

Large-Scale Deep Learning on Spark

DEVIEW
2015

이 주 열

LG CNS 정보기술연구원



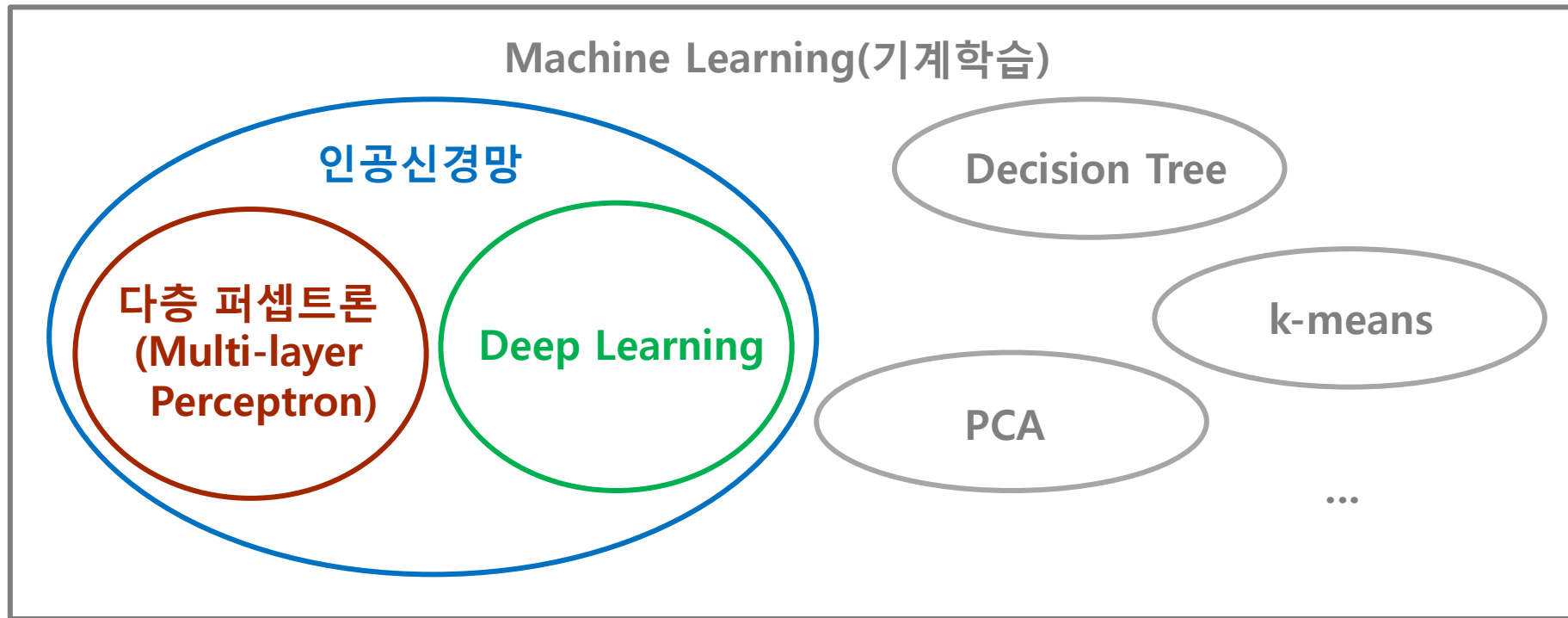
1. Deep Learning Overview
2. Large-Scale Deep Learning
3. What's Next?

