<table>
<tr><td colspan="3">과제명 (Title) : 연구주제 및 방향 설정</td></tr>
<tr><td>일자</td><td>연구내용</td><td>지도
대학원생</td></tr>
<tr><td>10/16</td><td>
<h2>연구주제 및 방향 설정</h2>
가진 능력
- 파이썬(pandas, numpy, matplotlib, sklearn), R(기초, dplyr, ggplot…), 자바(기초)
- Hadoop, Spark (기초), 쿼리문 기초
- Spring Framework 통한 웹 구현(기초)
- 학부수준의 통계지식(수리통계학,회귀분석,전산통계,시계열.EDA)
- kaggle, 공공데이터 포탈 통한 데이터 확보 능력

목표
- 실제 데이터를 활용한 딥러닝 모델의 구현과 활용을 통한 딥러닝 방법의 이해와 실제적인 활용방안 탐구

구체적인 활동
1) 공모전 활동(하나에서 두개 선택해서 진행예정)
1-1) MBN 빅데이터 아이디어
- https://www.thinkcontest.com/Contest/ContestDetail.html?id=13084
- 샘플 데이터만 제공…
- ~10/21까지(아이디어 계획서)
1-2) 암(cancer) 빅데이터 활용 경진대회
- https://www.thinkcontest.com/Contest/ContestDetail.html?id=13458
- 샘플 데이터만 제공
- ~10/28까지(아이디어 계획서)
1-3) L.point 빅데이터 컴퍼티션
- https://competition.lpoint.com/front/Guideline.tran
- 주제 : 디지털 행동기반 트렌드 예측
- ~1/16까지(1차 제출)

1-4) 대학생 인공지능 아이디어 경진대회
http://www.bitle.kr/board/view.bitle?boardId=BBS_0000006&menuCd=DOM_000000104001000000&orderBy=REGISTER_DATE%20DESC&startPage=1&dataSid=2076
- ~ 10/14일까지(아이디어 계획서)

2) 자체적 활동과 결과물 산출
2-1) 자체적으로 딥러닝 공부 후 결과 보고서 제작..
</td><td></td></tr>
<tr><td colspan="3">기록자: (서명)</td></tr>
</table>

과제명 (Title) : **연구주제 구체화**

일자	연구내용	지도 대학원생
10/16		서명

현재 지하철 광고패널은 **종일 같은 광고**를 띄우거나 **무작위적**으로 광고를 띄워 **효율이 낮은 방법**으로 광고를 하고 있다. 허나 디지털 패널광고의 경우, 시간대별로 간단히 **광고를 바꿔** 틀어줄 수 있으므로 분석결과를 활용하여 타켓팅 광고를 할 경우 **훨씬 효율적**이다.

사진1. 지하철 디지털 광고 패널

Ⅰ. 유동인구 데이터의 활용

시간별 지역구별 **유동인구 데이터**를 통해, 원하는 지역(지하철역이 있는)의 시간대별 **가장 많은 연령대를 파악**하여, 해당 시간대의 **광고 타겟층을 정함**

STD_YM	X_COORD	Y_COORD	INFLOW_C	AVG_00TN	AVG_01TN	AVG_02TN	AVG_03TN	AVG_04TN	AVG_05TN	AVG_06TN	AVG_07TN	AVG_08TN	AVG_09TN	AVG_10TN	AVG_11TN	AV
201505	996234.6	1812616	47190250	0	0	0	0	0	0	0	0	0	0	0	0	
201505	996583.5	1812359	47190250	0	0	0	0	0	0	0	0	0	0	0	0	
201505	995935.1	1812103	47190250	0	0	0	0	0	0	0	0	0	0	0	0	
201505	995985.1	1812462	47190250	0	0	0	0	0	0	0	0	0	0	0	0	
201505	996084.9	1812616	47190250	0	0	0	0	0	0	0	0	0	0	0	0	
201505	995735.8	1812565	47190250	0	0	0	0	0	0	0	0	0	0	0	0	
201505	995835.2	1811898	47190250	0	0	0	0	0	0	0	0	0	0	0	0	
201505	995585.9	1812154	30140109	0.15	0.12	0.1	0.11	0.1	0.13	0.17	0.34	0.44	0.61	0.6	0.57	
201505	995436.4	1812309	42110400	0	0	0	0	0	0	0	0	0	0	0	0	
201505	996084.8	1812206	42110400	0	0	0	0	0	0	0	0	0	0	0	0	
201505	995985.1	1812462	42110400	0	0	0	0	0	0	0	0	0	0	0	0	
201505	996384	1812205	42110400	0	0	0	0	0	0	0	0	0	0	0	0	
201505	996483.6	1811795	30140118	0	0	0	0	0	0	0	0	0.02	0.01	0.01	0.01	
201505	996034.8	1812052	45130134	0	0	0	0	0	0	0	0	0	0	0	0	
201505	996483.9	1812668	30140102	0	0	0	0	0	0	0	0	0.02	0.03	0.03	0.03	
201505	996184.4	1811949	43730360	0	0	0	0	0	0	0	0	0	0.02	0.01	0	
201505	996434.1	1812565	50110250	0	0	0	0	0	0	0	0	0	0	0	0	
201505	996234.4	1812154	11200107	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	
201505	996035.1	1812668	11200107	0	0	0	0	0	0	0	0	0	0	0	0	
201505	996184.6	1812514	11200107	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	
201505	996533.8	1812668	11200107	0	0	0	0	0	0	0	0	0	0	0	0	
201505	995636	1812514	11200107	0	0	0	0	0	0	0	0	0	0	0	0	
201505	996583.7	1812822	41285108	0	0	0	0	0	0	0	0	0	0	0	0	
201505	996134.5	1812051	30170112	0.02	0.02	0.04	0.01	0	0	0.01	0.08	0.16	0.19	0.25	0.27	
201505	996633.4	1812359	30170112	0.01	0.01	0.01	0	0	0	0.01	0.02	0.07	0.09	0.11	0.13	
201505	995586	1812360	30170112	0.02	0.01	0.01	0.01	0	0	0	0.05	0.1	0.11	0.13	0.13	
201505	995635.9	1812257	30110109	0.03	0.03	0.02	0.01	0	0	0.01	0.06	0.13	0.17	0.25	0.27	
201505	995735.6	1812206	30110109	0.03	0.03	0.03	0.02	0	0	0.01	0.05	0.11	0.14	0.2	0.24	
201505	995735.6	1812257	47930380	0	0	0	0	0	0	0	0	0	0	0	0	
201505	996084.6	1812000	27290124	0	0	0	0	0	0	0	0	0	0.01	0	0	
201505	995486.3	1812360	30140109	0.15	0.12	0.11	0.1	0.09	0.12	0.17	0.38	0.5	0.7	0.64	0.65	
201505	995685.8	1812360	41131112	0	0	0	0	0	0	0	0	0	0	0	0	
201505	995984.7	1811589	41197103	0	0	0	0	0	0	0	0	0	0	0	0	

