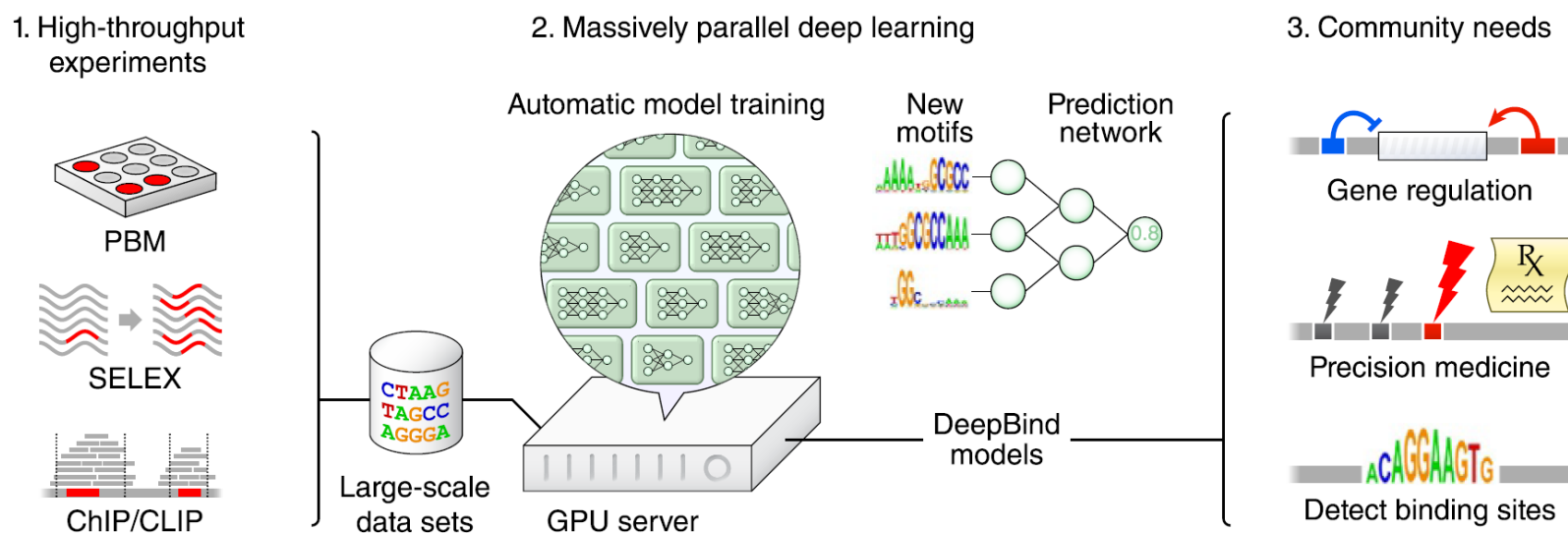


그림1. DeepBind's input data, training procedure and applications

DeepBind (2015)

DeepBind (2015) 모델 연구에 대한 개요 (그림1)

1. High-throughput experiments - 유전체, 분자생물학 실험 데이터 수집 및 가공 단계
2. Massively parallel deep learning - 실험 데이터를 사용한 머신러닝, 딥러닝 모델링 단계 (학습)
3. Community needs - 학습된 모델을 바탕으로 유의미한 분석을 수행하는 단계

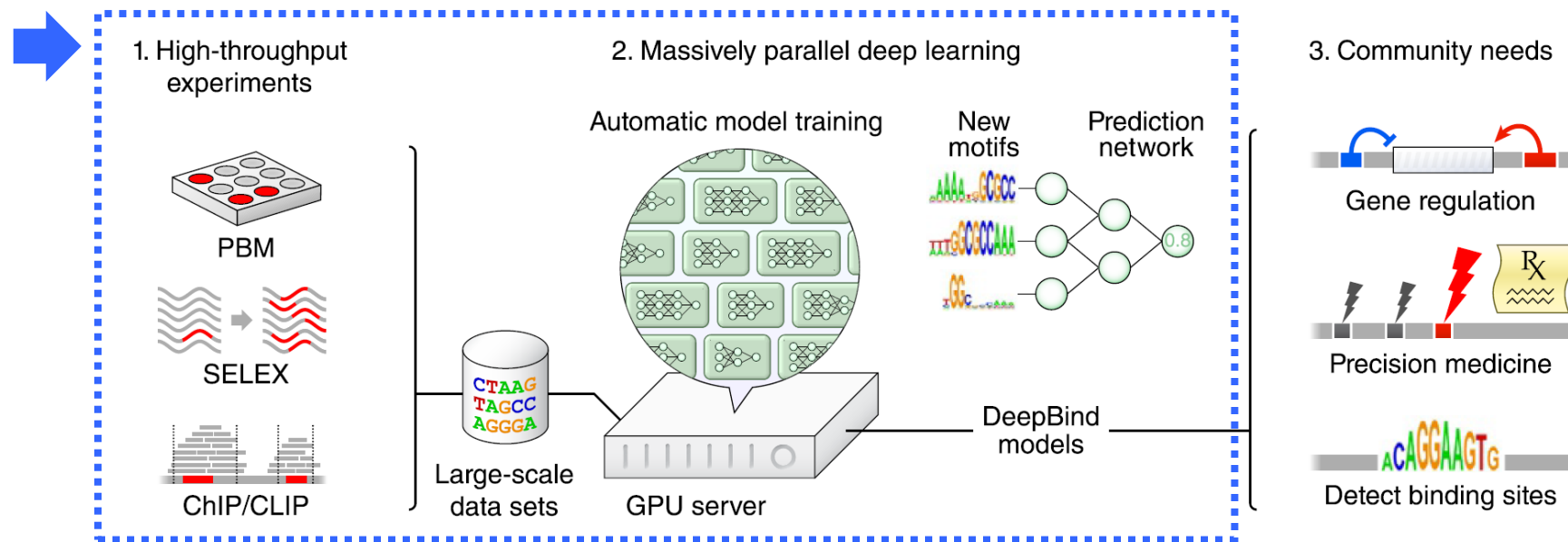


그림1. DeepBind's input data, training procedure and applications

High-throughput experiments, 용어 및 내용 정리

1. High-throughput experiments



PBM



SELEX



ChIP/CLIP

High-throughput

- 일반적으로 1회 정도 할 수 있던 일을 100회 이상 대용량고효율로 처리하는 것을 의미
- High-throughput platform을 갖추다 ~ 라는 표현으로 많이 사용

Sequencing

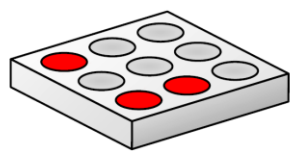
- 시퀀싱(sequencing)이란 DNA 서열을 분석하기 위해 생화학적 방법으로 모든 세포의 DNA 사슬을 구성하는 염기(A, C, G, T)가 결합된 순서를 분석하는 기술

NGS

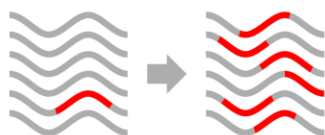
- Next Generation Sequencing 의 약자, 유전체의 염기서열을 고속으로 분석하는 기법
- 하나의 유전체를 수많은 조각으로 분해하여 각 조각을 동시에 읽은 후 전산 기술(ex. alignment algorithm)으로 조합하여 유전체 정보를 해독하는 방법

High-throughput experiments, 용어 및 내용 정리

1. High-throughput experiments



PBM



SELEX



ChIP/CLIP

PBM

- Protein (Binding) Microarray 실험의 약자 (high-throughput 실험)
- 특정 리간드(ligand)에 결합하는 수용체(receptor)를 검출하는데 사용 가능

SELEX

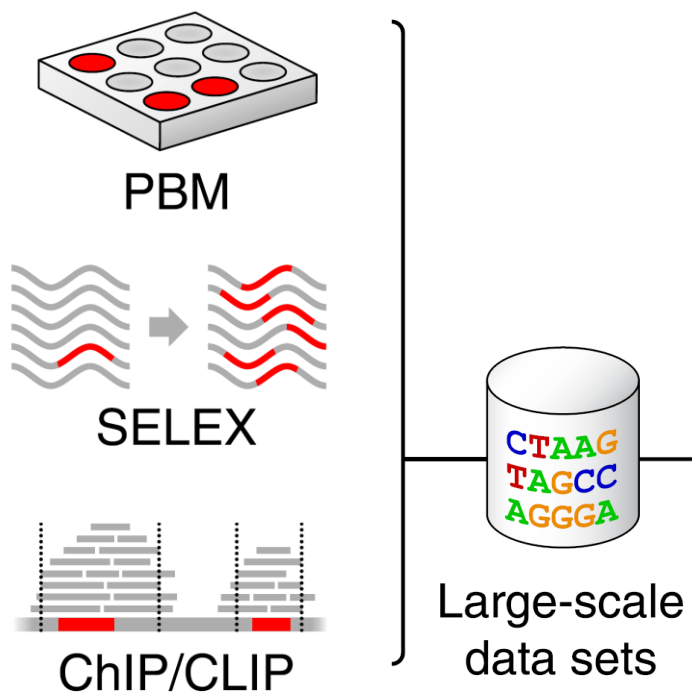
- 특정 수용체(ex. Protein)에 결합하는 인공적인 DNA/RNA 서열 생성 실험

