碩士學位論文

하둡 기반 제주 풍속 변화의 분석 플렛폼 구축

濟州大學校 大學院

電算統計學科

李東昱

2014年 12月

하둡 기반 제주 풍속 변화의 분석 플랫폼 구축

指導交綏 李 政 勳

李東昱

이 論文을 理學 碩士學位 論文으로 提出함

2014年 12月

李東昱의 理學 碩士學位 論文을 認准함

審査委員長 박 경 린

委 原 이 정 훈

委 原 송 준 모

濟州大學校 大學院

2014年 12月

Development of an analysis platform for Jeju wind speed based on Apache Hadoop

Dongwook Lee

(Supervised by professor Junghoon Lee)

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Science

2014. 12.

Department of Computer Science and Statistics
GRADUATE SCHOOL
JEJU NATIONAL UNIVERSITY

목 차

List of Tables

List of Figures

Abstract

Recently, according to the development of smart grid technologies and the penetration of electric vehicles, or EVs in short, renewable energy keeps drawing attention from many areas and stakeholders. Due to its intermittency, an accurate prediction in the possible power generation is indispensable for the efficient integration of a specific renewable energy into the power grid. This prediction must be built on the enormous amount of past records and the subsequent forecast models. In this regard, this thesis develops a data processing framework for wind speed history in Jeju, Republic of Korea, which has abundant wind energy all year round. The model begins with the investigation on the correlation between wind speed and wind direction, demonstrating how we can search core information from this framework. It maintains hour-by-hour wind speed records collected during the period of 2000 to 2009. An Apache Hadoop package working on our Linux server provides a flexible interface for raw wind speed streams, while Pig Latin allows a programmer to make a comprehensive script to specify how to filter the target fields. The filtered stream records, each of which essentially embraces wind speed and wind direction, are converted to a series of learning patterns and fed to an ANN (Artificial Neural Network) model consisting of input, hidden, and output layers. Here, the input node accounts for wind direction and the output node wind speed. The number of generated learning patterns is 28,030 as they are selected from the valid records during the period of 2000 to 2008. On the contrary, 2,970 records in 2009 are exploited to evaluate the accuracy of the developed correlation model. The accuracy analysis result shows that the maximum average error is 2.32m/s, while the lowest average error has been verified to be 0.96m/s. After all, our data processing framework cannot only efficiently cook the massive data generated from renewable energy sources but also combine sophisticated modeling strategies.

Abstract

최근 스마트그리드의 발전과 전기자동차의 확산에 따라 신재생에너지에 대한 관심이 증가하고 있다. 풍력과 같은 신재생에너지를 그리드에 결합하기 위해서는 정확한 발전 가능량 예측이 필수적인데 이는 막대한 양의 과거 데이터에 대한 분석과 이에 따르는 예측 모델 구성을 필요로 한다. 본 논문에서는 풍력이 풍부한 제주 지역에서 2000년부터 2009년까지 축적된 풍속 데이터를 기반으로 정밀한 풍속 모델을 개발하기 위하여 우선적으로 풍속과 풍향과의 연관성을 분석하기 위한 데이터 처리 프레임워클 구성한다. 우선적으로 시간마다 샘플링된 대용량의 풍속과 풍향이 원시 데이터를 계산환경으로 인터페이스하기 위하여 Apache Hadoop을 Linux 플랫폼에 설치하고 Pig 스크립트를 작성하여 필요한 데이터를 필터링한다. 이 정제된 데이터를 기반으로 인공신경망 (ANN; Artificial Neural Network)의 학습 패턴을 생성한 후 입력, 은닉, 출력 등 3 계층으로 구성된 ANN 모델을 구축한다. 이 과정에서 2000년부터 2008년까지의 1월 풍향을 입력변수로, 풍속의 출력변수로 설정하였으며 2009년의 데이터는 이 모델의 정확성을 분석하는데 사용하였다. 이에 28,030 개의 학습패턴이 생성되었으며 2,970 개의 테스트 패턴이 정확성 분석에 사용되었다. 그 결과 최대 평균 오류는 2.32m/s 이며, 최저 평균 오류는 0.96m/s를 검증했다. 결과적으로, 본 논문에서 구축한 데이터 처리 프레임워크는 신재생에너지에 관련된 대용량 데이터를 효율적으로 처리할 수 있을 뿐 아니라 정교한 모델링 기법과 결합할 수 있음을 보이고 있다.

I. 서 론

스마트 그리드는 전력망에 ICT 기술을 도입·융합하여 전력공급자와 소비자 간의 양방향 통신을 구축하고, 데이터를 실시간 교환하여 사용시간과 양을 제어할 수 있는 지능적인 전력망을 말한다[Ipakchi and Albuyeh, 2009, Silva et al. 2011]. 스마트 그리드로 인하여 분산형 전원체제 전환 및 환경오염과 온실효과를 발생하지 않는 신재생에너지의 필요성이 증가하였다. 우리나라에서 재생에너지는 석유, 석탄, 원자력 또는 천연가스가 아닌 에너지를 포괄하며 화석연료를 변환시켜 이용하거나 태양, 바람, 물, 지열, 생물유기체 등을 포함하는 재생 가능한 에너지를 변환시켜 이용하는 에너지이다. 국내 신에너지 및 재생에너지개발·이용·보급촉진법 2조에 따르면, 8개 분야의 재생에너지(태양열, 태양광발전, 바이오매스, 풍력, 소수력, 지열, 해양에너지, 폐기물에너지)와 3개 분야의 신에너지(연료전지, 석탄액화가스화, 수소에너지) 총11개 분야를 신재생에너지로 지정하고 있다[국회, 2014].

전 세계적으로 풍력발전은 전력생산을 위해 사용되는 신재생 에너지원 중 가장 빨리 성장하고 있는 분야로 새로 건설되는 풍력발전단지는 전체 전력 생산량에서 많은 부분을 차지해가고 있다[Moon and Kim, 2012]. 제주특별자치도 또한 최근에 수립한 지역에너지계획에 따라 풍력발전개발 2단계 사업으로 2019년까지 육상 350MW, 해상 1,000MW 규모의 풍력발전단지를 개발하고, 3단계 사업으로 해상에 1,000MW를 추가 개발하여 『Carbon Free Island 제주』 조성을 목표로 하고 있다[Jeju, 2009].

풍력에너지의 경제 효과는 에너지원인 바람의 변동성에 의해 크게 좌우된다. 신재생에너지원은 기상상태에 따라서 출력변동이 심하고 출력의 예측이나 조정이 어려워 생산계획을 수립하기 힘들다[Luickx et al. 2008]. 신재생에너지의 효율적으로 이용하기 위해서는 신재생에너지의 전력 출력을 안정화 기술이 필요하다.

그리고 스마트 그리드의 전력량 데이터는 200만 인구기준으로 하루에 22GBytes의 데이터를 생성할 것으로 전망된다[Shargal and Houseman. 2009]. 이 데이터가 월별, 연별로 축척이 되면 이로 인해 데이터는 기하급수적으로 증가할 것이다. 그러기 때문에 풍력 데이터도 관계형 데이터베이스로 저장하거나 분석하기가 어려워진다. 결과적으로, Hadoop과 같은 빅 데이터 저장 방법과 Pig같은 데이터 분석 시스템이 필요하다[Youk et al. 2014, Taft and Martini. 2012].

본 논문은 2000년도부터 2009년도의 제주특별자치도의 서쪽지역에서 수집된 풍력데이터를 대상으로, 리눅스에 빅 데이터를 처리하기 위하여 Hadoop을 설치하고, 빅 데이터 가공을 위하여 Pig의 스크립트 이용하여 샘플링 데이터를 처리하였다. 그리고 샘플링된 풍력데이터는 인공신경망(ANN; Artificial Neural Network)을 이용하여 예측 모델을 구축하고 결과를 평가하였다. 구체적으로 풍속 예측모델의 한 단계로서 풍향과 풍속 간의 연관성에 집중한다.

II. 배경과 관련 연구

본 장에서는 배경과 스마트 그리드에 대한 빅 데이터 처리에 관련된 관련 연구와 풍력에 대한 모델링에 관련된 연구에 대해서 고찰한다.