사진 2. 시간대별 유동인구 데이터

Ⅱ. 시계열 데이터의 분석

01~24 시까지 **시간별로 가장 많은 연령층이 무엇인지 파악**하기 위해 시간대별로 변화패턴을 보이는 **시계열자료를 예측하기 위한 모델**이 필요하다.

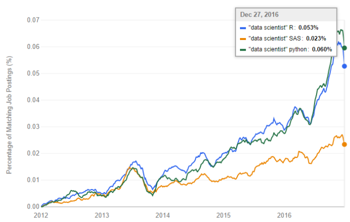


사진 3. 시계열 자료

Ⅲ. 시계열 예측을 위한 LSTM 모델

RNN 모델(순환신경망 모델)의 발전된 형태인 **LSTM(Long Short Term Memory networks) 모델은 이러한 시계열 자료의 예측에 뛰어난 성능**을 보인다. 따라서 LSTM 을 이용하여 시계열자료인 유동인구데이터를 분석, 예측하여 이를 기반으로 **해당시간의 광고 타겟층을 파악**한다.

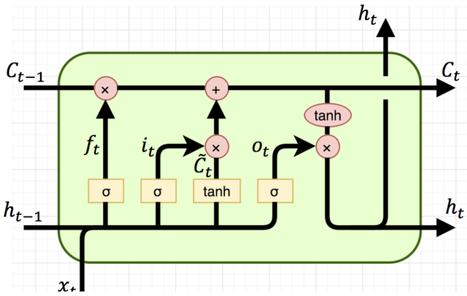


사진 4. LSTM 모델의 구조

Ⅳ. 모델의 작동 과정

1) 데이터 수집

- **API를 통한 데이터수집**과 Feature engineering **과정**을 파이썬 모듈화 하여 **주기적**으로 **크로울링**한다.

2) LSTM 모델의 학습

- api를 통한 데이터수집과 Feature engineering **과정**을 파이썬 모듈화 하여 **주기적**으로 **크로울링**한다.
- 데이터를 이용하여 **모델을 학습**시킨다.

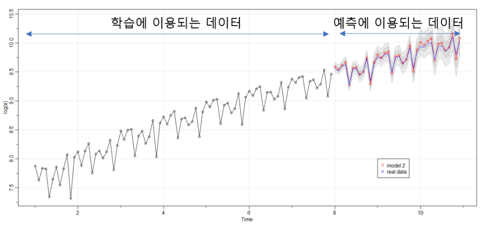


사진 5. 학습과정

Ⅴ. 학습된 모델 통한 예측 및 광고 추전

학습된 모델을 통해 예측된 유동인구를 바탕으로 해당 시간에 **가장 많은 연령층을 타겟으로 맞춤 광고**를 추천

08시	09시	10시	11시
10대 남 우세	20대 여 우세	40대 남 우세	60대 남녀 우세
게임, 공부	화장품	은행 상품	여행 패키지

표 1. 결과의 예시

기록자: (서명)

과제명 (Title) : LSTM paper review

일자	연구내용	지도 대학원생
10/23		

A Critical Review of Recurrent Neural Networks for Sequence Learning

Zachary C. Lipton
zlipton@cs.ucsd.edu

John Berkowitz
jaberkow@physics.ucsd.edu

Charles Elkan
elkan@cs.ucsd.edu

June 5th, 2015

Sequence 형태의 데이터를 위한 RNN model

Introduction

Why not use markov models

연산량이 너무 많다.

각 간격마다, 'state space'(상태공간)가 필요하다.
Sep가 길어질수록 알고리즘의 **시간복잡도**가
크게 증가 ➡ $O(N^2 T)$

$$P = \begin{pmatrix} 0 & \frac{4}{5} & 0 & \frac{1}{5} & 0 \\ \frac{1}{4} & 0 & \frac{1}{2} & \frac{1}{4} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{10} & \frac{2}{5} \\ 0 & 0 & 0 & 1 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}$$

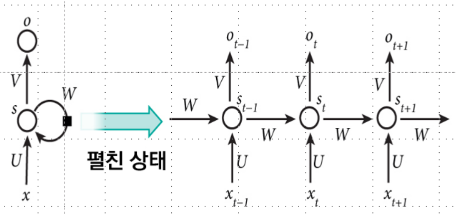
$$x_t \xrightarrow{\text{P(전이행렬)}} x_{t+1}$$

Recurrent neural network

RNN의 학습

BPTT
Backpropagation through time

기존의 **feedforward 방식**의 신경망에 적용되는 **'Backpropagation'** 알고리즘을 **펼친 상태의** RNN신경망에 동일하게 적용



기록자: (서명)

