Ghi chú của một coder

Vũ Anh

Tháng 01 năm 2018

Mục lục

Phần I

Khoa học dữ liệu

Chương 1

Data Science with Python

View online http://magizbox.com/training/ml$_d$ata$_p$ython/site/

The ability to analyze data with Python is critical in data science. Learn the basics, and move on to create stunning visualizations.

1.1 Get Started

Get Started with Ubuntu Requirements

numpy, scipy matplotlib pandas scikit-learn ipython Install pip

sudo apt-get install python-pip Install numpy scipy

sudo apt-get install python-numpy python-scipy python-matplotlib python-pandas python-sympy python-nose Install scikit-learn

pip install jupyter ipython pip install -U scikit-learn

1.2 Data Transformation

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dict of Series objects. It is generally the most commonly used pandas object

Create data frame Create new data frame from lists

import pandas as pd students = pd.DataFrame('name' : ["Kate", "John", "Tom", "Mark"], 'age' : [20, 21, 19, 18]) age name 0 20 Kate 1 21 John 2 19 Tom 3 18 Mark Load dataframe Load dataframe from datasets

import pandas as pd from sklearn import datasets iris$_d$ata = $datasets.load_iris()iris = pd.DataFrame(data = iris_data.data, columns = iris_data.feature_names)irisSelectionSelectbycolumnindex$

students.iloc[1:3, :] age name 1 21 John 2 19 Tom Filter students = pd.DataFrame('math' : [90, 80, 95, 50], 'physic' : [20, 50, 95, 60]) math physic 0 90 20 1 80 50 2 95 95 3 50 60 students[students['math'] > 85] math physic 0 90 20 2 95 95

students[students['math'] == students['physic']] math physic 2 95 95 Create new column students = pd.DataFrame('name' : ["Kate", "John", "Tom", "Mark"], 'age' : [20, 21, 19, 18]) students["birthyear"] = students.apply(lambda row: 2016 - row['age'], axis=1) students["birthyear"] = 2016 - students["age"]

age name birthyear 0 20 Kate 1996 1 21 John 1995 2 19 Tom 1997 3 18 Mark 1998 Delete column students = pd.DataFrame('name' : ["Kate", "John",

"Tom", "Mark"], 'age' : [20, 21, 19, 18]) students = students.drop('age', 1) References Wes McKinney, 10-minute tour of pandas: video, notebook DataFrame, Intro to Data Structures

1.3 Data Preperation

Normalization Example

import numpy from sklearn.preprocessing import normalize matrix = numpy.arange(0,27,3).reshape(3,3).a

array([[0., 3., 6.], [9., 12., 15.], [18., 21., 24.]])

normed$_matrix = normalize(matrix, axis = 1, norm = 'l1')$

[[0. 0.33333333 0.66666667] [0.25 0.33333333 0.41666667] [0.28571429 0.33333333 0.38095238]] Label Encoder Encode labels (categorical variables) with value between 0 and n$_classes - 1$.

import sklearn le = sklearn.preprocessing.LabelEncoder() le.fit(["paris", "paris", "tokyo", "amsterdam"]) LabelEncoder() list(le.classes$_)['amsterdam', 'paris', 'tokyo']le.transform([1tokyo], 1to$
$dimensionalnumpyarrayinpythonlessverbose?sklearn.preprocessing.LabelEncoder$

1.4 Data IO

This post shows how to import data to Python from numerous resources

CSV Read a csv file from local or from a server

import numpy as np import pandas as pd read data df = pd.read$_csv(1data.csv], header =$ $0)writedatadf.to_csv(1data.csv], header = 1, index = False)Excelimportpandasaspdreaddatadf =$ $pd.read_excel(1data.xls])writedatadf = pd.to_excel(1data.xls], index = False)Sqliteimportsqlite3$

DB$_NAME = 1db.sqlite3]SELECT_QUERY = 1SELECTpage_id, typeFROMservice_page]connecttosqlite$ $sqlite3.connect(DB_NAME)excutequerycursor = db_connector.execute(SELECT_QUERY)returndatasetdat$ $cursor.fetchall()Referencespandas.read_excelpandas.read_sqlitesqlite3.read_sqlite$

1.5 Numpy

NumPy Use the following import convention:

import numpy as np Creating Arrays a = np.array([1, 2, 3]) b = np.array([(1.5, 2, 3), (4, 5, 6)], dtype=float) c = np.array([[(1.5, 2, 3), (4, 5, 6)], [(3, 2, 1), (4, 5, 6)]], dtype=float) Initial Placeholders Create an array of zeros np.zeros((3, 4)) array([[0., 0., 0., 0.], [0., 0., 0., 0.], [0., 0., 0., 0.]]) Create an array of ones np.ones((2, 3, 4), dtype=np.int16)

array([[[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]],

[[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]]], dtype=int16) Create an array of evenly spaced values (step value) np.arange(10, 25, 5) array([10, 15, 20]) Create an array of evenly spaced values (number of samples) np.linspace(0, 2, 9) array([0. , 0.25, 0.5 , 0.75, 1. , 1.25, 1.5 , 1.75, 2.]) Create a constant array np.full((2, 2), 7) C:2-packages.py:301: FutureWarning: in the future, full((2, 2), 7) will return an array of dtype('int32') format(shape, fill$_value, array(fill_value).dtype), FutureWarning)array([[7., 7.], [7., 7.]$
$np.array([(1, 2), (3, 4)])b = np.array([(5, 6), (7, 8)])np.save('my_array', a)np.savez('arrays', a, b)np.load('arra$
$], 1)array([[1., 2., 3.], [4., 5., 6.]])a = np.array([(1.5, 2, 3), (4, 5, 6)], dtype = float)np.savetxt(1myarray.txt], a, a$
$]1)DataTypesSigned64-bitintegertypesnp.int64Stardarddouble-precisionfloatingpointnp.float32Comple$
$lengthstringtypenp.string_Fixed-lengthunicodetypenp.unicode_numpy.unicode_InspectingYourArraya=np.array([(1.5,2,3),(4$

a = np.random.random((3, 3)) a array([[0.07989823, 0.4180309 , 0.83932547], [0.06318651, 0.20509151, 0.08262809], [0.64938826, 0.531026 , 0.38633983]]) select the element at the 2nd index a[2] array([0.64938826, 0.531026 , 0.38633983]) select the element at row 0 column 2 a[1][2] a[1, 2] 0.08262808937797228 Slicing

select items at index 0 and 1 a[0:2] array([[0.07989823, 0.4180309 , 0.83932547], [0.06318651, 0.20509151, 0.08262809]]) select items at rớ 0 and 1 in column 1 a[0:2, 1] array([0.4180309 , 0.20509151]) select all items at row 0 a[1, ...] a[1,] array([0.06318651, 0.20509151, 0.08262809]) reversed array a a[::-1] array([[0.64938826, 0.531026 , 0.38633983], [0.06318651, 0.20509151, 0.08262809], [0.07989823, 0.4180309 , 0.83932547]]) Boolean indexing

select elements from a less than 0.5 a[a $<$ 0.5] array([0.07989823, 0.4180309 , 0.06318651, 0.20509151, 0.08262809, 0.38633983]) Fancy indexing

select elements (1,0), (0,1), (1, 2) and (0,0) a[[1, 0, 1, 0], [0, 1, 2, 0]] array([0.06318651, 0.4180309 , 0.08262809, 0.07989823]) select a subset of the matrix's rows and columns a[[1, 0, 1, 0]][:, [0, 1, 2, 0]] array([[0.06318651, 0.20509151, 0.08262809, 0.06318651], [0.07989823, 0.4180309 , 0.83932547, 0.07989823], [0.06318651, 0.20509151, 0.08262809, 0.06318651], [0.07989823, 0.4180309 , 0.83932547, 0.07989823]]) Array Manipulation Transposing Array a = np.random.random((2, 3)) a array([[0.57430709, 0.64401188, 0.12761183], [0.0726823 , 0.7951682 , 0.54114093]]) permulate array dimensions i = np.transpose(a) i array([[0.57430709, 0.0726823], [0.64401188, 0.7951682], [0.12761183, 0.54114093]]) permulate array dimensions i.T array([[0.57430709, 0.64401188, 0.12761183], [0.0726823 , 0.7951682 , 0.54114093]]) Changing Array Shape flatten the array a.ravel() array([0.57430709, 0.64401188, 0.12761183, 0.0726823 , 0.7951682 , 0.54114093]) reshape, but don't change data a.reshape(3, -2) array([[0.57430709, 0.64401188], [0.12761183, 0.0726823], [0.7951682 , 0.54114093]]) Adding/Removing Elements return a new array with shape (2, 6) a.resize(2, 3) a array([[0.57430709, 0.64401188, 0.12761183], [0.0726823 , 0.7951682 , 0.54114093]]) append items to an array h = np.random.random((2, 3)) print "h:", h g = np.random.random((2, 3)) print "g:", g np.append(h, g) h: [[0.67964404 0.09256795 0.90630423] [0.52906489 0.51567697 0.95132012]] g: [[0.03126344 0.84908154 0.74228134] [0.40333143 0.28595213 0.68416838]] array([0.67964404, 0.09256795, 0.90630423, 0.52906489, 0.51567697, 0.95132012, 0.03126344, 0.84908154, 0.74228134, 0.40333143, 0.28595213, 0.68416838]) insert items in an array a = np.random.random((1, 3)) print "a:", a np.insert(a, 1, 0.5) a: [[0.76135438 0.30331334 0.91866363]] array([0.76135438, 0.5 , 0.30331334, 0.91866363]) delete items from an array a = np.random.random((1, 3)) print "a:", a np.delete(a, [1]) a: [[0.1034073 0.93066432 0.49608264]] array([0.1034073 , 0.49608264]) Combining Arrays concatenate arrays a = np.random.random((1, 3)) print a b = np.random.random((1, 3)) print b np.concatenate((a, b), axis=0) [[0.34496986 0.59502574 0.43416152]] [[0.98921435 0.68832237 0.44286195]] array([[0.34496986, 0.59502574, 0.43416152], [0.98921435, 0.68832237, 0.44286195]]) stack arrays vertically (row-wise) a = np.random.random((1, 3)) print a b = np.random.random((2, 3)) print b np.vstack((a, b)) equivalent to np.r$_[a, b]$[[0.787938410.99234010.96372077]][[0.755370830.097813910.25327948]
wise)$a = np.random.random((3, 1))printab = np.random.random((3, 2))printbnp.hstack((a, b))$[[0.3372800
$np.random.random((3, 4))printa$[[0.642778160.759355990.649272470.80253242][0.876306640.197489310.5189
NumpyBasics

1.6 Data Wrangling

Learn about data wrangling with pandas

Tiny Data A foundation for wrangling in pandas

Create DataFrames Specify values for each column

import pandas as pd

df = pd.DataFrame(“a”: [4, 5, 6], “b”: [7, 8, 9], “c”: [10, 11, 12] , index=[1, 2, 3]) df a b c

1 4 7 10

2 5 8 11

3 6 9 12

Specify values for each row

df = pd.DataFrame([[4, 5, 6], [7, 8, 9], [10, 11, 12]], index=[1, 2, 3], columns=[“a”, “b”, “c”]) df a b c

1 4 5 6

2 7 8 9

3 10 11 12

Create DataFrame with a MultiIndex

df = pd.DataFrame(“a”: [4, 5, 6], “b”: [7, 8, 9], “c”: [10, 11, 12]) index = pd.MultiIndex.from$_tuples([(‘d’, 1), (‘d’, 2), (‘e’, 2)], names = [‘n’, ‘v’])dfabc$

0 4 7 10

1 5 8 11

2 6 9 12

Reshaping Data melt “Unpivots” a DataFrame from wide format to long format, optionally leaving identifier variables set.

import pandas as pd

df = pd.DataFrame(“a”: [4, 5], “b”: [7, 8], “c”: [10, 11]) df a b c

0 4 7 10

1 5 8 11

pd.melt(df) variable value

0 a 4

1 a 5

2 b 7

3 b 8

4 c 10

5 c 11

pivot Reshape data (produce a “pivot” table) based on column values. Uses unique values from index / columns to form axes of the resulting DataFrame.

df = pd.DataFrame(‘foo’: [‘one’,‘one’,‘one’,‘two’,‘two’,‘two’], ‘bar’: [‘A’, ‘B’, ‘C’, ‘A’, ‘B’, ‘C’], ‘baz’: [1, 2, 3, 4, 5, 6]) df bar baz foo

0 A 1 one

1 B 2 one

2 C 3 one

3 A 4 two

4 B 5 two

5 C 6 two

df.pivot(index=‘foo’, columns=‘bar’, values=‘baz’) bar A B C

foo

one 1 2 3

two 4 5 6

df.pivot(index='foo', columns='bar')['baz'] bar A B C
foo
one 1 2 3
two 4 5 6
concat Append rows of DataFrames
df1 = pd.DataFrame([['a', 1], ['b', 2]], columns=['letter', 'number']) df1 letter number
0 a 1
1 b 2
df2 = pd.DataFrame([['c', 3], ['d', 4]], columns=['letter', 'number']) pd.concat([df1, df2]) letter number
0 a 1
1 b 2
0 c 3
1 d 4
Append columns of DataFrames
df1 = pd.DataFrame([['a', 1], ['b', 2]], columns=['letter', 'number']) df1 letter number
0 a 1
1 b 2
df2 = pd.DataFrame([['bird', 'polly'], ['monkey', 'george']], columns=['animal', 'name']) df2 animal name
0 bird polly
1 monkey george
pd.concat([df1, df2], axis=1) letter number animal name
0 a 1 bird polly
1 b 2 monkey george
sort df = pd.DataFrame([['a', 10, 1], ['b', 10, 5], ['c', 30, 3]], columns=['name', 'age', 'score']) df name age score
0 a 10 1
1 b 10 5
2 c 30 3
order rows by values of a column (low to high)
$df.sort_values('age')name age score$
0 a 10 1
1 b 10 5
2 c 30 3
order rows by values of a column (high to low)
$df.sort_values('age', ascending = False)name age score$
2 c 30 3
0 a 10 1
1 b 10 5
order rows by values of two column
$df.sort_values(['age', 'score'], ascending = [False, False])name age score$
2 c 30 3
1 b 10 5
0 a 10 1
sort the index of a DataFrame
$df.sort_index()name age score$
0 a 10 1

1 b 10 5
2 c 30 3
Reset index of DataFrame to row numbers, moving index to columns
df.reset$_index()indexnameagescore$
0 0 a 10 1
1 1 b 10 5
2 2 c 30 3
drop drop columns from DataFrame
df.drop([‘age’, ‘score’], axis=1) name 0 a 1 b 2 c

1.7 Visualization

An introduction about data visualization techniques using Matplotlib and Seaborn.
Gallery line graph Line Graph
bar graph Bar Graph
pie graph Pie Graph
sratter plot Scatter Plot
References Patterns: The Data Visualisation Catalogue

Chương 2

Trí tuệ nhân tạo

View online http://magizbox.com/training/ai/site/

Artificial intelligence (AI) is the intelligence exhibited by machines or software. It is also the name of the academic field of study which studies how to create computers and computer software that are capable of intelligent behavior. Major AI researchers and textbooks define this field as "the study and design of intelligent agents", in which an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success. John McCarthy, who coined the term in 1955, defines it as "the science and engineering of making intelligent machines".

2.1 Autonomous Agents

limited ability to perceive its environment process the environment and calculate an action no global plan / leader Vehicles

Action / Selection Steering Locomotion Steering Behavior 1 2

Steering = Desired - Velocity

Seek Flow Filed Following Path Following Group Steering https://github.com/shiffman/The-Nature-of-Code-Examples/tree/master/chp06$_{a}gents$

Massive Battle: Coordinated Movement of Autonomous Agents

Craig Reynolds, Steering Behaviors For Autonomous Characters

2.2 Cellular Automator

https://www.youtube.com/watch?v=DKGodqDs9sAindex=1list=PLRqwX-V7Uu6YrWXvEQFOGbCt6cX8

Cellular Automata

Grid of cell Each cell has state, neighborhood cell state at time t defined by a function of neighborhood states at time t-1 Elementary Cellular Automata

2.3 Fractal

L-System

2.4 The Pac-Man project

Today I found an interesting AI project - The Pac-Man

http://ai.berkeley.edu/images/pacman$_game.gif$

Here is the project overview

The Pac-Man projects were developed for UC Berkeley's introductory artificial intelligence course, CS 188. They apply an array of AI techniques to playing Pac-Man. However, these projects don't focus on building AI for video games. Instead, they teach foundational AI concepts, such as informed state-space search, probabilistic inference, and reinforcement learning. These concepts underly real-world application areas such as natural language processing, computer vision, and robotics. We designed these projects with three goals in mind. The projects allow students to visualize the results of the techniques they implement. They also contain code examples and clear directions, but do not force students to wade through undue amounts of scaffolding. Finally, Pac-Man provides a challenging problem environment that demands creative solutions; real-world AI problems are challenging, and Pac-Man is too. In our course, these projects have boosted enrollment, teaching reviews, and student engagement. The projects have been field-tested, refined, and debugged over multiple semesters at Berkeley. We are now happy to release them to other universities for educational use. In the next part of this post, I will show my works on this project

Project 1: Search in Pacman

[caption id="" align="alignleft" width="231"]DFS[/caption]

[caption id="" align="alignleft" width="233"]BFS[/caption]

Chương 3

Học máy

View online http://magizbox.com/training/machinelearning/site/

- Vấn đề với HMM và CRF?
- Học MLE và MAP?