1. 배경

우리나라는 2004년 '전력IT종합대책'을 수립하여 스마트 그리드와 관련기초기술에 대한 개발과 연구를 시작하였다. 2008년 그린 에너지사업 발전전략 과제로 스마트 그리드가 선정되었고, 2009년 열린 G8 정상회의 기후변화포럼에서 세계적으로 온실가스 감축에 대하여 관심도가 높아졌고 한국의 스마트 그리드는 '세상을 바꾸는 7대 전환적 기술'로 선정 되었다. 이후 우리나라는 스마트 그리드 실증단지를 제주특별자치도로 선정하여 스마트 전력망, 스마트 양방향 전력통신, 스마트 운송, 스마트 신재생에너지, 스마트 전력 등 분야에 집중하고 있다. 그중에서도 신재생에너지는 미래의 에너지원으로 각광을 받고 있으나 날씨나 계절의 영향을 많이 받기 때문에 안정적으로 전력을 생산하고 분배하기가 어렵다. 예를 들면 풍력에너지는 에너지원인 바람의 변동성에 따라 크게 좌우된다[Lee et al. 2010]. 따라서 신재생에너지에 대한 더 정확한 예측이 필요하게 되었으며 여러 에너지원에 정보를 가공처리 할 수 있는 빅 데이터 기반의 시스템이 많이 연구 개발 되고 있다[Youk et al. 2014, Taft and Martini. 2012].

2. 관련 연구

2.1. 베스타스 윈드 시스템 (David, 2012)

덴마크의 베스타스에서는 발전기 부지 선정 모델을 도출하기 위해 사용하는 데이터 분석 프로세스 실행에 몇 주씩 소요 데다가 정확한 풍력발전기 부지 선정과 전력 예측을 위해 필요한 대량의 데이터를 충분히 지원하지 못하기 때문에 문제가 많이 발생하였다. 따라서 기존의 풍력발전기의 데이터를 축적하여 예측시

스템에 활용하는 방법을 추구했다. 이로 인해 풍력데이터의 빅 데이터를 활용하여 터빈 관리 및 배치를 효율적으로 추진하고, 바람의 방향 및 높이에 따른 변화요소, 날씨, 조수 간만의 차, 위성 이미지, 지리 데이터, 날씨 모델링 조사 등의 데이터를 결합하였다. IBM의 분석 솔루션과 슈퍼컴퓨터를 활용하여 바람의 방향, 높이에 따른 변화요소, 이력 등 주요 정보를 이용하여 동력장치의 위치를 결정함으로써 에너지 효율성을 개선하고, 동력장치의 배치를 고려하여 에너지 생산량 및 설치 전 투자수익률을 분석하였다. 풍력터빈 및 풍력단지에 지형 2.8TB에 이르는 기상데이터를 분석하는데 장시간 걸리던 풍력예측정보 모델링 시간을 1시간 이내로 단축하였다[David, 2012, Noh and Pack, 2014].

2.2 빅 데이터 환경의 공간 기반 에너지관리시스템 (Lee et al. 2013)

기존의 전력망에 정보기술(IT)를 접목하여 전력 공급자와 소비자가 실시간 정보를 활용하는 연구가 많이 진행되고 있다. 공장용 에너지관리시스템(FEMS; Factory Energy Management System), 빌딩용 에너지관리시스템(BEMS; Building Energy Management System), 주택용 에너지관리시스템(HEMS; Home Energy Management System) 등 사용하고 있는 각종 센서 정보들을 포함한 수집 가능한 데이터를 활용하여 공간 기반 에너지관리시스템(SEMS; Space-based Energy Managemen System)의 추론엔진의 자가 학습을 통해 전력 소모량을 SEMS를 사용하는 경우와 사용하지 않은 경우를 비교하였다. 결과적으로 SEMS를 적용한 공간에서의 전력 소모량을 최대 10~25% 절감시켰다[Lee et al. 2013].

2.3 Hadoop 에코시스템 (Lee et al. 2014c)

스마트 그리드 확산사업이 시행 되면서 많은 전력데이터가 수집되고 있다. 4개 지역단지내의 약 1,880여개소로부터 수집된 데이터는 1년에 약 400GB의 전력데이터가 수집되며 4TB의 자료가 수집되었다. 확장사업을 하거나 데이터 수집양이 많아지면 관계형 데이터베이스로 저장하거나 분석하기가 어려워진다. 따라서 병렬데이터 분석 시스템이 필요하며 그중에서 Hadoop이 이용되었다. 관계형 데이터베이스와 Hadoop으로 처리속도를 비교하였고 3~5배 Hadoop의 병렬데이터분석

시스템이 빠르다는 결과가 나왔다. 시간이 지나면 지날수록 데이터양은 기하학적으로 증가하는 것을 고려하면 관계형 데이터베이스로 처리가 불가능할 것이다. 그리고 Hadoop 에코시스템 중 예측모델인 마헛(mahout)은 기계 학습 알고리즘을 포함하고 있으며, 선형회귀분석을 이용하여 수요전력량 예측 값과 실제 사용량을 비교 평가한 내역은 상대적으로 수요전력량 예측과 비슷한 흐름이다[Lee et al. 2014c].

2.4 다계층 신경망을 풍력발전량 예측 (Hwang et al. 2012)

전력 생산량이 갑작스럽게 크게 변화(증가하거나 감소)하는 경우가 있는데 그걸 Ramp이라 한다. Ramp에 따라 전력생산량이 감소, 증가 할 수 있기 때문에 다계층 신경망을 이용하여 풍력 발전량 예측을 했다. 데이터는 2010년부터 04월 30일부터 05월 17일까지 제주특별자치도에서 측정한 데이터를 가지고 연구하였고 예측모델 구축에는 PRR(Power Ramp Rate) 뿐만 아니라 풍속, 풍향의 속성까지 총 세 가지 속성을 사용하였다. 10분 동안 전력 생산량의 증감율을 이용하여 예측모델 구축에 사용하였다. 4가지 모델을 적용하여 PRR의 속성을 사용하면 예측 정확도를 향상 시킬 수 있다는 결과를 얻었다[Hwang et al. 2012].

2.5 인공 신경망에 따라 풍속 모델링(Lee et al. 2012)

스마트 그리드 시스템에 풍력 발전의 정확한 예측을 하기 위한 목표로 인공 신경망을 이용하여 제주특별자치도 지역 풍향에 대한 풍속을 예측하였다. ANN 모델링을 하기 위해서 입력을 연속으로 5개 값과 하나의 출력으로 이루어져있으며 115 패턴으로 학습을 하고 각각 10, 20, 30, 40 은닉계층에 대해서 분석 하여 10개에서 가장 좋은 결과를 얻었다. 평가분석 결과 2000년~2010년 데이터 중 2004년 이후의 데이터는 실제 변화와 유사하였다. 풍력 발전의 정확한 추정은 스마트 그리드에 발전에 있어서 좋은 정보가 되고, 더 나아가 모델을 세분 할 수 있는 토대를 제공한다[Lee et al. 2012].

III. 제안 기법

본 장에서는 제안된 기법에 대해 설명한다. 우선 빅 데이터를 샘플링이 가능한 Hadoop 기반 데이터 처리 구조와 인공신경망(ANN; Artificial Neural Network)의 구조를 소개하고, 이 논문에서 사용될 기법을 자세한 동작을 설명한다.

1. Hadoop 기반 플랫폼

Hadoop은 빅 데이터 처리에 적합한 병렬처리 플랫폼이고[Hadoop], 구글의 분산 파일시스템(GFS; Google File System)[Ghemawat et al. 2003]에 논문 공개 후 본격적으로 개발되었다[White 2011]. Hadoop은 Apache에서 대용량의 웹 데이터를 처리하기 위해 연구개발하게 되면서 하부 컴포넌트 중 HDFS, HBase, Hive, Pig 등의 특성별로 지원하며, 데이터 처리방식으로 Map-Reduce 소프트웨어 프레임워크를 사용한다. 특징으로는 첫째, 오픈 소스이기 때문에 누구나 무료사용할 수 있다. 둘째, 데이터의 크기가 소스코드보다 훨씬 크기 때문에 소스코드를 데이터 파일 있는 곳으로 이동할 수 있다. 셋째, 스케일아웃 방식으로 저가 장비를 여러 대 사용하여 빅 데이터에 드는 유지보수 금액을 절약할 수 있다. 본 논문에서는 Standalone Mode로 설치하여 리눅스3.5.0에 Hadoop-1.2.1를 적용하였다.