과제명 (Title) : LSTM paper review-2

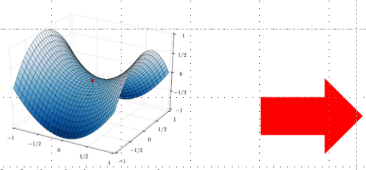
일자	연구내용	지도 대학원생

10/30

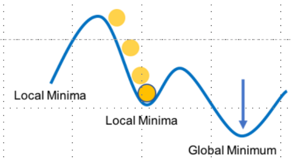
Recurrent neural network

문제점

Issue of local optima



n이 커짐에 따라 error의 Saddle point의 수가 기하급수적으로 **증가**

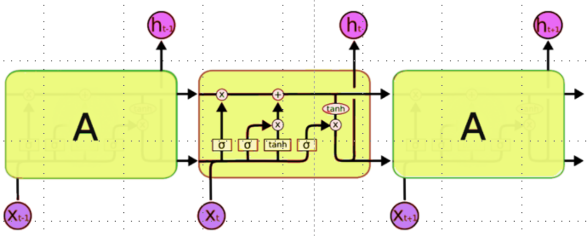


따라서 **지역해**에 수렴하기 쉬워져 **학습속도가 매우 느려진다.**

Modern RNN architectures

LSTM(Long short-term memory)

'memory cell'이라는 개념을 도입하여 **'vanishing gradient' 문제를 해결**하여 **긴term의 연관**을 가진 data를 **학습시키기에 유리**하다



Modern RNN architectures

Memory cell의 node와 gate

Input node g_c

x_t와 h_{t-1}을 받은 뒤 , tanh 활성함수를 거치는 구간

Input gate i_c

x_t와 h_{t-1}을 받은 뒤 , **sigmoid 활성함수를** 거치기 때문에 **0~1사이의 값**을 가진다. Input node 값에 곱해져 **신호를 자르거나 보내주는 gate의 역할**을 한다

Internal state s_c

x_t와 i_t를 원소끼리 곱해준 뒤 , 직전값 (s_{t-1},이를 **recurrent edge**를 가졌다고 표현) 을 더해준다.

이때 **W=1을 고정적**으로 가지기 때문에 **'gradient vanishing' 문제를 보이지 않은다.**

Forget gate f_c

$Internal\ state$를 계산할 때, s_{t-1}에 **곱해져 이전의 기억들을 잊게 해주는 역할**을 한다. **계속해서 새로운 데이터를 학습해야 하는 경우 유용**하다

Output gate o_c

Memory cell의 최종적인 출력값을 내보낼 때, s_t에 곱해진다. 이때 별도의 활성함수를 거치지 않는다.

기록자: (서명)

과제명 (Title) : 데이터 전처리-1		
일자	연구내용	지도 대학원생

11/2

simulation
100jy
2019 11 2
전체소스코드 : https://100jy.github.io/lstm_com/%EB%A7%88%ED%81%AC%EB%8B%A4%EC%9A%B4

가. data simulation(유동인구)
data simulation…

```
setwd("C:/Users/wnduq/Desktop")
data1<-read.csv("월평균 시간대별 유동인구.csv")
data2<-read.csv("월평균 연령별 유동인구.csv")
data3<-read.csv("월평균 요일별 유동인구.csv")
data4<-read.csv("월평균 성별 유동인구.csv")


##부전1동
data_time<-data1[which(data1$분석영역=="부전1동"),]
data_old<-data2[which(data2$분석영역=="부전1동"),]
data_days<-data3[which(data3$읍면동=="부전제1동"),]
data_sex<-data4[which(data4$읍면동=="부전제1동"),]
########################################################


###16년 화요일 missing(보간법)
weds<-round((data_days[which(data_days$년==2016&data_days$요일=="월"),]$평균.유동인구+

data_days[which(data_days$년==2016&data_days$요일=="수"),]$평균.유동인구)/2)

년=rep(2016,12)
구군=rep("부산진구",12)
읍명동=rep("부전제1동",12)
요일="화"
월=1:12
주중.주말=rep("주중",12)
평균.유동인구=weds
add=cbind(년,구군,읍명동,요일,월,주중.주말,평균.유동인구)
colnames(add)<-colnames(data_days)

data_days=rbind(data_days,add)

#######################################################################
(중략)
#############################################

factor=factor(rep(월,4),levels=name)
#####날짜 달아주기
##여기서부터 문제있음...
n=25536
시간=rep(rep(0:23,rep(7,24)),n/(7*24))
주차=rep(1:4,c(n/4,n/4,n/4,n/4))
요일=rep(1:7,n/7)
부전1동_유동_plat=cbind(c(tmp[,1],tmp[,2],tmp[,3],tmp[,4]),rep(월,4),주차,시간,요일)
dim(부전1동_유동_plat)
## [1] 25536     5
#24*28*(12*3+2)
uo=부전1동_유동_plat
uo=as.data.frame(uo)
str(uo)
## 'data.frame':    25536 obs. of  5 variables:
##  $ V1  : Factor w/ 25536 levels "14140.6349032871",..: 9755 9614 9370 9378 9557 9717 9559 9486 9282 9507 ...
##   ..- attr(*, "names")= chr  "" "" "" "" ...
##  $ V2  : Factor w/ 38 levels "201601","201602",..: 1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "names")= chr  "" "" "" "" ...
##  $ 주차: Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "names")= chr  "" "" "" "" ...
##  $ 시간: Factor w/ 24 levels "0","1","10","11",..: 1 1 1 1 1 1 1 2 2 2 ...
##   ..- attr(*, "names")= chr  "" "" "" "" ...
##  $ 요일: Factor w/ 7 levels "1","2","3","4",..: 1 2 3 4 5 6 7 1 2 3 ...
##   ..- attr(*, "names")= chr  "" "" "" "" ...
uo$V2=factor
uo$시간=as.integer(as.character(uo$시간))
부전1동_유동_plat=uo%>%
  arrange(V2,주차,요일,시간)
colnames(부전1동_유동_plat)[1]="유동인구"
부전1동_유동_plat$유동인구=as.integer(as.character(부전1동_유동_plat$유동인구))
time=1:25536
부전1동_유동_plat$time=time
24*12*28
## [1] 8064
```