Machine learning is a branch of science that deals with programming the systems in such a way that they automatically learn and improve with experience. Here, learning means recognizing and understanding the input data and making wise decisions based on the supplied data.

We can think of machine learning as approach to automate tasks like predictions or modelling. For example, consider an email spam filter system, instead of having programmers manually looking at the emails and coming up with spam rules. We can use a machine learning algorithm and feed it input data (emails) and it will automatically discover rules that are powerful enough to distinguish spam emails.

Machine learning is used in many application nowadays like spam detection in emails or movie recommendation systems that tells you movies that you might like based on your viewing history. The nice and powerful thing about machine learning is: It learns when it gets more data and hence it gets more and more powerful the more data we give them.

Có bao nhiêu thuật toán Machine Learning?

Có rất nhiều thuật toán Machine Learning, bài viết [Điểm qua các thuật toán Machine Learning hiện đại](https://ongxuanhong.wordpress.com/2015/10/22/diem-qua-cac-thuat-toan-machine-learning-hien-dai/) của Ông Xuân Hồng tổng hợp khá nhiều thuật toán. Theo đó, các thuật toán Machine Learning được chia thành các nhánh lớn như 'regression', 'bayesian', 'regularization', 'decision tree', 'instance based', 'dimesionality reduction', 'clustering', 'deep learning', 'neural networks', 'associated rule', 'ensemble'... Ngoài ra thì còn có các cheatsheet của [sklearn](http://scikit-learn.org/stable/tutorial/machine$_learning_map/index.html$).

Việc biết nhiều thuật toán cũng giống như ra đường mà có nhiều lựa chọn về xe cộ. Tuy nhiên, quan trọng là có task để làm, sau đó thì cập nhật SOTA của task đó để biết các công cụ mới.

Xây dựng model cần chú ý điều gì?

Khi xây dựng một model cần chú ý đến vấn đề tối ưu hóa tham số (có thể sử dụng [GridSearchCV](sklearn.model$_selection.GridSearchCV$))

Bài phát biểu này có vẻ cũng rất hữu ích [PYCON UK 2017: Machine learning libraries you'd wish you'd known about](https://www.youtube.com/watch?v=nDF7$_8FOhpI$).$C\emptyset cp\emptyset n$

* [DistrictDataLabs/yellowbrick](https://github.com/DistrictDataLabs/yellowbrick) (giúp visualize model được train bởi sklearn) * [marcotcr/lime](https://github.com/marcotcr/lime) (giúp inspect classifier) * [TeamHG-Memex/eli5](https://github.com/TeamHG-Memex/eli5) (cũng giúp inspect classifier, hỗ trợ nhiều model như xgboost, crfsuite, đặc biệt có TextExplainer sử dụng thuật toán từ eli5) * [rhiever/tpot](https://github.com/rhiever/tpot) (giúp tối ưu hóa pipeline) * [dask/dask](https://github.com/dask/dask (tính toán song song và lập lịch)

Ghi chú về các thuật toán trong xử lý ngôn ngữ tự nhiên tại [underthesea.flow/wiki](https://github.com/magizbox/underthesea.flow/wiki/Develop)

Framework để train, test hiện tại vẫn rất thoải mái sklearn. tensorboard cung cấp phần log cũng khá hay.

[Câu trả lời hay](https://www.quora.com/What-are-the-most-important-machine-learning-techniques-to-master-at-this-time/answer/Sean-McClure-3?srid=5O2u) cho câu hỏi [Những kỹ thuật machine learning nào quan trọng nhất để master?](https://www.quora.com/What are-the-most-important-machine-learning-techniques-to-master-at-this-time), đặc biệt là dẫn đến bài [The State of ML and Data Science 2017](https://www.kaggle.com/surveys/2017) của Kaggle.

Tài liệu học PGM

[Playlist youtube](https://www.youtube.com/watch?v=WPSQfOkb1M8amp;list=PL50E6E80E8525B59C khóa học Probabilistic Graphical Models của cô Daphne Koller. Ngoài ra còn có một [tutorial](http://mensxmachina.org/files/software/demos/bayesnetdemo.html) dở hơi ở đâu về tạo Bayesian network

[Chưa biết] Tại sao Logistic Regression lại là Linear Model?

Trong quyển Deep Learning, chương 6, trang 165, tác giả có viết

"' Linear models, such as logistic regression and linear regression, are appealing because they can be t eciently and reliably, either in closed form or with convex optimization "'

Mình tự hỏi tại sao logistic regression lại là linear, trong khi nó có sử dụng hàm logit (nonlinear)? Tìm hiểu hóa ra cũng có bạn hỏi giống mình trên [stats.stackexchange.com](https://stats.stackexchange.com/questions/93569/why-is-logistic-regression-a-linear-classifier). Ngoài câu trả lời trên stats.stackexchange, đọc một số cái khác [Generalized Linear Models, SPSS Statistics 22.0.0](https://www.ibm.com/support/knowl $IntroductiontoGeneralizedLinearModels$, $AnalysisofDiscreteData$, $PennsylvaniaStateUniversity$](*https* //*onlinecourses.science.psu.edu/stat*504/*node*/216)$cngvnchahiulm$.

Hiện tại chỉ hiểu là các lớp model này chỉ có thể hoạt động trên các tập linear separable, có lẽ do việc map input x, luôn có một liên kết linear $latexwx$, trước khi đưa vào hàm non-linear.

Các tập dữ liệu thú vị

Iris dataset: dữ liệu về hoa iris

Là một ví dụ cho bài toán phân loại

Weather problem: dữ liệu thời tiết. Có thể tìm được ở trong quyển Data Mining: Practical Machine Learning Tools and Techniques

Là một ví dụ cho bài toán cây quyết định

Deep Learning

Tài liệu Deep Learning

Lang thang thế nào lại thấy trang này [My Reading List for Deep Learning!](https://www.microsoft.com/en us/research/wp-content/uploads/2017/02/DL$_Reading_List.pdf$)$camtanhMicrosoft.Trong\emptyset$, $(\emptyset ngnhin)cDee$

Các layer trong deep learning [2]

Sparse Layers

[**nn.Embedding**](http://pytorch.org/docs/master/nn.htmlembedding) ([hướng dẫn](http://pytorch.org/tutorials/beginner/nlp/word$_e$$mbeddings_t$$utorial.html))$grepcode$: [$Shawn1993/cnn-text-classification-pytorch$](https : //github.com/Shawn1993/cnn-text-classification-pytorch/blob/master/model.py$L18$)Œ$ngvaitrnhmtlookuptable, mapmtwordvidenseve$

Convolution Layers

[**nn.Conv1d**](http://pytorch.org/docs/master/nn.htmlconv1d), [**nn.Conv2d**](http://pytorch.org/ [**nn.Conv3d**](http://pytorch.org/docs/master/nn.htmlconv3d) [1]$grepcode$: [$Shawn1993/cnn-text-classification-pytorch$](https : //github.com/Shawn1993/cnn-text-classification-pytorch/blob/master/model.py$L20-L24$), [$galsang/CNN-sentence-classification-pytorch$](https : //github.com/galsang/CNN - sentence - classification - pytorch/blob/master/model.py$L36 - L38$)

Các tham số trong Convolution Layer

* 'kernel$_s$$ize$'($haylfiltersize$)

Đối với NLP, kernel$_s$$izethngbngregion_s$$ize*word_d$$im(Øiviconv1d)hay(region_s$$ize, word_d$$im)Øiviconv2d$

<small>Quá trình tạo feature map đối với region size bằng 2</small>

* 'in$_c$$hannels$', '$out_c$$hannels$'($lslng$'$featuremaps$')

Kênh (channels) là các cách nhìn (view) khác nhau đối với dữ liệu. Ví dụ, trong ảnh thường có 3 kênh RGB (red, green, blue), có thể áp dụng convolution giữa các kênh. Với văn bản cũng có thể có các kênh khác nhau, như khi có các kênh sử dụng các word embedding khác nhau (word2vec, GloVe), hoặc cùng một câu nhưng biểu diễn ở các ngôn ngữ khác nhau.

* 'stride'

Định nghĩa bước nhảy của filter.

Hình minh họa sự khác biệt giữa các feature map đối với stride=1 và stride=2. Feature map đối với stride = 1 có kích thước là 5, feature map đối với stride = 3 có kích thước là 3. Stride càng lớn thì kích thước của feature map càng nhỏ.

Trong bài báo của Kim 2014, 'stride = 1' đối với 'nn.conv2d' và 'stride = word$_d$$im$'$Øivi$'$nn.conv1d$'

Toàn bộ tham số của mạng CNN trong bài báo Kim 2014,

| Description | Values | |—————|————| | input word vectors | Google word2vec | | filter region size | (3, 4, 5) | | feature maps | 100 | | activation function | ReLU | | pooling | 1-max pooling | | dropout rate | 0.5 | | $latexlamp; s = 22$ norm constraint | 3 |

Đọc thêm:

* [Lecture 13: Convolutional Neural Networks (for NLP). CS224n-2017](http://web.stanford.edu/class/cs2 2017-lecture13-CNNs.pdf) * [DeepNLP-models-Pytorch - 8. Convolutional Neural Networks](https://nbviewer.jupyter.org/github/DSKSD/DeepNLP-models-Pytorch/blob/master/notebooks/08.CNN-for-Text-Classification.ipynb) * [A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. Zhang 2015](https://arxiv.org/pdf/1510.03820.pdf)

BTS

22/11/2017 - Phải nói quyển này hơi nặng so với mình. Nhưng thôi cứ cố gắng vậy. 24/11/2017 - Từ hôm nay, mỗi ngày sẽ ghi chú một phần (rất rất nhỏ) về

Deep Learning [tại đây](https://docs.google.com/document/d/1KxDrw5s6uYHNLda7t0rhp0RM$_T$lUGxydQ-
Qi1JOPFr8/edit?usp = sharing)

[1] : [UnderstandingConvolutionalNeuralNetworksforNLP](http : //www.wildml.com/2015/11/unde
convolutional−neural−networks−for−nlp)[2] : [http : //pytorch.org/docs/master/nn.html](http :
//pytorch.org/docs/master/nn.html)

3.1 Machine Learning Process

The good life is a process, not a state of being. It is a direction not a destination.

Carl Rogers

I searched a framework fit for every data mining task, I found a good one from an article of Oracle.

And here is my summary. The data mining process has 4 steps:

Step 1. Problem Definition

This initial phase of a data mining project focuses on understanding the project objectives and requirements. Once you have specified the project from a business perspective, you can formulate it as a data mining problem and develop a preliminary implementation plan.

Step 2. Data Gathering Preparation

The data understanding phase involves data collection and exploration. As you take a closer look at the data, you can determine how well it addresses the business problem. You might decide to remove some of the data or add additional data. This is also the time to identify data quality problems and to scan for patterns in the data.

Data Access Data Sampling

Data Transformation

Data in the real world is dirty [3]. They are often incomplete (lacking attribute values, lacking certain attributes of interest, or containing only aggregate data), noisy (containing errors or outliers), inconsistent (containing discrepancies in codes or names). Step 3. Model Building In this phase, you select and apply various modeling techniques and calibrate the parameters to optimal values. If the algorithm requires data transformations, you will need to step back to the previous phase to implement them

Create Model Test Model

Evaluate Interpret Model

Some important questions [2]:

Is at least one of predictors useful in predicting the response? (F-statistics) Do all the predictors help to explain Y, or is only a subset of the predictors useful? (all subsets or best subsets) How well does the model fit the data? Given a set of predictor values, what response value should we predict, and how accurate is our prediction? Step 4. Knowledge Deployment Knowledge deployment is the use of data mining within a target environment. In the deployment phase, insight and actionable information can be derived from data. Model Apply Custom Reports External Applications References The Data Mining Process, Oracle Trevor Hastie and Rob Tibshirani, Model Selection and Qualitative Predictors, URL:https://www.youtube.com/watch?v=3T6RXmIHbJ4 Nguyen Hung Son, Data cleaning and Data preprocessing, URL:http://www.mimuw.edu.pl/ son/datamining/DM/4-preprocess.pdf

3.1.1 Problem Definition

This initial phase of a data mining project focuses on understanding the project objectives and requirements. Once you have specified the project from a business perspective, you can formulate it as a data mining problem and develop a preliminary implementation plan.

For example, your business problem might be: "How can I sell more of my product to customers?" You might translate this into a data mining problem such as: "Which customers are most likely to purchase the product?" A model that predicts who is most likely to purchase the product must be built on data that describes the customers who have purchased the product in the past. Before building the model, you must assemble the data that is likely to contain relationships between customers who have purchased the product and customers who have not purchased the product. Customer attributes might include age, number of children, years of residence, owners/renters, and so on.

3.1.2 Data Gathering

The data understanding phase involves data collection and exploration. As you take a closer look at the data, you can determine how well it addresses the business problem. You might decide to remove some of the data or add additional data. This is also the time to identify data quality problems and to scan for patterns in the data.

The data preparation phase covers all the tasks involved in creating the case table you will use to build the model. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order. Tasks include table, case, and attribute selection as well as data cleansing and transformation. For example, you might transform a DATE$_O F_B IRTH column to AGE; you might insert the average income in cases where

Additionally you might add new computed attributes in an effort to tease information closer to the surface of the data. For example, rather than using the purchase amount, you might create a new attribute: "Number of Times Amount Purchase Exceeds 500 in a 12 month time period." Customers who frequently make large purchases may also be rel

Thoughtful data preparation can significantly improve the information that can be discovered through data mining.

Data Sources Open Data

wikipedia dumps: https://dumps.wikimedia.org/other/pagecounts-raw/

3.1.3 Data Preprocessing

The quality of the data and the amount of useful information it contains affect greatly how well an algorithm can learn. Hence, it is important to preprocess the dataset before using it. The most common preprocessing steps are: removing missing values, converting categorical data into shape suitable for machine learning algorithm and feature scaling.

Missing Data Sometimes the samples in the dataset are missing some values and we want to deal with these missing values before passing it to the machine learning algorithm. There are a number of strategies we can follow

Remove samples with missing values: This approach is by far the most convenient but we may end up removing too many samples and by that we would be losing valuable information that can help the machine learning algorithm.

Imputing missing values: Instead of removing the entire sample we use interpolation to estimate the missing values. For example, we could substitute a missing value by the mean of the entire column. Categorical Data In general, features can be numerical (e.g. price, length, width, etc. . .) or categorical (e.g. color, size, etc..). Categorical features are further split into nominal and ordinal features.

Ordinal features can be sorted and ordered. For example, size (small, medium, large), we can order these sizes large > medium > small. While nominal features do not have an order for example, color, it doesn't make any sense to say that red is larger than blue.

Most machine learning algorithm require that you convert categorical features into numerical values. One solution would to assign each value a different number starting from zero. (e.g. small à 0 ,medium à 1 ,large à 2)

This works well for ordinal features but might cause problems with nominal features (e.g. blue à 0, white à 1, yellow à 2) because even though colors are not ordered the learning algorithm will assume that white is larger than blue and yellow is larger than white and this is not correct.

To get around this problem is to use one-hot encoding, the idea is to create a new feature for each unique value of the nominal feature.

In the above example, we converted the color feature into three new features Red, Green, Blue and we used binary values to indicate the color. For example, a sample with "Red" color is now encoded as (Red=1, Green=0, Blue=0)

Feature Scaling Why have we do Feature Scaling?