1. Deep Learning Overview

1.1 Deep Learning 배경

DEVVIEW
2015

Machine Learning과 Deep Learning

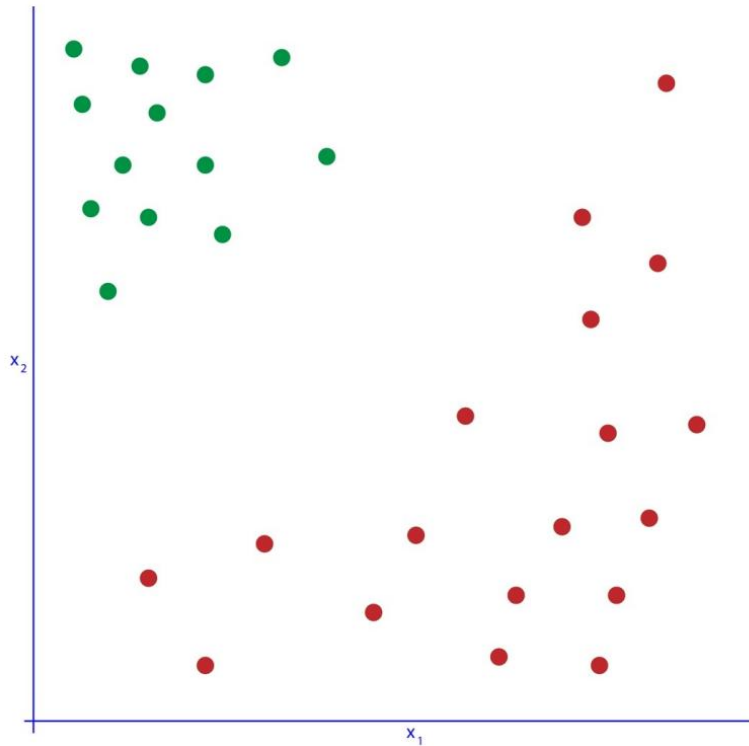


1.1 Deep Learning 배경

DEVIEW
2015

Machine Learning

- 주어진 데이터로 컴퓨터 훈련 (Learning)
- 데이터의 숨어있는 규칙을 추론 (Inference)
- 이를 토대로 새로운 데이터 예측

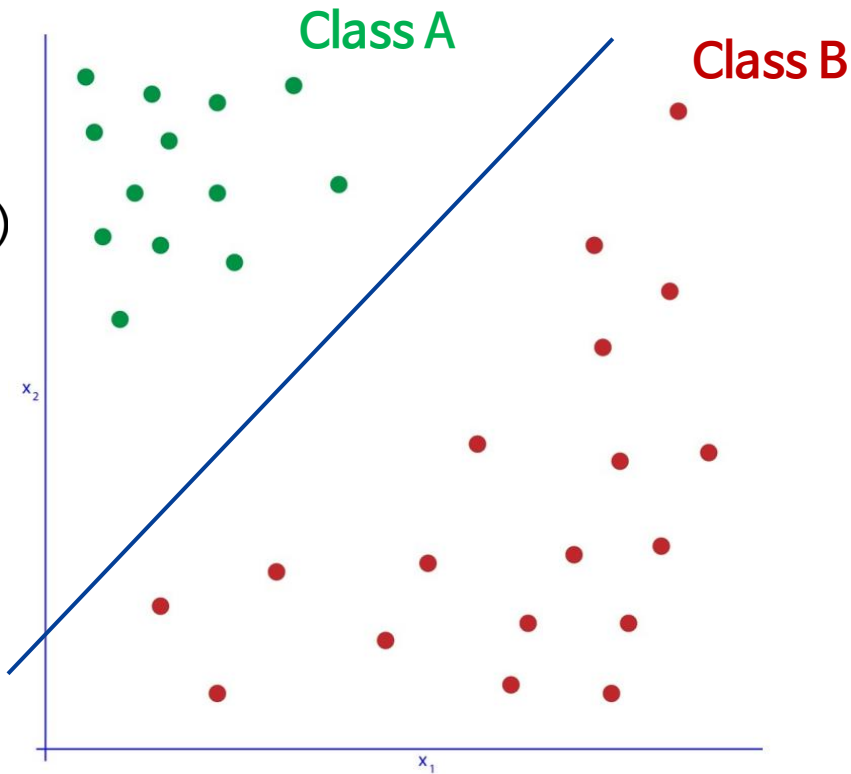


1.1 Deep Learning 배경

DEVIEW
2015

Machine Learning

- 주어진 데이터로 컴퓨터 훈련 (Learning)
- 데이터의 숨어있는 규칙을 추론 (Inference)
- 이를 토대로 새로운 데이터 예측

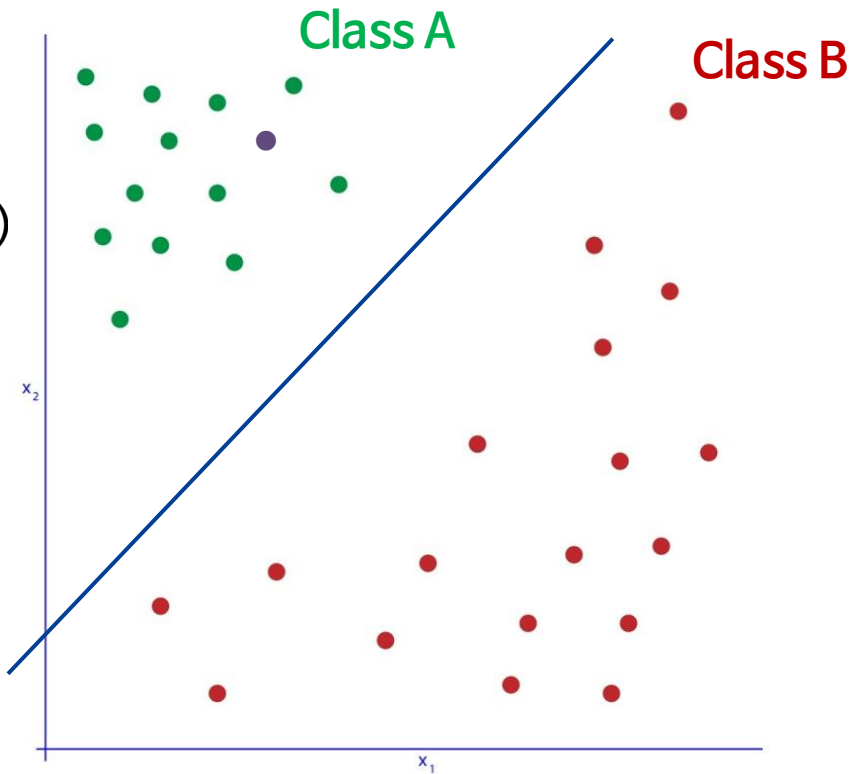


1.1 Deep Learning 배경

DEVIEW
2015

Machine Learning

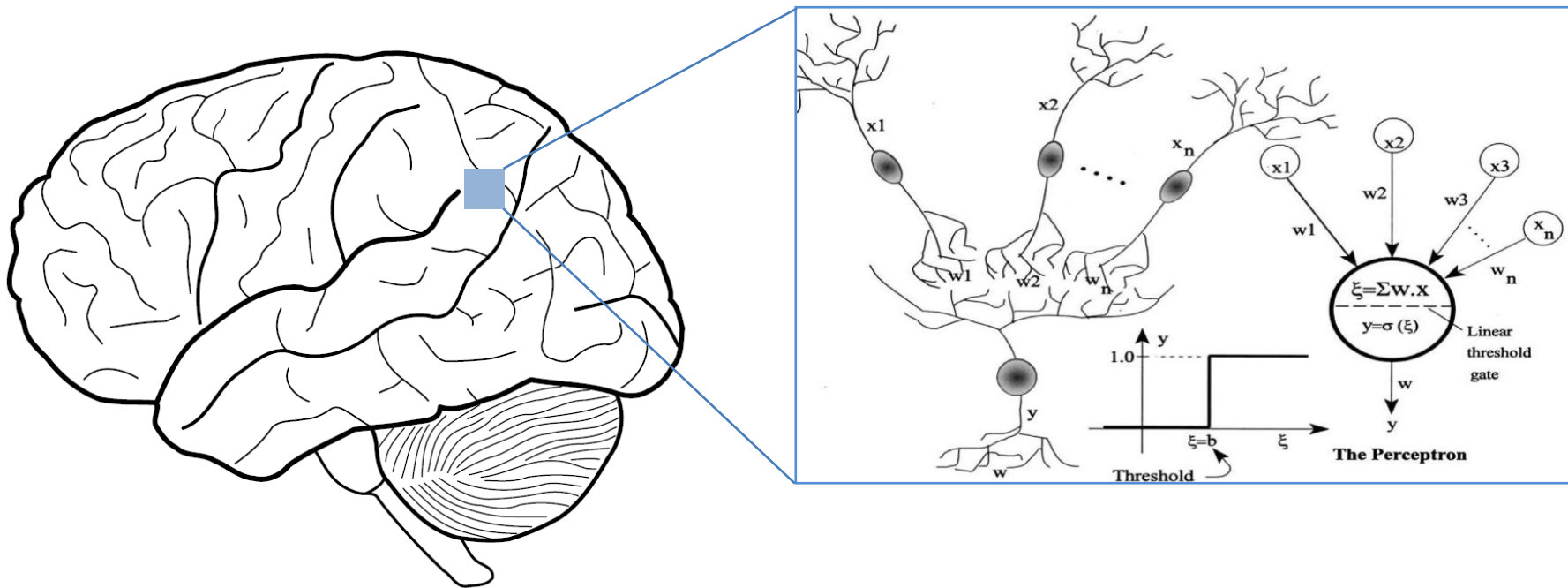
- 주어진 데이터로 컴퓨터 훈련 (Learning)
- 데이터의 숨어있는 규칙을 추론 (Inference)
- 이를 토대로 새로운 데이터 예측



1.1 Deep Learning 배경

DEVIEW
2015

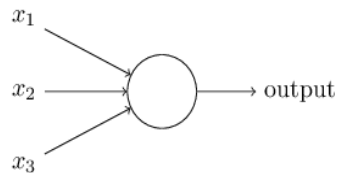
인공신경망 (Artificial Neural Network, ANN)



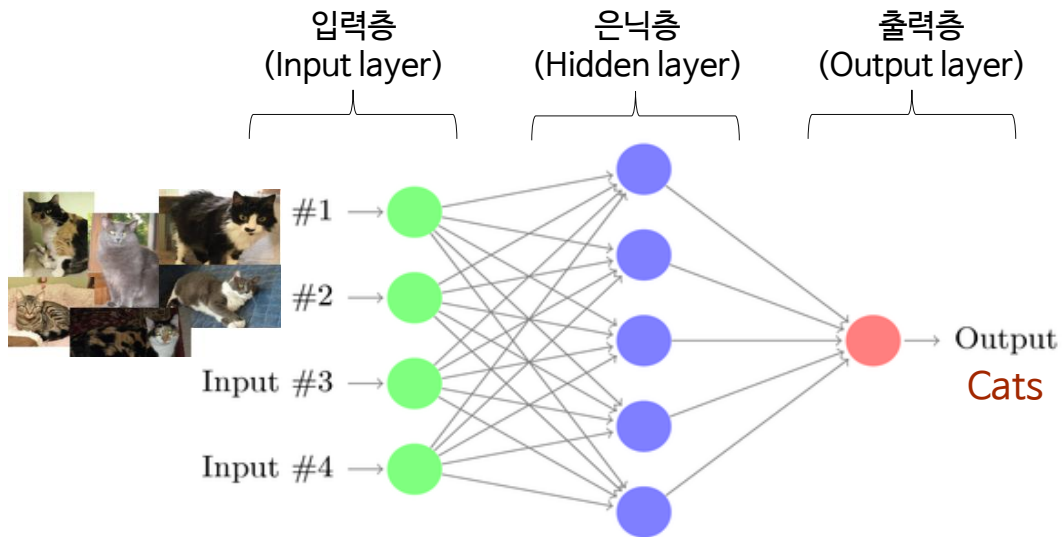
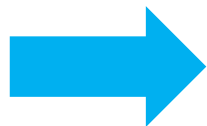
1.1 Deep Learning 배경

DEVIEW
2015

인공신경망 (Artificial Neural Network, ANN)



[단층 퍼셉트론]




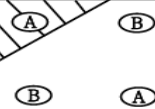

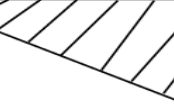

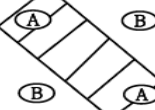



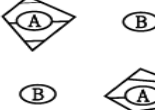
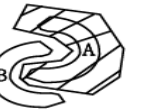
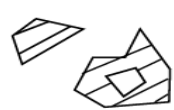
[다층 퍼셉트론]

1.1 Deep Learning 배경

DEVIEW
2015

인공신경망 (Artificial Neural Network, ANN)

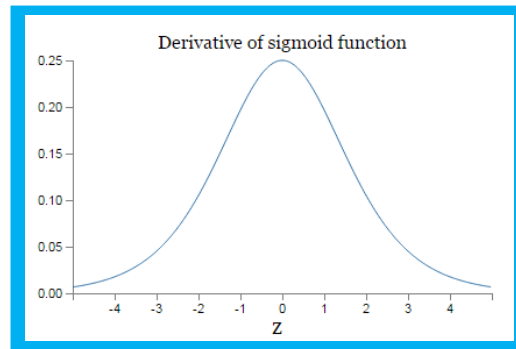
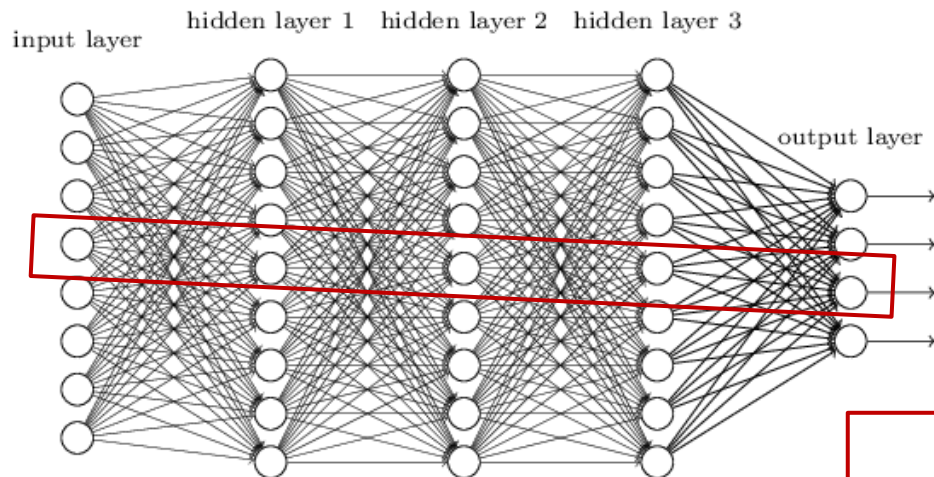
- 1943년, McCulloch과 Pitts : 최초 신경망
- 1958년, Rosenblatt : 단층 퍼셉트론
- 1969년, Minsky와 Papert : 한계 지적 (XOR)
- 1986년, Rumelhart 등 : 다층 퍼셉트론
- 2006년, Deep Learning

구 조	결정 구역	Exclusive -or	Classes with Meshed Regions	Most General Region Shapes
Single-layer 	Half Plane Bounded by Hyperplane			
Two-layer 	Convex Open or Closed Regions			
Three-layer 	Arbitrary (Complexity limited by Number of Units)			

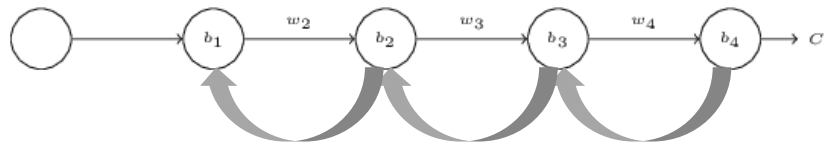
1.1 Deep Learning 배경

DEVIEW
2015

깊은 은닉층의 신경망 학습이 어려운 이유 : Vanishing Gradient!



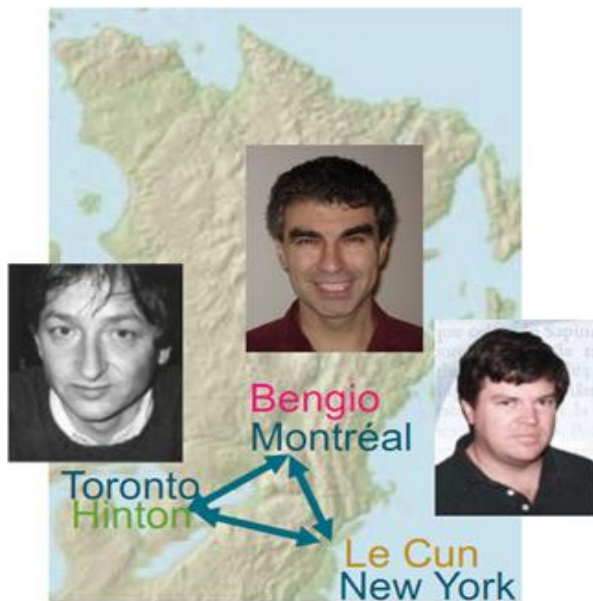
$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



1.1 Deep Learning 배경

DEVIEW
2015

Breakthrough! : 2006년 Deep Learning 핵심 논문 발표



- Hinton, Osindero & Teh
« A Fast Learning Algorithm for Deep Belief Nets », *Neural Computation*, 2006
- Bengio, Lamblin, Popovici, Larochelle
« Greedy Layer-Wise Training of Deep Networks », *NIPS'2006*
- Ranzato, Poultney, Chopra, LeCun
« Efficient Learning of Sparse Representations with an Energy-Based Model », *NIPS'2006*