기록자: (서명)

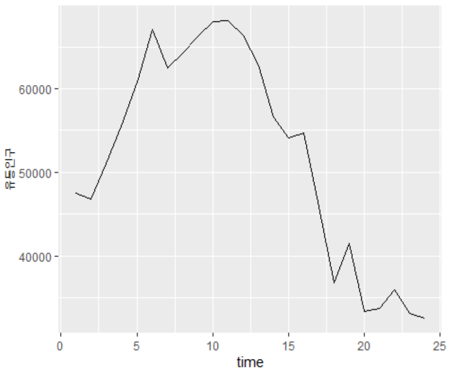
과제명 (Title) : 데이터 전처리-2

일자	연구내용	지도 대학원생

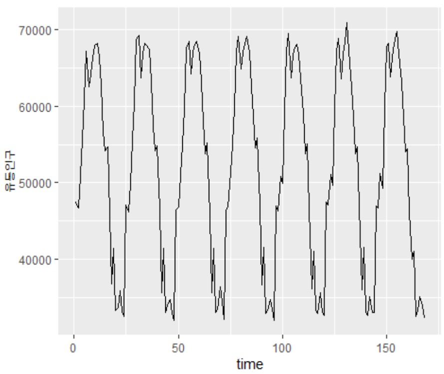
11/6

simulation
100jy
2019 11 6
전체소스코드 : https://100jy.github.io/lstm_com/%EB%A7%88%ED%81%AC%EB%8B%A4%EC%9A%B4

```
x11()
library(ggplot2)
###하루치
x11()
ggplot(부전1동_유동_plat[1:24,], aes(x=time, y=유동인구)) +
  geom_line()
```




```
## 일주일치
x11()
ggplot(부전1동_유동_plat[1:(24*7),], aes(x=time, y=유동인구)) +
  geom_line()
```



기록자: (서명)

과제명 (Title) : 데이터 전처리-3

일자	연구내용	지도 대학원생

11/12

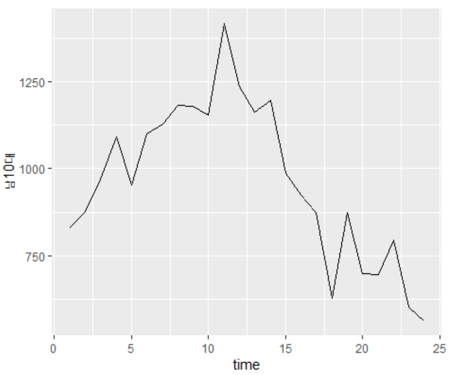
simulation
100jy

전체소스코드 : https://100jy.github.io/lstm_com/%EB%A7%88%ED%81%AC%EB%8B%A4%EC%9A%B4

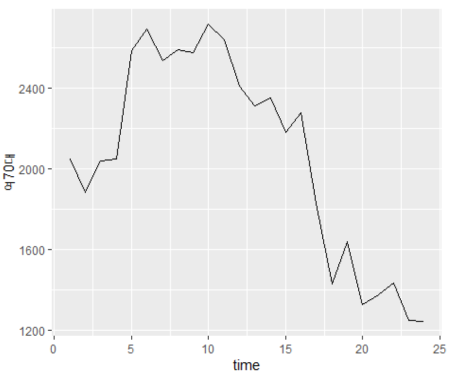
```
###################################################################################
###로그변환 해주면 좋을듯...
###새벽3 4 5시에 사람이 많은?????
#############성별연령별 컬럼 나누기
###일단위로 숫자뽑은뒤 비율로 변환...
###일단위로 랜덤하게 뽑기
###노이즈 섞어서 늘리고 이후에 비율화....
data sex<-data4[which(data4$읍면동=="부전제1동"),]
data old<-data2[which(data2$분석영역=="부전1동"),]
data old=data old[-c(1:27),]
##################
data sex$남자.전체.유동인구=as.integer(as.character(data sex$남자.전체.유동인구))
data sex$여자.전체.유동인구=as.integer(as.character(data sex$여자.전체.유동인구))
data sex$남성비율=(data sex$남자.전체.유동인구)/((data sex$남자.전체.유동인구)+(data sex$여자.전
체.유동인구))
data sex$여성비율=rep(1,38)-(data sex$남성비율)
p1=cbind(data sex$남성비율,data sex$여성비율)
p1=cbind(p1,p1,p1,p1,p1,p1,p1)
temp=cbind(data old[,3],data old[,3],data old[,4],data old[,4],data old[,5],data old[,5],
           data old[,6],data old[,6],data old[,7],data old[,7],data old[,8],data old[,8],
           data old[,9],data old[,9])*p1
data old=cbind(data old[,c(1:2)],temp)
colnames(data old)[c(3:16)]=c("10대남","10대여","20대남","20대여","30대남","30대여",
                              "40대남","40대여","50대남","50대여","60대남","60대여",
                              "70대남","70대여")

#############################################
gnor=function(x){
  rnorm(14*24*28,rep(as.integer(x[-c(1:2)]),rep(24*28,14)),50)
}
tmp=apply(data old,MARGIN = 1,FUN = gnor)
dim(tmp)
```

```
## [1] 9408   38
```

```
####(중략)

colnames(부전1동 유동 last)=c("남10대","여10대","이십대남","잇비대여","30대남","30대여",
                          "40대남","40대여","50대남","50대여","60대남","60대여",
                          "70대남","여70대","time")
부전1동 유동 last=as.data.frame(부전1동 유동 last)

###하루치(10대남)
x11()
ggplot(부전1동 유동 last[1:24,], aes(x=time, y= 남10대)) +
  geom_line()
```




```
x11()
ggplot(부전1동 유동 last[1:24,], aes(x=time, y= 여70대)) +
  geom_line()
```



기록자: (서명)

과제명 (Title) : 모델링

일자	연구내용	지도 대학원생
11/17	(아래 참조)	

전체 소스코드 : https://github.com/100jy/lstm_com/blob/master/pred._people_simple_ver..ipynb

LSTM모델을 통한 유동인구 예측(simple_ver.)