We have to predict the house prices base on 2 features:

House sizes (feet2) Number of bedrooms in the house And we relized that house sizes are about 1000 times the number of bedrooms. When features differ by orders of magnitude, first performing feature scaling can make gradient descent converge much more quickly.

Perform Feature Scaling

Subtract the mean value (the average value) of each feature from the dataset. After subtracting the mean, additionally scale (divide) the feature values by their respective "standard deviations." Function: x=xxx=xx where xx is the original feature vector, xx is the mean of that feature vector, and is its standard deviation. Feature Scaling Function implementation in Octave

function $[\mathrm{X}_norm, mu, sigma] = featureNormalize(X)X_norm = X; mu = zeros(1, size(X, 2)); sigma = zeros(1, size(X, 2));$

for i = 1:length(mu), mu(i) = mean(X(:,i)); end;

for i = 1:length(sigma), sigma(i) = std(X(:,i)); end;

$\mathrm{X}_norm = (X. - mu)./sigma; endRelatedReadingIntroductiontoMachineLearning$

3.1.4 Model Building

In this phase, you select and apply various modeling techniques and calibrate the parameters to optimal values. If the algorithm requires data transformations, you will need to step back to the previous phase to implement them

Create Model Test Model Evaluate Interpret Model Some important questions

Is at least one of predictors useful in predicting the response? (F-statistics) Do all the predictors help to explain Y, or is only a subset of the predictors useful? (all subsets or best subsets) How well does the model fit the data?

Given a set of predictor values, what response value should we predict, and how accurate is our prediction? Create Model First thing first, start with simple and fast model, then you known how difficult the problem is.

One import thing is create a well pipeline for your experiments, it is very helpful in turning features, model selection, save your experiment and write reports.

Feature Selections After train model, some model will give active features (such as CRF), it is clue for you to feature selection. If amount active features is too small compared to amount features, it is the problem. In this case the better way to enhance is try reduce amount of features and see how well this set fit data. Keep in mind the more number of features is, the complex model is, and it will make your model over fitting. Storing the model Number of active features: 5566 (35383) Number of active attributes: 4343 (20722) example after training crf model with python-crfsuite Test Model This phase determines how well the model fit data. See Evaluation for details.

What to do next In an interview Andrew Ng said about building machine learning model

"I often make an analogy to building a rocket ship. A rocket ship is a giant engine together with a ton of fuel. Both need to be really big. If you have a lot of fuel and a tiny engine, you won't get off the ground. If you have a huge engine and a tiny amount of fuel, you can lift up, but you probably won't make it to orbit. So you need a big engine and a lot of fuel.

The reason that machine learning is really taking off now is that we finally have the tools to build the big rocket engine — that is giant computers, that's our rocket engine. And the fuel is the data. We finally are getting the data that we need."

We need both big rocket engine and data to make our model works.

Related Reading Inside The Mind That Built Google Brain: On Life, Creativity, And Failure, huffingtonpost.com

3.1.5 Evaluation

Training vs Test Data We typically split the input data into learning and testing datasets. The then run the machine learning algorithm on the learning dataset to generate the prediction model. Later, we use the test dataset to evaluate our model.

It is important that the test data is separate from the one used in training otherwise we will be kind of cheating because may for example the generated model memorizes the data and hence if the test data is also part of the training data then our evaluation scores of the model will be higher than they actually are.

The data is usually split 75

In addition, when splitting the dataset, you need to maintaining class proportions and population statistics otherwise we will have some classes that are under represented in the training dataset and over represented in the test dataset.

For example, you may have 100 sample and a total of 80 samples are labeled with Class-A and the remaining 20 instances are labeled with Class-B. you want to make sure when splitting the data that you maintain this representation.

One way to avoid this problem and to make sure that all classes are represented in both training and testing datasets is stratification. It is the process of

rearranging the data as to ensure each set is a good representative of the whole. In our previous example, (80/20 samples), it is best to arrange the data such that in every set, each class comprises around 80:20 ratios of the two classes.

Cross Validation A crucial step when building our machine learning model is to estimate its performance on that that the model hadn't seen before. We want to make sure that the model generalizes well to new unseen data.

One case, the machine learning algorithm has different parameters and we want to tune these parameters to achieve the best performance. (Note: the parameters of the machine learning algorithm are called hyperparameters). Another case, sometimes we want to try out different algorithms and choose the best performing one. Below are some of the techniques used.

Holdout Method We simply split the data into training and testing datasets. We train the algorithm on the training dataset to generate a model. In order, to evaluate different algorithms we use the testing data to evaluate each algorithm.

However, if we reuse the same test dataset over and over again during algorithm selection, the test data has now come part of the training data. Hence, when we use the test data for the final evaluation the generated model is biased towards the test data and the performance score is optimistic.

Holdout Validation As before, we split the data into training and testing dataset. Then, the training data is further split into training and validation sets.

The training data is used to train different models. Then the validation data is used to compute performance of each of them and we select the best one. Finally, the model is then used for the test set to evaluate performance. The next figure illustrates this idea.

However, because we use the validation set multiple times, Holdout validation is sensitive to how we partition the data and that is what K-fold cross validation tries to solve.

K-fold cross validation Initially, we split the data into training and testing dataset. Furthermore, the training dataset is split into K chunks.

Suppose we will use 5-fold cross validation, the training data set is split into 5 chunks and the training phase will take place over 5 iterations. In each iteration we use one chunk as the validation dataset while the rest of the chunk are grouped together to form the training dataset.

This is very similar to Holdout validation except in each iteration the validation data is different and this removed the bias. Each iteration generates a score and the final score is the average score of all iteration. As before we select the best model and use the test data for the final performance evaluation.

Related Readings

Introduction to Machine Learning

3.2 Types of Machine Learning

There are three different types of machine learning: supervised, unsupervised and reinforcement learning. 4

Supervised Learning The goal of supervised learning is to learn a model from labelled training data that allows us to make predictions about future data. For supervised machine learning to work we need to feed the algorithm two things: the input data and our knowledge about it labels).

The spam filter example mentioned earlier is a good example of supervised learning; we have a bunch of emails (data) and we know whether each email is spam or not (labels).

Supervised learning can be divided into two subcategories:

Classification: It is used to predict categories or class labels based on past observations i.e. we have discrete variable you want to distinguish into discrete categorical outcome. For example, in the email spam filter system the output is discrete "spam" or "not spam". Regression: It is used to predict a continuous outcome. For example, to determine the price of houses and how it is affected by the number of rooms in that house. The input data is the house features (no. of rooms, location, size in square feet,) and the output is the price (the continuous outcome). Unsupervised Learning The goal of unsupervised learning is to discover hidden structure or patterns in unlabeled data and it can be divided into two subcategories

Clustering: It is used to organize information into meaningful clusters (subgroups) without having prior knowledge of their meaning. For example, the figure below shows how we can use clustering to organize unlabeled data into groups based on their features.

Dimensionality Reduction (Compression): It is used to reduce a higher dimension data into a lower dimension ones. To put it more clearly consider this example. A telescope has terabytes of data and not all of these data can be stored and so we can use dimensionality reduction to extract the most informative features of these data to be stored. Dimensionality reduction is also a good candidate to visualize data because if you have data in higher dimensions you can compress it to 2D or 3D to easily plot and visualize it.

Reinforcement Learning

The goal of reinforcement learning is to develop a system that improves its performance based on the interaction with a dynamic environment and there is a delayed feedback that act as a reward. i.e. reinforcement learning is learning by doing with a delayed reward. A classic example of reinforcement learning is a chess game, the computer decided a series of moves and the reward is the "win" or "lose" at the end the game.

You might think that this is similar to supervised learning where the reward is basically a label for the data but the core difference is this feedback/reward is not the truth but it is a measure of how well the action to achieving a certain goal.

Microsoft Azure Machine Learning 1

Machine Learning Cheat Sheet for scikit-learn 2

DLib C++ Library - Machine Learning Guide 3

Challenges Very much features ($>$ 100) Very much data ($>$ 1e9 items) Text Data, Images, Videos Training Times Accuracy, Over Fitting Machine learning algorithm cheat sheet for Microsoft Azure Machine Learning Studio

Machine Learning Cheat Sheet (for scikit-learn)

DLib C++ Library - Machine Learning Guide

Introduction to Machine Learning

3.3 How to learn a ML Algorithm?

1. Motivation

Each algorithm have its own motivation. It may a simple example to see how it work

2. Problem Definition

Where can we apply this algorithm? How did it work in real world applications

3. Mathematics Representation

Problem Equations, notations

We will discuss about mathematics representation of algorithm, notations we use for problem

4. Algorithm

We will discuss how to solve this mathematics problems

5. Examples

We will apply algorithm with a few examples (1-2 dimension is highly recommended, because we will plot these data and model easily)

In this section, we can see how well (bad) algorithm works with these data

6. Implementation Notice

We will give some notes about implement this algorithm to real world problems. What case we want to apply this algorithm? What case we don't?

7. Quiz

One way to rethink about problem is doing quiz.

8. Exercise

3.4 Linear Regression

linear regression — Hồi quy tuyến tính, là thuật toán machine learning cơ bản nhất, áp dụng trên dữ liệu số

In-Out

- Đầu vào: Continuous
- Đầu ra: Continuous

When to use

- Econometric Modeling
- Marketing Mix Model
- Customer Lifetime Value

Examples

Ex. Linear Regression with Boston Dataset

```
__author__ = 'rain'

from sklearn.datasets import load_boston
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression, Ridge
boston = load_boston()
data = boston['data']
X, y = data[:, :-1], data[:, -1]
```

```
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.3)
print boston['DESCR']
clf_linear = LinearRegression()
clf_linear.fit(X_train, y_train)
linear_score = clf_linear.score(X_test, y_test)
#-> 0.671
print(clf_linear.coef_)
print(clf_linear.intercept_)

clf_ridge = Ridge(alpha=1.0)
clf_ridge.fit(X_train, y_train)
# 0.674
ridge_score = clf_ridge.score(X_test, y_test)

print y_test
print clf_linear.predict(X_test)
print clf_ridge.predict(X_test)
```

3.5 Logistic Regression

logistic regression Mô hình đơn giản nhất của việc phân lớp, dựa vào hàm logistic

In-Out

- Đầu vào: continuos
- Đầu ra: True/False

Hyposthesis Representation

$$h_\theta(x) = g(\theta^T x) \text{ trong đó } g(z) = \frac{1}{1+e^z}$$

$g(z)$ là hàm sigmoid hay hàm logistic
$h_\theta(x)$ estimated probability of y=1 given x
In spam detection problem, h(x)=0.7 means it's 70% chance this email is spam.

Decision Boundary
Logistic Regression

Cost Function
cost(h(x),y)=ylog(h(x))(1y)log(1h(x)) cost(h(x),y)=ylog(h(x))(1y)log(1h(x))
Loss Function

$$J() = 1mi = 1mcost(h(x(i)), y(i)) = 1mi = 1my(i)logh(x(i))+(1y(i))log(1h(x(i)))$$

Gradient Descent
Gradient

$$J()j = 1mi = 1m(h(x(i))y(i))x(i)j$$

Predict

$$p(, X) = h(X)0.5$$

Regularization

6.1 Feature Mapping
Cost Function
mapFeature(x)=1x1x2x21x1x2x22x31x1x52x62 mapFeature(x)=[1x1x2x12x1x2x22x13x1x25x26]
6.2 Cost Function and Gradient Cost Function J()=1mi=1m[y(i)log(h(x(i)))(1y(i))log(1h(x(i)))]+2mj=1n2
J()=1mi=1m[y(i)log(h(x(i)))(1y(i))log(1h(x(i)))]+2mj=1nj2 Gradient
J()j=1mmi=1(h(x(i))y(i))x(i)jJ()j=1mi=1m(h(x(i))y(i))xj(i) for j=0j=0
J()j=(1mmi=1(h(x(i))y(i))x(i)j)+mjJ()j=(1mi=1m(h(x(i))y(i))xj(i))+mj for
j1j1
Example: Bank Marketing Data Set

```
import statsmodels.api as sm
import pandas as pd
from statsmodels.tools.tools import categorical
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
from sklearn.metrics import confusion_matrix
import numpy
from sklearn.tree import DecisionTreeClassifier


def get_data():
  return pd.read_csv("./bank/bank-full.csv", header=0, sep=";")

data = get_data()

data.job = LabelEncoder().fit_transform(data.job)
data.marital = LabelEncoder().fit_transform(data.marital)
data.education = LabelEncoder().fit_transform(data.education)
data.default = LabelEncoder().fit_transform(data.default)
data.housing = LabelEncoder().fit_transform(data.housing)
data.loan = LabelEncoder().fit_transform(data.loan)
data.month = LabelEncoder().fit_transform(data.month)
data.contact = LabelEncoder().fit_transform(data.contact)
data.poutcome = LabelEncoder().fit_transform(data.poutcome)

X = data.iloc[:, :-1]
y = data.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

clf = LogisticRegression()
clf.fit(X_train, y_train)
score = clf.score(X_test, y_test)

print confusion_matrix(y_test, clf.predict(X_test))
```

 # [[11807 203]
 # [1243 311]]

Examples: Affair Dataset, Logistic Regression with scikit-learn Linear Regression vs Logistic Regression vs Poisson Regression

3.6 Classification

A very familiar example is the email spam-catching system: given a set of emails marked as spam and not-spam, it learns the characteristics of spam emails and is then able to process future email messages to mark them as spam or not-spam.

The technique used in the above example of email spam-catching system is one of the most common machine learning techniques: classification (actually, statistical classification). More precisely it is a supervised statistical classification. Supervised because the system needs to be first trained using already classified training data as opposed to an unsupervised system where such training is not done.

A supervised learning system that performs classification is known as a learner or, more commonly, a classifier.

The classifier is first fed training data in which each item is already labeled with the correct label or class. This data is used to train the learning algorithm, which creates models that can then be used to label/classify similar data.

Formally, given a set of input items, and a set of labels/classes, and training data is the label/class for $latexx_i$, a classifier is a mapping from X to Y $latexf(T, x) = y$.

3.7 Binary Classification

Algorithms 1 Two-class SVM 100 features, linear model

Two-class Logistic Regression Fast training, linear model Two-class Bayes point machine Fast training, linear model Two-class random forest Accuracy, fast training Two-class boosted decision tree Accuracy, fast training Two-class neural network Accuracy, long training times Multiclass Classification

Introduction 2 In machine learning, multiclass or multinomial classification is the problem of classifying instances into one of the more than two classes (classifying instances into one of the two classes is called binary classification).

While some classification algorithms naturally permit the use of more than two classes, others are by nature binary algorithms; these can, however, be turned into multinomial classifiers by a variety of strategies.

Multiclass classification should not be confused with multi-label classification, where multiple labels are to be predicted for each instance.

Algorithms 1

Multiclass Logistic Regression Multiclass SVM Multiclass Neural Network Multiclass Decision Forest Multiclass Decision Jungle

Confusion Matrix

sklearn plot confusion matrix with labels 3

import matplotlib.pyplot as plt
def plot_confusion_matrix(cm,

```
                                title ='Confusion matrix',
                               cmap=plt.cm.Blues, labels=None):
  fig  = plt.figure()
  ax = fig.add_subplot(111)
  cax = ax.matshow(cm)
  plt.title( title )
  fig.colorbar(cax)
if  labels:
  ax.set_xticklabels(['']  + labels)
  ax.set_yticklabels(['']  + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

3.8 Multilabel Classification

Introduction

In machine learning, multi-label classification and the strongly related problem of multi-output classification are variants of the classification problem where multiple target labels must be assigned to each instance. Multi-label classification should not be confused with multiclass classification, which is the problem of categorizing instances into one of more than two classes. Formally, multi-label learning can be phrased as the problem of finding a model that maps inputs x to binary vectors y, rather than scalar outputs as in the ordinary classification problem.

There are two main methods for tackling the multi-label classification problem:[1] problem transformation methods and algorithm adaptation methods. Problem transformation methods transform the multi-label problem into a set of binary classification problems, which can then be handled using single-class classifiers. Algorithm adaptation methods adapt the algorithms to directly perform multi-label classification. In other words, rather than trying to convert the problem to a simpler problem, they try to address the problem in its full form.

Implements

Multiclass and multilabel algorithms SVM Multi-label classification

Multiclass classification

sklearn plot confusion matrix with labels

3.9 Clustering

Using K-Means to cluster wine dataset Recently, I joined Cluster Analysis course in coursera. The content of first week is about Partitioning-Based Clustering Methods where I learned about some cluster algorithms based on distance such as K-Means, K-Medians and K-Modes. I would like to turn what I learn into practice so I write this post as an excercise of this course.

In this post, I will use K-Means for clustering wine data set which I found in one of excellent posts about K-Mean in r-statistics website.

Meet the data

The wine data set contains the results of a chemical analysis of wines grown in a specific area of Italy. Three types of wine are represented in the 178 samples, with the results of 13 chemical analyses recorded for each sample. The Type variable has been transformed into a categoric variable.