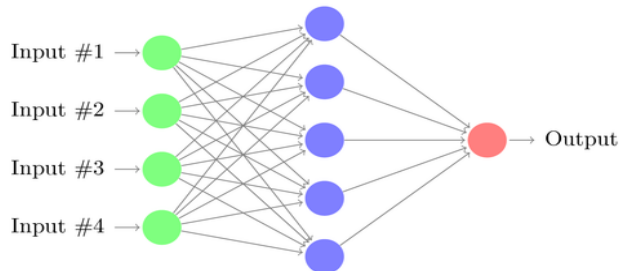
1.2 Deep Learning 정의

DEVIEW
2015

은닉층이 많은 깊은 신경망(Deep Neural Network, DNN)을
학습시키는 알고리즘

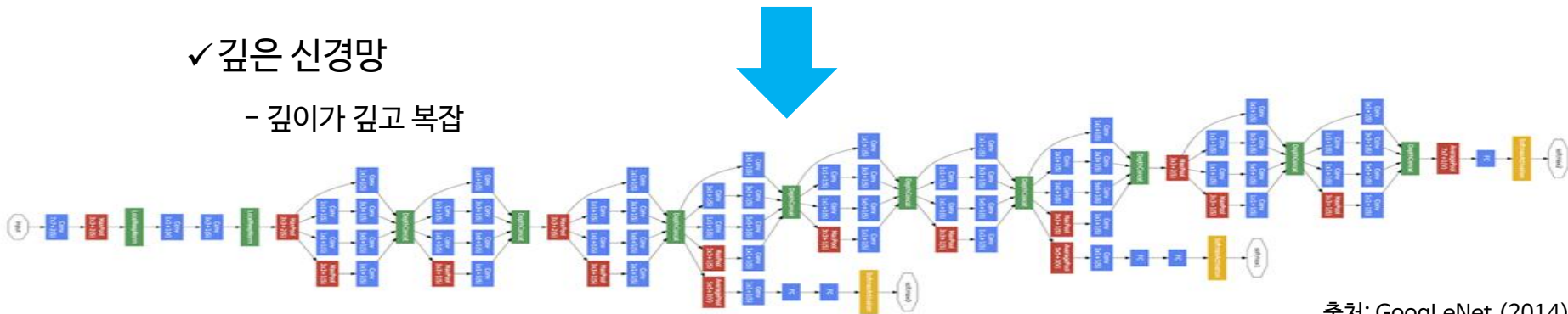
✓ 기존 인공신경망

- 깊이가 얇고 단순



✓ 깊은 신경망

- 깊이가 깊고 복잡



출처: GoogLeNet (2014)

1.2 Deep Learning 정의

DEVIEW
2015

어떻게 가능한가?

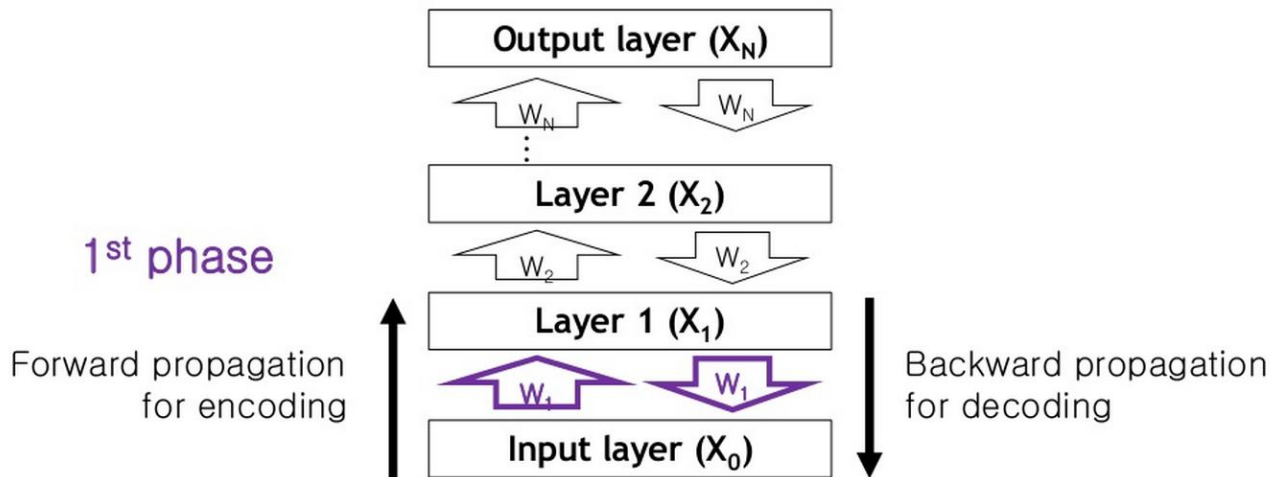
- 사전 학습(Pre-Training) → DBN(Deep Belief Networks)
- 각 Layer의 노드간 연결 수 제한 → CNN(Convolutional Neural Networks)

1.2 Deep Learning 정의

DEVIEW
2015

DBN은 어떻게 작동하나?

- 순방향 : 하위 특징에서 상위 특징 추출 (인코딩)
- 역방향 : 상위 특징으로부터 하위 특징 생성 (디코딩)
- 인코딩 후 디코딩으로 입력 패턴을 최대한 복원하도록 각 Layer 별 학습 → 사전 학습



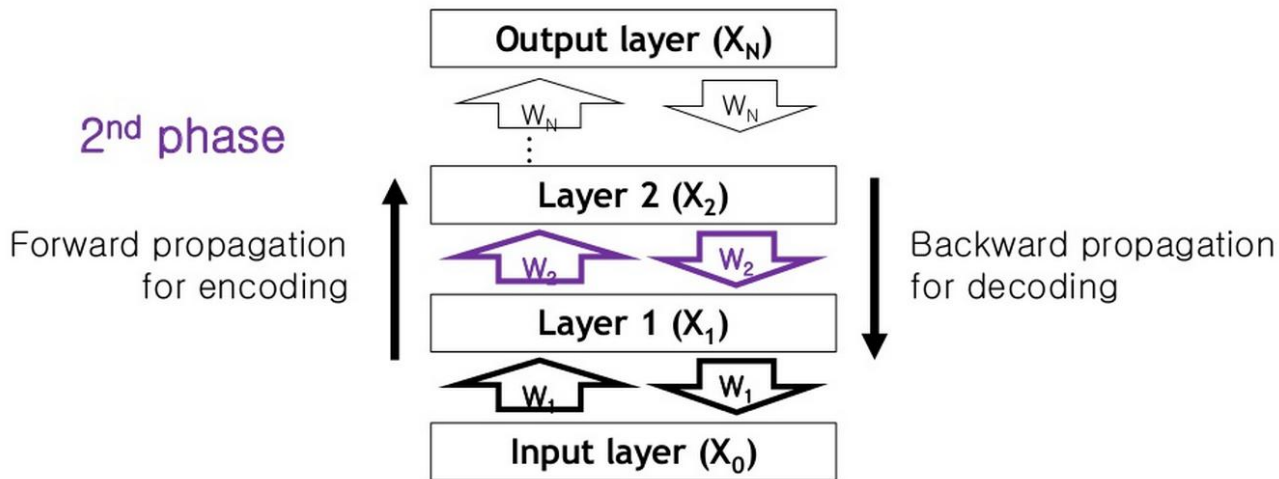
출처: 김인중 (2014), "Deep Learning: 기계학습의 새로운 트렌드"

1.2 Deep Learning 정의

DEVIEW
2015

DBN은 어떻게 작동하나?

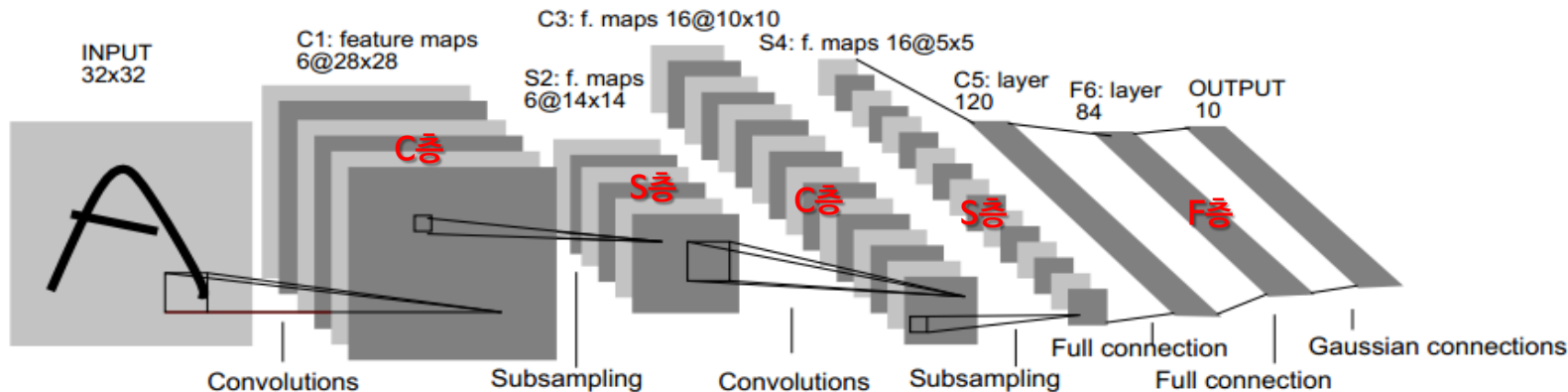
- 순방향 : 하위 특징에서 상위 특징 추출 (인코딩)
- 역방향 : 상위 특징으로부터 하위 특징 생성 (디코딩)
- 인코딩 후 디코딩으로 입력 패턴을 최대한 복원하도록 각 Layer 별 학습 → 사전 학습



1.2 Deep Learning 정의

DEVIEW
2015

CNN은 어떻게 작동하나?



- C, S 층은 제한된 수의 노드(Node)간 연결로 Vanishing gradient problem 완화

- 1) C 층: 일정 크기의 지역에서 특징 추출 (마스크: 에지, 코너, 텍스처 추출)
- 2) S 층: 패턴의 이동, 회전, 크기 변화에 대한 불변성 (다운샘플링으로 해상도 낮춤)
- 3) F 층: 다층 퍼셉트론 신경망으로 인식, 분류

2.

Large-Scale Deep Learning

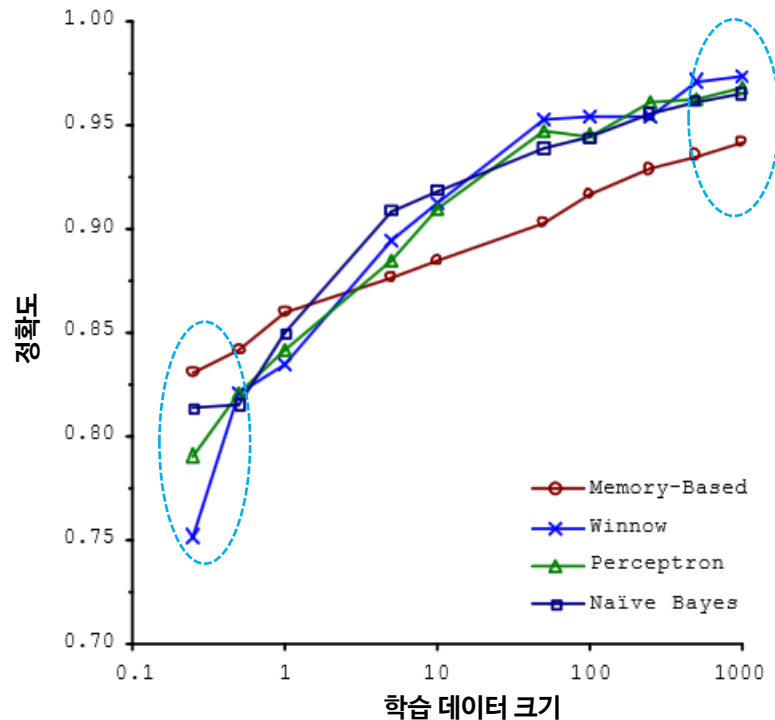
2.1 Why Large-Scale Deep Learning?