1.1 HDFS(Hadoop Distributed File System)

HDFS은 빅 데이터 처리를 높이기 위한 분산 파일 시스템이다. 크게 마스터-슬레이브 구조로 동작하며, 마스터의 네임노드가 슬레이브 안에 있는 데이터노드에 파일의 위치, 분할한 내역, 복제한 내역을 관리한다[Geum et al. 2010]. 클라이언트에서 데이터가 넘어오면 일반적으로 64MB 크기의 블록 단위로 나누어서 저장처리를 하며, 데이터블록의 손실에 대한 안전방안으로 네임노드에서는 데이터노드에 데이터를 저장할 때 예비로 다른 데이터노드에 동일한 데이터블록을 복사

한다. 평상시에는 데이터노드들은 주기적으로 네임노드에게 상태정보 신호를 발생하여 데이터노드 상태를 보고한다[Ohohta 2014].

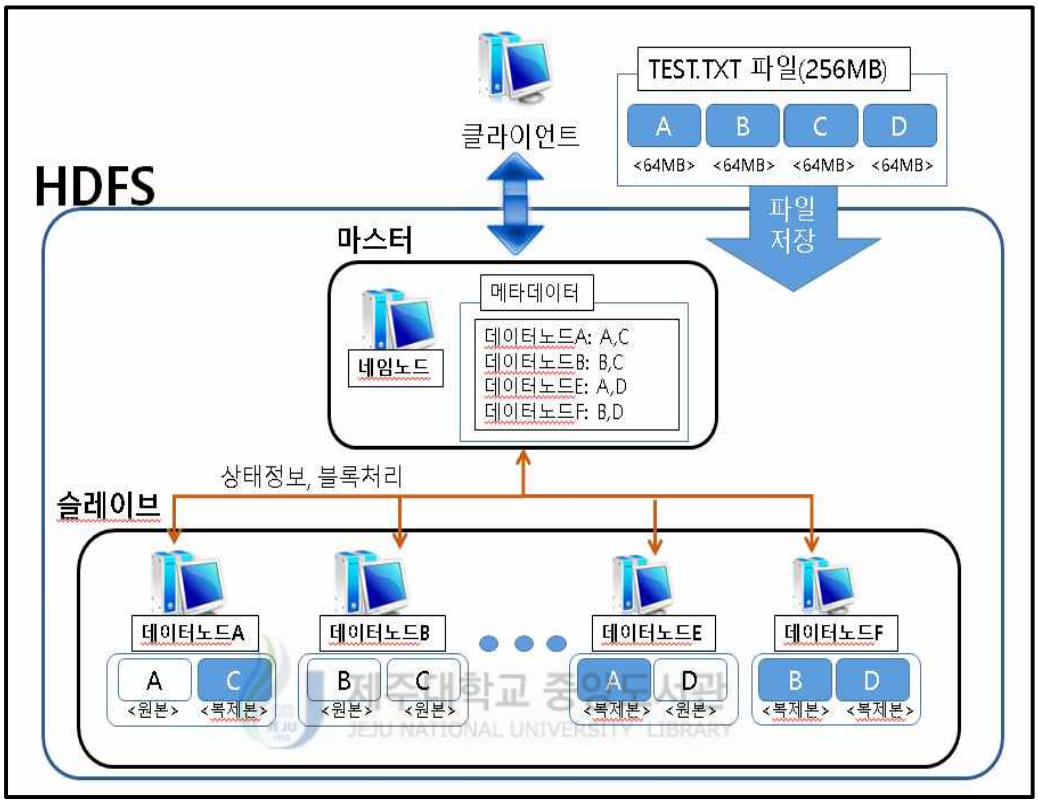


그림 1. HDFS 구성도

<그림 1>은 HDFS 구성도이며 TEST.TXT 파일을 클라이언트에서 HDFS을 통하여 저장하는 방법을 표현했다. TEST.TXT 파일의 크기는 256MB이고, 64MB 크기의 블록 단위로 나누고 A 블록을 데이터노드 A에 저장할 때 데이터노드 E에 복제본을 같이 저장 처리한다. 그리고 데이터노드들의 블록정보는 네임노드가 관리한다. 본 논문에서는 여러 풍력단지 및 풍력발전기에 데이터 보관방식에 각 TCP/IP로 연결하여 데이터 구축이 가능할 것이라고 가설을 설정하고 테스트서버를 설정하였다.

1.2 Pig

Pig는 야후에서 처음 개발되었으며 Pig Latin이라는 데이터셋 플로우 제어 언어를 제공하며 HDFS에 저장된 대용량 데이터셋을 분석이 가능한 순차적 프로그래밍 모델로 하이레벨 프로그래밍 언어 중 하나이다[Pig]. Pig Latin은 내부 컴파일러에 의해 Map-Reduce 프로그램으로 컴파일되며, 훨씬 더 작은 양의 코드로도 더 높은 생산성을 갖는다. 부족한 기능이 있을 경우 해당 기능을 사용자 정의 함수(UDF; User Defined Function)로 Java, Python, JavaScript 등 여러 스크립트 언어로도 작성이 가능하다. <그림 2>는 풍력 데이터를 LOAD 명령으로 호출하고 FOREACH 로 데이터 값을 가져온다. 그 후 FILTER를 이용하여 1월 데이터를 분류하고, 오류를 제거한 후 연별로 데이터를 나눠서 파일로 저장하였다.

```
log = LOAD '/home/seakove/wind/winddata.txt' USING PigStorage('\t')
AS (station: int, stamp: datetime, mon: int,dir: double, speed: double);
proj1 = FOREACH log GENERATE station, GetYear(stamp) as year, GetHour(stamp),mon,
dir, speed;
proj2 = FILTER proj1 BY (mon ==  1);
proj5 = FILTER proj2 BY (dir >  -0.1);
proj6 = FILTER proj5 BY (year <  2009);
proj7 = FILTER proj5 BY (year ==  2009);
proj09 = FOREACH proj6 GENERATE dir,speed;
proj10 = FOREACH proj7 GENERATE dir,speed;
STORE proj09 INTO 'winddata2008';
STORE proj10 INTO 'winddata2009';
```


그림 2. Pig 풍력 데이터 연도별 1월 데이터 샘플링(wind.pig)

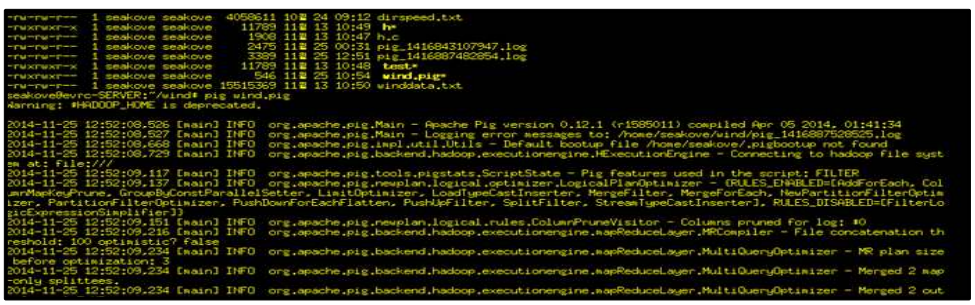
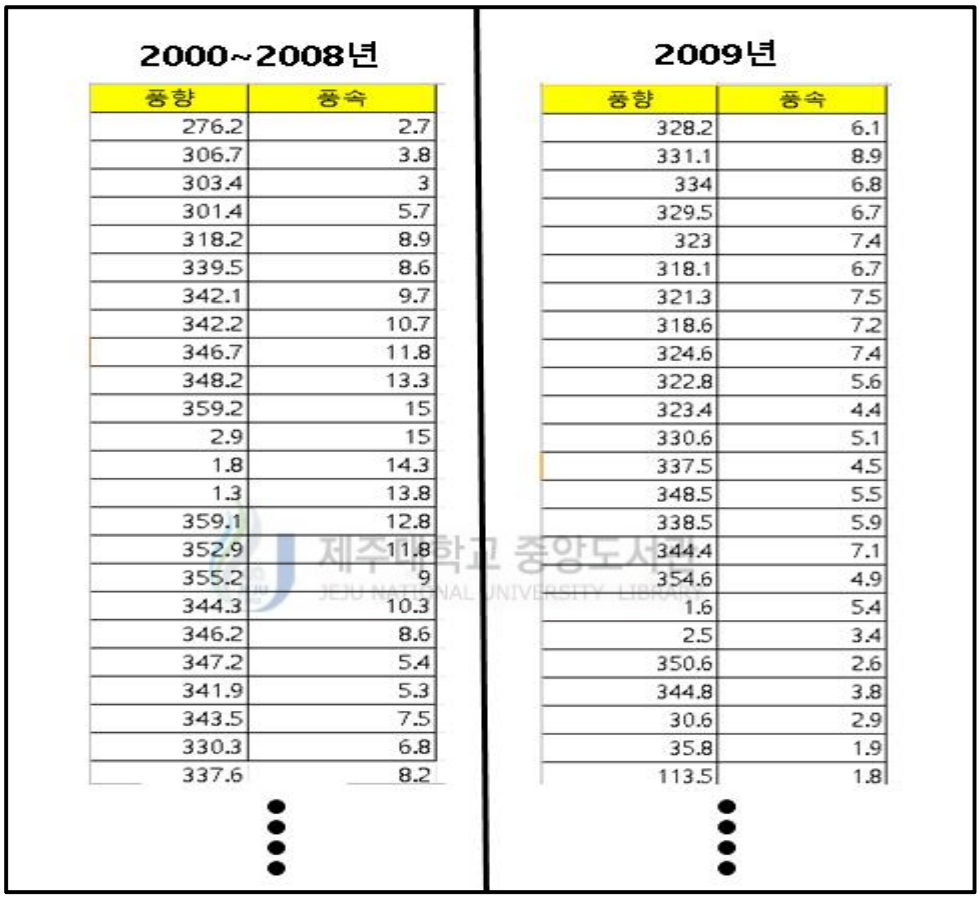