간단한 구조의 LSTM model을 활용하여 유동인구 데이터를 예측

In [ ]:
```
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import pandas as pd
import os
from sklearn.preprocessing import MinMaxScaler
```

In [31]:
```
from tensorflow.python.keras.layers import LSTM
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
from tensorflow.python.keras import backend as K
from tensorflow.python.keras import optimizers
from tensorflow.python.keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard, ReduceL
ROnPlateau
```

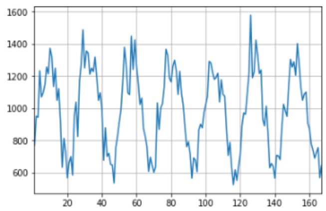
In [50]:
```
df=pd.read_csv("C:/Users/wndug/Desktop/유동인구_부전1동.csv",encoding='euc-kr')
df.shape
```

Out[50]: (25536, 17)

In [ ]:
```
df["남10대"].plot()
df["남50대"].plot()
df["여20대"].plot()
```

In [51]:
```
#일주일치
df["남10대"][1:24*7].plot(grid=True)
```

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x218c69e4f60>



살짝 과적합되는 느낌... (1. 배치사이즈 조절

1. dropout 적용
2. ...)

In [63]:
```
##데이터
#scaling
scaler = MinMaxScaler(feature_range=(0, 1))
tmp_data = scaler.fit_transform(df["남10대"][:,None])

#2일치로 1시간 예측(슬라이드 윈도우)
look_back = 24*2

def create_dataset(dataset, look_back):
    dataX, dataY = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i + look_back)]
        dataX.append(a)
        dataY.append(dataset[i + look_back])
    return np.array(dataX), np.array(dataY)

data={}
data=create_dataset(tmp_data,look_back)

x_data=data[0]
y_data=data[1]
num_data = len(x_data)



##test,train 분할(1년치만 학습)

num_train = 24*28*24
num_test = 24*28*1

x_train = x_data[0:num_train]
x_test = x_data[num_train:num_train+num_test-1]

y_train = y_data[0:num_train]
y_test = y_data[num_train:num_train+num_test-1]

num_x_signals = x_data.shape[1]
num_x_signals

num_y_signals = 1
num_y_signals
```

기록자: (서명)

과제명 (Title) : 모델링-2		
일자	연구내용	지도 대학원생
11/20	전체 소스코드 : https://github.com/100jy/lstm_com/blob/master/pred._people_simple_ver..ipynb	

```
##모델링(simple_ver.)


K.clear_session()

model = Sequential() # Sequeatial Model
model.add(LSTM(20, input_shape=(24+2, 1))) # (timestep, feature)
model.add(Dense(1)) # output = 1
model.compile(loss='mean_squared_error', optimizer='adam')
model.summary()


x_train = x_train.reshape(x_train.shape[0],24+2,1)


##callback
early_stop = EarlyStopping(monitor='loss', patience=1, verbose=1)
callback_tensorboard = TensorBoard(log_dir='./last_M/',histogram_freq=0,write_graph=True)


model.fit(x_train, y_train, epochs=100,
          batch_size=50, verbose=1, callbacks=[early_stop,callback_tensorboard])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 20)                1760
_________________________________________________________________
dense (Dense)                (None, 1)                 21
=================================================================
Total params: 1,781
Trainable params: 1,781
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
16128/16128 [==============================] - 42s 3ms/sample - loss: 0.0112
Epoch 2/100
16128/16128 [==============================] - 43s 3ms/sample - loss: 0.0043
Epoch 3/100
16128/16128 [==============================] - 45s 3ms/sample - loss: 0.0031
Epoch 4/100
16128/16128 [==============================] - 42s 3ms/sample - loss: 0.0022
Epoch 5/100
16128/16128 [==============================] - 42s 3ms/sample - loss: 0.0020
Epoch 6/100
```

학습이후 예측값과 실제값을 비교하여 본다.

그림을 통한 비교

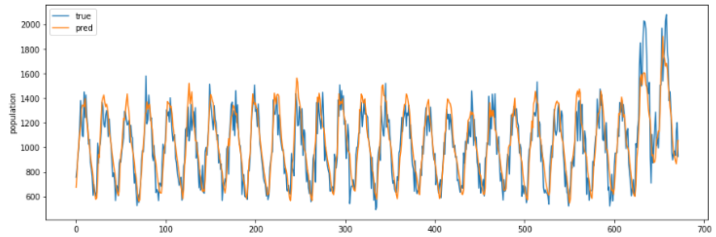
In [70]:

```
##예측

def plot_comparison(start_idx, length, train):

    if train:
        x = x_train
        y_true = y_train
    else:
        x = x_test.reshape(x_test.shape[0],24+2,1)
        y_true = y_test
    end_idx = start_idx + length
    x = x[start_idx:end_idx]
    y_true = y_true[start_idx:end_idx]
    y_pred = model.predict(x)


    signal_pred = scaler.inverse_transform(y_pred)
    signal_true = scaler.inverse_transform(y_true)
    plt.figure(figsize=(15,5))
    plt.plot(signal_true, label='true')
    plt.plot(signal_pred, label='pred')

    plt.ylabel("population")
    plt.legend()
    plt.show()

###training set에대해
plot_comparison(0, length=24*28, train=True)
```