DEVIEW
2015

왜 Large-Scale이 필요한가?

1. 학습 데이터 증가

- 알고리즘 보다 많은 학습 데이터가 승리¹⁾
- Big Data 시대 도래
 - '13년 전세계 데이터²⁾: 총 4.4 ZB (4.4조 GB)
 - 매년 40% 이상 증가 예상



1) 출처: Banko and Brill (2011), "Scaling to Very Large Corpora for Natural Language Disambiguation"

2) 출처: IDC (2014), "The Digital Universe of Opportunities"

2.1 Why Large-Scale Deep Learning?

DEVIEW
2015

왜 Large-Scale이 필요한가?

2. 알고리즘 모델의 크기

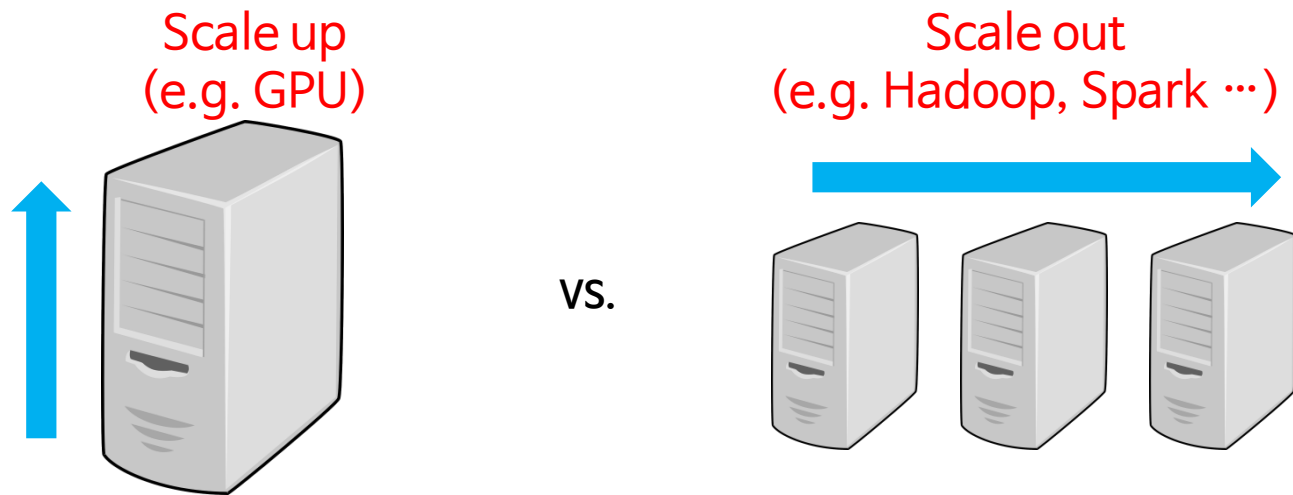
- 점점 깊어지는 DNN → 계산량 ↑, 파라미터 개수 ↑
 - AlexNet('12년) ~ 8 Layer
 - GoogLeNet('14년) ~ 24 Layer (3배 증가)



2.1 Why Large-Scale Deep Learning?

DEVIEW
2015

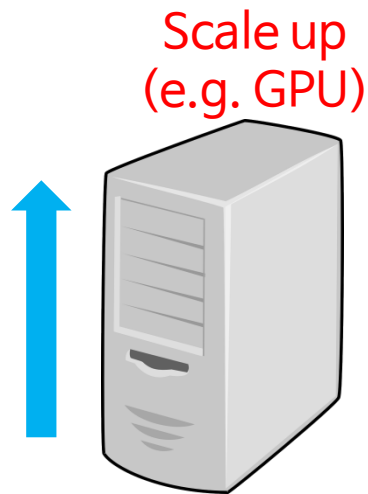
Large-Scale Deep Learning을 위한 선택



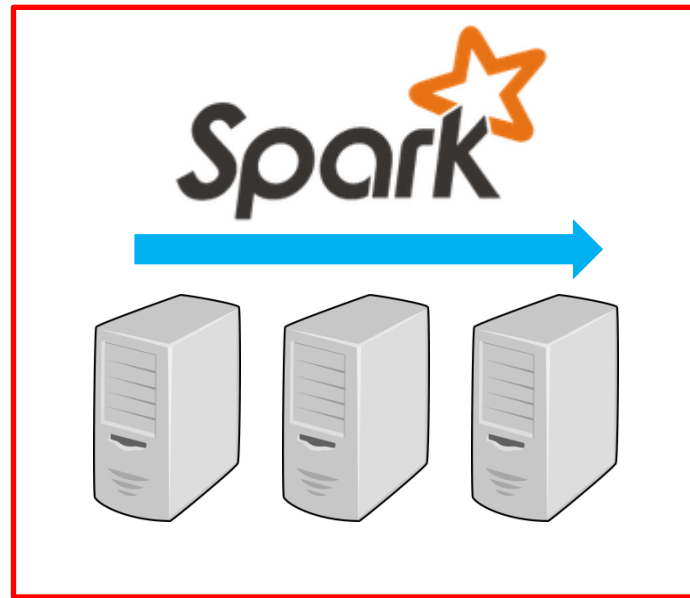
2.1 Why Large-Scale Deep Learning?

DEVIEW
2015

Large-Scale Deep Learning을 위한 선택



vs.



2.2 Why Spark?

DEVIEW
2015

GPU 대신 Spark 고려 이유

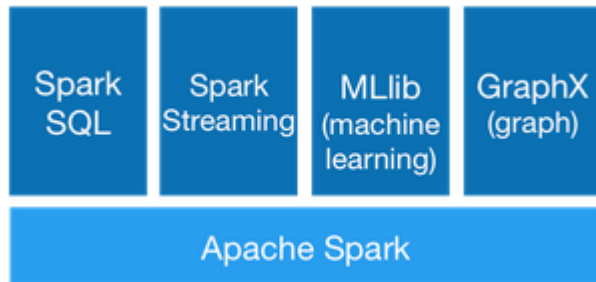
1. Hadoop 등의 기존 대용량 시스템 활용

- 기존 HDFS에 들어있는 데이터 분석
- 시스템 확장 용이
- GPU 시스템 추가 도입 및 데이터 이관 부담 없음

2. 통합된 데이터 처리, 분석 시스템과 환경

- e.g. Streaming → SQL → Machine Learning을 하나의 플랫폼에서!

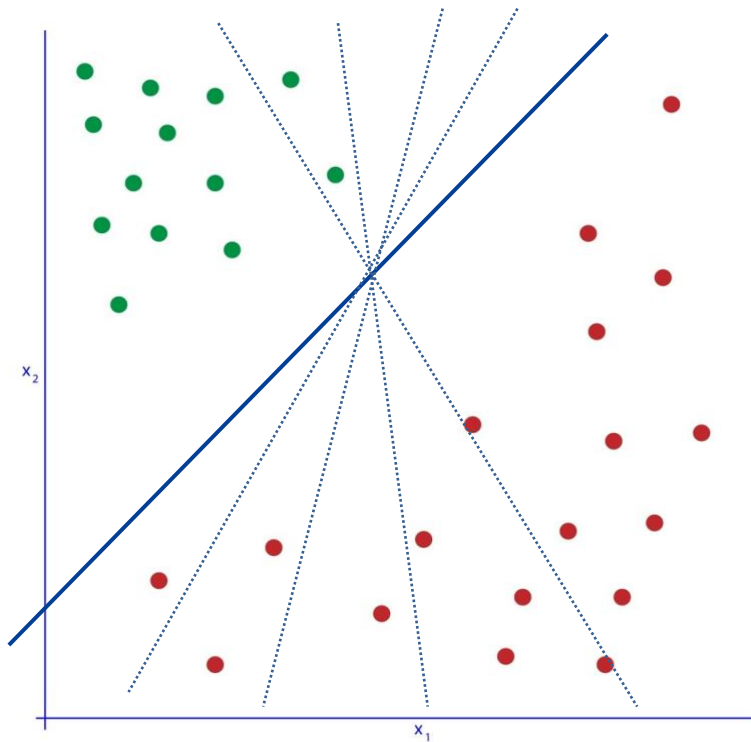
3. GPU 메모리 한계 극복 (GPU 메모리 사이즈 : 24GB, K80)



2.2 Why Spark?

DEVIEW
2015

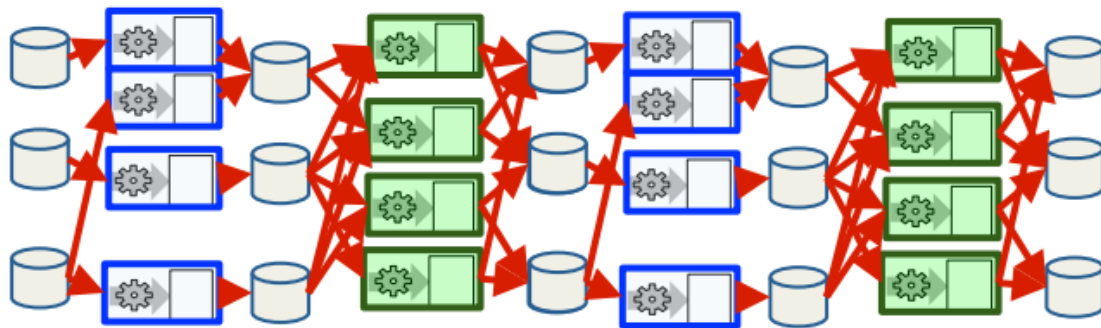
Hadoop MapReduce vs. Iterative 알고리즘 (Machine Learning)



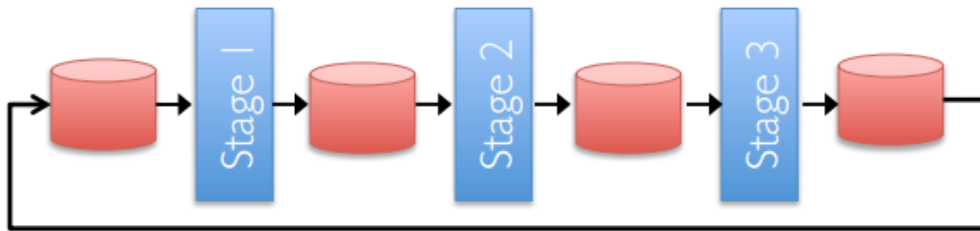
2.2 Why Spark?