그림 3. Pig 실행 (wind.pig)

		풍향	풍속	건수
2000년-2008년 1월	최소값	0°	0m/s	25,030건
	최대값	360°	15m/s	
2009년 1월	최소값	0°	0m/s	2,970건
	최대값	360°	12.6m/s	

표 1. Pig 샘플링 데이터 내역



2000~2008년

풍향	풍속
276.2	2.7
306.7	3.8
303.4	3
301.4	5.7
318.2	8.9
339.5	8.6
342.1	9.7
342.2	10.7
346.7	11.8
348.2	13.3
359.2	15
2.9	15
1.8	14.3
1.3	13.8
359.1	12.8
352.9	11.8
355.2	9
344.3	10.3
346.2	8.6
347.2	5.4
341.9	5.3
343.5	7.5
330.3	6.8
337.6	8.2

2009년

풍향	풍속
328.2	6.1
331.1	8.9
334	6.8
329.5	6.7
323	7.4
318.1	6.7
321.3	7.5
318.6	7.2
324.6	7.4
322.8	5.6
323.4	4.4
330.6	5.1
337.5	4.5
348.5	5.5
338.5	5.9
344.4	7.1
354.6	4.9
1.6	5.4
2.5	3.4
350.6	2.6
344.8	3.8
30.6	2.9
35.8	1.9
113.5	1.8

그림 4. Pig를 이용한 풍력 데이터 샘플링 결과 형태

위 <표 1>를 보면 2000년부터 2008년 1월 데이터 총 25,030건의 데이터가 추출되었으며 풍향에 대한 최소값 0°부터 360°, 풍속에는 0m/s부터 15m/s 까지 샘플링되었다. 그리고 2009년 1월 데이터는 총 2,970건이 추출되었고 풍향에 대한 최소값 0°부터 최대값 360°, 그리고 풍속은 0m/s부터 12.6m/s까지 샘플링되었다. <그림 4>은 2000년부터 2008년의 데이터파일명은 part-m-00000로 정의하였고 풍향과 풍속은 소수점까지 표현하였다. 그리고 2009년 데이터는 파일명 Pre로 변

환시키고 풍향과 풍속은 앞과 동일하게 표현하였다.

1.3 Hadoop을 이용하여 데이터 샘플링 전체 구성도

Hadoop을 이용하여 풍력데이터를 학습데이터와 추후 예측결과 데이터를 분류하여 샘플링을 하였다. 그리고 <그림 5>처럼 여러 풍력단지 및 지역이 분리 되었을 때 데이터 샘플링 전체 구성도를 표현하였다. 병렬 분산 처리방식이라 서버만 증가 하면 많은 데이터도 수집가능하고 샘플링을 통하여 여러 가지 분석을 할 수 있다.

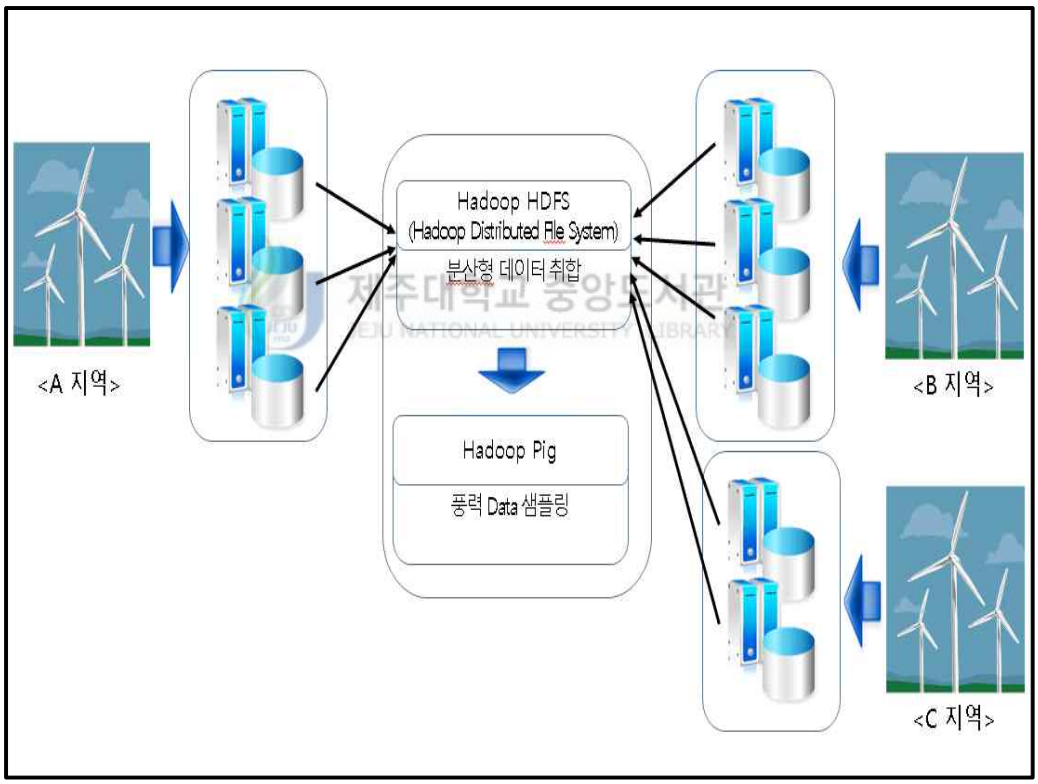


그림 5. Hadoop을 이용한 풍력데이터 연도별 1월 데이터 샘플링 구성도

2. ANN 기반 예측모형

인공신경망(ANN)은 인간의 뇌와 같이 수많은 신경세포가 연결되어 있는 형태를 컴퓨터 시뮬레이션으로 표현하는 것을 목표로 하는 수학 모델이다[Lee et al. 2014b]. 인공신경망은 반복학습을 통하여 예측 값의 정확도가 높아진다. 하지만 어느 정도 학습이 진행되면 예측 값은 더 이상 변하지 않는다. 본 논문에서는 <그림 6>에서 보는 바와 같이 1개의 입력과 1개의 출력, 레이어(입력1, 히든1, 출력1)는 3개, 히든뉴런은 30개 갖고 있다. 그리고 제주도 풍력 데이터를 2000년부터 2008년까지 1월 풍속데이터를 학습을 시키고 2009년 1월은 학습을 시키지 않고, 예측내용과 비교 자료로 사용하였다. ANN 라이브러리로는 FANN(Fast Artificial Neural Network)을 사용한다[Nissen 2005].