ChIP-seq

- 특정 단백질이 결합(binding)하는 DNA서열을 NGS로 시퀀싱
- 유전체 전체에서 결합 가능한 motif 를 탐색하기위해 사용

High-throughput Sequencing Data

1. High-throughput experiments



Large-scale data sets

- 표적 단백질에 결합한 DNA/RNA 서열 정보를 포함한 텍스트 파일들

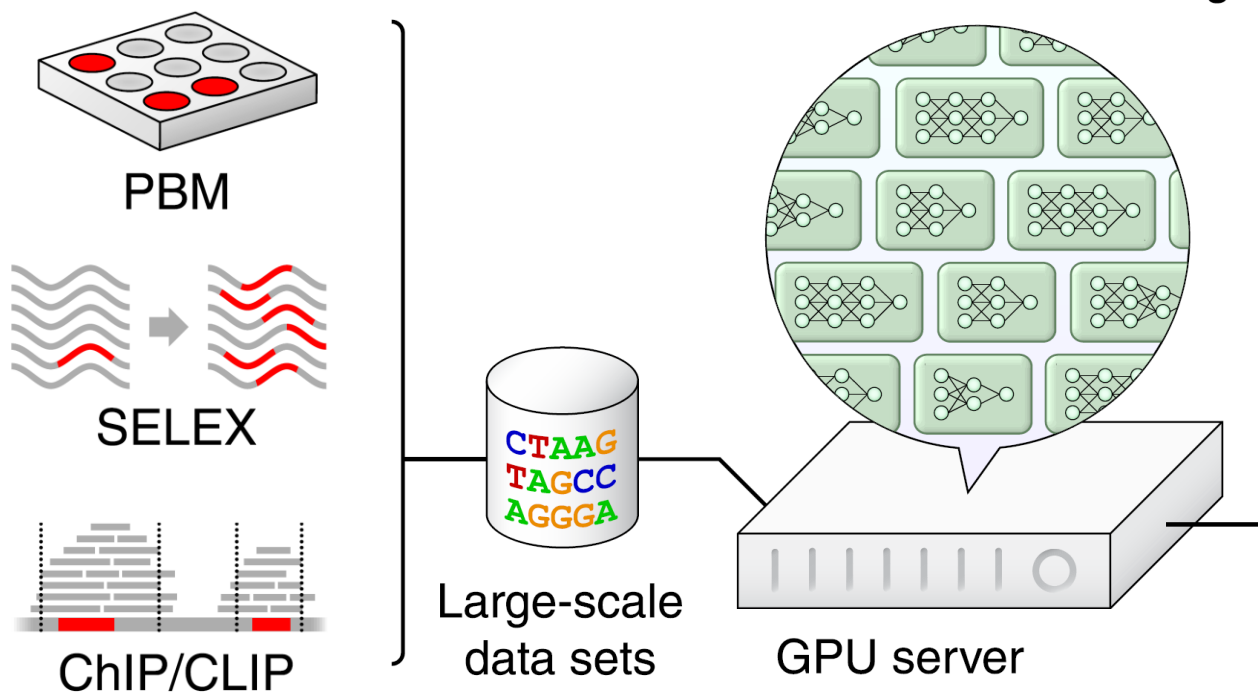
```

1 FoldID=EventID*seq*Bound
2 A→seq_00001_peak→GCTGGGAAGCCTTCGGGTGTCCCAGAACTCACTGGCAAGGATCCCTCTCTGAACACAGCATTTCGGGAACCCCTTCAGTAGGAGTCTCATTGTTTTCAGC→1
3 A→seq_00003_peak→GAGTTCTTTGATGTGCAACAAGCTGAGAGTCCAGCTGAAGGATTTCCACACATGATTACATTCAAAGGGTTTTCTCCTGTGTGAGTTCTCTGATGAGCA→1
4 A→seq_00005_peak→ACCCTACAAATGTCCCAGAGTGTGGGAAGAGATTGAGCAGCAGCTCTCACCTTATTCAGCATCACAGATCACATACAGGTGAAAAACCATATGAATGTTCTG→1
5 A→seq_00007_peak→TTTGCCACATTCTCATTGTTAGGGTTTCTGCCAGTATGAATTCTCTATGCTGAATAAGGTATGAGAACAGGTAAAAAGCTTGGCCATTCTCTAC→1
6 A→seq_00009_peak→GCCTGAGTCAGCATCAGCTGATTACACTGGAGAGAAGCCTTATAAATGCAACAAGTGTACAAAAGCCTTTGGTTGAGTTACGGCTTATTCGCCATCAG→1
7 A→seq_00011_peak→ACTCAGCTCATCAGGCATCTGCAAAATTCACACTGGGGAAGAGCCTGACAAATGCAATCAGTGCAATAAAGCCTTTGCAAGGAGCTCCTACCTTGTGATGCA→1
8 A→seq_00013_peak→CAGTGACTGTGGAAAAGCCTTCATCGATCAGTCATCCCTTAAGAAACACACACGCTCTCACACTGGAGAGAAGCCTTATGAGTGAACAGGTGGAAAGT→1
9 A→seq_00015_peak→GTGATCGGTGAGTCTTATCCAGCACAGAGAACTCATCTGGGAGCGGCATATGAGTGAACGAATGTGGGAAGCCTTTGGCTACTGCTCAGCCCTG→1
10 A→seq_00017_peak→CAGTGACCGTCTCTCTCAACCAGCACGAGCGAACTCA
11 A→seq_00019_peak→TGAGACAACCTGAAAGCTTTGCCACATTCCATGCACACAT
12 A→seq_00021_peak→GCCAGCGATATTCTGAAAGGTTTCTGACACTGGTTGCA
13 A→seq_00023_peak→TAATGACTGTGGGGAAGACTTTAGTCACATTACAGACTT
14 A→seq_00025_peak→CCCTATGCGTGTGCTGAATGTGATAAAGCCTTCAGCCGG
15 A→seq_00027_peak→CTACACATCAGAGAAGTCATACTGGAGAGAGACCTACA
16 A→seq_00029_peak→ATGCCGAACAAAATTTGCAGGTGGGTAAAAAGCCTTTC
17 A→seq_00031_peak→AGACAGGAGCTGTCCCTGAAGGCTTTGCCACACCGATTAT
18 A→seq_00033_peak→TAAGTTCTGAGCCAGCAGTAAAGGCCCTCCACATTCTCT
19 A→seq_00035_peak→AAACCTATGAATGTGAAGAATGTGGGAAGCCTTCAGT
20 A→seq_00037_peak→TAGGGCTTCTCTCAGTATGAGTTCTCTGTTGTTCACTA
21 A→seq_00039_peak→CCAGAGAATTATACAGGAGAGCAGCCTATGAGTGAAT
22 A→seq_00041_peak→AACATCAGAGAATTATACACTGGAGAGAACCCTATAAAT
23 A→seq_00043_peak→AGACCCTACGAATGTGAAGGAATGTAGGAAAGCCTTCAGC
24 A→seq_00045_peak→CAACATCTGAGGATTATACACTGGAGAGAAGCCTATAAAT
25 A→seq_00047_peak→GTGGCTCAGCACTTACTAATCATCAGAGAATTACACTGT
26 A→seq_00049_peak→GAGAACACACACAGGGGAGAAATCGTATGAATGTCTGCA
27 A→seq_00051_peak→GCATTCACTGAATCTTCTGTGCTTAAACGACATGAGAGA
28 A→seq_00053_peak→TTGTCACTTCAGTGCATTCAATAATGTTTCAAGTTCTGTA
29 A→seq_00055_peak→TGTGGCAAGAGCCTTCAACAGGGCTTACATCTCACTGGA
30 A→seq_00057_peak→GTACTGAGTGCAGAAAACCTTCAGCCAGAGGTCAACTC

A→seq_00001_peak→GCTGGGAAGCCTTCGGGTGTCCCAGAACTCA
A→seq_00003_peak→GAGTTCTTTGATGTGCAACAAGCTGAGAGTCCAGCTGAAGGATTTCCACACATGATTACATTCAAAGGGTTTTCTCCTGTGTGAGTTCTCTGATGAGCA
A→seq_00005_peak→ACCCTACAAATGTCCCAGAGTGTGGGAAGAGATTGAGCAGCAGCTCTCACCTTATTCAGCATCACAGATCACATACAGGTGAAAAACCATATGAATGTTCTG
A→seq_00007_peak→TTTGCCACATTCTCATTGTTAGGGTTTCTGCCAGTATGAATTCTCTATGCTGAATAAGGTATGAGAACAGGTAAAAAGCTTGGCCATTCTCTAC
A→seq_00009_peak→GCCTGAGTCAGCATCAGCTGATTACACTGGAGAGAAGCCTTATAAATGCAACAAGTGTACAAAAGCCTTTGGTTGAGTTACGGCTTATTCGCCATCAG
A→seq_00011_peak→ACTCAGCTCATCAGGCATCTGCAAAATTCACACTGGGGAAGAGCCTGACAAATGCAATCAGTGCAATAAAGCCTTTGCAAGGAGCTCCTACCTTGTGATGCA
A→seq_00013_peak→CAGTGACTGTGGAAAAGCCTTCATCGATCAGTCATCCCTTAAGAAACACACACGCTCTCACACTGGAGAGAAGCCTTATGAGTGAACAGGTGGAAAGT
A→seq_00015_peak→GTGATCGGTGAGTCTTATCCAGCACAGAGAACTCATCTGGGAGCGGCATATGAGTGAACGAATGTGGGAAGCCTTTGGCTACTGCTCAGCCCTG
A→seq_00017_peak→CAGTGACCGTCTCTCTCAACCAGCACGAGCGAACTCA
A→seq_00019_peak→TGAGACAACCTGAAAGCTTTGCCACATTCCATGCACACAT
A→seq_00021_peak→GCCAGCGATATTCTGAAAGGTTTCTGACACTGGTTGCA
A→seq_00023_peak→TAATGACTGTGGGGAAGACTTTAGTCACATTACAGACTT
A→seq_00025_peak→CCCTATGCGTGTGCTGAATGTGATAAAGCCTTCAGCCGG
A→seq_00027_peak→CTACACATCAGAGAAGTCATACTGGAGAGAGACCTACA
A→seq_00029_peak→ATGCCGAACAAAATTTGCAGGTGGGTAAAAAGCCTTTC
A→seq_00031_peak→AGACAGGAGCTGTCCCTGAAGGCTTTGCCACACCGATTAT
A→seq_00033_peak→TAAGTTCTGAGCCAGCAGTAAAGGCCCTCCACATTCTCT
A→seq_00035_peak→AAACCTATGAATGTGAAGAATGTGGGAAGCCTTCAGT
A→seq_00037_peak→TAGGGCTTCTCTCAGTATGAGTTCTCTGTTGTTCACTA
A→seq_00039_peak→CCAGAGAATTATACAGGAGAGCAGCCTATGAGTGAAT
A→seq_00041_peak→AACATCAGAGAATTATACACTGGAGAGAACCCTATAAAT
A→seq_00043_peak→AGACCCTACGAATGTGAAGGAATGTAGGAAAGCCTTCAGC
A→seq_00045_peak→CAACATCTGAGGATTATACACTGGAGAGAAGCCTATAAAT
A→seq_00047_peak→GTGGCTCAGCACTTACTAATCATCAGAGAATTACACTGT
A→seq_00049_peak→GAGAACACACACAGGGGAGAAATCGTATGAATGTCTGCA
A→seq_00051_peak→GCATTCACTGAATCTTCTGTGCTTAAACGACATGAGAGA
A→seq_00053_peak→TTGTCACTTCAGTGCATTCAATAATGTTTCAAGTTCTGTA
A→seq_00055_peak→TGTGGCAAGAGCCTTCAACAGGGCTTACATCTCACTGGA
A→seq_00057_peak→GTACTGAGTGCAGAAAACCTTCAGCCAGAGGTCAACTC
    
```