DEVIEW
2015

Hadoop MapReduce는 Machine Learning에 효율적이지 않다



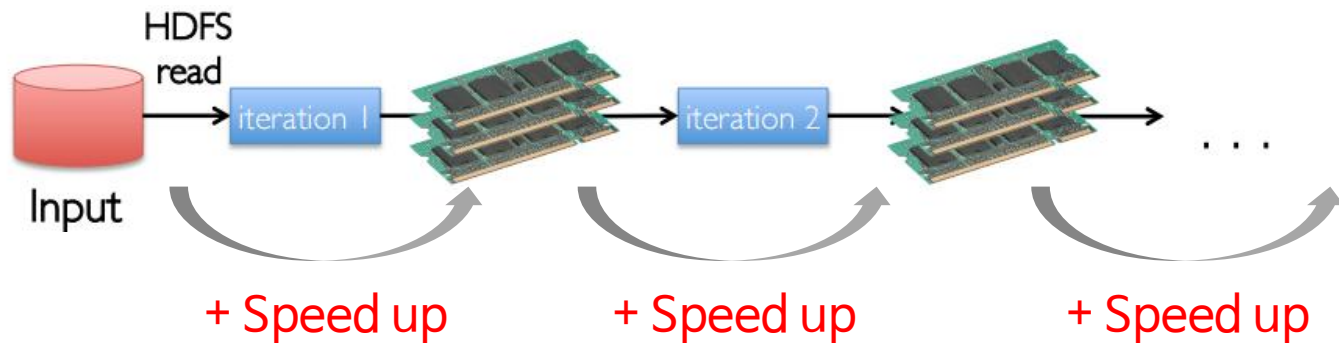
Disk I/O is
very slow!



2.2 Why Spark?

DEVIEW
2015

Machine Learning, Iteration, Spark → 성공적!

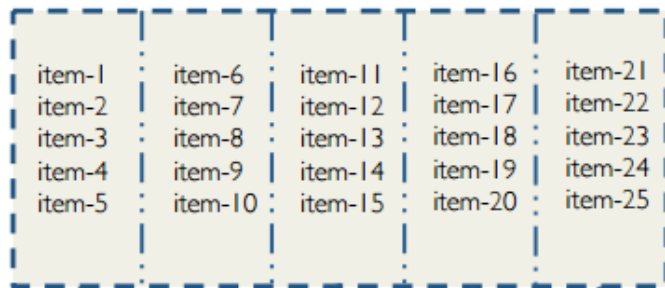


2.3 Deep Learning on Spark

DEVIEW
2015

데이터 병렬 처리로 Large-Scale Deep Learning 구현

RDD split into 5 partitions



more partitions = more parallelism

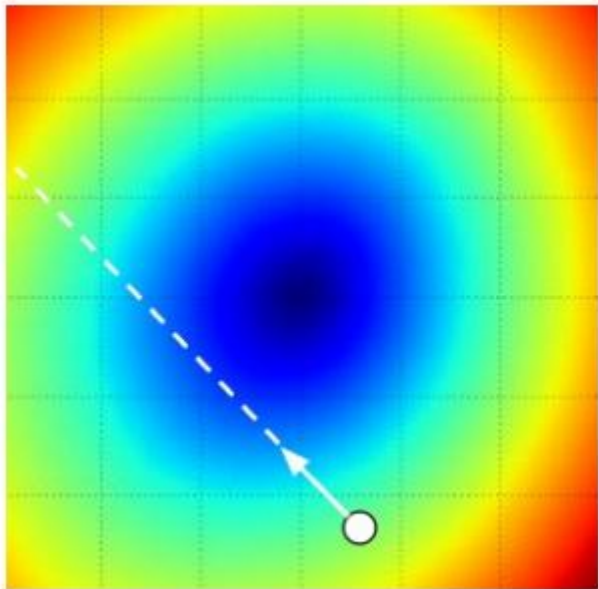


2.3 Deep Learning on Spark

DEVIEW
2015

Learning → Optimization: 정답과의 오차를 최소화하는 파라미터 찾기

Gradient Descent

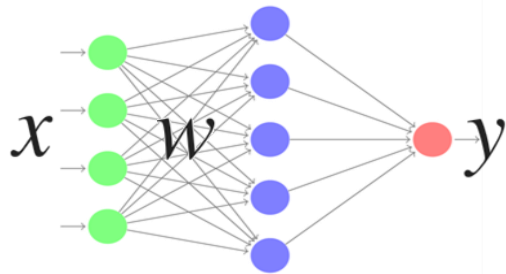


출처 : Fei-Fei (2015)

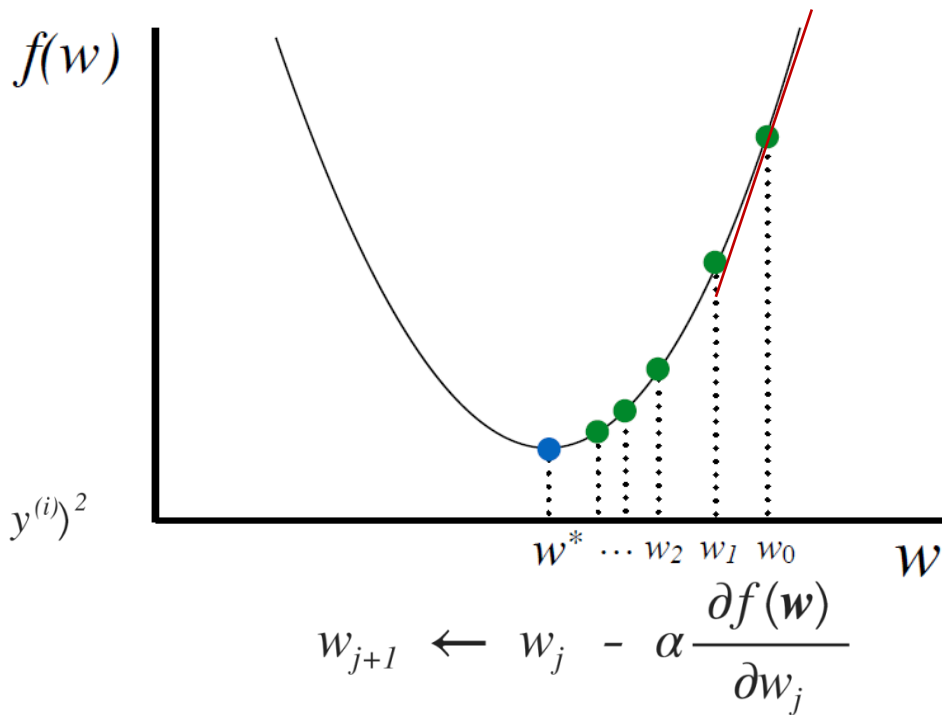
2.3 Deep Learning on Spark

DEVIEW
2015

대표적 Optimization 알고리즘: Gradient Descent



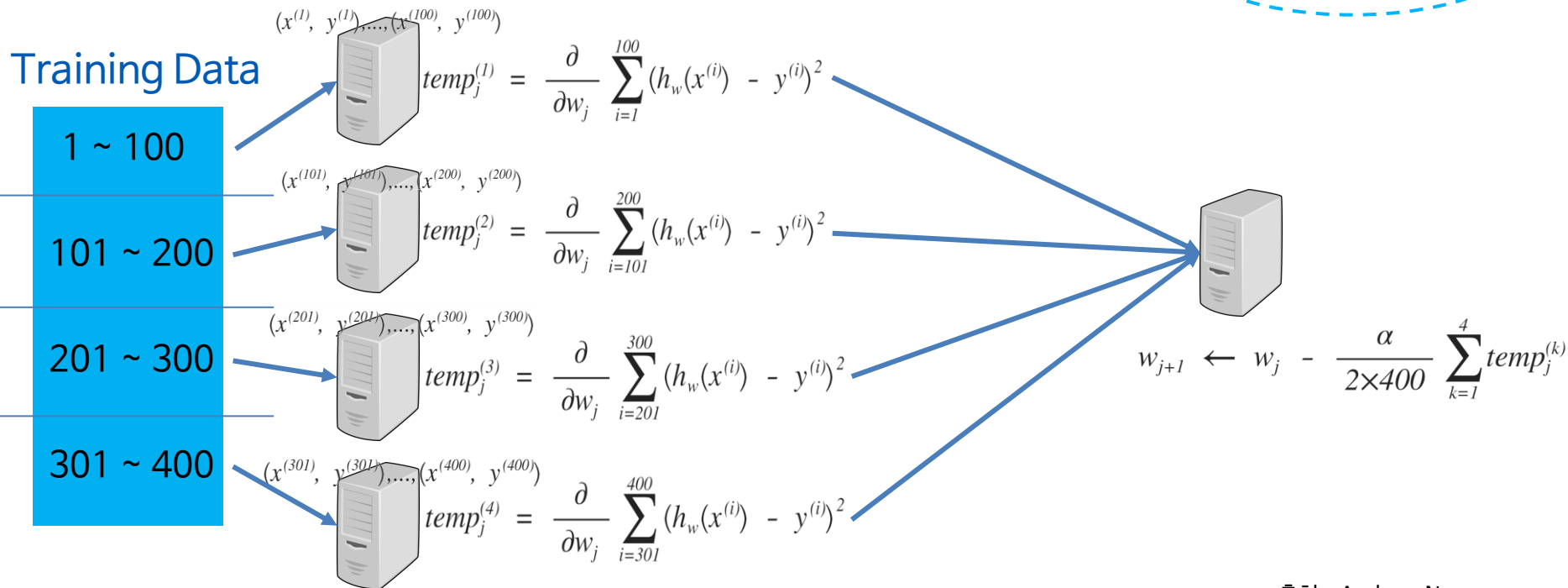
- DNN 모델: $h_w(x)$
- 파라미터: w
- Cost 함수: $f(w) = \frac{1}{2n} \sum_{i=1}^n (h_w(x^{(i)}) - y^{(i)})^2$



2.3 Deep Learning on Spark

DEVIEW
2015

Gradient 분산 병렬 계산 하기: $w_{j+1} \leftarrow w_j - \frac{\alpha}{2 \times 400} \frac{\partial}{\partial w_j} \sum_{i=1}^{400} (h_w(x^{(i)}) - y^{(i)})^2$



2.3 Deep Learning on Spark

DEVIEW
2015

Spark 커뮤니티의 Deep Learning 구현 현황

1. 다층 퍼셉트론(Multi-Layer Perceptron) Pipeline 개발 완료 (ML 패키지, '15년 7월)
 - Spark 1.5.0 릴리즈('15년 9월 9일)
 - JIRA [SPARK-9471]: <https://issues.apache.org/jira/browse/SPARK-9471>
2. Deep Learning 구현 제안
 - JIRA [SPARK-5575]: <https://issues.apache.org/jira/browse/SPARK-5575>
3. Optimization (분산 병렬)
 - Stochastic Gradient Descent, L-BFGS 구현