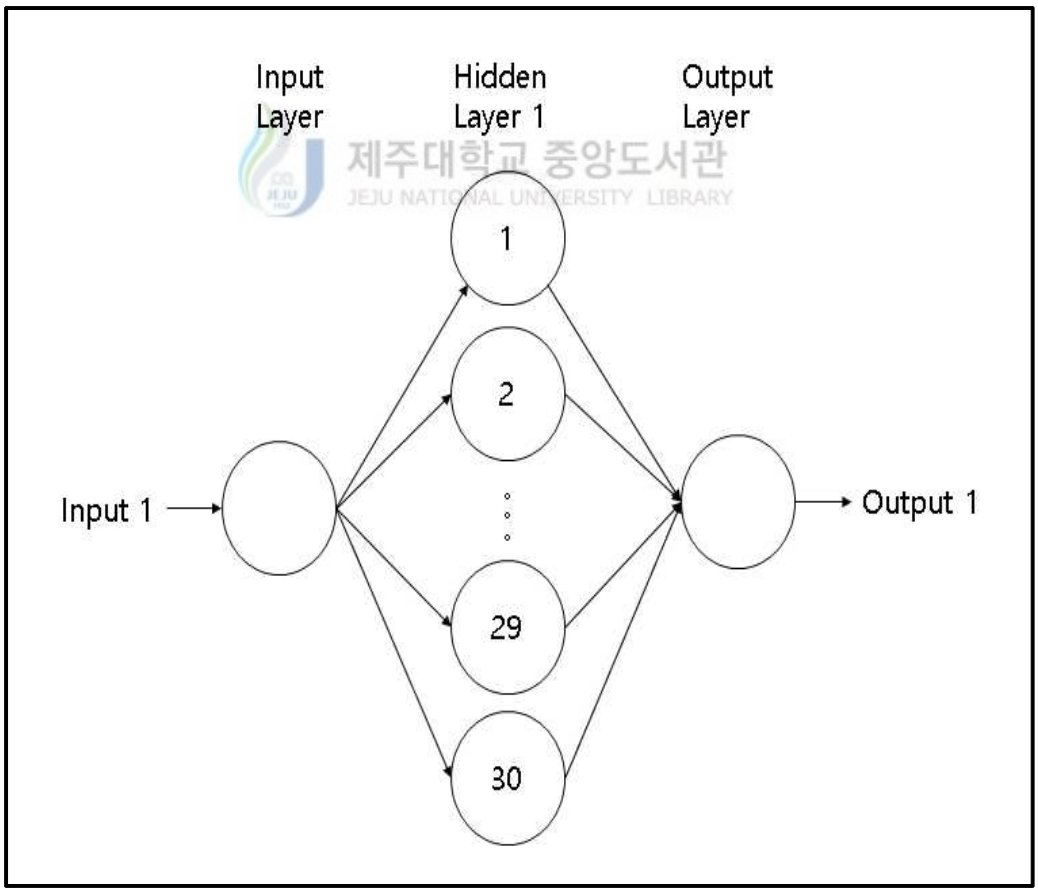


그림 6. ANN 구성도

2.1 ANN 데이터 변환 작업

```
#include <stdio.h>
main() {
	FILE *fp, *ofp;
	int NP=500;
	double pat[Len]={0.0};
	int gen=0, i= 0, dir=0, cnt=0;;
	double a = 0.0 , b= 0.0;
	double speed= 0.0;
	double dirmax=0.0, speedmax = 0.0;
	fp = fopen("part-m-00000", "r");
	while (! feof(fp)) {
		fscanf(fp,"%lf%lf", &a, &b);
		if (a > dirmax) dirmax = a;
		if (b > speedmax) speedmax = b;
	}
	fclose(fp);
	printf("******************************** \n");
	printf("dirmax : %lf	\n", dirmax);
	printf("speedmax : %lf	\n",  speedmax);
	printf("******************************** \n");

	fp = fopen("part-m-00000", "r");
	ofp = fopen("learn.txt", "w");
	fprintf(ofp, "00000 1 1 %lf %lf\n",dirmax,speedmax);
	while (! feof(fp)) {
		if (fscanf(fp,"%lf%lf", &a, &b)==EOF) break;
		fprintf(ofp, "%lf %lf \n",a, b/speedmax );
		cnt++;
	}
	fclose(fp);
	fseek(ofp, 0, SEEK_SET);
	fprintf(ofp, "%5d 1 1 %lf %lf\n", cnt-1,dirmax,speedmax);
	fclose(ofp);
}
```

그림 7. ANN 학습 패턴의 생성 코드

FANN 라이브러리의 인공신경망을 사용하기 위해서는 데이터 값이 0~1 사이에 값으로 변경하여야 한다[Lee et al. 2012, Lee et al. 2014a]. 데이터 변환을 위하여 C언얼 작성한 변환 프로그램은 <그림 7>에 보이고 있으며 이의 실행화면은 <그림 8>에 보이고 있다. 본 논문에서는 리눅스 기반으로 실행하였다.

```
-rwxrwxr-x  1 seakove seakove  90689 11월 25 12:18 trace*
-rw-rw-r--  1 seakove seakove   1359 11월 25 12:41 trace.c
seakove@evrc-SERVER:~/Neural/winddata$ gcc -o genpat genpat.c
seakove@evrc-SERVER:~/Neural/winddata$ ./genpat
********************************
dirmax : 360,000000
speedmax : 15,000000
********************************
seakove@evrc-SERVER:~/Neural/winddata$
```

그림 8. ANN을 사용하기 위한 데이터 변환 과정

또, <그림 9>에서 보면 풍속을 0~1 사이에 값으로 변경하였다. 11번째 로우 데이터의 예에서 보면 풍향 359.2° 풍속 15인 데이터가 변경 후 1로 변환되고, 343.5° 풍속 7.5는 데이터 변경 후 0.5 로 변환했다.

원본 데이터

풍향	풍속
276.2	2.7
306.7	3.8
303.4	3
301.4	5.7
318.2	8.9
339.5	8.6
342.1	9.7
342.2	10.7
346.7	11.8
348.2	13.3
359.2	15
2.9	15
1.8	14.3
1.3	13.8
359.1	12.8
352.9	11.8
355.2	9
344.3	10.3
346.2	8.6
347.2	5.4
341.9	5.3
343.5	7.5
330.3	6.8
337.6	8.2

변경 데이터

풍향	풍속
276.2	0.18
306.7	0.253333
303.4	0.2
301.4	0.38
318.2	0.593333
339.5	0.573333
342.1	0.646667
342.2	0.713333
346.7	0.786667
348.2	0.886667
359.2	1
2.9	1
1.8	0.953333
1.3	0.92
359.1	0.853333
352.9	0.786667
355.2	0.6
344.3	0.686667
346.2	0.573333
347.2	0.36
341.9	0.353333
343.5	0.5
330.3	0.453333
337.6	0.546667

그림 9. ANN 학습 패턴의 생성 결과

2.2 인공신경망 학습

<그림 10>은 FANN 라이브러리 이용하여 C언어로 코딩한 내용이다. fann_create_standard로 인하여 신경망을 구성하며 int numHidden=30은 은닉형 뉴런을 30개를 배치하고, 총 3개의 레이어를 구성한다. fann_train_on_file를 이용하여 학습할 데이터 learn.txt 파일을 5,000번 학습하거나 오류률 0.00001% 미만에 적합할 때 ANN 파일의 생성을 완료한다.

```
#include <stdio.h>
#include "../FANN/fann.h"

void main(int argc, char *argv[]) {
        struct fann *ann;
        int numHidden=30;
        if (argc > 1)
                numHidden = atoi(argv[1]);
        ann= fann_create_standard(3, 1, numHidden, 1);
        fann_train_on_file (ann, "learn.txt", 5000, 100, 0.00001);
        fann_save(ann, "dirspeed5000.ann");
        fann_destroy(ann);
}
```