How to find motifs? 🤔

1. High-throughput experiments



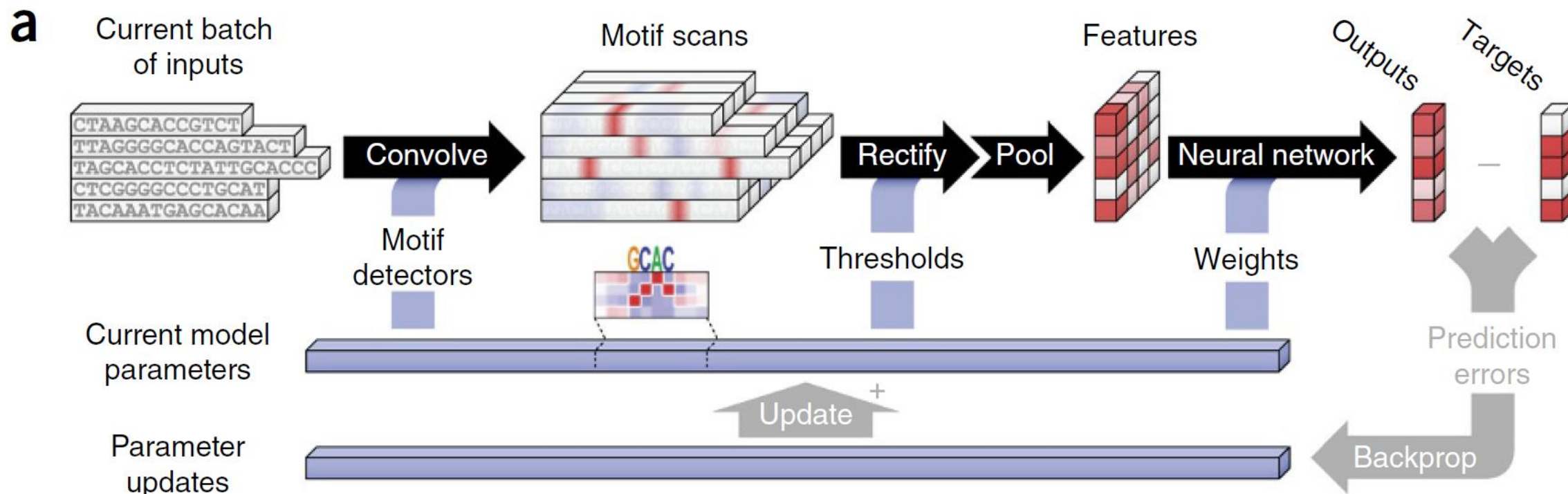
Automatic model training

- 표적 단백질에 결합한 DNA/RNA 서열들을 학습시킨다???