2.3 Deep Learning on Spark

DEVIEW
2015

Deep Learning/Machine Learning 개발 및 실행 환경

1. BLAS (Basic Linear Algebra Subprograms)

- OpenBLAS 설치 (netlib-java)
- Spark 빌드 시 “netlib-lgpl” Maven Profile 지정

2. Spark 설정 조절

- conf/spark-default.conf
 - spark.executor.memory # 기본값 1GB
 - spark.driver.memory # 기본값 1GB
 - spark.akka.frameSize # 기본값 128, 수천 개 map/reduce 실행 시 ↑

2.3 Deep Learning on Spark

DEVIEW
2015

Deep Belief Networks (DBN) on Spark 구현

GradientDescent.scala

```
def pretrain(
  data: RDD[(SV, SV)],
  batchSize: Int,
  numIteration: Int,
  dbn: DBN,
  fraction: Double,
  learningRate: Double,
  weightCost: Double): DBN = {
  val stackedRBM = dbn.stackedRBM
  val numLayer = stackedRBM.innerRBMs.length
  StackedRBM.train(data.map(_._1), batchSize, numIteration, stackedRBM,
    fraction, learningRate, weightCost, numLayer - 1)
  dbn
}
```

broad cast W

```
def finetune(data: RDD[(SV, SV)],
  batchSize: Int,
  numIteration: Int,
  dbn: DBN,
  fraction: Double,
  learningRate: Double,
  weightCost: Double): DBN = {
  MLP.train(data, batchSize, numIteration, dbn.mlp,
    fraction, learningRate, weightCost)
  dbn
}
```

```
var regVal = updater.compute(
  weights, Vectors.dense(new Array[Double](weights.size)), 0, 1, regParam)._2

for (i <- 1 to numIterations) {
  val bcWeights = data.context.broadcast(weights)
  // Sample a subset (fraction miniBatchFraction) of the total data
  // compute and sum up the subgradients on this subset (this is one map-reduce)
  val (gradientSum, lossSum, miniBatchSize) = data.sample(false, miniBatchFraction, 42 + i)
  .treeAggregate((BDV.zeros[Double](n), 0.0, 0L))(
    seqOp = (c, v) => {
      // c: (grad, loss, count), v: (label, features)
      val l = gradient.compute(v._2, v._1, bcWeights.value, Vectors.fromBreeze(c._1))
      (c._1, c._2 + 1, c._3 + 1)
    },
    combOp = (c1, c2) => {
      // c: (grad, loss, count)
      (c1._1 += c2._1, c1._2 + c2._2, c1._3 + c2._3)
    })

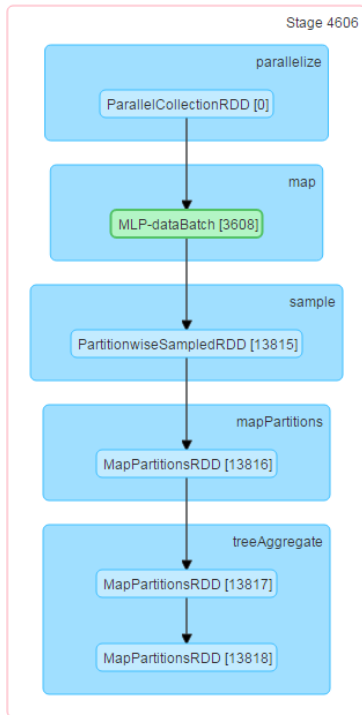
  if (miniBatchSize > 0) {
    /**
     * NOTE(Xinghao): lossSum is computed using the weights from the previous iteration
     * and regVal is the regularization value computed in the previous iteration as well.
     */
    println(f"loss $i ${((lossSum / miniBatchSize + regVal) * 1.6f}")
    stochasticLossHistory.append(lossSum / miniBatchSize + regVal)
    val update = updater.compute(
      weights, Vectors.fromBreeze(gradientSum / miniBatchSize.toDouble), stepSize, i, regParam)
    weights = update._1
    regVal = update._2
  } else {
    logWarning(s"Iteration ($i/$numIterations). The size of sampled batch is zero")
  }
}
```

출처 : <https://github.com/witgo/spark/blob/Sent2vec/mllib/src/main/scala/org/apache/spark/mllib/neuralNetwork/DBN.scala>

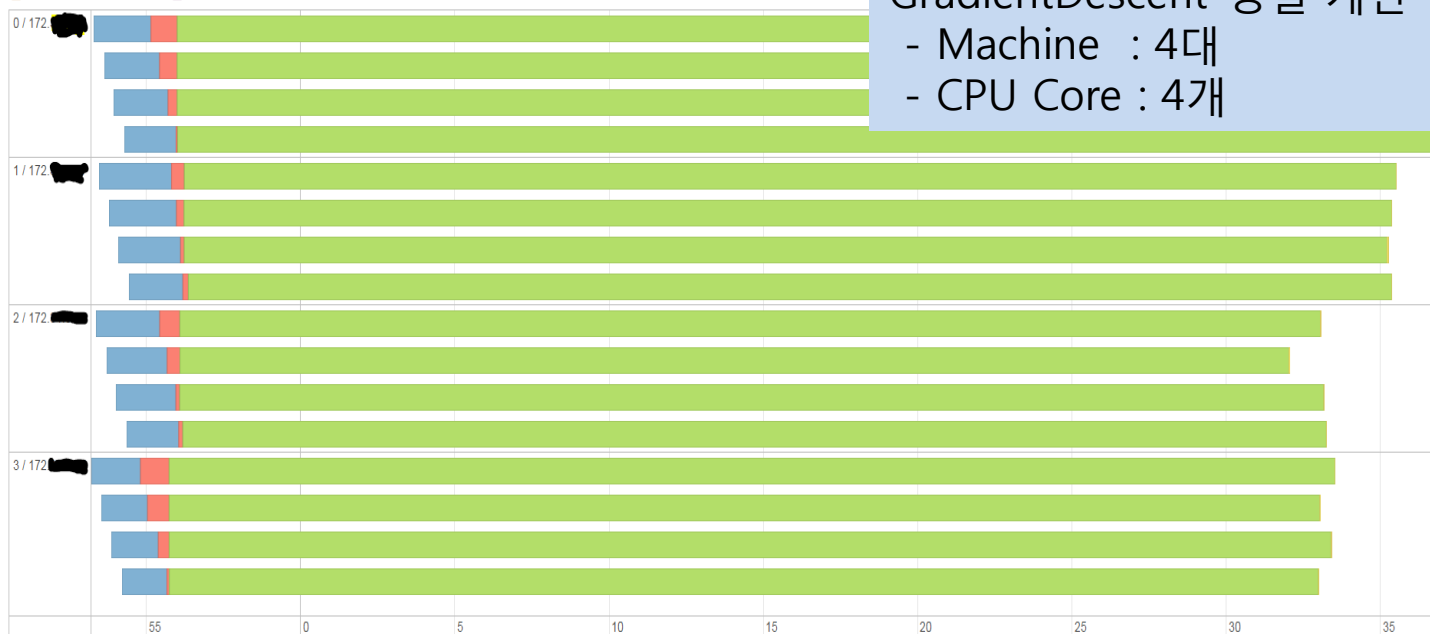
2.3 Deep Learning on Spark

DEVIEW
2015

Deep Belief Networks (DBN) on Spark 구현



■ Scheduler Delay
■ Task Deserialization Time
■ Shuffle Read Time
■ Executor Computing Time
■ Shuffle Write Time
■ Result Serialization Time
■ Getting Result Time



GradientDescent 병렬 계산
- Machine : 4대
- CPU Core : 4개

2.3 Deep Learning on Spark

DEVIEW
2015

Deep Belief Networks (DBN) on Spark 테스트

1. 환경

- AWS: m4.xlarge (Haswell 2.4GHz, vCPU 4, 메모리 16GB, SSD, 750Mbps) 5대
- Spark 1.5.0-SNAPSHOT, JVM 1.7.0, OpenBLAS 0.2.14

2. 대상

- MNIST (학습 데이터: 60,000, 테스트 데이터: 10,000)
 - Hinton 교수 DBN 논문: 1.25% test error¹⁾



1) 출처: Hinton (2006), "A fast learning algorithm for deep belief nets"

2.3 Deep Learning on Spark

DEVIEW
2015

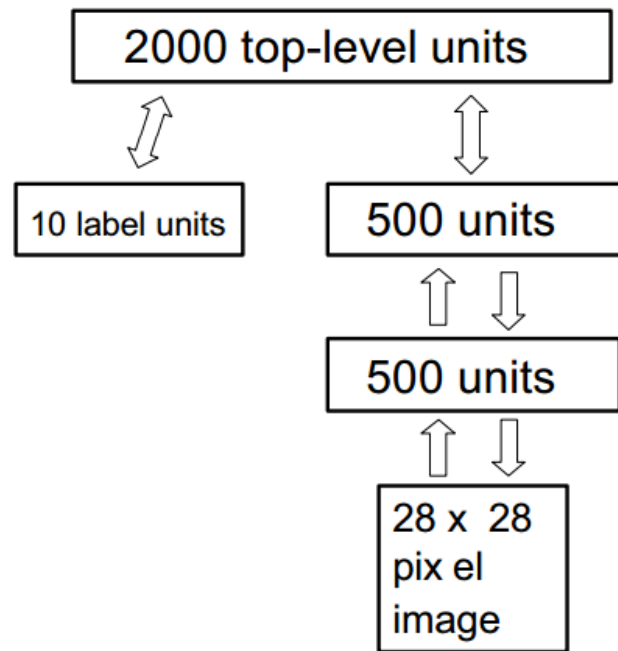
Deep Belief Networks (DBN) on Spark 테스트

3. 학습 모델 : 784 X 500 X 500 X 2000 X 10

- 784 입력 (28 x 28 픽셀 이미지, MNIST)
- 은닉 층 500 X 500 X 2000
- 10 출력 (0~9)
- $\alpha = 0.002$, # of Iter = 2000