```
data(wine, package=&quot;rattle&quot;)
head(wine)
```

```
 Type Alcohol Malic Ash Alcalinity Magnesium Phenols
1 1 14.23 1.71 2.43 15.6 127 2.80
2 1 13.20 1.78 2.14 11.2 100 2.65
3 1 13.16 2.36 2.67 18.6 101 2.80
4 1 14.37 1.95 2.50 16.8 113 3.85
5 1 13.24 2.59 2.87 21.0 118 2.80
6 1 14.20 1.76 2.45 15.2 112 3.27
 Flavanoids Nonflavanoids Proanthocyanins Color Hue
1 3.06 0.28 2.29 5.64 1.04
2 2.76 0.26 1.28 4.38 1.05
3 3.24 0.30 2.81 5.68 1.03
4 3.49 0.24 2.18 7.80 0.86
5 2.69 0.39 1.82 4.32 1.04
6 3.39 0.34 1.97 6.75 1.05
 Dilution Proline
1 3.92 1065
2 3.40 1050
3 3.17 1185
4 3.45 1480
5 2.93 735
6 2.85 1450
```

Explore **and** Preprocessing Data

Let's see structure of wine data set

```
\begin{ lstlisting }[language=R]
str(wine)
```

```
 &apos;data.frame&apos;: 178 obs. of 14 variables:
 $ Type : Factor w/ 3 levels &quot;1&quot;,&quot;2&quot;,&quot;3&quot;: 1
     ↪ 1 1 1 1 1 1 1 1 1 ...
 $ Alcohol : num 14.2 13.2 13.2 14.4 13.2 ...
 $ Malic : num 1.71 1.78 2.36 1.95 2.59 1.76 1.87 2.15 1.64 1.35 ...
 $ Ash : num 2.43 2.14 2.67 2.5 2.87 2.45 2.45 2.61 2.17 2.27 ...
 $ Alcalinity : num 15.6 11.2 18.6 16.8 21 15.2 14.6 17.6 14 16 ...
 $ Magnesium : int 127 100 101 113 118 112 96 121 97 98 ...
 $ Phenols : num 2.8 2.65 2.8 3.85 2.8 3.27 2.5 2.6 2.8 2.98 ...
 $ Flavanoids : num 3.06 2.76 3.24 3.49 2.69 3.39 2.52 2.51 2.98 3.15 ...
 $ Nonflavanoids : num 0.28 0.26 0.3 0.24 0.39 0.34 0.3 0.31 0.29 0.22 ...
 $ Proanthocyanins: num 2.29 1.28 2.81 2.18 1.82 1.97 1.98 1.25 1.98 1.85
     ↪ ...
 $ Color : num 5.64 4.38 5.68 7.8 4.32 6.75 5.25 5.05 5.2 7.22 ...
 $ Hue : num 1.04 1.05 1.03 0.86 1.04 1.05 1.02 1.06 1.08 1.01 ...
```

```
$ Dilution : num 3.92 3.4 3.17 3.45 2.93 2.85 3.58 3.58 2.85 3.55 ...
$ Proline : int 1065 1050 1185 1480 735 1450 1290 1295 1045 1045 ...
```

Wine data set contains 1 categorical variables (label) and 13 numerical variables. But these numerical variables is not scaled, I use scale function for scaling and centering data and then assign it as training data.

data.train lt;- scale(wine[-1]) Data is already centered and scaled.