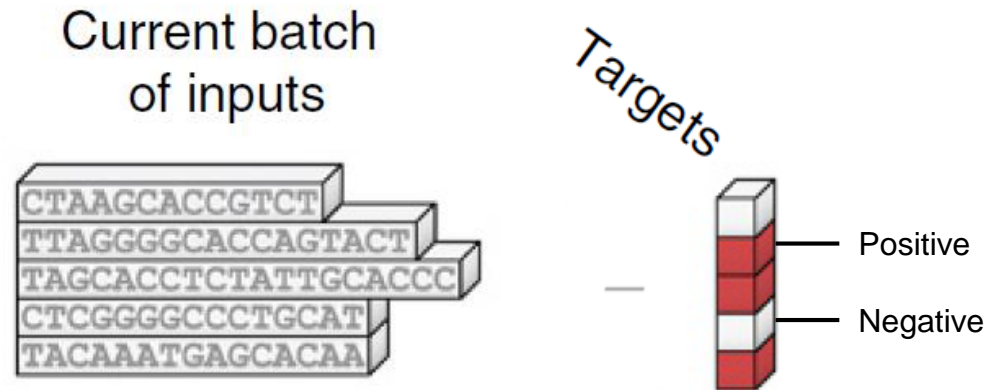
```

1 FoldID=EventID=seq=Bound
2 A→seq_00001_peak→GCTGGGAAGCCTTCGGGTGTCCAGAACCTCACTGGCAAGGATCCCTCTCTGAACACAGCATTTTGGGAACCCCTTCAGTAGAGTCCTCATTGTTTCAGC→1
3 A→seq_00003_peak→GAGTTCCTTTGATGTGCAACAAGCTGAGATCCAGTGAAGGATTTCCACACTGATTACATCAAGGTTTTCTCTGTGTGAGTCTCTGATGAGCA→1
4 A→seq_00005_peak→ACCCTACAAATGTCCCAGAGTGTGGGAAGATTCAGCAGCAGCTCTCACCTTATTCAGCATCAGACATCACATACAGGTGAAAAACCATATGAATGTTCTG→1
5 A→seq_00007_peak→TTTGCCACATTCCTCACATTTGTAGGGTTTTCGCCAGATGAATTTCTTATGCTGAATAGGATGAGAACAGGTGAAAAAGCTTTGCCACATTCCTCAC→1
6 A→seq_00009_peak→GCCTGAGTCAGCATCAGCTGATTCACACTGGAGAGAAGCTTATAAATGCAACAAGGTGACAAAGCTCTTGGTGTGATGTCACGGCTTATTCGCCATCAG→1
7 A→seq_00011_peak→ACTCAGCTCATCAGGCATCTGCAAAATTCACACTGGGAGAAGCGTACAAATGCAATGCAATGAAGGCTTTGCAAGGAGCTCTCCTACTTGTGATGCA→1
8 A→seq_00013_peak→CAGTGACTGTGGAAAAGCCTTCATCGATCAGTCATCCCTTAAGAAACACACAGCTCTCACACTGGAGAGAAGCCTTATGAGGTAAACAGTGTGGAAGT→1
9 A→seq_00015_peak→GTGATCGGTGAGTCTTATCCAGCACAGAGAATCTACTGGGAGCGGCTATAGGTGTAACGAATGTGGGAAAAGCCTTTGGCTACTGCTCAGCCCTG→1
10 A→seq_00017_peak→CAGTGACCGTTCCTCTCTCAACACAGCAGAGCACTCACAGCGGAGAGACCTTATGAATGTAAGCAGTGTGGGAGAGCCTTCAGCAGAGGCTTTCCC→1
11 A→seq_00019_peak→TGAGACAACGAAAGCTTTGCCACATTCATGACACATAAGGTTTCTCTCAGGTGAGGTTTCTTATGAGTAATGAAGGCAATTGAGTAAAGGAGCT→1
12 A→seq_00021_peak→GCCAGCGATATTCCTGAAAAGTTTCTGACACTGGTTCAGCGCATAGGGCTTCTCGCCAGTATGAAATCCTCTGATGCCGTGAGAGTACGCACTCTGATGA→1
13 A→seq_00023_peak→TAATGACTGTGGGGAAGCTTTAGTCACATTACAGACTTTACTGACCATCAGAGGATCATCTAGCAGAGAACCCTATGATGTTGAGCAGGCTTTTAACT→1
14 A→seq_00025_peak→CCCTATGCTGTGCTGAATGTGATAAGGCTTCAGCGGAGCTTTTCCCTCATTCTACATCAGAGAGCTCATCTGGAGAGAACCCTATGATGATGAAGT→1
15 A→seq_00027_peak→CTACACATCAGAGAAGTCATCTGGAGAGAGACCTACAAATGTGAAGATGTGGCAAGCTTCACTCTAGGTCATACCTCCTACATCATCGGAGAAGA→1
16 A→seq_00029_peak→ATGCCGAACAAAATTTGCAGGTGGGTAAAAGCCTTCCACATCTCTCAAAACATAGGGCTTCTCCAGGATGAACTCCTATGTCGAGTGAAGGAAG→1
17 A→seq_00031_peak→AGACAGGAGCTGCTCCCTGAAGGCTTTTGCACACGATTAACACTGTAGGGCTTCTCACCAGATATGAGTCTCTGCTGAGGAGGAGGCTTCCCTGATGTTCT→1
18 A→seq_00033_peak→TAAGTCTGAGCCACGACTAAAGGCTTCCACATTCCTTACATTGATGAGGTTTCTCTCAGTATGAGTCTCAGATGTTCAAGGTGTGAGGACCA→1
19 A→seq_00001_peak→GCTGGGAAGCCTTCGGGTGTCCAGAACCTCACTGGCAAGGATCCCTCTCTGAACACAGCATTTTGGGAACCCCTTCAGTAGAGTCCTCATTGTTTCAGC→1
20 A→seq_00003_peak→GAGTTCCTTTGATGTGCAACAAGCTGAGATCCAGTGAAGGATTTCCACACTGATTACATCAAGGTTTTCTCTGTGTGAGTCTCTGATGAGCA→1
21 A→seq_00005_peak→ACCCTACAAATGTCCCAGAGTGTGGGAAGATTCAGCAGCAGCTCTCACCTTATTCAGCATCAGACATCACATACAGGTGAAAAACCATATGAATGTTCTG→1
22 A→seq_00007_peak→TTTGCCACATTCCTCACATTTGTAGGGTTTTCGCCAGATGAATTTCTTATGCTGAATAGGATGAGAACAGGTGAAAAAGCTTTGCCACATTCCTCAC→1
23 A→seq_00009_peak→GCCTGAGTCAGCATCAGCTGATTACACACTGGGAGAAGCGTACAAATGCAATGCAATGAAGGCTTTGCAAGGAGCTCTCCTACTTGTGATGCA→1
24 A→seq_00011_peak→ACTCAGCTCATCAGGCATCTGCAAAATTCACACTGGGAGAAGCGTACAAATGCAATGCAATGAAGGCTTTGCAAGGAGCTCTCCTACTTGTGATGCA→1
25 A→seq_00013_peak→CAGTGACTGTGGAAAAGCCTTCATCGATCAGTCATCCCTTAAGAAACACACAGCTCTCACACTGGAGAGAAGCCTTATGAGGTAAACAGTGTGGAAGT→1
26 A→seq_00015_peak→GTGATCGGTGAGTCTTATCCAGCACCAATGCTGATCGGTGAGTCTTATCCAGCACCAATGCTGATCGGTGAGTCTTATCCAGCACCAATGCTGATCGGTGAGTCTTATCCAGCACCA→1
27 A→seq_00017_peak→CAGTGACCGTTCCTCTCTCAACCAGCACACTGAGTGAAGGCTTCTCACCAGATATGAGTCTCTGCTGAGGAGGAGGCTTCCCTGATGTTCT→1
28 A→seq_00019_peak→TGAGACAACGAAAGCTTTGCCACATTCATGACACATAAGGTTTCTCTCAGGTGAGGTTTCTTATGAGTAATGAAGGCAATTGAGTAAAGGAGCT→1
29 A→seq_00021_peak→GCCAGCGATATTCCTGAAAAGTTTCTGACACTGGTTCAGCGCATAGGGCTTCTCGCCAGTATGAAATCCTCTGATGCCGTGAGAGTACGCACTCTGATGA→1
30 A→seq_00023_peak→TAATGACTGTGGGGAAGACTTTAGTCACATTACAGACTTTACTGACCATCAGAGGATCATCTAGCAGAGAACCCTATGATGTTGAGCAGGCTTTTAACT→1
    
```

Sol. Deep Convolutional Neural Network & Binary Classification



Sol. Deep Convolutional Neural Network & Binary Classification



- ① 실험 결과, 발생한 서열들은 Positive class, 그렇지 못한 서열들 (없으면 무작위 서열)은 Negative class로 지정

Sol. Deep Convolutional Neural Network & Binary Classification



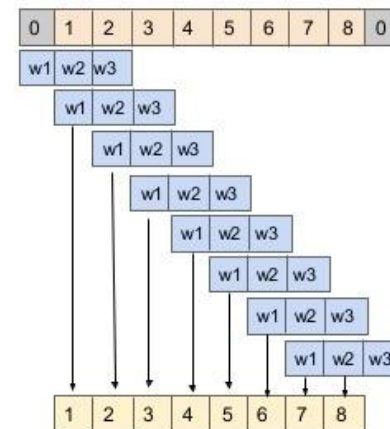
- ① 실험 결과, 발생한 서열들은 Positive class, 그렇지 못한 서열들 (없으면 무작위 서열)은 Negative class로 지정
- ② 만약 어떤 방법(혹은 모델)이 이 두가지 class들을 분류할 수 있다면 해당 모델은 입력 서열에서 유의미한 패턴 정보를 파악한 것

Sol. Deep Convolutional Neural Network & Binary Classification



1D Convolutions

When we add zero padding, we normally do so on both sides of the sequence (as in image padding)

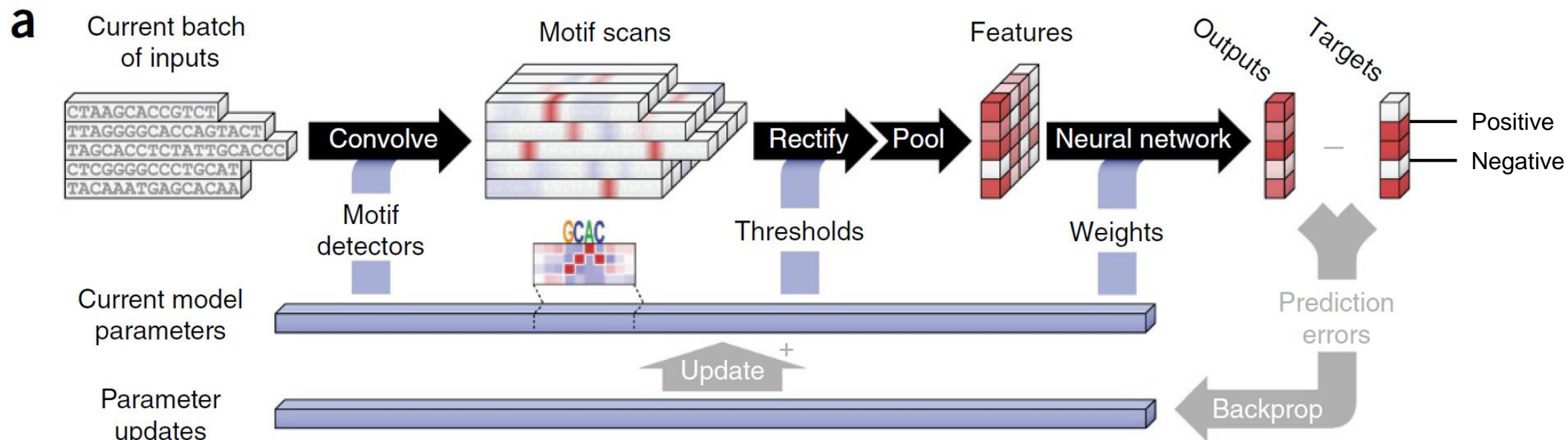


The length result of the convolution is well known to be:
 $\text{seqlength} - \text{kwidth} + 1 = 10 - 3 + 1 = 8$

So the output matrix will be (8, 100) because we had padding

- ① 실험 결과, 발생한 서열들은 Positive class, 그렇지 못한 서열들 (없으면 무작위 서열)은 Negative class로 지정
- ② 만약 어떤 방법(혹은 모델)이 이 두가지 class들을 분류할 수 있다면 해당 모델은 입력 서열에서 유의미한 패턴 정보를 파악한 것
- ③ 머신러닝/딥러닝 방법론으로 이를 접근한다면, 어느 모델 구조 (architecture)가 가장 적합한가? → 1D-CNN

Sol. Deep Convolutional Neural Network & Binary Classification



모델은 입력 서열에서 유의미한 패턴 정보를 파악한 것

③ 머신러닝/딥러닝 방법론으로 이를 접근한다면, 어느 모델 구조 (architecture)가 가장 적합한가? → 1D-CNN

④ 결론: 1D-CNN 모델과 실험 데이터를 바탕으로 이진분류모델을 학습시키면 CNN 구조의 filter 정보로 motif를 추론할 수 있다

<http://tools.genes.toronto.edu/deepbind/>

DeepBind

type search tags here

Welcome to the searchable database of DeepBind models!