4. 결과

- Test error: 1.96%



2.3 Deep Learning on Spark

DEVIEW
2015

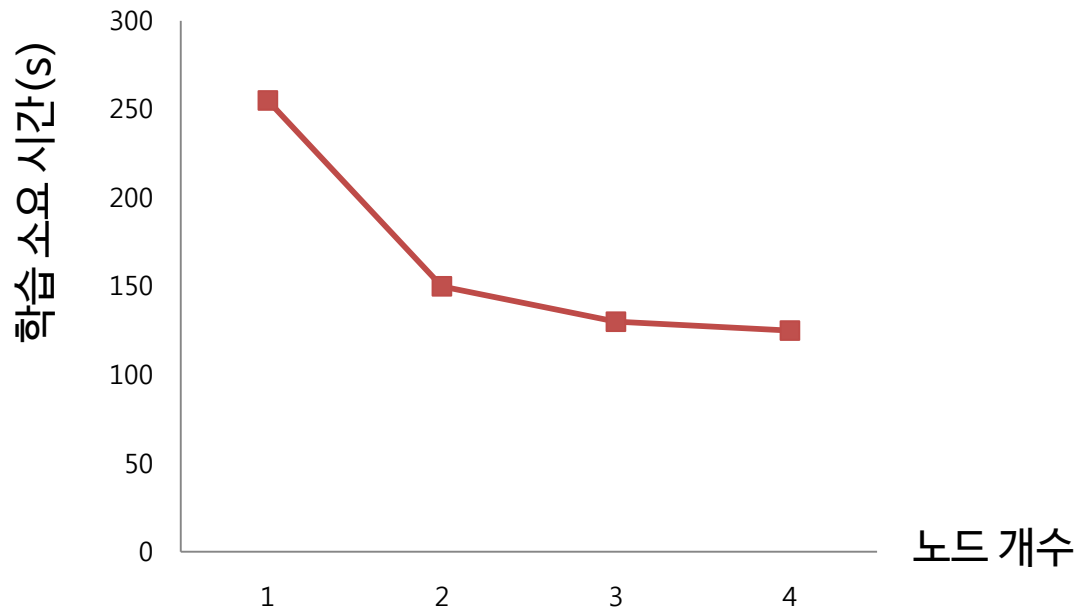
Deep Belief Networks (DBN) on Spark 확장성 테스트

Worker Node 개수	Training 소요 시간 (s)	비고
1	255	<ul style="list-style-type: none">• 총 5대 Node• Master 1, Worker 4• 60000 학습 데이터
2	150	
3	130	
4	125	

2.3 Deep Learning on Spark

DEVIEW
2015

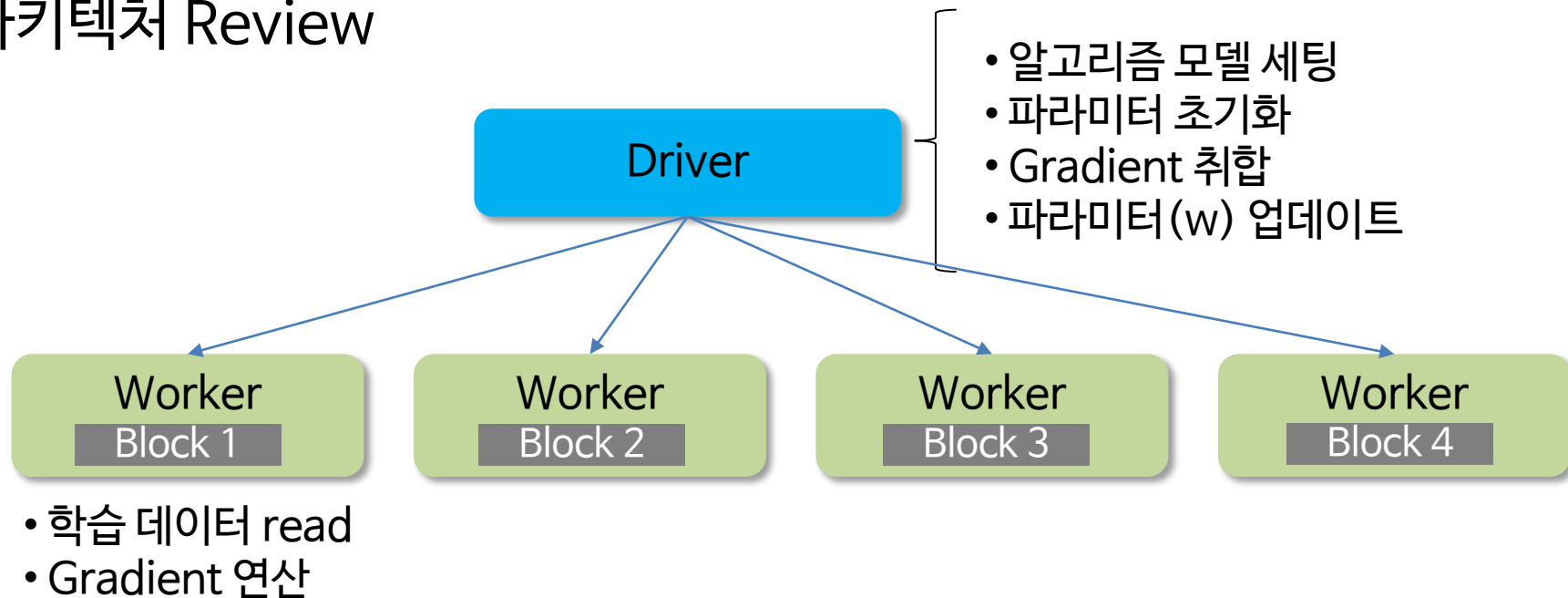
Deep Belief Networks (DBN) on Spark 확장성 테스트



2.3 Deep Learning on Spark

DEVIEW
2015

아키텍처 Review



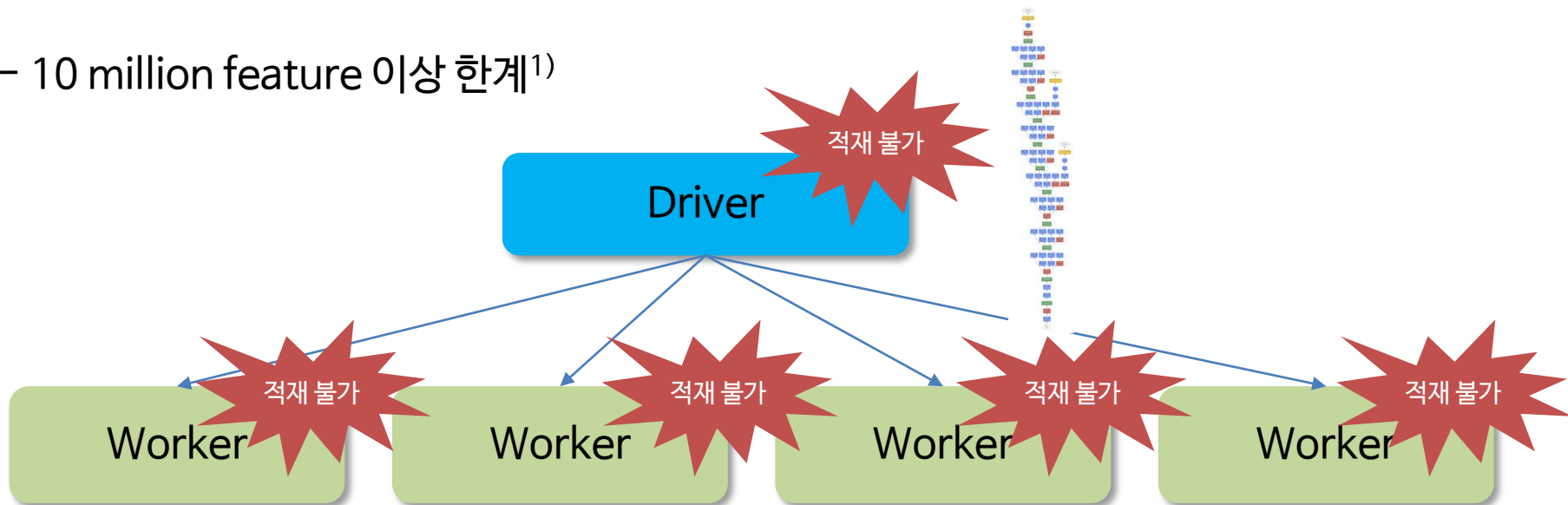
3. What's Next?

3.1 Data Parallelism 한계

DEVIEW
2015

DNN 모델이 단일 머신의 메모리보다 크다면 대안 필요

- 10 million feature 이상 한계¹⁾



1) 출처: <https://issues.apache.org/jira/browse/SPARK-4590>

3.2 Google의 DistBelief

DEVIEW
2015

NIPS¹⁾ 2012 논문 : Large Scale Distributed Deep Networks

1. 수십억 파라미터의 Deep Network를 1만 CPU 코어에서 학습 가능
2. 기존 대비 30대 더 큰 모델
3. ImageNet²⁾ state-of-art 정확도 ('12년)
4. 음성인식 train 속도 GPU 대비 10배
5. Downpour SGD, Sandblaster L-BFGS

Large Scale Distributed Deep Networks

Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen,
Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato,
Andrew Senior, Paul Tucker, Ke Yang, Andrew Y. Ng
{jeff, gcorrado}@google.com
Google Inc., Mountain View, CA

Abstract

Recent work in unsupervised feature learning and deep learning has shown that being able to train large models can dramatically improve performance. In this paper, we consider the problem of training a deep network with billions of parameters using tens of thousands of CPU cores. We have developed a software framework called *DistBelief* that can utilize computing clusters with thousands of machines to train large models. Within this framework, we have developed two algorithms for large-scale distributed training: (i) Downpour SGD, an asynchronous stochastic gradient descent procedure supporting a large number of model replicas, and (ii) Sandblaster, a framework that supports a variety of distributed batch optimization procedures, including a distributed implementation of L-BFGS. Downpour SGD and Sandblaster L-BFGS both increase the scale and speed of deep network training. We have successfully used our system to train a deep network 30x larger than previously reported in the literature, and achieves state-of-the-art performance on ImageNet, a visual object recognition task with 16 million images and 21k categories. We show that these same techniques dramatically accelerate the training of a more modestly-sized deep network for a commercial speech recognition service. Although we focus on and report performance of these methods as applied to training large neural networks, the underlying algorithms are applicable to any gradient-based machine learning algorithm.