그림 10. ANN을 학습 코드

```
seakove@evrc-SERVER:"/Neural/winddata# ./genpat
********************************
dirmax : 360.000000
speedmax : 15.000000
********************************
seakove@evrc-SERVER:"/Neural/winddata#
seakove@evrc-SERVER:"/Neural/winddata#
seakove@evrc-SERVER:"/Neural/winddata#
seakove@evrc-SERVER:"/Neural/winddata# gcc -o learn learn.c ../FANN/doublefann.c -lm
seakove@evrc-SERVER:"/Neural/winddata# ./learn
1
Max epochs     5000. Desired error: 0.0000100000.
Epochs            1. Current error: 0.0909446031. Bit fail 6877.
Epochs          100. Current error: 0.0275245570. Bit fail 397.
Epochs          200. Current error: 0.0275308322. Bit fail 409.
Epochs          300. Current error: 0.0275152456. Bit fail 402.
Epochs          400. Current error: 0.0275053829. Bit fail 401.
Epochs          500. Current error: 0.0275173020. Bit fail 411.
Epochs          600. Current error: 0.0275167599. Bit fail 395.
Epochs          700. Current error: 0.0275072698. Bit fail 396.
Epochs          800. Current error: 0.0275130924. Bit fail 406.
Epochs          900. Current error: 0.0275129583. Bit fail 411.
Epochs         1000. Current error: 0.0274886880. Bit fail 400.
Epochs         1100. Current error: 0.0274839867. Bit fail 401.
Epochs         1200. Current error: 0.0274733286. Bit fail 400.
Epochs         1300. Current error: 0.0274664294. Bit fail 391.
Epochs         1400. Current error: 0.0274701510. Bit fail 398.
Epochs         1500. Current error: 0.0274713915. Bit fail 400.
Epochs         1600. Current error: 0.0274643358. Bit fail 402.
Epochs         1700. Current error: 0.0274617784. Bit fail 392.
Epochs         1800. Current error: 0.0274633355. Bit fail 400.
Epochs         1900. Current error: 0.0274580866. Bit fail 396.
Epochs         2000. Current error: 0.0274631083. Bit fail 397.
Epochs         2100. Current error: 0.0274642836. Bit fail 400.
Epochs         2200. Current error: 0.0274582077. Bit fail 400.
Epochs         2300. Current error: 0.0274616051. Bit fail 405.
Epochs         2400. Current error: 0.0274589397. Bit fail 392.
```

그림 11. ANN의 학습과정

<그림 11>은 위에 설명한내용처럼 5,000번 반복 학습을 하거나 Mean square error가 0.00001%보다 작아지면 프로세스가 ANN 파일을 생성하고 종료가 된다. 현재 오류율이 0.02% 미만으로 내려가지 않아 5,000번 반복 수행 결과가 생성되었다.

```
FANN_FLO_2.1
num_layers=3
learning_rate=0.700000
connection_rate=1.000000
network_type=0
learning_momentum=0.000000
training_algorithm=2
...
cascade_activation_functions_count=10
cascade_activation_functions=3 5 7 8 10 11 14 15 16 17
cascade_activation_steepnesses_count=4
cascade_activation_steepnesses=2.50000000000000000000e-01
 5.00000000000000000000e-01 7.50000000000000000000e-01 1.00000000000000000000e+00
layer_sizes=2 31 2
scale_included=0
neurons (num_inputs, activation_function, activation_steepness)=
(0, 0, 0.00000000000000000000e+00) (0, 0, 0.00000000000000000000e+00)
(2, 4, 5.00000000000000000000e-01) (2, 4, 5.00000000000000000000e-01)
...
connections (connected_to_neuron, weight)=(0, -1.71237314492922454434e-02)
 (1, 4.82506455674434509007e+00) (0, -9.86839824211866029069e-02)
(1, -3.32397963252730743733e+00) (0, 3.46935472680456102879e-01)
(31, 8.10517180355675215253e-01) (32, -1.27491186957526647650e-01)
...
```

그림 12. ANN 라이브러리가 자체 생성한 텍스트 파일

2.3 모델링 결과

FANN로 학습한 후 모델링 결과를 출력한다. 예측에 대한 FANN 라이브러리는 fann_run이며 학습한 내용처럼 입력 1개를 받고 출력 1개를 하는 방식이다. <그림 13>는 2000년부터 2008년도까지의 데이터로 학습한 내용의 예측결과와 기존 데이터 사이에 유사도를 판단하기 위해서 작성하였다.

```
TraceSpeed() {
    FILE *fp , *ofp;
    struct fann *ann = fann_create_from_file("dirspeed5000.ann");
    double seq[0];
    double dirmax=0.0, speedmax = 0.0;
    int i=0, cnt =0;
    double *output, next;
    fp = fopen("learn.txt", "r");
    ofp = fopen("learndata.csv", "w");
    fscanf(fp, "%d%d%d%lf%lf", &i, &i, &i ,&dirmax , &speedmax);

    while (! feof(fp)) {
        if (fscanf(fp,"%lf%lf", &seq[0], &next)==EOF) break;
        output = fann_run(ann, seq);
       fprintf(ofp, "%3.1lf,%5.1lf,%5.1lf\n",seq[0], next*Max, (*output)*Max);
    }
    fclose(fp);
    fclose(ofp);
}
```


그림 13. 2000년-2008년 ANN을 학습결과 비교

```
PredictSpeed() {

    FILE *fp, *ofp;
    struct fann *ann = fann_create_from_file("dirspeed5000.ann");
    double seq[0];
    double dirmax=0.0, speedmax = 0.0;
    int i;
    double *output, next;
    fp = fopen("pre", "r");
    ofp = fopen("predata.csv", "w");

    while (! feof(fp)) {
        if (fscanf(fp,"%lf%lf", &seq[0], &next)==EOF) break;
        output = fann_run(ann, seq);
        fprintf(ofp, "%3.1lf,%5.1lf,%5.1lf\n",seq[0], next, (*output)*Max);
    }
    fclose(fp);
    fclose(ofp);
}
```

그림 14. 2009년 ANN을 학습결과 비교

<그림 14>는 2000년부터 2008년도까지의 데이터로 학습한 내용의 예측결과와 2009년 데이터 사이에 유사도를 판단하기 위해서 작성하였다.

그림 15. ANN을 학습결과 비교처리

IV. 성능평가

본 장에서는 풍력데이터 학습 내용과 예측결과에 따른 비교와 오류정도를 비교 분석한다.

1. 풍향에 따른 예측풍속과 기존풍속 데이터 비교

2000년-2008년

풍향	풍속	예측풍속
276.2	2.7	3.9
306.7	3.8	4.5
303.4	3	4.5
301.4	5.7	4.5
318.2	8.9	4.7
339.5	8.6	5
342.1	9.7	5.1
342.2	10.7	5.1
346.7	11.8	5.1
348.2	13.3	5.1
359.2	15	5.2
2.9	15	4.7
1.8	14.3	4.8
1.3	13.8	4.8
359.1	12.8	5.2
352.9	11.8	5.1
355.2	9	5.1
344.3	10.3	5.1
346.2	8.6	5.1
347.2	5.4	5.1
341.9	5.3	5.1
343.5	7.5	5.1
330.3	6.8	4.9
337.6	8.2	5

2009년

풍향	풍속	예측풍속
328.2	6.1	4.9
331.1	8.9	4.9
334	6.8	5
329.5	6.7	4.9
323	7.4	4.8
318.1	6.7	4.7
321.3	7.5	4.8
318.6	7.2	4.7
324.6	7.4	4.8
322.8	5.6	4.8
323.4	4.4	4.8
330.6	5.1	4.9
337.5	4.5	5
348.5	5.5	5.1
338.5	5.9	5
344.4	7.1	5.1
354.6	4.9	5.1
1.6	5.4	4.8
2.5	3.4	4.7
350.6	2.6	5.1
344.8	3.8	5.1
30.6	2.9	3.6
35.8	1.9	3.4
113.5	1.8	2.5

그림 16. 2000년~2008년 풍속데이터와 2009년 풍속데이터 예측데이터 비교

2000년부터 2008년까지 풍속데이터와 2009년 풍속데이터에 두 데이터 모두 FANN로 2000년부터 2008년까지 학습하고, 그 결과에 따른 예측 풍속을 각 각에 같이 ROW데이터 형식으로 생성했다. 두 개의 데이터 모두 기존에 데이터는 범

위가 상하 많은 변동이 있고, 예측풍속은 조금씩만 미동이 있는 점이 보인다.