```
summary(data.train)
 Alcohol Malic
 Min. :−2.42739 Min. :−1.4290
 1st Qu.:−0.78603 1st Qu.:−0.6569
 Median : 0.06083 Median :−0.4219
 Mean : 0.00000 Mean : 0.0000
 3rd Qu.: 0.83378 3rd Qu.: 0.6679
 Max. : 2.25341 Max. : 3.1004
 Ash Alcalinity
 Min. :−3.66881 Min. :−2.663505
 1st Qu.:−0.57051 1st Qu.:−0.687199
 Median :−0.02375 Median : 0.001514
 Mean : 0.00000 Mean : 0.000000
 3rd Qu.: 0.69615 3rd Qu.: 0.600395
 Max. : 3.14745 Max. : 3.145637
 Magnesium Phenols
 Min. :−2.0824 Min. :−2.10132
 1st Qu.:−0.8221 1st Qu.:−0.88298
 Median :−0.1219 Median : 0.09569
 Mean : 0.0000 Mean : 0.00000
 3rd Qu.: 0.5082 3rd Qu.: 0.80672
 Max. : 4.3591 Max. : 2.53237
 Flavanoids Nonflavanoids
 Min. :−1.6912 Min. :−1.8630
 1st Qu.:−0.8252 1st Qu.:−0.7381
 Median : 0.1059 Median :−0.1756
 Mean : 0.0000 Mean : 0.0000
 3rd Qu.: 0.8467 3rd Qu.: 0.6078
 Max. : 3.0542 Max. : 2.3956
 Proanthocyanins Color
 Min. :−2.06321 Min. :−1.6297
 1st Qu.:−0.59560 1st Qu.:−0.7929
 Median :−0.06272 Median :−0.1588
 Mean : 0.00000 Mean : 0.0000
 3rd Qu.: 0.62741 3rd Qu.: 0.4926
 Max. : 3.47527 Max. : 3.4258
 Hue Dilution
 Min. :−2.08884 Min. :−1.8897
 1st Qu.:−0.76540 1st Qu.:−0.9496
 Median : 0.03303 Median : 0.2371
 Mean : 0.00000 Mean : 0.0000
 3rd Qu.: 0.71116 3rd Qu.: 0.7864
 Max. : 3.29241 Max. : 1.9554
```

Proline
Min. :−1.4890
1st Qu.:−0.7824
Median :−0.2331
Mean : 0.0000
3rd Qu.: 0.7561
 # > Max. : 2.963

Model Fitting Now the fun part begins. I use NbClust function to determine what is the best number of clusteres k for K-Means

nc lt;- NbClust(data.train, min.nc=2, max.nc=15, method=quot;kmeansquot;) barplot(table(nc$Best.n[1,]), xlab = quot;Numero of Clustersquot;, ylab = quot;Numberof Criteriaquot;, ma quot;Numberof Clusters Chosenby 26 Criteriaquot;)

According to the graph, we can find the best number of clusters is 3. Beside NbClust function which provides 30 indices for determing the number of clusters and proposes the best clustering scheme, we can draw the sum of square error (SSE) scree plot and look for a bend or elbow in this graph to determine appropriate k

wss lt;- 0 for (i in 1:15) wss[i] lt;- sum(kmeans(data.train, centers=i)$withinss) plot(1 : 15, wss, type = quot;bquot;, xlab = quot;Numberof Clustersquot;, ylab = quot;Withingroupssumofsquare

Both two methods suggest k=3 is best choice for us. It's reasonsable if we take notice that the original data set also contains 3 classes.

Fit the model We now fit wine data to K-Means with k = 3

fit.km lt;- kmeans(data.train, 3) Then interpret the result

fit.km

K-means clustering with 3 clusters of sizes 51, 65, 62

Cluster means: Alcohol Malic Ash Alcalinity 1 0.1644436 0.8690954 0.1863726 0.5228924 2 -0.9234669 -0.3929331 -0.4931257 0.1701220 3 0.8328826 -0.3029551 0.3636801 -0.6084749 Magnesium Phenols Flavanoids Nonflavanoids 1 -0.07526047 -0.97657548 -1.21182921 0.72402116 2 -0.49032869 -0.07576891 0.02075402 -0.03343924 3 0.57596208 0.88274724 0.97506900 -0.56050853 Proanthocyanins Color Hue Dilution 1 -0.77751312 0.9388902 -1.1615122 -1.2887761 2 0.05810161 -0.8993770 0.4605046 0.2700025 3 0.57865427 0.1705823 0.4726504 0.7770551 Proline 1 -0.4059428 2 -0.7517257 3 1.1220202

Clustering vector: [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 [26] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 [51] 3 3 3 3 3 3 3 3 3 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 3 2 [76] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 [101] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 3 2 2 2 [126] 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 [151] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 [176] 1 1 1

Within cluster sum of squares by cluster: [1] 326.3537 558.6971 385.6983 (between$_S S/total_S S = 44.8$

Available components:

[1] quot;clusterquot; quot;centersquot; quot;totssquot; [4] quot;withinssquot; quot;tot.withinssquot; quot;betweenssquot; gt; [7] quot;sizequot; quot;iterquot; quot;ifaultquot The result shows information about cluster means, clustering vector, sum of square by cluster and available components. Let's do some visualizations to see how data set is clustered.

First, I use plotcluster function from fpc package to draw discriminant projection plot

library(fpc) plotcluster(data.train, fit.km$cluster$)

We can see the data is clustered very well, there are no collapse between clusters. Next, we draw parallel coordinates plot to see how variables contributed in each cluster

library(MASS) parcoord(data.train, fit.km$cluster$)

We can extract some insights from above graph suc as black cluster contains wine with low flavanoids value, low proanthocyanins value, low hue value. Or green cluster contains wine which has dilution value higher than wine in red cluster.

Evaluation Because the original data set wine also has 3 classes, it is reasonable if we compare these classes with 3 clusters fited by K-Means

confuseTable.km lt;- table(wine$Type$, $fit.km$cluster) confuseTable.km 1 2 3 1 0 0 59 2 3 65 3 gt; 3 48 0 We can see only 6 sample is missed. Let's use randIndex from flexclust to compare these two parititions - one from data set and one from result of clustering method.

library(flexclust) randIndex(ct.km) ARI 0.897495 It's quite close to 1 so K-Means is good model for clustering wine data set.

References Choosing number of cluster in K-Means, http://stackoverflow.com/a/15376462/1036500 K-means Clustering (from "R in Action"), http://www.r-statistics.com/2013/08/k-means-clustering-from-r-in-action/ Color the cluster output in r, http://stackoverflow.com/questions/1538696 the-cluster-output-in-r

3.10 Ensemble

Ensemble Algorithms 1 Ensemble methods are models composed of multiple weaker models that are independently trained and whose predictions are combined in some way to make the overall prediction.

Much effort is put into what types of weak learners to combine and the ways in which to combine them. This is a very powerful class of techniques and as such is very popular.

Boosting Bootstrapped Aggregation (Bagging) AdaBoost Stacked Generalization (blending) Gradient Boosting Machines (GBM) Gradient Boosted Regression Trees (GBRT) Random Forest XGBoost XGBoost is short for eXtreme gradient boosting.

Features 1 Easy to use Easy to install Highly developed R/python for users Efficiency Automatic parallel computation on a single machine Can be run on a cluster. Accuracy Good results for most data sets Feasibility Customized object and evaluation Turnable parameters Xgboost Optimization 2 You can use xgb.plot$_i$$mportanttodecidehowmanyfeaturesinyourmodel.Usexgb.cv(example)insteadofxgb.trainwithwat$
$//www.kaggle.com/c/otto-group-product-classification-challenge/forums/t/12947/achieve-0-50776-on-the-leaderboard-in-a-minute-with-xgboost?page=5$

Installation Installation in Windows 64bit, Python 2.7, Anaconda

git clone https://github.com/dmlc/xgboost git checkout 9bc3d16 Open project in xgboost/windows with Visual Studio 2013 In Visual Studio 2013, open Configuration Manager..., choose Release in Active solution configuration choose x64 in Active solution platform Rebuild xgboost, xgboost$_w$$rapperCopyallfileinxgboost/windows/x64/Releasefold$
$package, runcommandpythonsetup.pyinstallCheckxgboostbyrunningcommandpython-c"importxgboost"ExamplesMulticlassclassification$:

Understanding XGBoost Model on Otto Dataset

Resources http://www.slideshare.net/ShangxuanZhang/xgboost youtube, Kaggle Winning Solution Xgboost algorithm – Let us learn from its author

Notes on Parameter Tuning

3.11 Dimensionality Reduction

Dimensionality Reduction Algorithms Like clustering methods, dimensionality reduction seek and exploit the inherent structure in the data, but in this case in an unsupervised manner or order to summarise or describe data using less information.

This can be useful to visualize dimensional data or to simplify data which can then be used in a supervized learning method. Many of these methods can be adapted for use in classification and regression.

Principal Component Analysis (PCA) Principal Component Regression (PCR) Partial Least Squares Regression (PLSR) Sammon Mapping Multidimensional Scaling (MDS) Projection Pursuit Linear Discriminant Analysis (LDA) Mixture Discriminant Analysis (MDA) Quadratic Discriminant Analysis (QDA) Flexible Discriminant Analysis (FDA) t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) 1 is a (prize-winning) technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. The technique can be implemented via Barnes-Hut approximations, allowing it to be applied on large real-world datasets. We applied it on data sets with up to 30 million examples. The technique and its variants are introduced in the following papers:

L.J.P. van der Maaten. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research 15(Oct):3221-3245, 2014. PDF [Supplemental material] L.J.P. van der Maaten and G.E. Hinton. Visualizing Non-Metric Similarities in Multiple Maps. Machine Learning 87(1):33-55, 2012. PDF L.J.P. van der Maaten. Learning a Parametric Embedding by Preserving Local Structure. In Proceedings of the Twelfth International Conference on Artificial Intelligence Statistics (AI-STATS), JMLR WCP 5:384-391, 2009. PDF L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(Nov):2579-2605, 2008. PDF [Supplemental material] [Talk]

3.12 Anomaly Detection

Motivation and Examples Algorithms Evaluation AD: Examples Problem motivation 1 Anomaly detection is a reasonably commonly used type of machine learning application Can be thought of as a solution to an unsupervised learning problem But, has aspects of supervised learning What is anomaly detection? Imagine you're an aircraft engine manufacturer As engines roll off your assembly line you're doing QA Measure some features from engines (e.g. heat generated and vibration) You now have a dataset of x1 to xm (i.e. m engines were tested) Say we plot that dataset Next day you have a new engine An anomaly detection method is used to see if the new engine is anomalous (when compared to the previous engines) If the new engine looks like this; Probably OK - looks like the ones we've seen before But if the engine looks like this Uh oh! - this looks like an

anomalous data-point More formally We have a dataset which contains normal (data) How we ensure they're normal is up to us In reality it's OK if there are a few which aren't actually normal Using that dataset as a reference point we can see if other examples are anomalous How do we do this? First, using our training dataset we build a model We can access this model using p(x) This asks, "What is the probability that example x is normal" Having built a model if $latexp(x_{test}) < \epsilon$ –> flag this as an anomaly if $latexp(x_{test}) \geq \epsilon$ –> this is OK is some threshold probability value which we define, depending on how sure we need/want to be We expect our model to (graphically) look something like this; i.e. this would be our model if we had 2D data Examples 1 Fraud detection Users have activity associated with them, such as Length on time on-line Location of login Spending frequency Using this data we can build a model of what normal users' activity is like What is the probability of "normal" behavior? Identify unusual users by sending their data through the model Flag up anything that looks a bit weird Automatically block cards/transactions Manufacturing Already spoke about aircraft engine example Monitoring computers in data center If you have many machines in a cluster Computer features of machine $latexx_1$ = memory use $latexx_2$ = number of disk accesses/sec $latexx_3$ = CPU load In addition to the measurable features you can also define your own complex features $latexx_4$ = CPU load/network traffic If you see an anomalous machine Maybe about to fail Look at replacing bits from it

3.13 Recomendation System

ntroduction 2 Two motivations for talking about recommender systems

Important application of ML systems Many technology companies find recommender systems to be absolutely key Think about websites (amazon, Ebay, iTunes genius) Try and recommend new content for you based on passed purchase Substantial part of Amazon's revenue generation Improvement in recommender system performance can bring in more income Kind of a funny problem In academic learning, recommender systems receives a small amount of attention But in industry it's an absolutely crucial tool Talk about the big ideas in machine learning Not so much a technique, but an idea As soon, features are really important There's a big idea in machine learning that for some problems you can learn what a good set of features are So not select those features but learn them Recommender systems do this - try and identify the crucial and relevant features Example - predict movie ratings You're a company who sells movies You let users rate movies using a 1-5 star rating To make the example nicer, allow 0-5 (makes math easier) You have five movies And you have four users Admittedly, business isn't going well, but you're optimistic about the future as a result of your truly outstanding (if limited) inventory

To introduce some notation

n_u - Number of users (called $?^{nu}$ occasionally as we can't subscript in superscript) n_m - Number of movies $r(i,j)$ - 1 if user j has rated movie i (i.e. bitmap) $y(i,j)$ - rating given by user j to move i (defined only if $latexr(i,j) = 1$) So for this example $n_u = 4$ $n_m = 5$

Summary of scoring Alice and Bob gave good ratings to rom coms, but low scores to action films Carol and Dave game good ratings for action films but low ratings for rom coms We have the data given above The problem is

as follows Given $latexr(i, j)$ and $latexy^{(i,j)}$ - go through and try and predict missing values (?s) Come up with a learning algorithm that can fill in these missing values KDD 2015 Tutorial: Shlomo Berkovsky and Jill Freyne, Web Personalisation and Recommender Systems

1. Approaches 1

Attribute-based Recommendations

You like action movies, starring Clint Eastwood, you might like "Good, Bad and the Ugly" (Netflix)

Item Hierachy

You bought Printer you will also need ink (Bestbuy)

Association Rules

Content-Based Recommender Collaborative Filtering - Item-Item Similarity

You like Godfather so you will like Scarface (Netflix)

Collaborative Filtering - User-User Similarity

People like you who bought beer also bought diapers (Target)

Social+Interest Graph Based

Your friends like Lady Gaga so you will like Lady Gaga (Facebook, Linkedin)

Model Based

Training SVM, LDA, SVD for implicit features.

2. Challenges Kaggle Challenge: Million Song Dataset Challenge

3. Articles How Big Data is used in Recommendation Systems to change our lives 4. Recommendation Interface 4.1 Type of Input predictions recommendations filtering organic vs explicit presentation 4.2 Type of Output explicit implicit Apriori https://en.wikipedia.org/wiki/Apriori$_algorithm$

https://github.com/asaini/Apriori

Item item collaborative filtering Works when |U| » |I|

items dont change much RS: Examples Google News

http://1.bp.blogspot.com/$_7ZYqYi4xigk/TCuWLmXhdjI/AAAAAAAAGVI/umfi5tHpBr0/s1600/G$
$News + Redesign + June + 30 + 2010 + AM + PT.jpg$

RS: Association Rules

Content Based Recommendation User–User Collaborative Filtering User - User 1 User user look simular in row space

$p_{u,i} = \overline{r_u} + \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \overline{r'_u})}{\sum_{u' \in N} |s(u,u')|} s = 2$

http://files.grouplens.org/papers/FnT

mlclass lecture notes, Recommender Systems

Chương 4

Probabilistic Graphical Model

View online http://magizbox.com/training/probabilistic$_g raphical_m odels/site/$

Probabilistic graphical models (PGMs) are a rich framework for encoding probability distributions over complex domains: joint (multivariate) distributions over large numbers of random variables that interact with each other. These representations sit at the intersection of statistics and computer science, relying on concepts from probability theory, graph algorithms, machine learning, and more. They are the basis for the state-of-the-art methods in a wide variety of applications, such as medical diagnosis, image understanding, speech recognition, natural language processing, and many, many more. They are also a foundational tool in formulating many machine learning problems.

4.1 Representation

Probabilistic graphical models (PGMs) are a rich framework for encoding probability distributions over complex domains: joint (multivariate) distributions over large numbers of random variables that interact with each other.

These representations sit at the intersection of statistics and computer science, relying on concepts from probability theory, graph algorithms, machine learning, and more. They are the basis for the state-of-the-art methods in a wide variety of applications, such as medical diagnosis, image understanding, speech recognition, natural language processing, and many, many more. They are also a foundational tool in formulating many machine learning problems.

4.2 Foundation: Graph

Perhaps the most pervasive concept in this book is the representation of a probability distribution using a graph as a data structure. In this section, we survey some of the basic concepts in graph theory used in the book.

1 Nodes and Edges A graph is a data structure K consisting of a set of nodes and a set of edges. Throughout most this book, we will assume that the set of nodes is X = X1,...,Xn. A pair of nodes Xi,Xj directed edge can be connected by

a directed edge Xi Xj or an undirected edge Xi—Xj. Thus, the set undirected edge of edges E is a set of pairs, where each pair is one of Xi Xj, Xj Xi, or Xi—Xj, for Xi,Xj X , i < j. We assume throughout the book that, for each pair of nodes Xi,Xj, at most one type of edge exists; thus, we cannot have both Xi Xj and Xj Xi, nor can we have Xi Xj and Xi—Xj.2 The notation Xi Xj is equivalent to Xj Xi, and the notation X j—Xi is equivalent to Xi—Xj. We use Xi Xj to represent the case where Xi and X j are connected via some edge, whether directed (in any direction) or undirected. In many cases, we want to restrict attention to graphs that contain only edges of one kind directed graph or another. We say that a graph is directed if all edges are either Xi Xj or Xj Xi. We usually denote directed graphs as G. We say that a graph is undirected if all edges are Xi—Xj. undirected graph We denote undirected graphs as H. We sometimes convert a general graph to an undirected graph by ignoring the directions on the edges. Definition 2.11 Given a graph K = (X, E), its undirected version is a graph H = (X, E0) where E0 = X—Y : graph's undirected version X Y E. Whenever we have that Xi Xj E, we say that Xj is the child of Xi in K, and that child Xi is the parent of Xj in K. When we have Xi—Xj E, we say that Xi is a neighbor of parent neighbor Xj in K (and vice versa). We say that X and Y are adjacent whenever X Y E. We use PaX to denote the parents of X, ChX to denote its children, and NbX to denote its neighbors. We define the boundary of X, denoted BoundaryX, to be PaX NbX; for DAGs, this set is boundary simply X's parents, and for undirected graphs X's neighbors.3 Figure 2.3 shows an example of a graph K. There, we have that A is the only parent of C, and F,I are the children of C. The degree only neighbor of C is D, but its adjacent nodes are A,D,F,I. The degree of a node X is the number of edges in which it participates. Its indegree is the number of directed edges Y X. indegree The degree of a graph is the maximal degree of a node in the graph. 2. Note that our definition is somewhat restricted, in that it disallows cycles of length two, where Xi Xj Xi, and allows self-loops where Xi Xi. 3. When the graph is not clear from context, we often add the graph as an additional argument.

2 Subgraphs In many cases, we want to consider only the part of the graph that is associated with a particular subset of the nodes. Definition 2.12 Let K = (X , E), and let X X . We define the induced subgraph K[X] to be the graph (X, E0) induced subgraph where E0 are all the edges X Y E such that X, Y X. For example, figure 2.4a shows the induced subgraph K[C, D, I]. A type of subgraph that is often of particular interest is one that contains all possible edges. Definition 2.13 A subgraph over X is complete if every two nodes in X are connected by some edge. The set X complete subgraph is often called a clique; we say that a clique X is maximal if for any superset of nodes Y X, clique Y is not a clique. Although the subset of nodes X can be arbitrary, we are often interested in sets of nodes that preserve certain aspects of the graph structure. Definition 2.14 We say that a subset of nodes X X is upwardly closed in K if, for any X X, we have that upward closure BoundaryX X. We define the upward closure of X to be the minimal upwardly closed subset

Y that contains X. We define the upwardly closed subgraph of X, denoted K+[X], to be the induced subgraph over Y , K[Y]. For example, the set A, B, C, D, E is the upward closure of the set C in K. The upwardly closed subgraph of C is shown in figure 2.4b. The upwardly closed subgraph of C, D, I is shown in figure 2.4c.

3 Paths and Trails Using the basic notion of edges, we can define dierent

types of longer-range connections in the graph.

Definition path

We say that X1,...,XkX1,...,Xk form a path in the graph K=(X,E)K=(X,E) if, for every i=1,...,k1i=1,...,k1, we have that either XiXi+1XiXi+1 or XiXi+1XiXi+1. A path is directed if, for at least one i, we have XiXi+1XiXi+1.

Definition trail

We say that X1,...,XkX1,...,Xk form a trail in the graph K=(X,E)K=(X,E) if, for every i=1,...,k1i=1,...,k1, we have that XiXi+1XiXi+1.

In the graph KK of figure 2.3, A,C,D,E,IA,C,D,E,I is a path, and hence also a trail. On the other hand, A,C,F,G,DA,C,F,G,D is a trail, which is not a path.

Definition connected graph

A graph is connected if for every Xi,XjXi,Xj there is a trail between XiXi and XjXj.

We can now define longer-range relationships in the graph.

Definition ancestor, descendant