주1) NIPS : Neural Information Processing Systems 학회

주2) 1천6백만 이미지, 2만 카테고리, 10억개 파라미터

3.2 Google의 DistBelief

DEVIEW
2015

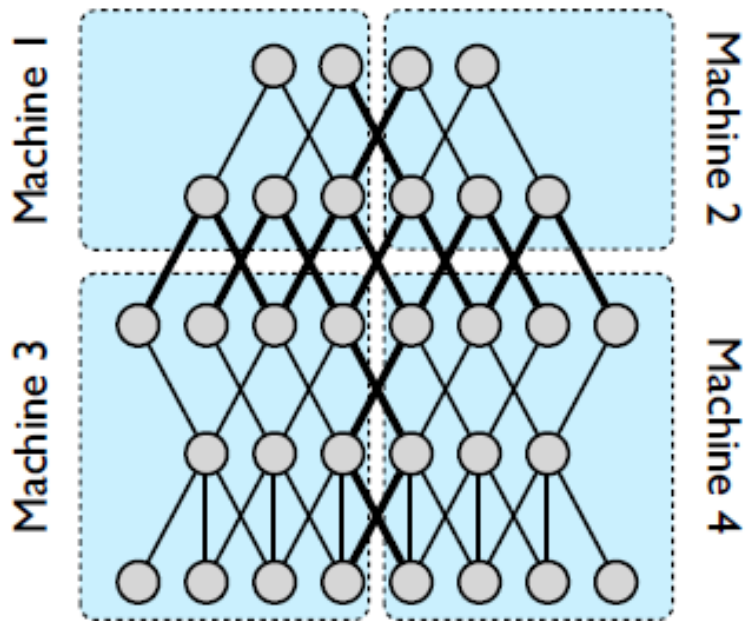
DistBelief의 특징

1. Data & Model Parallelism
2. Distributed Optimization Algorithms
 - Downpour SGD
 - Sandblaster L-BFGS
 - **Parameter server** 개념 (Model replica들이 파라미터 공유)

3.2 Google의 DistBelief

DEVIEW
2015

DistBelief의 Model Parallelism

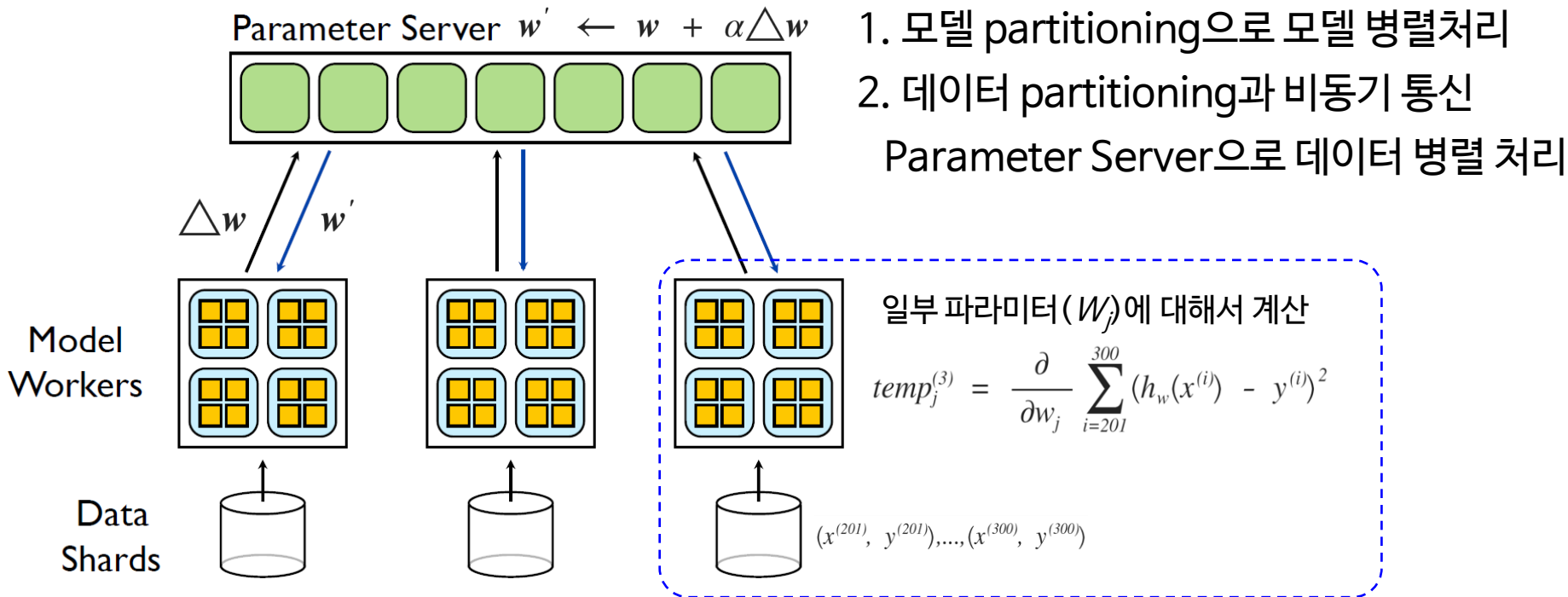


1. DNN 모델을 각 머신에서 분산 처리
2. 모델 파라미터는 각 머신별 나뉘어져 처리

3.2 Google의 DistBelief

DEVIEW
2015

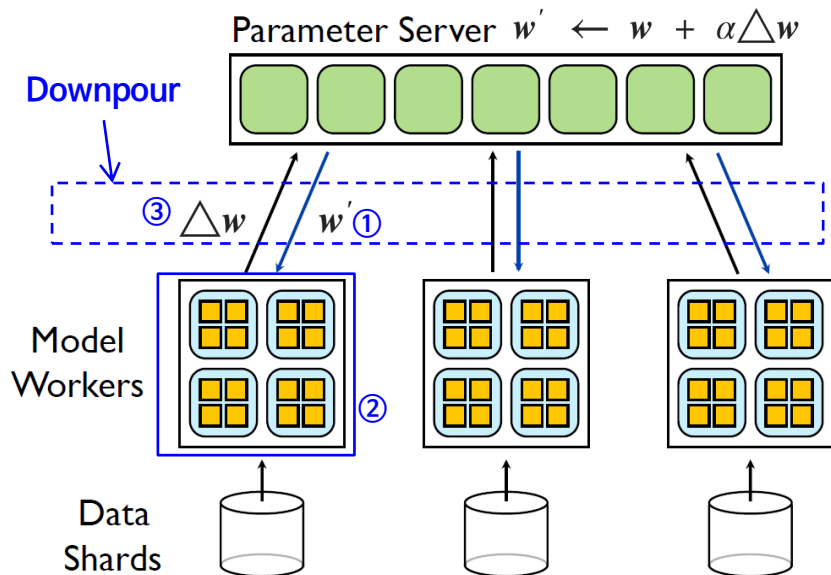
DistBelief의 Model + Data Parallelism



3.2 Google의 DistBelief

DEVIEW
2015

DistBelief의 Downpour SGD



```
main
global parameters, accruedgradients
step ← 0
accruedgradients ← 0
while true
  if (step mod  $n_{fetch}$ ) == 0
    then STARTASYNCHRONOUSLYFETCHINGPARAMETERS(parameters) ①
  data ← GETNEXTMINIBATCH()
  gradient ← COMPUTEGRADIENT(parameters, data) ②
  accruedgradients ← accruedgradients + gradient
  parameters ← parameters -  $\alpha * gradient$ 
  if (step mod  $n_{push}$ ) == 0
    then STARTASYNCHRONOUSLYPUSHINGGRADIENTS(accruedgradients) ③
  step ← step + 1
```

3.3 Parameter Server

DEVIEW
2015

분산 Machine Learning을 위한 다양한 Parameter Server 존재

1. CMU Machine Learning

<http://parameterserver.org>

2. Factorbird (Stanford Univ.)

3. Petuum

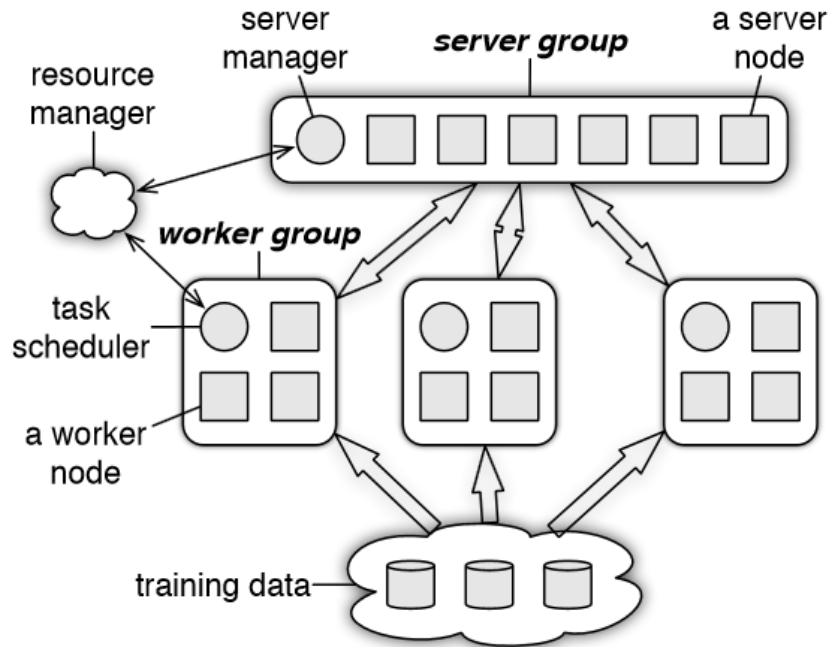
<http://petuum.github.io>

4. Vowpal Wabbit (Yahoo!, MS)

<http://hunch.net/~vw>

5. Paracel

<http://paracel.io/>



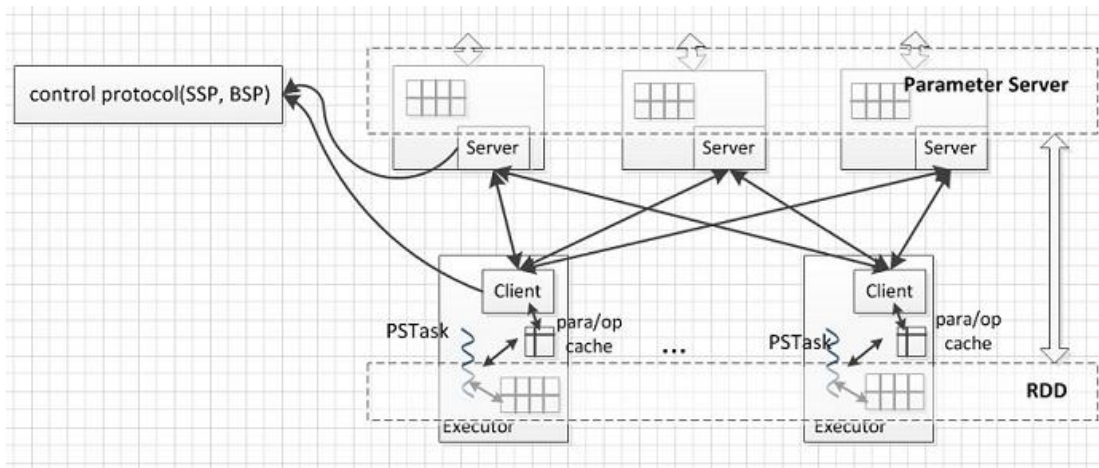
3.4 Parameter Server on Spark

DEVIEW
2015

Spark 2.x를 목표로 Parameter Server 제안

Parameter Server 프로토타입

- JIRA [SPARK-6932]: <https://issues.apache.org/jira/browse/SPARK-6932>
- Parameter Server에 파라미터를 분산 저장하기 위한 테이블 자료구조 필요



3.5 Wrap-Up

광대한 Net에 Spark의 잠재력은 무한!

Deep Learning Tool 선호도¹⁾

Tool	선호도	비고
Pylearn2	18.6%	Python
Theano	16.9%	Python
Caffe	9.9%	C++
Torch	9.1%	Lua
CUDA-ConvNet	5.7%	C++
DeepLearning4J	4.1%	Java
Others	35.8%	-

1) 출처: KDnuggets, <http://www.kdnuggets.com/2015/06/popular-deep-learning-tools.html>



Q&A

Thank You

이주열 (zenithon@gmail.com)