기록자: (서명)

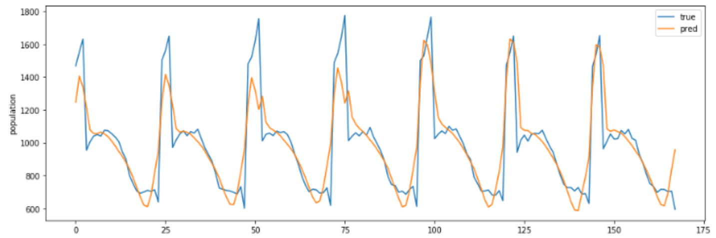
과제명 (Title) : 모델링-3

일자	연구내용	지도 대학원생
11/24	**전체 소스코드** : https://github.com/100jy/lstm_com/blob/master/pred._people_simple_ver..ipynb	

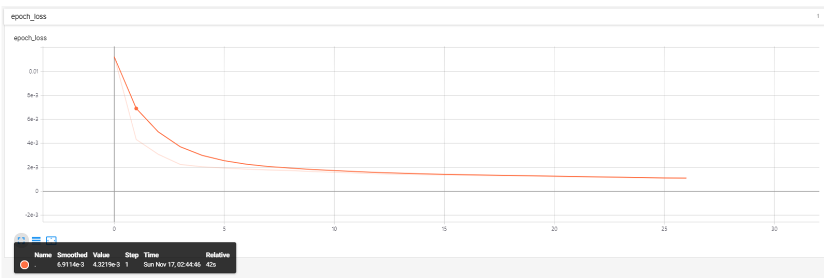


training set에 대한 적합을 본다.

In [69]:
```
###predict
plot_comparison(0, length=24*7, train=False)
```



test set에 대한 predict값과 실제값을 비교하여 본다.



기록자: (서명)

과제명 (Title) : 모델링-4

일자	연구내용	지도 대학원생
11/27	(see below)	

전체 소스코드 : https://github.com/100jy/lstm_com/blob/master/pred._people_simple_ver..ipynb

import modules

In [1]:
```
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import pandas as pd
import os
from sklearn.preprocessing import MinMaxScaler
```

In [2]:
```
from tensorflow.python.keras.layers import LSTM
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
from tensorflow.python.keras import backend as K
from tensorflow.python.keras import optimizers
from tensorflow.python.keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard, ReduceLROnPlat
eau
```

유동인구 data 불러오기

In [16]:
```
df=pd.read_csv("C:/Users/wnduq/Desktop/유동인구_최종/유동인구_부전1동.csv",encoding='euc-kr')
df
```

Out[16]:

	Unnamed: 0	남10대	여10대	남20대	여20대	남30대	여30대	남40대
0	1	905.809767	867.335419	4223.613759	4277.240753	4180.953422	4115.471699	4088.688921
1	2	915.703151	826.247217	4280.180663	4325.748017	4017.522110	4218.581633	4161.656972
2	3	931.680078	1219.264692	4464.528678	4539.528097	4431.449251	4490.067514	4167.246254
3	4	1082.323717	1134.101183	4988.650671	4997.976477	4934.094061	4870.464613	4540.073216
4	5	1300.297861	975.962514	5307.215472	5343.265054	5183.591120	5131.022234	5168.861198
...	...	...	...	...	...	...	...	...
25531	25532	942.032794	962.913564	3963.286317	4052.804238	3696.428185	3781.943405	3456.584867
25532	25533	918.533195	940.286290	3857.965039	3947.819767	3601.931740	3684.245235	3364.934030
25533	25534	755.560838	773.292828	3174.366754	3246.801462	2962.212012	3028.732201	2769.674694
25534	25535	648.474448	663.432829	2722.174284	2785.433673	2540.465927	2598.176705	2373.822485
25535	25536	561.792929	574.673690	2363.368697	2415.689786	2204.094138	2254.964674	2060.085687

25536 rows × 17 columns

입력데이터,출력데이터 생성

48시간 간격의 sliding window 생성, minmax saciling으로 학습속도 높인다.

In [17]:
```
#feature생성..

scaler = MinMaxScaler(feature_range=(0, 1))
df = pd.DataFrame(scaler.fit_transform(np.array(df.iloc[:,1:15])))


def create_dataset(dataset, look_back):

    total=[]
    x_data=pd.DataFrame(np.zeros((25487,48)))
    y_data=pd.DataFrame(np.zeros((25487,1)))


    for j in range(14) :

        tmp_data = np.array(dataset.iloc[:,j:j+1])
        dataX, dataY = [], []
        for i in range(len(tmp_data)-look_back-1):
            a = tmp_data[i:(i + look_back)]
            dataX.append(a)
            dataY.append(tmp_data[i + look_back])

        x=pd.DataFrame((np.array(dataX).reshape(25487,48)))
        y=pd.DataFrame((np.array(dataY).reshape(25487,1)))
        x_data=pd.concat([x_data,x],axis=1)
        y_data=pd.concat([y_data,y],axis=1)


    return  x_data,y_data

#2일치로 1시간 예측(슬라이드 윈도우)
look_back = 24*2
temp=create_dataset(df,look_back)
x_data=temp[0]
y_data=temp[1]


x_data.columns=range(720)
y_data.columns=range(15)

x_data=np.array(x_data.drop(x_data.columns[0:48], axis='columns'))
y_data=np.array(y_data.drop(y_data.columns[[0]], axis='columns'))
x_data.shape,y_data.shape
```

Out[17]: ((25487, 672), (25487, 14))

기록자: (서명)

과제명 (Title) : 모델링-5

일자	연구내용	지도 대학원생

11/30

전체 소스코드 : https://github.com/100jy/lstm_com/blob/master/pred._people_simple_ver..ipynb

모델생성 및 학습

trainset(2년),testset(2개월)으로 LSTM모델학습, 배치당 20분소요...