We say that XX is an ancestor of YY in K=(X,E)K=(X,E), and that YY is a descendant of XX, if there exists a directed path X1,...,XkX1,...,Xk with X1=XX1=X and Xk=YXk=Y. We use DescendantsXDescendantsX to denote X's descendants, AncestorsXAncestorsX to denote X's ancestors, and NonDescendantsXNonDescendantsX to denote the set of nodes in XDescendantsXXDescendantsX.

In our example graph K, we have that F,G,IF,G,I are descendants of CC. The ancestors of CC are AA, via the path A,C,A,C, and BB, via the path B,E,D,CB,E,D,C.

A final useful notion is that of an ordering of the nodes in a directed graph that is consistent with the directionality its edges.

Definition topological ordering

Let G=(X,E)G=(X,E) be a graph. An ordering of the nodes X1,...,XnX1,...,Xn is a topological ordering relative to KK if, whenever we have XiXjEXiXjE, then i<ji<j.

Appendix A.3.1 presents an algorithm for finding such a topological ordering.

4 Cycles and Loops Note that, in general, we can have a cyclic path that leads from a node to itself, making that node its own descendant.

Definition 2.20 A cycle in K is a directed path X1, . . . , Xk where X1 = Xk. A graph is acyclic if it contains no cycle acyclic cycles. For most of this book, we will restrict attention to graphs that do not allow such cycles, since it is quite dicult to define a coherent probabilistic model over graphs with directed cycles. DAG A directed acyclic graph (DAG) is one of the central concepts in this book, as DAGs are the basic graphical representation that underlies Bayesian networks. For some of this book, we also use acyclic graphs that are partially directed. The graph K of figure 2.3 is acyclic. However, if we add the undirected edge A—E to K, we have a path A, C, D, E, A from A to itself. Clearly, adding a directed edge E A would also lead to a cycle. Note that prohibiting cycles does not imply that there is no trail from a node to itself. For example, K contains several trails: C, D, E, I, C as well as C, D, G, F, C. An acyclic graph containing both directed and undirected edges is called a partially directed PDAG acyclic graph or PDAG. The acyclicity requirement on a PDAG implies that the graph can be chain component decomposed into a directed graph of chain components, where the nodes within each chain component are connected to each other only with undirected edges. The acyclicity of a PDAG guarantees us that we can

order the components so that all edges point from lower-numbered components to higher-numbered ones. Definition 2.21 Let K be a PDAG over X . Let K1, . . . , K‘ be a disjoint partition of X such that: • the induced subgraph over Ki contains no directed edges; • for any pair of nodes X Ki and Y Kj for i < j, an edge between X and Y can only be a directed edge X Y . chain component Each component Ki is called a chain component. chain graph Because of its chain structure, a PDAG is also called a chain graph. Example 2.6 In the PDAG of figure 2.3, we have six chain components: A, B, C, D, E, F, G, H, and I. This ordering of the chain components is one of several possible legal orderings. Note that when the PDAG is an undirected graph, the entire graph forms a single chain component. Conversely, when the PDAG is a directed graph (and therefore acyclic), each node in the graph is its own chain component.

Dierent from a cycle is the notion of a loop: Definition 2.22 A loop in K is a trail X1, . . . , Xk where X1 = Xk. A graph is singly connected if it contains loop singly connected no loops. A node in a singly connected graph is called a leaf if it has exactly one adjacent node. leaf A singly connected directed graph is also called a polytree. A singly connected undirected graph is polytree called a forest; if it is also connected, it is called a tree. forest tree We can also define a notion of a forest, or of a tree, for directed graphs. Definition 2.23 A directed graph is a forest if each node has at most one parent. A directed forest is a tree if it is also connected. Note that polytrees are very dierent from trees. For example, figure 2.5 shows a graph that is a polytree but is not a tree, because several nodes have more than one parent. As we will discuss later in the book, loops in the graph increase the computational cost of various tasks. We conclude this section with a final definition relating to loops in the graph. This definition will play an important role in evaluating the cost of reasoning using graph-based representations. Definition 2.24 Let X1—X2— · · · —Xk—X1 be a loop in the graph; a chord in the loop is an edge connecting chordal graph Xi and Xj for two nonconsecutive nodes Xi, Xj. An undirected graph H is said to be chordal if any loop X1—X2— · · · —Xk—X1 for k 4 has a chord. Thus, for example, a loop A—B—C—D—A (as in figure 1.1b) is nonchordal, but adding an edge A—C would render it chordal. In other words, in a chordal graph, the longest “minimal loop” (one that has no shortcut) is a triangle. Thus, chordal graphs are often also called triangulated triangulated. graph We can extend the notion of chordal graphs to graphs that contain directed edges. Definition 2.25 A graph K is said to be chordal if its underlying undirected graph is chordal.

4.3 Bayesian Network

A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest. When used in conjunction with statistical techniques, the graphical model has several advantages for data analysis. One, because the model encodes dependencies among all variables, it readily handles situations where some data entries are missing. Two, a Bayesian network can be used to learn causal relationships, and hence can be used to gain understanding about a problem domain and to predict the consequences of intervention. Three, because the model has both a causal and probabilistic semantics, it is an ideal representation for combining prior knowledge (which often comes in causal form) and data. Four, Bayesian statistical methods in conjunction with

bayesian networks offer an efficient and principled approach for avoiding the overfitting of data. In this paper, we discuss methods for constructing Bayesian networks from prior knowledge and summarize Bayesian statistical methods for using data to improve these models. With regard to the latter task, we describe methods for learning both the parameters and structure of a Bayesian network, including techniques for learning with incomplete data. In addition, we relate Bayesian-network methods for learning to techniques for supervised and unsupervised learning. We illustrate the graphical-modeling approach using a real-world case study.

A Non-Causal Bayesian Network Example Figure 1 shows a simple Bayesian network, which consists of only two nodes and one link. It represents the JPD of the variables Eye Color and Hair Color in a population of students (Snee, 1974). In this case, the conditional probabilities of Hair Color given the values of its parent node, Eye Color, are provided in a CPT. It is important to point out that this Bayesian network does not contain any causal assumptions, i.e. we have no knowledge of the causal order between the variables. Thus, the interpretation of this network should be merely statistical (informational).

A Causal Network Example Figure 2 illustrates another simple yet typical Bayesian network. In contrast to the statistical relationships in Figure 1, the diagram in Figure 2 describes the causal relationships among the seasons of the year (X1X1), whether it is raining (X2X2), whether the sprinkler is on (X3X3), whether the pavement is wet (X4X4), and whether the pavement is slippery (X5X5). Here, the absence of a direct link between X1X1 and X5X5, for example, captures our understanding that there is no direct influence of season on slipperiness. The influence is mediated by the wetness of the pavement (if freezing were a possibility, a direct link could be added).

A Dynamic Bayesian Network Example Entities that live in a changing environment must keep track of variables whose values change over time. Dynamic Bayesian networks capture this process by representing multiple copies of the state variables, one for each time step. A set of variables Xt1Xt1 and XtXt denotes the world state at times t-1 and t respectively. A set of evidence variables Et denotes the observations available at time t. The sensor model P(Et|Xt)P(Et|Xt) is encoded in the conditional probability distributions for the observable variables, given the state variables. The transition model P(Xt|Xt1)P(Xt|Xt1) relates the state at time t-1 to the state at time t. Keeping track of the world means computing the current probability distribution over world states given all past observations, i.e. P(Xt|E1,. . . ,Et)P(Xt|E1,. . . ,Et).

Dynamic Bayesian networks (DBN) are a generalization of Hidden Markov Models (HMM) and Kalman Filters (KF). Every HMM and KF can be represented with a DBN. Furthermore, the DBN representation of an HMM is much more compact and, thus, much better understandable. The nodes in the HMM represent the states of the system, whereas the nodes in the DBN represent the dimensions of the system. For example, the HMM representation of the valve system in Figure 2.3 is made of 26 nodes and 36 arcs, versus 9 nodes and 11 arcs in the DBN (Weber and Jouffe, 2003).

4.4 Template Models for Bayesian Networks

In many cases, we need to model distributions that have a recurring structure. In this module, we describe representations for two such situations. One is temporal scenarios, where we want to model a probabilistic structure that holds constant over time; here, we use Hidden Markov Models, or, more generally, Dynamic Bayesian Networks. The other is aimed at scenarios that involve multiple similar entities, each of whose properties is governed by a similar model; here, we use Plate Models.

Temporal Models Our focus in this section is on modeling dynamic settings, where we are interested in reasoning about the state of the world as it evolves over time. We can model such settings in terms of a system state system state, whose value at time t is a snapshot of the relevant attributes (hidden or observed) of the system at time t. We assume that the system state is represented, as usual, as an assignment of values to some set of random variables X . We use X (t) i to represent the instantiation of the variable Xi at time t. Note that Xi itself is no longer a variable that takes a value; rather, it is a template variable template variable. This template is instantiated at dierent points in time t, and each Xi (t) is a variable that takes a value in Val(Xi). For a set of variables X X , we use X (t1:t2) (t1 $<$ t2) to denote the set of variables X (t) : t [t1,t2]. As usual, we use the notation x(t:t0) for an assignment of values to this set of variables.

Each "possible world" in our probability space is now a trajectory: an assignment of values to each variable X (t) i for each relevant time t. Our goal therefore is to represent a joint distribution over such trajectories. Clearly, the space of possible trajectories is a very complex probability space, so representing such a distribution can be very dicult. We therefore make a series of simplifying assumptions that help make this representational problem more tractable.

Dynamic Bayesian Networks

Directed Probabilistic Models for Object-Relational Domains Based on the framework described in the previous section, we now describe template-based representation languages that can encode directed probabilistic models.

Plate Models We begin our discussion by presenting the plate model, the simplest and best-established of the object-relational frameworks. Although restricted in several important ways, the plate modeling framework is perhaps the approach that has been most commonly used in practice, notably for encoding the assumptions made in various learning tasks. This framework also provides an excellent starting point for describing the key ideas of template-based languages and for motivating some of the extensions that have been pursued in richer languages.

In the plate formalism, object types are called plates. The fact that multiple objects in the class share the same set of attributes and same probabilistic model is the basis for the use of the term "plate," which suggests a stack of identical objects. We begin with some motivating examples and then describe the formal framework.

Examples Example 1 The simplest example of a plate model, shown in figure 6.6, describes multiple random variables generated from the same distribution. In this case, we have a set of random variables X(d) (dD)X(d) (dD) that all have the same domain Val(X) and are sampled from the same distribution. In a plate representation, we encode the fact that these variables are all generated

from the same template by drawing only a single node X(d) and enclosing it in a box denoting that d ranges over D, so that we know that the box represents an entire "stack" of these identically distributed variables. This box plate is called a plate, with the analogy that it represents a stack of identical plates.

4.5 Factor Graph

A factor graph is a bipartite graph representing the factorization of a function.

Each edge in graph defines a function

Definition A factor graph is a bipartite graph representing the factorization of a function.

Related Readings [1]: Factor Graph, wikipedia.org

4.6 Inference

This addresses the question of probabilistic inference: how a PGM can be used to answer questions.

Even though a PGM generally describes a very high dimensional distribution, its structure is designed so as to allow questions to be answered efficiently. The course presents both exact and approximate algorithms for different types of inference tasks, and discusses where each could best be applied. The (highly recommended) honors track contains two hands-on programming assignments, in which key routines of the most commonly used exact and approximate algorithms are implemented and applied to a real-world problem.

4.7 Learning

This course addresses the question of learning: how a PGM can be learned from a data set of examples.

The course discusses the key problems of parameter estimation in both directed and undirected models, as well as the structure learning task for directed models. The (highly recommended) honors track contains two hands-on programming assignments, in which key routines of two commonly used learning algorithms are implemented and applied to a real-world problem.

4.8 An Introduction to UnBBayes

UnBBayes is a probabilistic network framework written in Java. It has both a GUI and an API with inference, sampling, learning and evaluation. It supports Bayesian networks, influence diagrams, MSBN, OOBN, HBN, MEBN/PR-OWL, PRM, structure, parameter and incremental learning.

Features Probabilistic Networks: Bayesian Network (BN) Junction Tree Likelihood Weighting Gibbs Influence Diagram (ID) Multiply Sectioned Bayesian Network (MSBN) Hybrid Bayesian Network (HBN) Gaussian Mixture - Propagation under development Object-Oriented Bayesian Network (OOBN) FOL Probabilistic Network: Multi-Entity Bayesian Network (MEBN) Probabilistic

Ontology Language (PR-OWL) Learning Bayesian Network: K2 B CBL-A CBL-B Incremental Learning Sampling Logic Likelihood Weighting Gibbs Classification Performance Evaluation Evaluation using Logic Sampling Evaluation using Likelihood Weighting Sampling Installation Go to https://sourceforge.net/projects/unbbayes/files/latest/dow 4.21.18.$ziptounbbayes-4.21.18folderOpenunbbayes-4.21.18folder$, $doubleclicktounbbayes.batunbbayes-4.21.18open$

Official Videos In this section, I add some official videos from unbbayes team. There are overview

Overview In this video we are going to show the basic function we have in UnBBayes. This is the first of many tutorials we have been creating to support the demand for documentation on how to use UnBBayes. We hope this will help UnBBayes' user community to grow even more.

Bayesian Network In this video we are going to show how to create and compile a Bayesian Network (BN) in UnBBayes. This is our second of many video tutorials we have been creating to support the demand for documentation on how to use UnBBayes. We hope this will help UnBBayes' user community to grow even more.

UnBBayes Performance Evaluation for Multi-Sensor Classification Systems In this video we are going to show how to do a performance evaluation for multi-sensor classification systems in UnBBayes. It has been a while we do not post new videos, but hopefully this third one is just one more of many tutorials we will have available to support the demand for documentation on how to use UnBBayes. We hope this will help UnBBayes' user community to grow even more.

Probabilistic Ontology Modeling Using UnBBayes In this video we discuss how to model probabilistic ontologies using PR-OWL/MEBN in UnBBayes. This session was a video conference between PhD students from the Institute of Business Administration (http://www.iba.edu.pk) and Rommel Carvalho from George Mason University (http://www.gmu.edu).

4.9 Medical Domain Data

We have provided you with a joint probability distribution of symptons, conditions and diseases based on the "flu" example in class. Certain diseases are more likely than others given certain symptons, and a model such as this can be used to help doctors make a diagnosis. (Don't actually use this for diagnosis, though!). The ground-truth joint probability distribution consists of twelve binary random variables and contains 212212 possible configurations (numbered 0 to 4095), which is small enough that you can enumerate them exhaustively. The variables are as follows:

(0) IsSummer true if it is the summer season, false otherwise. (1) HasFlu true if the patient has the flu. (2) HasFoodPoisoning true if the patient has food poisoning. (3) HasHayFever true if patient has hay fever. (4) HasPneumonia true if the patient has pneumonia. (5) HasRespiratoryProblems true if the patient has problems in the respiratory system. (6) HasGastricProblems true if the patient has problems in the gastro-intestinal system. (7) HasRash true if the patient has a skin rash. (8) Coughs true if the patient has a cough. (9) IsFatigued true if the patient is tired and fatigued. (10) Vomits true if the patient has vomited. (11) HasFever true if the patient has a high fever. You can download all the

data here. The archive contains two files:

joint.dat: The true joint probability distribution over the twelve binary variables. Since each variable is binary, we can represent a * full variable assignment as a bitstring. This file lists all $2^12 assignments (one in each line) as pairs "Integer Probability" where "Integer" 0 and true = 1, an assignment to all variables results in a 12-bit binary number (with the index of the variables sho The dataset consists of samples from the above probability distribution. Each line of the file contains a complete eas$

4.10 Optical Word Recognition

We will be studying the computer vision task of recognizing words from images. The task of recognizing words is usually decomposed to recognition of individual characters from their respective images (optical character recognition, OCR), and hence inferring the word. However character recognition is often a very difficult task, and since each character is predicted independent of its neighbors, its results can often contain combinations of characters that may not be possible in English. In this homework we will augment a simple OCR model with additional factors that capture some intuitions based on character co-occurences and image similarities.

The undirected graphical model for recognition of a given word is given in the figure above. It consists of two types of variables:

Image Variables: These are observed images that we need to predict the corresponsing character of, and the number of these image variables for a word is the number of characters in the word. The value of these image variables is an observed image, represented by an integer id (less than 1000). For the description of the model, assume the id of the image at position i is represented by img(i). Character Variables: These are unobserved variables that represent the character prediction for each of the images, and there is one of these for each of the image variables. For our dataset, the domain of these variables is restricted to the ten most frequent characters in the English language (e,t,a,o,i,n,s,h,r,d [ciation]), instead of the complete alphabet. For the discussion below, assume the predicted character at position i is represented by char(i). The model for a word w will consist of len(w) observed image ids, and the same number of unobserved character variables. For a given assignment to these character variables, the model score will be specified using three types of factors:

OCR Factors, oo : These factors capture the predictions of a character-based OCR system, and hence exist between every image variable and its corresponding character variable. The number of these factors of word w is len(w). The value of factor between an image variable and the character variable at position i is dependent on img(i) and char(i), and is stored in ocr.dat file described in the data section. Transition Factors, tt : Since we also want to represent the co-occurence frequencies of the characters in our model, we add these factors between all consecutive character variables. The number of these factors of word w is len(w)-1. The value of factor between two character variables at positions i and i+1 is dependent on char(i) and char(i+1), and is high if char(i+1) is frequently preceded by char(i) in english words. These values are given to you in trans.dat file described in the data section. Skip Factors, ss : Another intuition that we would like to capture in our model is that similar images in a word always represent the same character. Thus our model score should be higher if it predicts the same characters for similar images. These factors exist between