Example searches:

`gata selex type:tf` - GATA TFs trained on SELEX data

`"mus musculus" type:rbp` - mouse RBPs

For more help, see frequently asked questions.

Score your own sequences!

[deepbind-v0.11-win32.zip](#)

[deepbind-v0.11-linux.tgz](#)

[deepbind-v0.11-osx.tgz](#)

[Copy IDs](#)

[« prev](#)

[next »](#)

Protein	Type	Species	Family	Experiment	Logos	ID
A1CF	RBP	Gallus gallus	RRM	RNAcompete		D00288.001
A1CF	RBP	Homo sapiens	RRM	RNAcompete		D00084.001
A2BP1	RBP	Drosophila melanogaster	RRM	RNAcompete		D00175.001
Ahctf1	TF	Mus musculus	AT hook	PBM		↔ D00053.001

2. 실습 및 예시 코드 – ML/DL for sequencing data

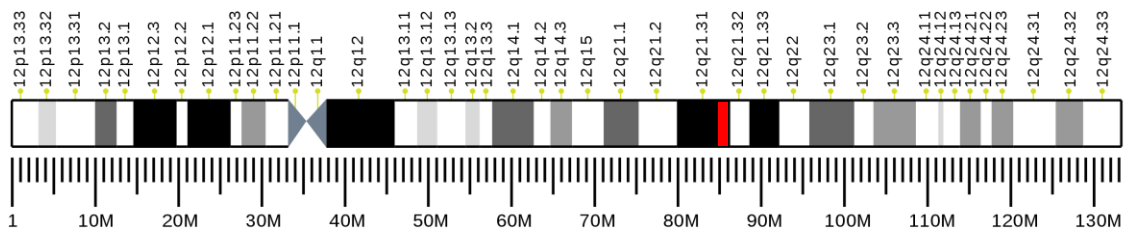
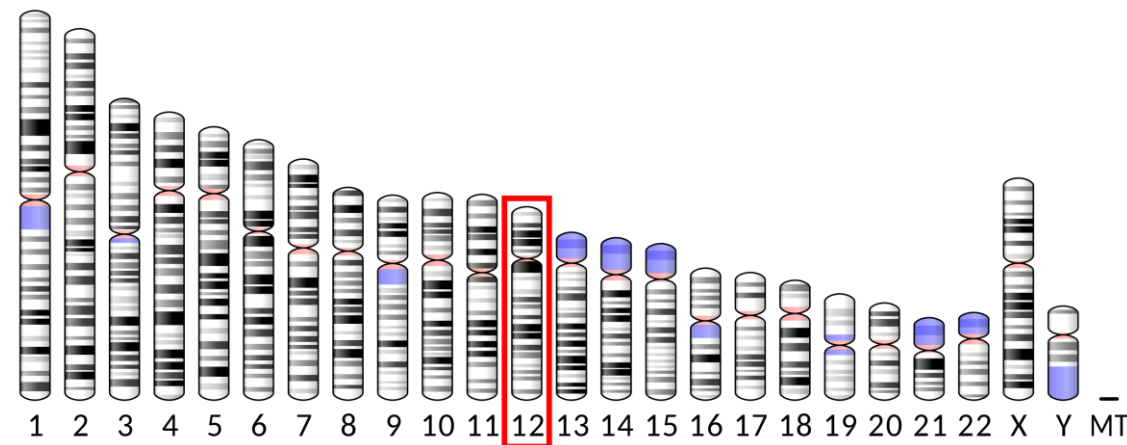
실습 목표 확인, 데이터 준비, etc ...

실습 과제 목표

1. 이진 분류 모델 학습을 위한 sequencing 데이터 준비하기
2. Sequencing 데이터를 다양한 머신러닝 모델에 사용하기
3. 이진 분류 모델의 학습에 대한 올바른 평가 수행하기

※ Motif 찾는 사후 분석은 제외

Download raw data



Alx1

ALX homeobox protein 1 is a [protein](#) that in humans is encoded by the *ALX1* [gene](#).

- ID: **D00289.002**
- Type: **TF**
- Species: **Mus musculus**
- Family: **Homeodomain**
- Experiment: **SELEX**
- Model: **deepbind 0.1**
- Cite: **PMID 23332764**
- Wiki: <https://en.wikipedia.org/wiki/ALX1>
- DeepBind info: ([link](#))

Sequence Dataset for Binary Classifier (Source: DeepBind)

- Download type-A ([link](#))
- Download type-B ([link](#))

About the raw data sequences

- Alx1_DBD_TAAAGC20NCG_3_Z_B.seq (학습 및 평가용 데이터)
- Alx1_DBD_TAAAGC20NCG_3_Z_A.seq (테스트용 데이터)

Alx1_DBD_TAAAGC20NCG_3_Z_B.seq

FoldID	EventID	seq	Bound
B	seq_000000_peak	AATTAACCTACTAATTGGAT	1
B	seq_000000_shuf	ACTAAAACCTGGATATTATT	0
B	seq_000002_peak	TAATTGAGGTAACGCTTTCC	1
B	seq_000002_shuf	TCTTTGGTATTAACGAAGCC	0
B	seq_000004_peak	CCTTTGCGAAATTAAGTTAA	1

Alx1_DBD_TAAAGC20NCG_3_Z_A.seq

FoldID	EventID	seq	Bound
A	seq_000001_peak	GCAGATAATCTAATTACCCC	1
A	seq_000003_peak	CTCAGTCCTCGTCTCGATGG	1
A	seq_000005_peak	TCATAATCTAATTACGCTCG	1
A	seq_000006_peak	GACTTCCTCAATCTAATTAG	1
A	seq_000011_peak	GCAGTTAATCTAATTAACCG	1

입력 데이터(input feature)

출력 데이터(label)

테스트 데이터 셋 내의 출력 데이터 (label)는 Positive class만 존재

머신러닝 모델에 입력으로 넣기 위해 numerical form으로 변환이 필요

Integer array로 변환하면 코드에서 직접 사용가능

Load the downloaded raw data

- Alx1_DBD_TAAAGC20NCG_3_Z_B.seq (학습 및 평가용 데이터)
- Alx1_DBD_TAAAGC20NCG_3_Z_A.seq (테스트용 데이터)

```
from collections import defaultdict
import numpy as np

def raw_seq_parser(path):
    f = open(path, "r")
    seqs = []
    seq_lens = []
    labels = []
    flines = f.readlines()
    cols = None
    for i, line in enumerate(flines):
        if not i:
            cols = line.replace("\n", "").split("\t")
        else:
            fold_id, event_id, seq, bound = line.replace("\n", "").split("\t")
            seqs.append(seq)
            labels.append(int(bound))
            seq_lens.append(len(seq))

    num_pos = sum(labels)
    num_neg = len(labels) - num_pos
    print("Load {} finish ({} lines, #positive: {}, #negative: {})".format(path, len(labels), num_pos, num_neg))
    print("- sequence sizes : {}".format(set(seq_lens)))

    return seqs, labels
```

path of raw data

TRAIN_RAW = "Alx1_DBD_TAAAGC20NCG_3_Z_B.seq"

TEST_RAW = "Alx1_DBD_TAAAGC20NCG_3_Z_A.seq"