In [19]:

```
##test,train 분할(2년치만 학습)

num_train = 24*28*24
num_test = 24*28*2

x_train = x_data[0:num_train]
x_test = x_data[num_train:num_train+num_test-1]

y_train = y_data[0:num_train]
y_test = y_data[num_train:num_train+num_test-1]

num_x_signals = x_data.shape[1]
num_x_signals

num_y_signals = 1
num_y_signals



##모델링(simple_ver.)



K.clear_session()

model = Sequential() # Sequestial Model
model.add(LSTM(512, input_shape=(24*2*14,1))) # (timestep*feature,1)
model.add(Dense(14)) # output = 14
model.compile(loss='mean_squared_error', optimizer='adam')
model.summary()



x_train = x_train.reshape(x_train.shape[0],24*2*14,1)


##callback
early_stop = EarlyStopping(monitor='loss', patience=1, verbose=1)
callback_tensorboard = TensorBoard(log_dir='./last_f/',histogram_freq=0,write_graph=True)


model.fit(x_train, y_train, epochs=100,
          batch_size=50, verbose=1, callbacks=[early_stop,callback_tensorboard])#gpu메모리부족 batch_size=5
0한계...
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 512)               1052672
_________________________________________________________________
dense (Dense)                (None, 14)                7182
=================================================================
Total params: 1,059,854
Trainable params: 1,059,854
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
16128/16128 [==============================] - 1312s 81ms/sample - loss: 0.0169
Epoch 2/100
16128/16128 [==============================] - 1297s 80ms/sample - loss: 0.0041
Epoch 3/100
16128/16128 [==============================] - 1297s 80ms/sample - loss: 0.0026
Epoch 4/100
16128/16128 [==============================] - 1296s 80ms/sample - loss: 0.0024
Epoch 5/100
16128/16128 [==============================] - 1297s 80ms/sample - loss: 0.0023
Epoch 6/100
16128/16128 [==============================] - 1299s 81ms/sample - loss: 0.0021
Epoch 7/100
16128/16128 [==============================] - 1300s 81ms/sample - loss: 0.0019
Epoch 8/100
16128/16128 [==============================] - 1297s 80ms/sample - loss: 0.0016
Epoch 9/100
16128/16128 [==============================] - 1297s 80ms/sample - loss: 0.0016
Epoch 10/100
16128/16128 [==============================] - 1296s 80ms/sample - loss: 0.0015
Epoch 11/100
16128/16128 [==============================] - 1297s 80ms/sample - loss: 0.0013
Epoch 12/100
16128/16128 [==============================] - 1295s 80ms/sample - loss: 0.0014
Epoch 00012: early stopping
```

19]: <tensorflow.python.keras.callbacks.History at 0x25a26c283c8>

기록자: (서명)

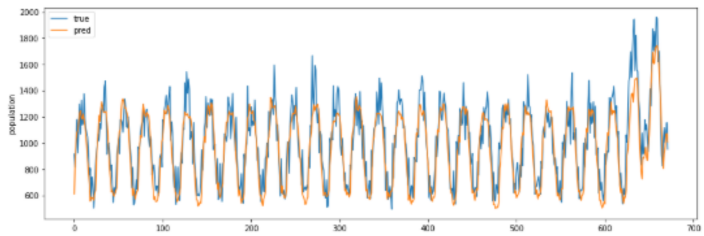
과제명 (Title) : 모델링-6

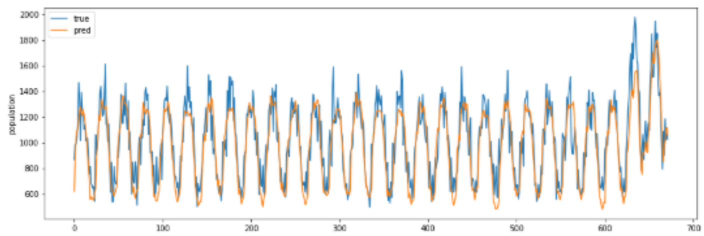
일자	연구내용	지도 대학원생
12/5	전체 소스코드 : https://github.com/100jy/lstm_com/blob/master/pred._people_simple_ver..ipynb	

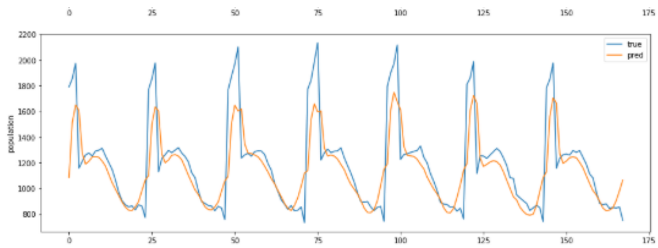
ploting 통한 예측력 확인

trainingset에 대한 적합, prediction의 정확도 확인

In [22]:
```
##예측

def plot_comparison(start_idx, length, train):

    if train:
        x = x_train
        y_true = y_train
    else:
        x = x_test.reshape(x_test.shape[0],24*2*14,1)
        y_true = y_test
    end_idx = start_idx + length
    x = x[start_idx:end_idx]
    y_true = y_true[start_idx:end_idx]
    y_pred = model.predict(x)
    y_pred=scaler.inverse_transform(y_pred)
    y_true=scaler.inverse_transform(y_true)

    for signal in range(14):
        signal_pred = y_pred[:,signal]
        signal_true = y_true[:,signal]
        plt.figure(figsize=(15,5))
        plt.plot(signal_true, label='true')
        plt.plot(signal_pred, label='pred')

        plt.ylabel("population")
        plt.legend()
        plt.show()


###training set에대해
plot_comparison(0, length=24*28, train=True)
###predict
plot_comparison(0, length=24*7, train=False)
```