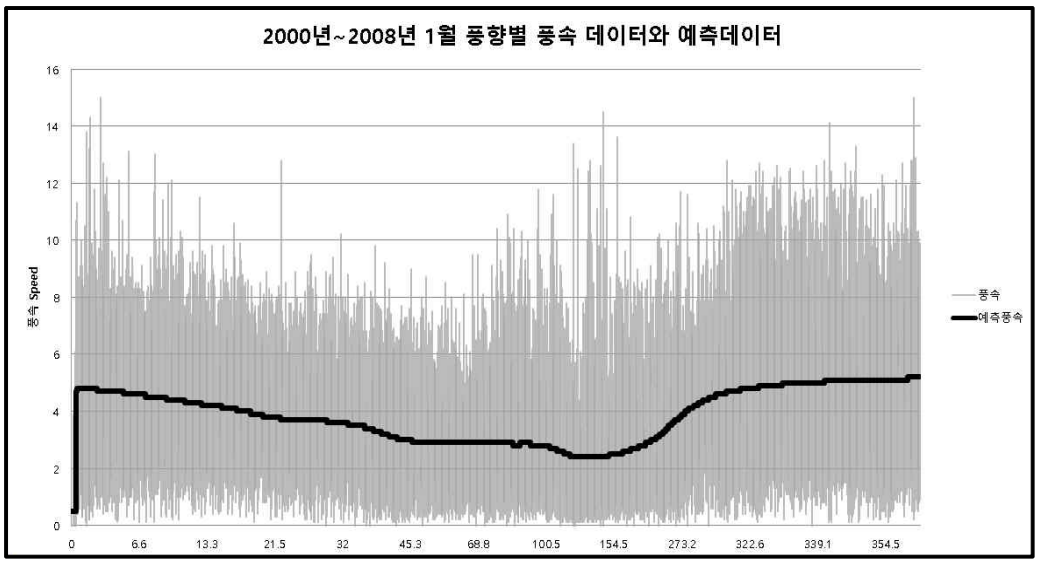


그림 17. 2000년~2008년 풍속데이터와 예측데이터 비교

<그림 17>에서 세로축은 풍속을 나타내며 가로축은 풍향을 나타낸다. 2000년~2008년 1월 풍속 데이터가 풍속의 범위가 0m/s부터 15m/s까지 데이터가 변동함을 알 수 있고, 연도 마다 강수량, 기후, 날씨 등 기상조건에 따라 많이 영향받기 때문에 풍속에 대한 범위가 넓어지며 예측 규칙을 찾기 힘들 것처럼 보였으나, FANN 통하여 예측한 데이터인 실선의 라인에 대한 흐름이 비슷하게 진행기 때문에 학습이 0.02%이하 오류에 대해서도 어느 정도는 예측 가능할 수 있다는 걸 보여준다.

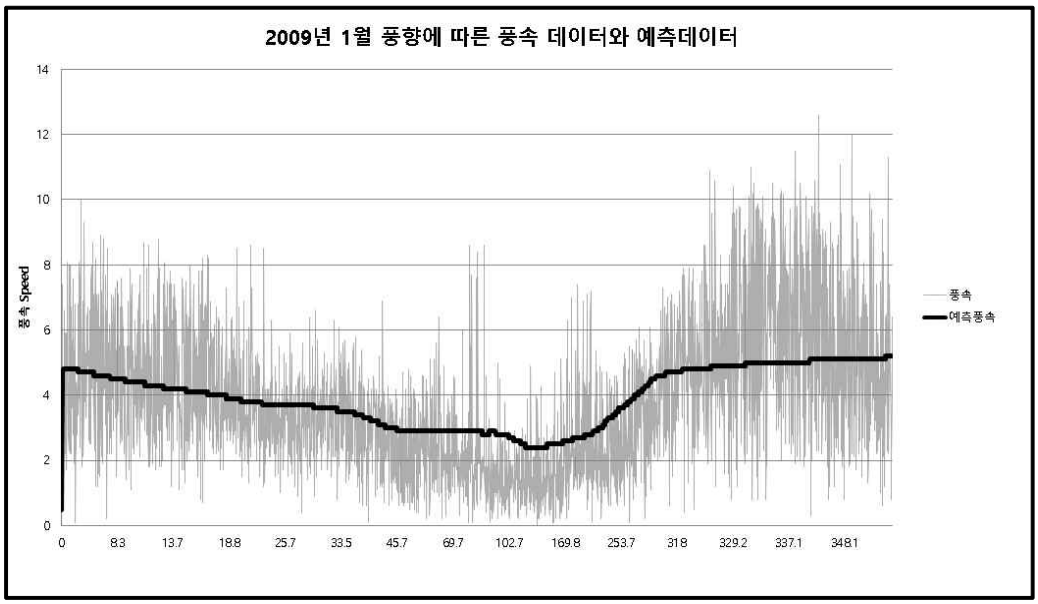


그림 18. 2009 풍속데이터와 예측데이터 비교

이번에는 2009년 1월 데이터를 비교분석했다. <그림 18>를 보면 기존 2000년부터 2008년까지 풍속데이터를 비교한 <그림 17> 보다는 선명하게 구분이 가는 걸 확인할 수 있다. 그리고 이 그래프 보면 약 풍향이 [약 0°부터20°] 와 [약 300°부터 360°]까지 풍속이 높다는 걸 알 수 있고 그에 따른 FANN 예측데이터와 유사하다는 걸 볼 수 있다.

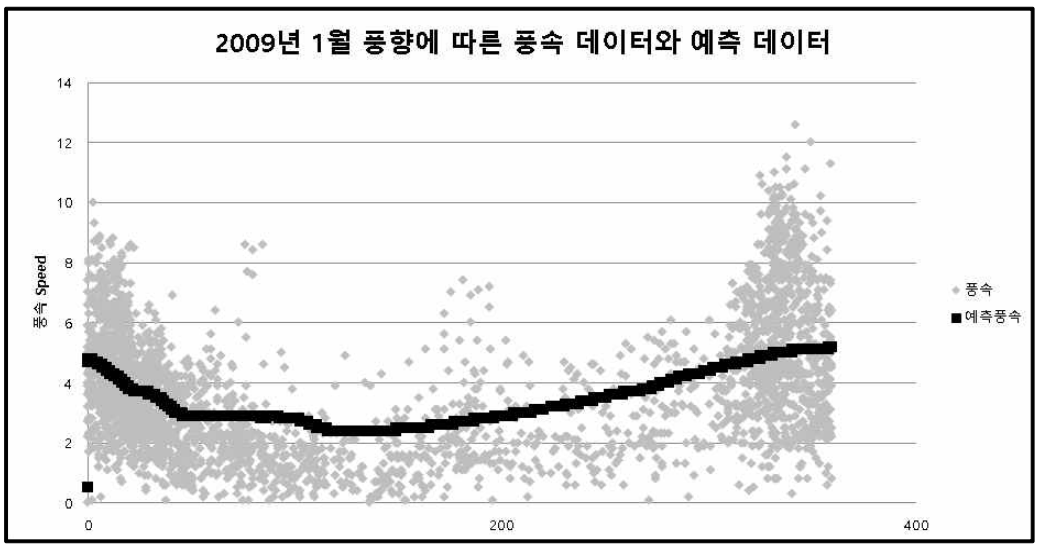


그림 19. 2009 풍속데이터와 예측데이터 분포도

<그림 19>에서 분포도가 북쪽방향에 대해서 많이 있는 걸 볼 수 있다. 우리나라는 여름이 되면 북태평양 기단이 북서쪽으로 확대하므로, 북태평양 고기압 연변에 정체해 있던 장마 정선이 북상한다. 겨울에는 시베리아 대륙의 한랭한 대륙성 고기압과 알류샨 근해의 저기압이 서고동저형의 기압 배치를 이루면 한랭 건조한 북서 계절풍이 분다.