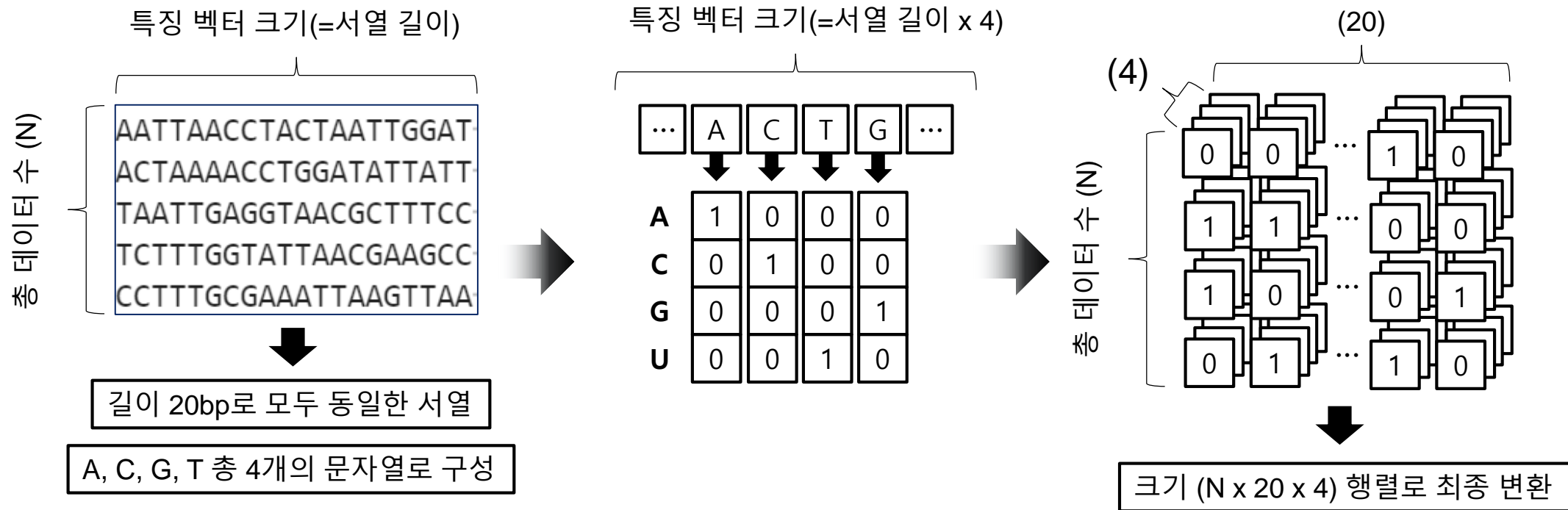
train_seqs, train_labels = raw_seq_parser(TRAIN_RAW)

test_seqs, test_labels = raw_seq_parser(TEST_RAW)

Load Alx1_DBD_TAAAGC20NCG_3_Z_B.seq finish (255508 lines, #positive: 127754, #negative: 127754)
- sequence sizes : {20}
Load Alx1_DBD_TAAAGC20NCG_3_Z_A.seq finish (128012 lines, #positive: 128012, #negative: 0)
- sequence sizes : {20}

Sequence Encoding (one-hot encoding)

- 별도의 변환 없이, 데이터 그 자체를 표현할 수 있는 간단한 방법
- 보통 classification 데이터의 label 변환에 자주 사용된다 (ex. 4 classes, class-4 \rightarrow [0,0,0,1])
- 생물정보학 분야에서 염기서열을 표현할 때 많이 사용되나, 길이가 다른 서열들을 표현하기엔 어려움




Sequence Encoding (one-hot encoding)

```
def onehot_encoder(seqs, letters="ACGT"):
    enc = defaultdict(lambda: np.array([1/len(letters)]*len(letters))) # default. [0.25,0.25,0.25,0.25]
    for i,l in enumerate(letters):
        f = np.zeros(len(letters))
        f[i] = 1
        enc[l] = f

    onehot_features = []
    for seq in seqs:
        feature = []
        for c in seq.upper():
            feature.append(enc[c])
        onehot_features.append(feature)
    onehot_features = np.array(onehot_features)
    print("One-hot encoding finished, with feature shape {}".format(onehot_features.shape))
    return onehot_features

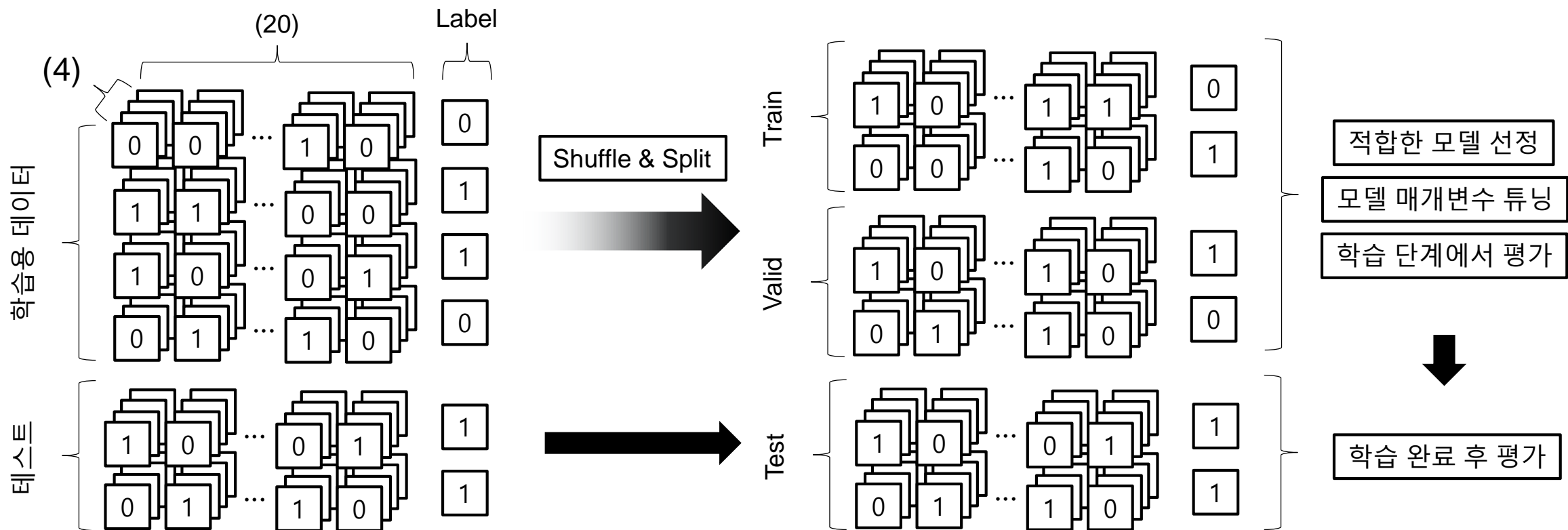
X = onehot_encoder(train_seqs)
test_x = onehot_encoder(test_seqs)
```



One-hot encoding finished, with feature shape (255508, 20, 4)
One-hot encoding finished, with feature shape (128012, 20, 4)

Split Training & Validation datasets

- 테스트 데이터 셋은 모델 학습 후 최종 평가 시 사용
- 학습 데이터를 학습용(training) 및 평가용(validation)으로 분리하여 overfitting 방지 및 hyper-parameter tuning에 사용
- Scikit-learn 패키지의 train_test_split ([ref-link](#))으로 쉽게 코딩 가능



Split Training & Validation datasets

- 테스트 데이터 셋은 모델 학습 후 최종 평가 시 사용
- 학습 데이터를 학습용(training) 및 평가용(validation)으로 분리하여 overfitting 방지 및 hyper-parameter tuning에 사용
- Scikit-learn 패키지의 train_test_split ([ref-link](#))으로 쉽게 코딩 가능

```
from sklearn.model_selection import train_test_split

X, y = np.array(X), np.array(y) # convert python-list to numpy-array

train_x, valid_x, train_y, valid_y = train_test_split(X, y,
                                                    test_size=0.33, # ratio of size of train-test set splitting
                                                    random_state=42) # random-seed

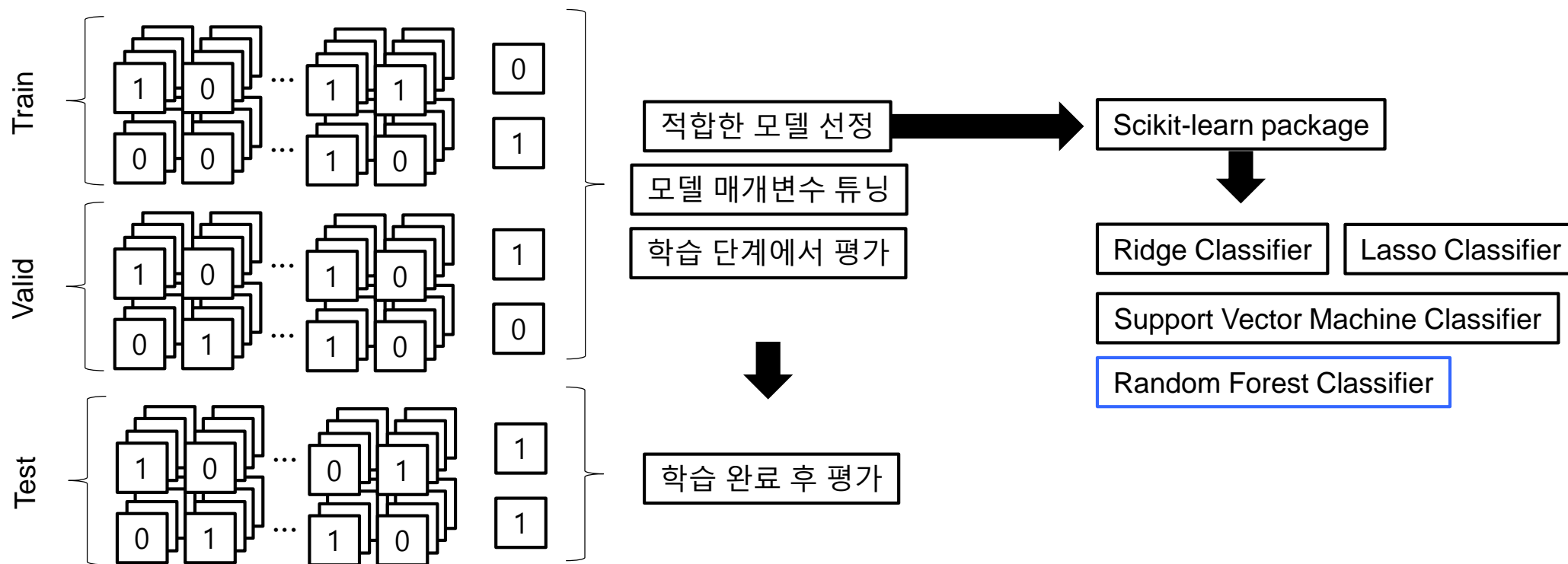
print("Train set | x: {}, y: {} \nValid set | x: {}, y: {}".format(train_x.shape,
                                                                    train_y.shape,
                                                                    valid_x.shape,
                                                                    valid_y.shape))
```



```
Train set | x: (171190, 20, 4), y: (171190,)
Valid set | x: (84318, 20, 4), y: (84318,)
```