모델저장

In [24]:
```
from tensorflow.python.keras.models import load_model
model.save("LSTM_COM_MODEL_부전1동.h5")
```

기록자: (서명)

과제명 (Title) : 시계열 분석

일자	연구내용	지도 대학원생

12/9

전체 소스코드
: https://github.com/100jy/lstm_com/

longterm_timedependecy

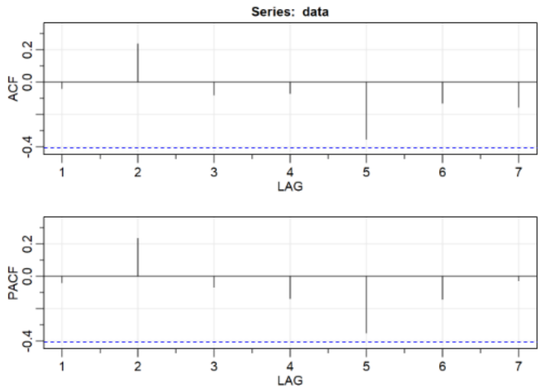
100jy

2019 11 16

ACF그림

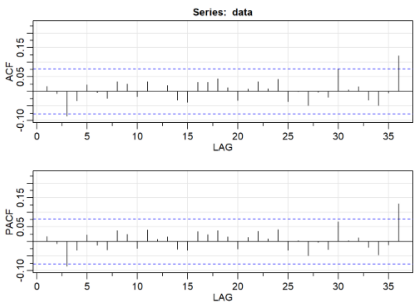
긴term의 time lag을 가진 dependency의 존재확인

```
library("astsa")
setwd("C:/Users/wnduq/Desktop")
부전1동_유동_last<-read.csv("유동인구_부전1동.csv")

#하루간 데이터의 상관(패턴존재x)
data=부전1동_유동_last$label[1:24]
data=data+rnorm(24,0,0.1)
acf2(data)
```




```
##        ACF  PACF
## [1,] -0.04 -0.04
## [2,]  0.24  0.23
## [3,] -0.08 -0.07
## [4,] -0.07 -0.14
## [5,] -0.35 -0.35
## [6,] -0.13 -0.14
## [7,] -0.15 -0.03
```

```
#한달간 데이터의 상관(제법 유의한 상관있을 것으로 보이는 lag이 보임..)
data=부전1동_유동_last$label[1:(24*28)]
data=data+rnorm(24*28,0,0.1)
acf2(data)
```

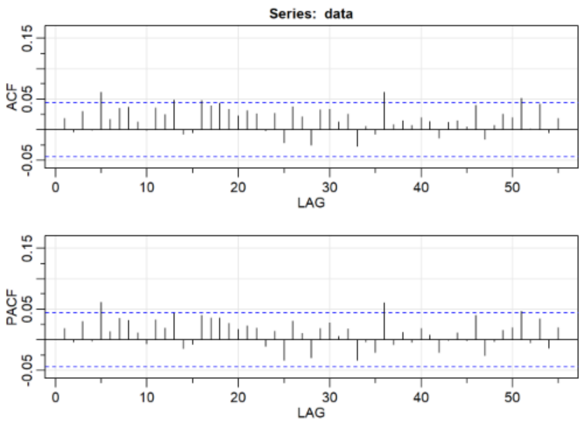



```
##         ACF  PACF
##  [1,]  0.02  0.02
##  [2,] -0.01 -0.01
##  [3,] -0.09 -0.09
##  [4,] -0.03 -0.03
##  [5,]  0.02  0.02
##  [6,]  0.00 -0.01
##  [7,] -0.02 -0.03
##  [8,]  0.03  0.04
##  [9,]  0.03  0.02
## [10,] -0.02 -0.02
## [11,]  0.03  0.04
## [12,]  0.00  0.01
## [13,]  0.02  0.02
## [14,] -0.03 -0.03
## [15,] -0.04 -0.03
## [16,]  0.03  0.03
```

기록자: (서명)

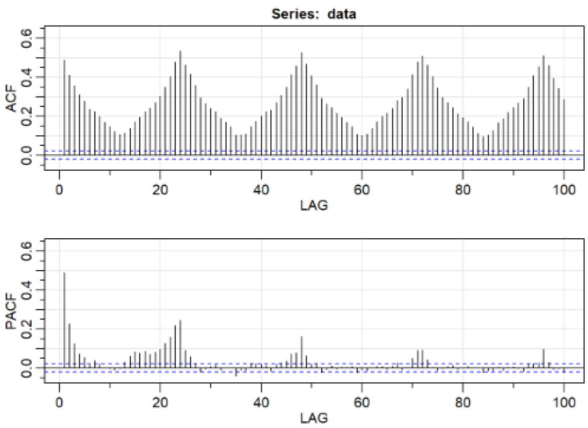
과제명 (Title) : 시계열 분석-2

일자	연구내용	지도 대학원생
12/10	전체 소스코드 : https://github.com/100jy/lstm_com/	




```
##        ACF  PACF
## [1,]  0.02  0.02
## [2,]  0.00  0.00
## [3,]  0.03  0.03
## [4,]  0.00  0.00
## [5,]  0.06  0.06
## [6,]  0.02  0.01
```

```
#일변간 데이터의 상관(긴term을 가지고 뚜렷한 상관을 보임)
data=부천1동_유동_last$label[1:(24*28*12)]
data=data+rnorm(24*12*28,0,0.1)
acf2(data)
```




```
##       ACF  PACF
## [1,] 0.49  0.49
## [2,] 0.41  0.23
## [3,] 0.36  0.13
## [4,] 0.31  0.07
## [5,] 0.26  0.05
## [6,] 0.23  0.01
## [7,] 0.22  0.04
## [8,] 0.20  0.02
```

기록자: (서명)