every pair of image variables that have the same id, i.e. this factor exist between all i,j, i!=j such that img(i)==img(j). The value of this factor depends on char(i) and char(j), and is 5.0 if char(i)==char(j), and 1.0 otherwise. You can download all the data here. The archive contains the following files:

ocr.dat: Contains the output predictions of a pre-existing OCR system for the set of thousand images. Each row contains three tab separated values "id a prob" and represents the OCR system's probability that image id represents character aa, p(char=a|img=id)=probp(char=a|img=id)=prob. Use these values directly as the value of the factor between image and character variables at position ii, o(image(i)=id,char(i)=a)=probo(image(i)=id,char(i)=a)=prob. Since there are 10 characters and 1000 images, the total number of rows in this file is 10,000. trans.dat: Stores the factor potentials for the transition factors. Each row contains three tab-separated values "a b value" that represents the value of factor when the previous character is "a" and the next character is "b", i.e. (char(i)=a, char(i+1)=b) = value. The number of rows in the file is 100 (10*10). data.dat (and truth.dat): Dataset to run your experiments on (see Core Tasks below). The observed dataset (data.dat) consists observed images of one word on each row. The observed images for a word are represented by a sequence of tab-separated integer ids ("id1 id2 id3"). The true word for these observed set of images is stored the respective row in truth.dat, and is simply a string ("eat"). For the core task (3) below, you should iterate through both the files together to ensure you have the true word along with the observed images. Extra files (bicounts.dat, allwords.dat, allimagesX.dat): These files are not necessary for the core tasks, but may be useful for further fun and your own exploration. allwords.dat and allimagesX.dat are larger versions of data.dat and truth.dat, i.e. they contain all possible words that can be generated from our restricted set of alphabet, and five samples of their observed image sequences (one in each file). You can run inference on these if you like, but is likely to take 15-20 times longer than the small dataset. bicount.dat is in the same format as trans.dat, but instead of storing inexplicable potentials, it stores the joint probability of the co-occurences of the characters. Core Task 1. Graphical Model: Implement the graphical model containing the factors above. For any given assignment to the character variables, your model should be able to calculate the model score. Implemention should allow switching between three models:

OCR model: only contains the OCR factors Transition model: contains OCR and Transition factors Combined model: containing all three types of factors Note: To avoid errors arising from numerical issues, we suggest you represent the factors in the log-space and take sums as much as possible, calculating the log of the model score.

2. Exhaustive Inference: Using the graphical model, write code to perform exhaustive inference, i.e. your code should be able to calculate the probability of any assignment of the character and image variables. To calculate the normalization constant Z for the word w, you will need to go through all possible assignments to the character variables (there will be 10len(w)10len(w) of these).

3. Model Accuracy: Run your model on the data given in the file data.dat. For every word in the dataset, pick the assignment to character variables that has the highest probability according to the model, and treat this as the model prediction for the word. Using the truth given in truth.dat, compare the accuracy of the model predictions using the following three metrics: 1. Character-wise accuracy: Ratio of correctly predicted characters to total number of characters 2.

Word-wise accuracy: Ratio of correctly predicted words to total number of words 3. Average Dataset log-likelihood: For each word given in data.dat, calculate the log of the probability of the true word according to the model. Compute the average of this value for the whole dataset.

Compare all of the three models described in (1) using these three metrics. Also give some examples of words that were incorrect by the OCR model but consequently fixed by the Transition model, and examples of words that were incorrect by the OCR, partially corrected by the Transition model, and then completely fixed by the Combined model.

Chương 5

Học sâu

24/11/2017 - Từ hôm nay, mỗi ngày sẽ ghi chú một phần (rất rất nhỏ) về Deep Learning

22/11/2017 - Phải nói quyển này hơi nặng so với mình. Nhưng thôi cứ cố gắng vậy.

Tài liệu cũ: http://magizbox.com/training/deep_learning/site/

Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence.

5.1 Deep Feedforward Networks

deep feedforward networks

Deep feedforward networks, (hay còn gọi là feedforward neural networks) hoặc multilayer perceptrons (MLPs) là mô hình deep learning cơ bản nhất. Mục tiêu của một feedforward network là xấp xỉ một hàm $f*$. Ví dụ, với bài toán phân loại, $y = f*(x)$ chuyển đầu vào x thành một nhãn y. Một feedforward network định nghĩa một ánh xạ $y = f(x, \theta)$ và học giá trị của θ để xấp xỉ hàm $f*$ tốt nhất.

Mô hình được gọi là feedforward vì giá trị của x đang lan truyền qua mạng đến output, mà không có chiều ngược lại. Các mô hình neural có sự lan truyền ngược lại được gọi là recurrent neural network.

Feedforward là nền tảng cho rất nhiều mô hình mạng neural hiện đại. Ví dụ, các ứng dụng trong việc nhận diện ảnh thường áp dụng mô hình convolutional, là một dạng đặc biệt của feedforward network. Feedforward network cũng là nền tảng cho mạng recurrent network, có nhiều ứng trọng xử lý ngôn ngữ.

Feedforward neural networks được gọi là network vì nó thường được biểu diễn bởi một đồ thị gồm nhiều tầng, định nghĩa các hàm được kết hợp với nhau như thế nào. Ví dụ, chúng ta có 3 hàm f(1)(x), f(2)(x), f(3)(x) được đặt với nhau thành một chuỗi, hình thành hàm f(x) = f(3)(f(2)(f(1)(x))). Cấu trúc chuỗi này là dạng phổ biến nhất trong mạng neural. Trong trường hợp này f(1)(x) được gọi là layer 1, f(2)(x) được gọi là layer 2, và cứ tiếp tục như vậy. Độ dài của chuỗi thể hiện độ sâu của mô hình. Thuật ngữ "deep learning" xuất phát từ điều này. Layer cuối cũng được goi là output layer. Trong mạng neural, mục tiêu là học hàm f*(x). Ứng với mỗi giá trị đầu vào x, sẽ có tương ứng một giá trị y. Mục tiêu của mạng là tìm các tham số để y xấp xỉ với f*(x). Nhưng việc

học của hàm không hoàn toàn phụ thuộc vào x, y, mà do learning algorithm quyết định. Vì dữ liệu huấn luyện không chỉ rõ giá trị ở những layer ở giữa, nên chúng được gọi là các hidden layers Cuối cùng, mạng neural được gọi là neural vì nó lấy cảm hứng từ ngành khoa học thần kinh. Mỗi hidden layer trong mạng thường là các giá trị vector. Mỗi phần tử trong vector sẽ quyết định việc hành xử thế nào với các input đầu vào và đưa ra output.

Ta có thể coi mỗi layer chứa rất nhiều units hoạt động song song, đóng vai trò như một hàm ánh xạ vector sang giá trị số. Mặc dù các kiến trúc của mạng neural lấy rất nhiều ý tưởng từ ngành khoa học thần kinh, tuy nhiên, mục tiêu của mạng neural không phải để mô phỏng bộ não. Một cách nhìn tốt hơn là xem mạng neural như một máy học, được thiết kế với các điểm nhấn từ những gi chúng ta biết về não người. Một cách để hiểu mạng neural là bắt đầu với các mô hình linear, và xem xét các hạn chế của các mô hình này. Linear models, như logistic regression hay linear regression, hấp dẫn ở chỗ chúng có thể fit rất hữu quả và đang tin cậy, với các close form và tối ưu hóa lồi. Một hàn chế hiển nhiên của linear là các hàm linear, nên model không thể hiểu tương tác giữa các input. Để mở rộng lienar model để biểu diễn hàm nonlinear của x, chúng ta có thể áp dụng linear model không phải cho x mà cho (x), trong đó là một nonlinear transformer. Có thể xem như một tập các feature mô tả x, hoặc một cách để biểu diễn x. Vấn đề là chọn mapping . Có 3 cách thông dụng, chọn một generic như RBF, thiết kế bằng tay, hoặc học .

Nội dung chương này:

- 1. Ví dụ cơ bản củ feedforward network
- 2. Design decisions cho việc thiết kế mạng feedforward network
- 3. Choose activation functions trong hidden layer
- 4. Thiết kế mạng neural, bao nhiêu layers, các layer kết nối với nhau như thế nào, có bao nhiêu unit trong một layer
  - chosing the optimizer
  - cost function
  - form of the output units
- 5. Thuật toán Back-propagation
- 6. Góc nhìn lịch sử

Tham khảo Chapter 6, Deep Learning Book Neural Network Zoo http://www.asimovinstitute.org/neural-network-zoo/

5.2 Thực hành: Bắt đầu với Tensorflow

Tensorflow

Install tensorflow in Windows Anaconda Anaconda is the leading open data science platform powered by Python. The open source version of Anaconda is a high performance distribution of Python and R and includes over 100 of the most popular Python, R and Scala packages for data science.

Step 1: Download the Anaconda installer

Step 2: Double click the Anaconda installer and follow the prompts to install to the default location.

After a successful installation you will see output like this:

CUDA Toolkit 8.0 The NVIDIA CUDA Toolkit provides a comprehensive development environment for C and C++ developers building GPU-accelerated applications. The CUDA Toolkit includes a compiler for NVIDIA GPUs, math libraries, and tools for debugging and optimizing the performance of your applications. You'll also find programming guides, user manuals, API reference, and other documentation to help you get started quickly accelerating your application with GPUs.

Step 1: Verify the system has a CUDA-capable GPU.

Step 2: Download the NVIDIA CUDA Toolkit.

Step 3: Install the NVIDIA CUDA Toolkit.

Step 4: Test that the installed software runs correctly and communicates with the hardware.

cuDNN

The NVIDIA CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers. cuDNN is part of the NVIDIA Deep Learning SDK.

Step 1: Register an NVIDIA developer account

Step 2: Download cuDNN v5.1, you will get file like that cudnn-8.0-windows7-x64-v5.1.zip

Step 3: Copy CUDNN files to CUDA install

Extract your cudnn-8.0-windows7-x64-v5.1.zip file, and copy files to corresponding CUDA folder

In my environment, CUDA installed in C:FilesGPU Computing Toolkit8.0, you must copy append three folders bin, include, lib

Install Tensorflow Package

CPU TensorFlow environment

conda create –name tensorflow python=3.5 activate tensorflow conda install -y jupyter scipy pip install tensorflow GPU TensorFlow environment

conda create –name tensorflow-gpu python=3.5 activate tensorflow-gpu conda install -y jupyter scipy pip install tensorflow-gpu word2vec Example Step 1: Download word2vec example from github

dir

02/06/2017 11:45 DIR . 02/06/2017 11:45 DIR .. 02/06/2017 10:12 9,144 word2vec$_b$*asic.pyStep*2 : *Runword*2*vec*$_b$*asicexample*

activatetensorflow − *gpu* python word2vec$_b$*asic.py*

Found and verified text8.zip Data size 17005207 Most common words (+UNK) [['UNK', 418391], ('the', 1061396), ('of', 593677), ('and', 416629), ('one', 411764)] Sample data [5241, 3082, 12, 6, 195, 2, 3136, 46, 59, 156] ['anarchism', 'originated', 'as', 'a', 'term', 'of', 'abuse', 'first', 'used', 'against'] 3082 originated -> 5241 anarchism 3082 originated -> 12 as 12 as -> 6 a 12 as -> 3082 originated 6 a -> 195 term 6 a -> 12 as 195 term -> 2 of 195 term -> 6 a Initialized Average loss at step 0 : 288.173675537 Nearest to its: nasl, tinkering, derivational, yachts, emigrated, fatalism, kingston, kochi, Nearest to into: streetcars, neglecting, deutschlands, lecture, realignment, bligh, donau, medalists, Near-

est to state: canterbury, exceptions, disaffection, crete, westernmost, earthly, organize, richland,

5.3 Tài liệu Deep Learning

Lang thang thế nào lại thấy trang này My Reading List for Deep Learning! của một anh ở Microsoft. Trong đó, (đương nhiên) có Deep Learning của thánh Yoshua Bengio, có một vụ hay nữa là bài review "Deep Learning" của mấy thánh Yann Lecun, Yoshua Bengio, Geoffrey Hinton trên tạp chí Nature. Ngoài ra còn có nhiều tài liệu hữu ích khác.

5.4 Các layer trong deep learning

5.4.1 Sparse Layers

nn.Embedding (hướng dẫn)

grep code: Shawn1993/cnn-text-classification-pytorch

Đóng vai trò như một lookup table, map một word với dense vector tương ứng

5.4.2 Convolution Layers

nn.Conv1d, nn.Conv2d, nn.Conv3d)

grep code: Shawn1993/cnn-text-classification-pytorch, galsang/CNN-sentence-classification-pytorch

Các tham số trong Convolution Layer

* $kernel\_size$ (hay là filter size)

Đối với NLP, $kernel\_size$ thường bằng $region\_size * word\_dim$ (đối với conv1d) hay ($region\_size$, $word\_dim$) đối với conv2d

<small>Quá trình tạo feature map đối với region size bằng 2</small>

* ‘in$_c$$hannels$‘, ‘$out_c$$hannels$‘($lslng$‘$featuremaps$‘)

Kênh (channels) là các cách nhìn (view) khác nhau đối với dữ liệu. Ví dụ, trong ảnh thường có 3 kênh RGB (red, green, blue), có thể áp dụng convolution giữa các kênh. Với văn bản cũng có thể có các kênh khác nhau, như khi có các kênh sử dụng các word embedding khác nhau (word2vec, GloVe), hoặc cùng một câu nhưng biểu diễn ở các ngôn ngữ khác nhau.

* ‘stride‘

Định nghĩa bước nhảy của filter.

Hình minh họa sự khác biệt giữa các feature map đối với stride=1 và stride=2. Feature map đối với stride = 1 có kích thước là 5, feature map đối với stride = 3 có kích thước là 3. Stride càng lớn thì kích thước của feature map càng nhỏ.

Trong bài báo của Kim 2014, ‘stride = 1‘ đối với ‘nn.conv2d‘ và ‘stride = word$_d$$im$‘$Øivi$‘$nn.conv1d$‘

Toàn bộ tham số của mạng CNN trong bài báo Kim 2014,

| Description | Values | |———————|———————| | input word vectors | Google word2vec | | filter region size | (3, 4, 5) | | feature maps | 100 | | activation function | ReLU | | pooling | 1-max pooling | | dropout rate | 0.5 | | *latexlamp*; $s = 22$ norm constraint | 3 |

Đọc thêm:

* [Lecture 13: Convolutional Neural Networks (for NLP). CS224n-2017](http://web.stanford.edu/class/cs2 2017-lecture13-CNNs.pdf) * [DeepNLP-models-Pytorch - 8. Convolutional Neural Networks](https://nbviewer.jupyter.org/github/DSKSD/DeepNLP-models-Pytorch/blob/master/notebooks/08.CNN-for-Text-Classification.ipynb) * [A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. Zhang 2015](https://arxiv.org/pdf/1510.03820.pdf)

5.5 Recurrent Neural Networks

What are RNNs? The idea behind RNNs is to make use of sequential information. In a traditional neural network we assume that all inputs (and outputs) are independent of each other. But for many tasks that's a very bad idea. If you want to predict the next word in a sentence you better know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps (more on this later). Here is what a typical RNN looks like:

A recurrent neural network and the unfolding in time of the computation involved in its forward computation

A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: Nature The above diagram shows a RNN being unrolled (or unfolded) into a full network. By unrolling we simply mean that we write out the network for the complete sequence. For example, if the sequence we care about is a sentence of 5 words, the network would be unrolled into a 5-layer neural network, one layer for each word. The formulas that govern the computation happening in a RNN are as follows:

xtxt is the input at time step tt. For example, x1x1 could be a one-hot vector corresponding to the second word of a sentence. ststst is the hidden state at time step tt. It's the "memory" of the network. ststst is calculated based on the previous hidden state and the input at the current step: st=f(Uxt+Wst1)st=f(Uxt+Wst1). The function ff usually is a nonlinearity such as tanh or ReLU. s1s1, which is required to calculate the first hidden state, is typically initialized to all zeroes. otot is the output at step tt. For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary. ot=softmax(Vst)ot=softmax(Vst). There are a few things to note here:

You can think of the hidden state ststst as the memory of the network. ststst captures information about what happened in all the previous time steps. The output at step otot is calculated solely based on the memory at time tt. As briefly mentioned above, it's a bit more complicated in practice because ststst

typically can't capture information from too many time steps ago. Unlike a traditional deep neural network, which uses different parameters at each layer, a RNN shares the same parameters (UU, VV, WW above) across all steps. This reflects the fact that we are performing the same task at each step, just with different inputs. This greatly reduces the total number of parameters we need to learn. The above diagram has outputs at each time step, but depending on the task this may not be necessary. For example, when predicting the sentiment of a sentence we may only care about the final output, not the sentiment after each word. Similarly, we may not need inputs at each time step. The main feature of an RNN is its hidden state, which captures some information about a sequence. What can RNNs do? RNNs have shown great success in many NLP tasks. At this point I should mention that the most commonly used type of RNNs are LSTMs, which are much better at capturing long-term dependencies than vanilla RNNs are. But don't worry, LSTMs are essentially the same thing as the RNN we will develop in this tutorial, they just have a different way of computing the hidden state. We'll cover LSTMs in more detail in a later post. Here are some example applications of RNNs in NLP (by non means an exhaustive list).

Language Modeling and Generating Text Given a sequence of words we want to predict the probability of each word given the previous words. Language Models allow us to measure how likely a sentence is, which is an important input for Machine Translation (since high-probability sentences are typically correct). A side-effect of being able to predict the next word is that we get a generative model, which allows us to generate new text by sampling from the output probabilities. And depending on what our training data is we can generate all kinds of stuff. In Language Modeling our input is typically a sequence of words (encoded as one-hot vectors for example), and our output is the sequence of predicted words. When training the network we set ot=xt+1ot=xt+1 since we want the output at step tt to be the actual next word.

Research papers about Language Modeling and Generating Text:

Recurrent neural network based language model Extensions of Recurrent neural network based language model Generating Text with Recurrent Neural Networks Machine Translation Machine Translation is similar to language modeling in that our input is a sequence of words in our source language (e.g. German). We want to output a sequence of words in our target language (e.g. English). A key difference is that our output only starts after we have seen the complete input, because the first word of our translated sentences may require information captured from the complete input sequence.

RNN for Machine Translation

RNN for Machine Translation. Image Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

Research papers about Machine Translation:

A Recursive Recurrent Neural Network for Statistical Machine Translation Sequence to Sequence Learning with Neural Networks Joint Language and Translation Modeling with Recurrent Neural Networks Speech Recognition Given an input sequence of acoustic signals from a sound wave, we can predict a sequence of phonetic segments together with their probabilities.

Research papers about Speech Recognition:

Towards End-to-End Speech Recognition with Recurrent Neural Networks Generating Image Descriptions Together with convolutional Neural Networks, RNNs have been used as part of a model to generate descriptions for unlabeled

images. It's quite amazing how well this seems to work. The combined model even aligns the generated words with features found in the images.

Deep Visual-Semantic Alignments for Generating Image Descriptions. Source: http://cs.stanford.edu/people/karpathy/deepimagesent/

Training RNNs Training a RNN is similar to training a traditional Neural Network. We also use the backpropagation algorithm, but with a little twist. Because the parameters are shared by all time steps in the network, the gradient at each output depends not only on the calculations of the current time step, but also the previous time steps. For example, in order to calculate the gradient at t=4t=4 we would need to backpropagate 3 steps and sum up the gradients. This is called Backpropagation Through Time (BPTT). If this doesn't make a whole lot of sense yet, don't worry, we'll have a whole post on the gory details. For now, just be aware of the fact that vanilla RNNs trained with BPTT have difficulties learning long-term dependencies (e.g. dependencies between steps that are far apart) due to what is called the vanishing/exploding gradient problem. There exists some machinery to deal with these problems, and certain types of RNNs (like LSTMs) were specifically designed to get around them.

RNN Extensions Over the years researchers have developed more sophisticated types of RNNs to deal with some of the shortcomings of the vanilla RNN model. We will cover them in more detail in a later post, but I want this section to serve as a brief overview so that you are familiar with the taxonomy of models.

Bidirectional RNNs are based on the idea that the output at time t may not only depend on the previous elements in the sequence, but also future elements. For example, to predict a missing word in a sequence you want to look at both the left and the right context. Bidirectional RNNs are quite simple. They are just two RNNs stacked on top of each other. The output is then computed based on the hidden state of both RNNs.

Deep (Bidirectional) RNNs are similar to Bidirectional RNNs, only that we now have multiple layers per time step. In practice this gives us a higher learning capacity (but we also need a lot of training data).