Binary Classifier

- 기본적인 binary classifier 학습 준비 과정 및 평가 단계를 모두 숙지하기 위해 해당 실습에서는 기본적인 머신러닝 분류 모델만 사용 (ref. scikit-learn package)




Initialize a Random Forest Classifier

- 정형 데이터에서 대체로 robust하게 동작하는 Random Forest 분류 모델을 사용

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(class_weight = "balanced", # just for class imbalance
                              n_estimators = 200, # number of decision trees (hyper-param)
                              max_depth = 10) # maximum tree depth (hyper-param)

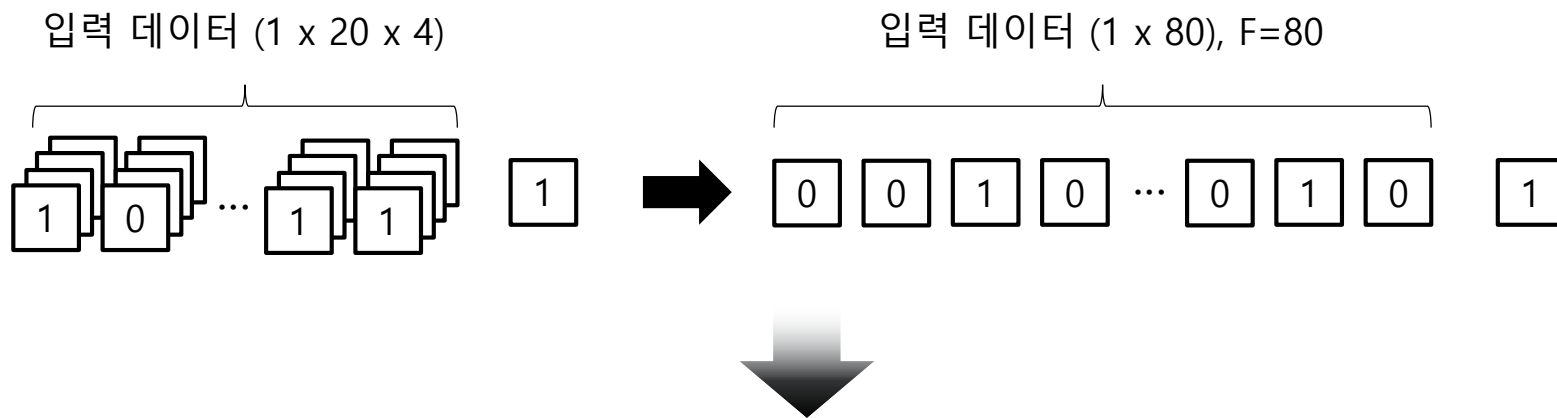
model # print model description
```



```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight='balanced',
                        criterion='gini', max_depth=10, max_features='auto',
                        max_leaf_nodes=None, max_samples=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=200,
                        n_jobs=None, oob_score=False, random_state=None,
                        verbose=0, warm_start=False)
```

Train the Random Forest Classifier

- Random Forest 모델 입력으로 사용하기 위해 입력 데이터 행렬 형식 변환 ($N \times F$) where $F = \text{\#features}$



```
train_x = train_x.reshape(-1, 80)
valid_x = valid_x.reshape(-1, 80)
test_x = test_x.reshape(-1, 80)
print(train_x.shape, valid_x.shape, test_x.shape)
```

(171190, 80) (84318, 80) (128012, 80)

Train the Random Forest Classifier

- Random Forest 모델 입력으로 사용하기 위해 입력 데이터 행렬 형식 변환 ($N \times F$) where $F = \text{\#features}$
- 데이터 수가 많아서 일부만 사용

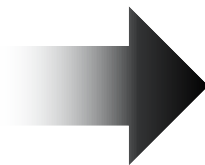


```
train_x = train_x.reshape(-1, 80)[:5000]
valid_x = valid_x.reshape(-1, 80)[:1000]
test_x = test_x.reshape(-1, 80)[:100]

train_y = np.array(train_y[:5000])
valid_y = np.array(valid_y[:1000])
test_y = np.array(test_y[:100])

print(train_x.shape, valid_x.shape, test_x.shape)
print(train_y.shape, valid_y.shape, test_y.shape)

(5000, 80) (1000, 80) (100, 80)
(5000,) (1000,) (100,)
```



```
model.fit(train_x, train_y)

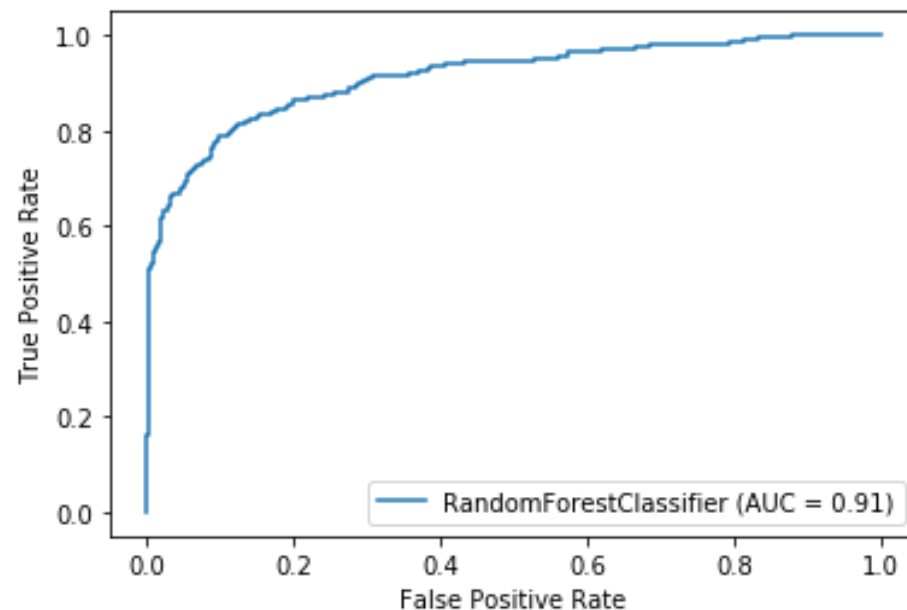
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight='balanced',
                        criterion='gini', max_depth=10, max_features='auto',
                        max_leaf_nodes=None, max_samples=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=200,
                        n_jobs=None, oob_score=False, random_state=None,
                        verbose=0, warm_start=False)
```


Model Evaluation

- 학습된 이진 분류 모델의 성능을 평가
- Receiver operating characteristic (ROC) Area Under the Curve (AUC) 값이 대체로 많이 사용됨
- ROC-AUC, AUC로 불리는 이 값은 0 ~ 1 사이 값을 가지며 1에 가까울수록 좋다
- 참고자료 : https://en.wikipedia.org/wiki/Receiver_operating_characteristic
- Scikit-learn 패키지를 사용하면 plot 까지 바로 제공하는 함수를 사용 가능

```
import matplotlib.pyplot as plt
from sklearn import datasets, metrics, model_selection