2. 예측풍속에 대한 오류패턴 분석

그림 20. 2000년부터 2008년 1월 풍력 평균오류 데이터

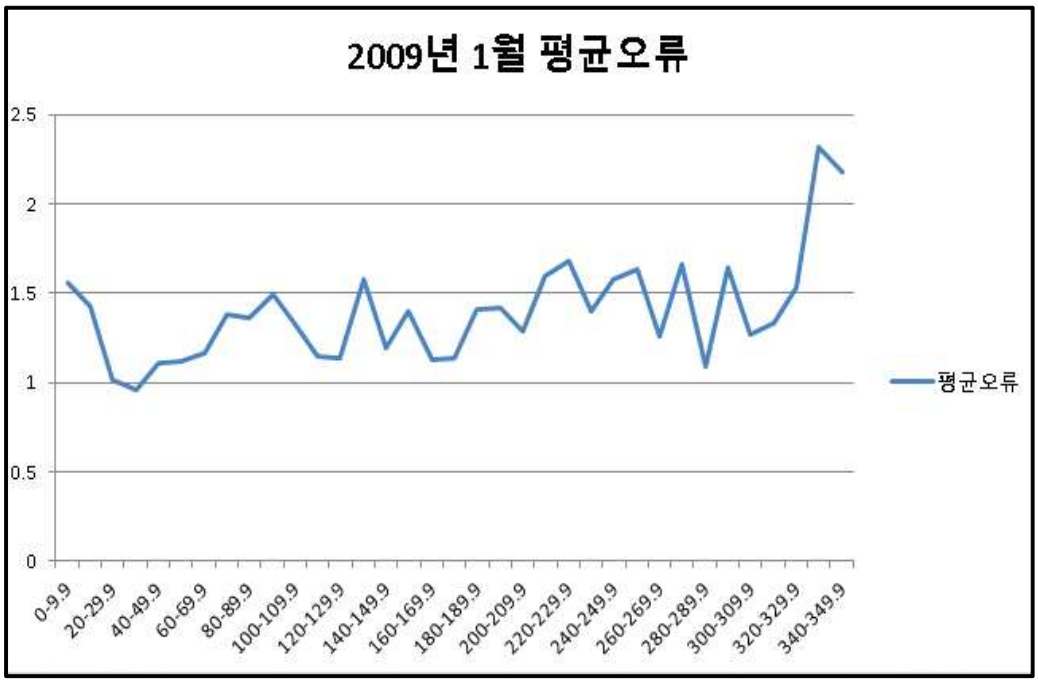


그림 21. 2009년 1월 풍력 평균오류 데이터

FANN에서 나온 예측풍속에 대한 평균오류데이터이다. 세로는 평균오류이다. 가로는 풍향을 10° 단위로 구분하여 그룹을 만들어서 평균오류를 측정하였다. <그림 20>에서 보면 풍향이 320°부터 360°까지에 대한 평균오류가 높다는 걸 보여주며, 최고는 풍향 320°-320.9°구간에서는 2.19m/s을 보여주며 최소는 풍향 110°-119.9°구간에는 1.122m/s에 오류 평균을 보여준다. <그림 21> 2009년 1월 풍력 데이터에서는 최고는 풍향 330°-339.9°구간에서는 2.32m/s 와 최소는 풍향 30°-39.9°구간에서는 0.96m/s를 보여준다.

V. 결 론

본 논문에서는 우리는 제주특별자치도의 풍력데이터를 효율적으로 처리하며, 풍향에 대한 풍속 예측을 할 수 있는 Hadoop 기반으로 제주 풍속 변화의 분석 플랫폼 구축을 하였다. Hadoop은 대규모 데이터를 분산 병렬 처리할 수 있는 기반 환경과 샘플링 기법을 제공하고 있다. 스마트 그리드에서도 분산 전력 체계를 추구하고 시간이 지나면서 전력데이터양이 빠른 속도로 증가하기 때문에 Hadoop의 HDFS이 많은 빅 데이터 연구에 응용이 가능하며, Hadoop의 하부 컴포넌트 Pig를 사용하여 빅 데이터를 샘플링 쉽게 접근할 수 있다. 그리고 2000년부터 2009년까지 풍향, 속도 데이터를 가지고 풍향에 따른 풍속의 예측모델을 ANN을 이용하여 구축하고 성능평가를 하였다. FANN으로 학습 시킨 모델이 2009년 1월 풍향에 따른 풍속데이터와 성능평가를 실시하였다. 그 결과 최대 평균 오류는 2.32m/s 이며, 최저 평균 오류는 0.96m/s를 검증했다. 우리는 성능평가 결과로 2000년부터 2008년 풍력데이터로 학습한 ANN는 2009년도 데이터를 유사하게 예측한다고 판단했다. 추후 다른 연관성에 대해서 조금 더 연구를 진행하며 더 정확한 예측이 가능한 모델이 완성된다고 생각하다.

VI. 참고문헌

[Ipakchi and Albuyeh, 2009] Ipakchi, A., Albuyeh,, F., 2009. Grid of the future, IEEE Power & Energy Magazine, pp. 52-62.

[Silva et al. 2011] Silva, D., Xinghuo, Y., Alahakoon, D., Holmes, G., 2011. Semi-supervised classification of characterized patterns for demand forecasting using smart electricity meters, Electrical Machines and Systems (ICEMS), 2011 International Conference on, pp. 1-6.

[국회, 2014] 대한민국 국회, 2014. 신에너지 및 재생에너지 개발・이용・보급 촉진법 2조, 산업기술혁신 촉진법, 시행 2011.11.25.

[Moon and Kim, 2012] Moon, D., Kim, S., 2012. Study on Artificial Neural Network Based Fault Detection Schemes for Wind Turbine System, Journal of Korean Institute of Intelligent Systems, Vol. 22, No. 5, October 2012, pp. 603-609

[Jeju, 2009] Jeju Special Self-Governing Province, 2009. Fundamental Renewable Energy Supply Development Plan of Jeju.

[Luickx et al. 2008] Luickx, P., Delarue, E.,Dhaeseleer, W., 2008. Considerations on the backup of wind power : Operational backup, Applied Energy, Vol. 85, No. 9, pp.787-799.

[Shargal and Houseman. 2009] Shargal, M., Houseman, D., 2009. The Big Picture of your Coming Smart Grid, Smart Grid News, 5 March.

[Youk et al. 2014] Youk, C., Lim, H., Lee, J., 2014. A proposal of an energy efficiency management system using Big data technology, 한국정보과학회 2014 한국컴퓨터종합학술대회 논문집 (2014.6.), pp.1997-1999.

[Taft and Martini. 2012] Taft, J., Martini, P. D., 2012. Ultra Large Scale Power System Control Architecture,Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES, pp.1 - 6.

[Lee et al. 2010] Lee, Y., Yoo, M., Choi, H., Kim, Y, Seo, Y., 2010, A study on the Conceptual Design for the Real-time wind Power Prediction System in Jeju, 전기학회논문지 제59권 제12호 (2010.12.), pp.2202-2211.

[David, 2012] David, P., 2012. Lords of the Data Storm: Vestas and IBM Win Big Data Award, [Oline] he Big Data Hub: Understanding big data for the enterprise, http://www.ibmbigdatahub.com/blog/lords-data-storm-vestas-and-ibm-win-big-data-award

[Noh and Pack, 2014] Noh, K., Park, S., 2014.An Exploratory Study on Application Plan of Big Data to Manufacturing Execution System, 디지털융복합연구 제12권 제1호 (2014.1.)., pp 305-311.

[Lee et al. 2013] Lee, Y., Heo, J., Choi, Y., 2013. A Study for Space-based Energy Management System to Minimizing Power Consumption in the Big Data Environments, 한국인터넷방송통신학회 논문지 13(6), pp.229-235.

[Lee et al. 2014c] Lee, W., Lee, I., On, B., Choi, J., 2014.A Study on Big Data System to Analyze Smart Grid Energy Data , 정보과학회지 제32권 제9호 (2014.9.), pp. 35-41.

[Hwang et al. 2012] Hwang, M., Jin, C., Yun, U., Kim, K., Ryu, K., 2012. Building of Prediction Model of Wind Power Generationusing Power Ramp Rate, 한국컴퓨터정보학회논문지 17(1), 211-218.

[Lee et al 2012] Lee, J., Park, G., Kim, E., Kim, Y., Lee, I., Wind Speed Modeling based on Artificial Neural Networks for Jeju Area , IJCA Vol. 5 No. 2 (2012.6.), pp. 81-88

[Hadoop] Hadoop , onlien, http://hadoop.apache.org/

[Ghemawat et al. 2003] Ghemawat, S., Gobioff, H., Leung, S., 2003, The Google file system , SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles, pp. 29-43.

[White 2011] White, T., 2011, Hadop : The Definitve Guide, O'reily, Book,

[Geum et al. 2010] Geum, T., Kim, S., Lee, S., 2010. Hadoop 설치와 애플리케

이션의 구동, 한국컴퓨터정보학회지 제18권 제1호 (2010.6.), pp.9-16.
[Ohohta 2014] Ohohta, K., 2014, 빅 데이터 시대의 하둡 완벽 입문 오픈 소스 분산 처리 환경 구축 ,제이펍.

[Pig] Pig , online, http://pig.apache.org/

[Lee et al. 2014a] Lee, J., Lee, D., Park, G., State-of-Charge stream processing and modeling for electric vehicle-based trips, International Journal of Control and Automation (ISSN: 2005-4297), Vol. 7, No.9, pp. 399-410.

[Lee et al. 2012] Lee, J., Park, G., Lee, D., 2012. Comparative performance analysis of relocation policies for electric vehicle sharing systems, CCIS 353, pp. 195-201.

[Lee et al. 2014b] Lee, J., Lee, D., Park, G., 2014. State-of-charge Stream processing for electric vehicles, Advanced Technology Letters (Proc. International Conference on Information Technology and Computer Science), Vol. 51, pp. 242-245.

[Nissen 2005] Nissen, S., 2005, Neural network made simple, Software 2.0