metrics.plot_roc_curve(model, valid_x, valid_y)
plt.show()
```



Model Evaluation

- Scikit-learn / classification_report / 학습된 (이진) 분류 모델의 성능을 종합적으로 평가
- Positive class 뿐인 테스트 데이터 셋을 평가할 때 ROC-AUC를 사용할 수 없음 (직접 실행해보는 것을 추천)
- Precision, Recall, F1-score 같은 기본적인 분류 모델 평가 수치를 간편하게 출력해주는 함수 (classification_report)

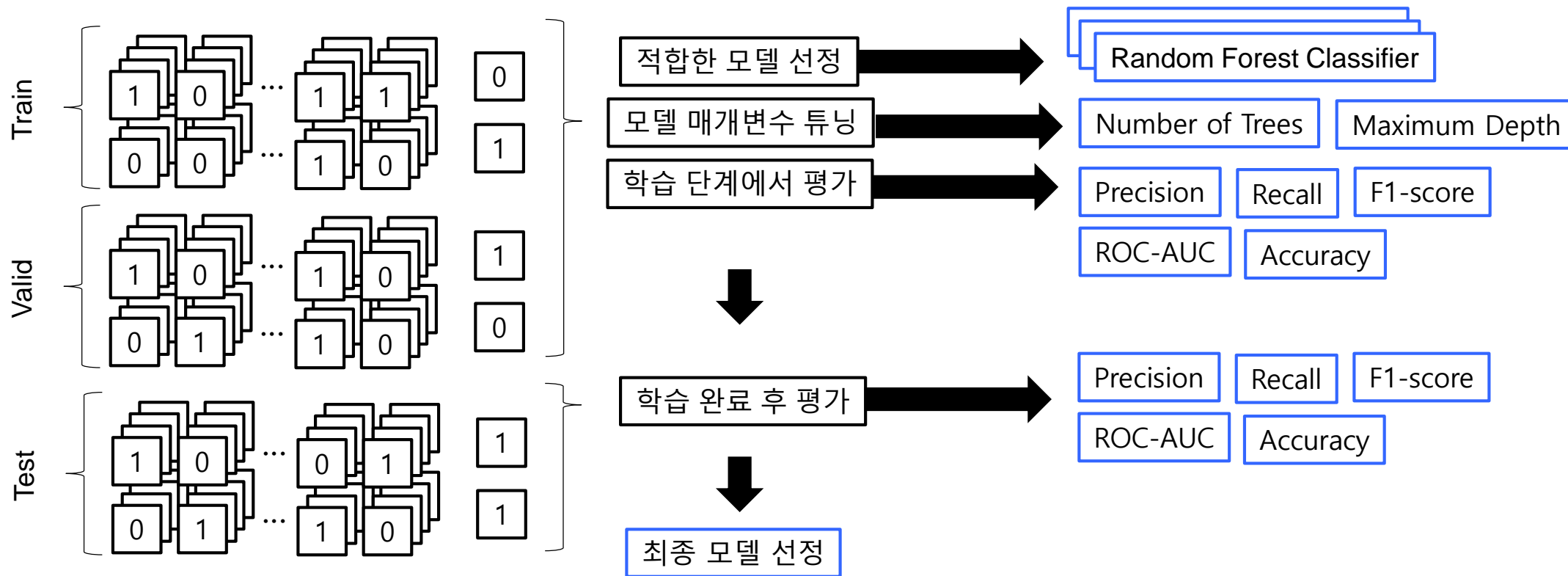
```
valid_pred_y = model.predict(valid_x)
print(metrics.classification_report(valid_y, valid_pred_y))
```



	precision	recall	f1-score	support
0	0.82	0.87	0.84	2567
1	0.85	0.80	0.83	2433
accuracy			0.84	5000
macro avg	0.84	0.84	0.84	5000
weighted avg	0.84	0.84	0.84	5000

Model Selection

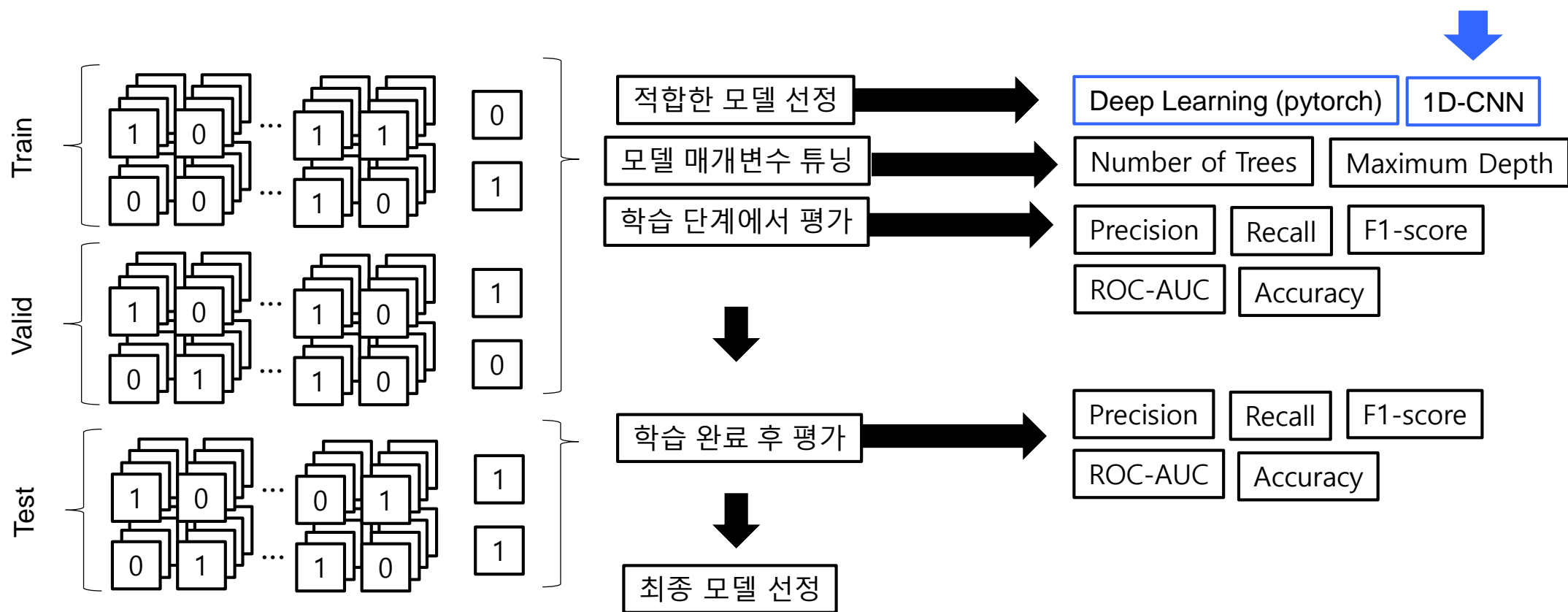
- 다양한 머신러닝 모델을 평가 및 비교
- 각 모델의 매개변수를 변경해보며 성능을 평가 및 비교
- 학습된 모델을 테스트 데이터로 성능 평가 및 비교



3. 후속 연구 소개 – Deep Learning & Motif analysis (간단히)

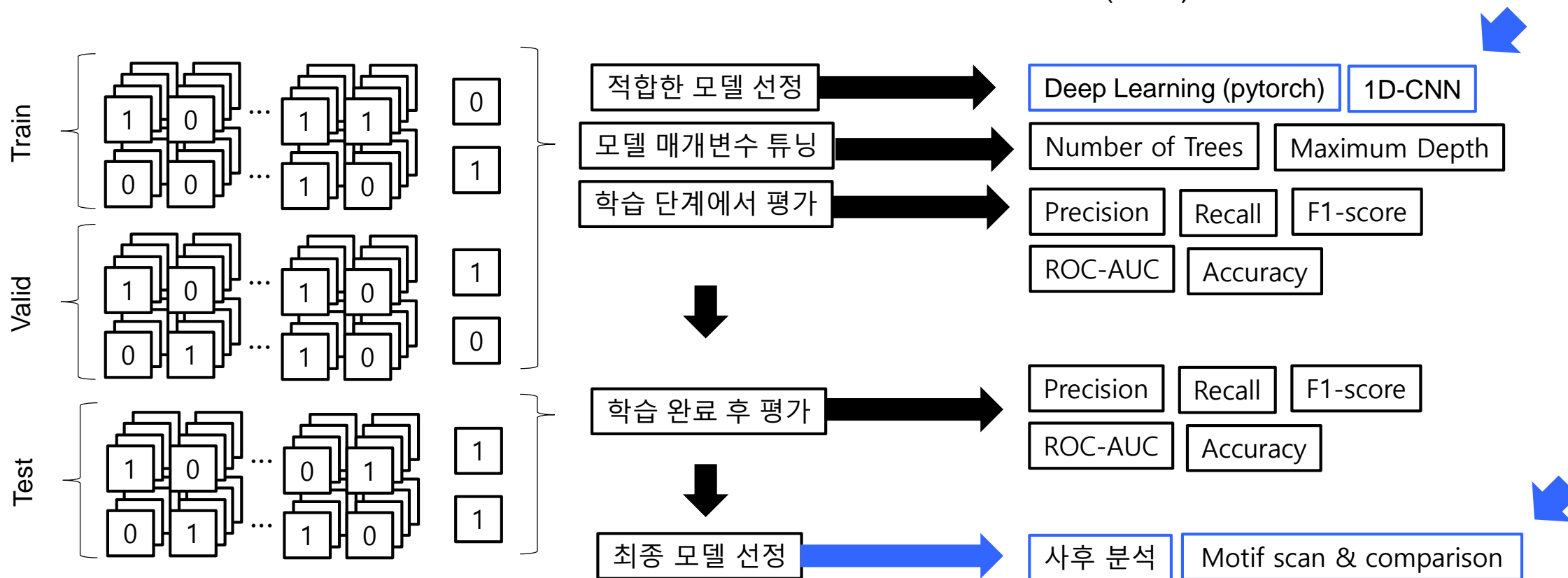
1D-CNN (Convolutional Neural Network)

- 모델 평가 및 선별 방식은 크게 변하지 않음
- 학습 모델을 딥 러닝 모델 사용



1D-CNN (Convolutional Neural Network) + Motif scan

- 모델 평가 및 선별 방식은 크게 변하지 않음
- 학습 모델을 딥 러닝 모델 사용
- 1D-Convolutional Neural Network를 사용하여 motif 를 찾는 것 까지 진행 (예정)



Thank you

Q & A



DeepBind Example full script ([link](#))