Deep Bidirectional RNN LSTM networks are quite popular these days and we briefly talked about them above. LSTMs don't have a fundamentally different architecture from RNNs, but they use a different function to compute the hidden state. The memory in LSTMs are called cells and you can think of them as black boxes that take as input the previous state ht1ht1 and current input xtxt. Internally these cells decide what to keep in (and what to erase from) memory. They then combine the previous state, the current memory, and the input. It turns out that these types of units are very efficient at capturing long-term dependencies. LSTMs can be quite confusing in the beginning but if you're interested in learning more this post has an excellent explanation.

Conclusion So far so good. I hope you've gotten a basic understanding of what RNNs are and what they can do. In the next post we'll implement a first version of our language model RNN using Python and Theano. Please leave questions in the comments!

$[^1] : [Understanding Convolutional Neural Networks for NLP](http://www.wildml.com/2015/11/unde$
$convolutional-neural-networks-for-nlp)[^2] : [http://pytorch.org/docs/master/nn.html](http:$
$//pytorch.org/docs/master/nn.html)$

Chương 6

Xử lý ngôn ngữ tự nhiên

Bản lưu cũ http://magizbox.com/training/natural_language_processing/site/

Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages.

6.1 Introduction to Natural Language Processing

Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages.

NLP is related to the area of human–computer interaction. Many challenges in NLP involve: natural language understanding, enabling computers to derive meaning from human or natural language input; and others involve natural language generation.

The input and output of an NLP system can be either speech or written text.

Components of NLP

There are two components of NLP as given

Natural Language Understanding (NLU): this task mapping the given input in natural language into useful representations and analyzing different aspects of the language. Natural Language Generation (NLG): In the process of producing meaningful phrases and sentences in the form of natural language form some internal representation. It involves text planning retrieve the relevant content from knowledge base, sentence planning choose required words, forming meaningful phrases, setting tone of the sentence, text realization map sentence plan into sentence structure. Difficulties

Natural Language has an extremely rich form and structure. It is very ambiguous. There can be different levels of ambiguity

Lexical ambiguity: it is at very primitive level such as word-level. For example, treating the word “board” as noun or verb? Syntax level ambiguity: A sentence be parsed in different ways. For example, “He lifted the beetle with the red cap?” - did he use cap to lift the beetle or he lifted a beetle that had

red cap? Referential ambiguity: referring to something using pronouns. For example, Rima went to Gauri. She said "I am tired". - Exactly who is tired? One input can mean different meanings. Many inputs can mean the same thing.

6.2 Natural Language Processing Tasks

The analysis of natural language is broken into various board levels such as phonological, morphological, syntactic, semantic, pragmatic and discourse analysis.

Phonological Analysis Phonology is analysis of spoken language. Therefore, it deals with speech recognition and generation. The core task of speech recognition and generation system is to take an acoustic waveform as input and produce as output, a string of words. The phonology is a part of natural language analysis, which deals with it. The area of computational linguistics that deals with speech analysis is computational phonology

Example: Hans Rosling's shortest TED talk

Original Sound

0:00 / 0:52

Text X means unknown but the world is pretty known it's seven billion people have seven stones. One billion can save money to fly abroad on holiday every year. One billion can save money to keep a car or buy a car. And then three billion they save money to pay the by be a bicycle or perhaps a two-wheeler. And two billion they are busy saving money to buy shoes. In the future they will get rich and these people we move over here, these people will move over here, we will have two billion more in the world like this and the question is whether the rich people over there are prepared to be integrated in the world with 10 bilions people. Auto generated sound

0:00 / 0:36

Morphological Analysis It is the most elementary phase of NLP. It deals with the word formation. In this phase, individual words are analyzed according to their components called "morphemes". In addition, non-word taken such as punctuation, etc. are separated from words. Morpheme is basic grammatical building block that makes words.

The study of word structure is refereed to as morphology. In natural language processing, it is done in morphological analysis. The task of breaking a word into its morphemes is called morphological parsing. A morpheme is defined as minimal meaningful unit in a language, which cannot be further broken into smaller units.

Example: word fox consists a single morpheme, as it cannot be further resolved into smaller units. Whereas word cats consists two morphemes, the morpheme "cat" and morpheme "s" indicating plurality.

Here we defined the term meaningful. Though cat can be broken in "c" and "at", but these do not relate with word "cat" in any sense. Thus word "cat" will be dealt with as minimum meaningful unit.

Morphemes are traditionally divided into two types

(i) "free morphemes", that are able to act as words in isolation (e.g., "thing", "permanent", "local") (ii) "bound morphemes", that can operate only as part of other words (e.g., "is" 'ing' etc) The morpheme, which forms the center part of the world, is also called "stem". In English, a word can be made up of one or more

morphemes, e.g., word - thing -> stem "think" word - localize -> stem "local", suffix "ize" word - denationalize -> prefix "de", stem "nation", suffix "al", "ize" The computational tool to perform morphological parsing is finite state transducer. A transducer performs it by mapping between the two sets of symbols, and a finite state transducer does it with finite automaton. A transducer normally consists of four parts: recognizer, generator, translator, and relator. The output of the transducer becomes a set of morphemes.

Lexical Analysis In this phase of natural language analysis, validity of words according to lexicon is checked. Lexicon stands for dictionary. It is a collection of all possible valid words of language along with their meaning.

In NLP, the first stage of processing input text is to scan each word in sentence and compute (or look-up) all the relevant linguistic information about that word. The lexicon provides the necessary rules and data for carrying out the first stage analysis.

The details of words, like their type (noun, verb and adverb, and other details of nouns and verb, etc.) are checked.

Lexical analysis is dividing the whole chunk of text into paragraphs, sentences, and words.

Syntactic Analysis Syntax refers to the study of formal relationships between words of sentences. In this phase the validity of a sentence according to grammar rules is checked. To perform the syntactic analysis, the knowledge of grammar and parsing is required. Grammar is formal specification of rules allowable in the language, and parsing is a method of analyzing a sentence to determine its structure according to grammar. The most common grammar used for syntactic analysis for natural languages are context free grammar (CFG) also called phase structure grammar and definite clause grammar. These grammars are described in detail in a separate actions.

Syntactic analysis is done using parsing. Two basic parsing techniques are: top-down parsing and bottom-up parsing.

Semantic Analysis In linguistics, semantic analysis is the process of relating syntactic structures, from the levels of phrases, clauses, sentences and paragraphs to the level of the writing as a whole, to their language-independent meanings. It also involves removing features specific to particular linguistic and cultural contexts, to the extent that such a project is possible.

The elements of idiom and figurative speech, being cultural, are often also converted into relatively invariant meanings in semantic analysis. Semantics, although related to pragmatics, is distinct in that the former deals with word or sentence choice in any given context, while pragmatics considers the unique or particular meaning derived from context or tone. To reiterate in different terms, semantics is about universally coded meaning, and pragmatics the meaning encoded in words that is then interpreted by an audience

Discourse Analysis The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence.

Topics of discourse analysis include:

The various levels or dimensions of discourse, such as sounds, gestures, syntrax, the lexicon, style, rhetoric, meanings, speech acts, moves, strategies, turns, and other aspects of interaction Genres of discourse (various types of discourse in politics, the media, education, science, business, etc.) The relations between text (discourse) and context The relations between discourse and power The re-

lations between discourse and interaction The relations between discourse and cognition and memory Pragmatic Analysis During this, what was said is reinterpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge.

Sentiment Analysis MetaMind, @RichardSocher

Named Entity Recognition KDD 2015 Tutorial: Automatic Entity Recognition and Typing from Massive Text Corpora - A Phrase and Network Mining Approach

Relationship Extraction AlchemyAPI

6.3 Natural Language Processing Applications

Information Retrieval (IR) Information retrieval (IR) is the activity of obtaining information resources relevant to an information need from a collection of information resources. Searches can be based on metadata or on full-text (or other content-based) indexing.

Information Extraction (IE) Information extraction (IE) is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents. In most of the cases this activity concerns processing human language texts by means of natural language processing (NLP).

Machine Translation Machine translation, sometimes referred to by the abbreviation MT (not to be confused with computer-aided translation, machine-aided human translation (MAHT) or interactive translation) is a sub-field of computational linguistics that investigates the use of software to translate text or speech from one language to another.

Question Answering (QA) Question answering (QA) is a computer science discipline within the fields of information retrieval and natural language processing (NLP), which is concerned with building systems that automatically answer questions posed by humans in a natural language.

6.4 Spelling Correction

For instance, we may wish to retrieve documents containing the term carrot when the user types the query carot. Google reports (http://www.google.com/jobs/britney.html) that the following are all treated as misspellings of the query britney spears: britian spears, britney's spears, brandy spears and prittany spears

We look at two steps to solving this problem: the first based on edit distance and the second based on k-gram overlap. Before getting into the algorithmic details of these methods, we first review how search engines provide spell-correction as part of a user experience.

Implementing spelling correction There are two basic principles underlying most spelling correction algorithms.

Of various alternative correct spellings for a mis-spelled query, choose the nearest one. This demands that we have a notion of nearness or proximity between a pair of queries. When two correctly spelled queries are tied (or nearly tied), select the one that is more common. For instance, grunt and grant both seem equally plausible as corrections for grnt. Then, the algorithm should choose

the more common of grunt and grant as the correction. The simplest notion of more common is to consider the number of occurrences of the term in the collection; thus if grunt occurs more often than grant, it would be the chosen correction. A different notion of more common is employed in many search engines, especially on the web. The idea is to use the correction that is most common among queries typed in by other users. The idea here is that if grunt is typed as a query more often than grant, then it is more likely that the user who typed grnt intended to type the query grunt. Corpus Birkbeck spelling error corpus

References How to Write a Spelling Corrector. Peter Norvig. 2007 Statistical Natural Language Processing in Python. Peter Norvig. 2007 Spelling correction. Introduction to Information Retrieval. 2008

6.5 Word Vectors

Discrete Representation Use a taxonomy like WordNet that has hypernyms (is-a) relationships

from nltk.corpus import wordnet as wn panda = wn.synset("panda.n.01") hyper = lambda s: s.hypernyms() list(panda.closure(hyper)) [Synset('procyonid.n.01'), Synset('carnivore.n.01'), Synset('placental.n.01'), Synset('mammal.n.01'), Synset('vertebrate.n.01'), Synset('chordate.n.01'), Synset('animal.n.01'), Synset('organism.n.01'), Synset('living$_thing.n.01'), Synset('w$

Great as resource but missing nuances, e.g. synonyms: adept, expert, good, practiced, proficient, skillful? Missing new words (impossible to keep up to date): wicked, badass, nifty, crack, ace, wizard, genius, ninjia Subjective Requires human labor to create and adapt Hard to compute accurate word similarity Word2Vec Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.

Word2vec was created by a team of researchers led by Tomas Mikolov at Google. The algorithm has been subsequently analysed and explained by other researchers. Embedding vectors created using the Word2vec algorithm have many advantages compared to earlier algorithms like Latent Semantic Analysis.

Main Idea of Word2Vec

Instead of capturing cooccurrence counts directly,, Predict surrounding words of every word Both are quite similar, see "Glove: Global Vectors for Word Representation" by Pennington et at. (2014) and Levy and Goldberg (2014)... more later. Faster and can easily incorporate a new sentence/document or add a word to the vocabulary. Detail of Word2Vec

Predict surrounding words in a window of length m of every word. Objective function: Maximize the log probability of any context word given the current cenetr word: J()=1Tt=1Tmjm,j0log p(wt+j|wt) J()=1Tt=1Tmjm,j0log p(wt+j|wt) where represents all variables we optimize

Predict surrounding words in a window of length m of every word For

p(wt+j|wt)p(wt+j|wt) the simplest first formulation is p(o|c)=exp(uTovc)Ww=1exp(uTwvc) p(o|c)=exp(uoTvc)w=1Wexp(uwTvc) where o is the outside (or output) word id, c is the center word id, u and v are "center" and "outside" vectors of o and c

Every word has two vectors! This is essentially "dynamic" logistic regression Linear Relationships in word2vec

These representations are very good at encoding dimensions of similarity!

Analogies testing dimensions of similarity can be solved quite well just by doing vector subtraction in the embedding space Syntactically

xapplexapplesxcarxcarsxfamilyxfamiliesxapplexapplesxcarxcarsxfamilyxfamilies Similarly for verb and adjective morphological forms Semantically (Semeval 2012 task 2)

xshirtxclothingxchairxfurniturexshirtxclothingxchairxfurniture xkingxmanxqueenxwomanxkingxmanxqueenxwoman GloVe Project

Highlights Training Model Overview GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Pre-trained Model fastText

Pre-trained word vectors for 294 languages, trained on Wikipedia using fastText. These vectors in dimension 300 were obtained using the skip-gram model described in Bojanowski et al. (2016) with default parameters.

glove

Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/

Language: English

Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, 300d vectors, 822 MB download): glove.6B.zip Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, 200d vectors, 1.42 GB download): glove.twitter.27B.zip word2vec-GoogleNews-vectors

Language: English

Pre-trained Google News corpus (3 billion running words) word vector model (3 million 300-dimension English word vectors).

Word Analogies Test for linear relationships, examined by Mikolov et at. (2014)

Suggested Readings Simple Word Vector representations: word2vec, GloVe. cs224d.stanford.edu. Last Accessed: 2017-02-01. FastText and Gensim word embeddings. rare-technologies.com. Last Accessed: 2016-08-31. Distributed Representations of Words and Phrases and their Compositionality. papers.nips.cc. Last Accessed: 2013-12-05. Efficient Estimation of Word Representations in Vector Space. arxiv.org. Last Accessed: 2013-01-16

6.6 Conditional Random Fields in Name Entity Recognition

In this tutorial, I will write about how to using CRF++ to train your data for name entity recognition task.

Environment:

Ubuntu 14.04 Install CRF++ Download CRF++-0.58.tar.gz

Extact CRF++-0.58.tar.gz file

Navigate to the location of extracted folder through

Install CRF++ from source

./configure make sudo make install ldconfig Congratulations! CRF++ is install

$\mathrm{crf}_{l}earnTrainingCRFTotrainaCRFusingCRF++, youneed2things$:

A template file: where you define features to be considered for training A training data file: where you have data in CoNLL format $\mathrm{crf}_{l}earn{-}ttemplate_{f}iletrain_{d}ata_{f}ilemodel$

$\mathrm{crf}_{l}earn - ttemplatetrain.txtmodelAbinaryofmodelisproduce.$

To test this model, on a testing data

$\mathrm{crf}_{t}est - mmodeltestfile > output.txt$

$\mathrm{crf}_{t}est{-}mmodeltest.txt > output.txtReferencesConditionalRandomFields$: $InstallingCRF{+}{+}onUbuntuConditionalRandomFieldsTrainingandTestingusingCRF{+}{+}$

6.7 Entity Linking

In natural language processing, entity linking, named entity linking (NEL), named entity disambiguation (NED), named entity recognition and disambiguation (NERD) or named entity normalization (NEN) is the task of determining the identity of entities mentioned in text. More precise, it is the task of linking entity mentions to entries in a knowledge base (e.g., DBpedia, Wikipedia)

Entity linking requires a knowledge base containing the entities to which entity mentions can be linked. A popular choice for entity linking on open domain text are knowledge-bases based on Wikipedia, in which each page is regarded as a named entity. NED using Wikipedia entities has been also called wikification (see Wikify! an early entity linking system]). A knowledge base may also be induced automatically from training text or manually built.

NED is different from named entity recognition (NER) in that NER identifies the occurrence or mention of a named entity in text but it does not identify which specific entity it is

Examples Example 1:

For example, given the sentence “Paris is the capital of France”, the idea is to determine that “Paris” refers to the city of Paris and not to Paris Hilton or any other entity that could be referred as “Paris”.

Example 2:

Give the sentence “In Second Debate, Donald Trump and HIllary Clinton Spar in Bitter, Personal Terms”, the idea is to determine that “Donald Trump” refer to an American politician, and “Hillary Clinton” refer to 67th United States Secretary of State from 2009 to 2013.

Architecture

Mention detection: Identification of text snippets that can potentially be linked to entities Candidate selection: Generating a set of candidate entities for each mention Disambiguation: Selecting a single entity (or none) for each mention, based on the context Mention detection

Goal: Detect all "linkable" phrases

Challenges:

Recall oriented: Do not miss any entity that should be link Find entity name variants (e.g. "jlo" is name variant of [Jennifer Lopez]) Filter out inappropriate ones (e.g. "new york" matches >2k different entities) COMMON APPROACH Build a dictionary of entity surface forms entities with all names variants Check all document n-grams against the dictionary the value of n is set typically between 6 and 8 Filter out undesired entities Can be done here or later in the pipeline Examples

Candidate Selection

Goal: Narrow down the space of disambiguation possibilities

Balances between precision and recall (effectiveness vs. efficiency)

Often approached as ranking problem: keeping only candidates above a score/rank threshold for downstream processing.

COMMONNESS Perform the ranking of candidate entities based on their overall popularity, i.e., "most command sense"

Examples

Commonness can be pre-computed and stored in the entity surface form dictionary. Follows a power law with a long tail of extremely unlikely senses; entities at the tail end of distribution can be safely discarded (e.g., 0.001 is sensible threshold)

Disambiguation

Baseline approach: most common sense

Consider additional types of evidence: prior importance of entities and mentions, contextual similarity between the text surrounding the mention and the candidate entity, coherence among all entity linking decisions in the document.

Combine these signals: using supervised learning or graph-based approaches

Optionally perform pruning: reject low confidence or semantically meaning less annotations.

References "Entity Linking". wikipedia "Entity Linking". Krisztian Balog, University of Stavanger, 10th Russian Summer School in Information Retrieval. 2016 "An End-to-End Entity Linking Approach for Tweets". Ikuya Yamada, Hideaki Takeda, Yoshiyasu Takefuji. 2015

Chương 7

Nhận dạng tiếng nói

19/01/2018: Hôm nay thực sự quá mệt với bạn Kaldi. Mãi không thể decode với các thuộc tính LDA-MLLT được? Hỏi mãi ở kaldi-help mà không có reply

05/01/2018 - "điên đầu" với Sphinx và HTK. HTK thì đã bỏ rồi vì quá lằng nhằng. Sphinx thì setup được đối với dữ liệu nhỏ rồi. Nhưng không thể làm nó hoạt động với dữ liệu của VIVOS. Chắc hôm nay sẽ switch sang Kaldi vậy.

26/12/2017 - Automatic Speech Recognition 100. Sau mấy ngày "vật lộn" với code base của Truong Do, thì cuối cùng cũng produce voice được. Cảm giác rất thú vị. Quyết định làm luôn ASR. Tìm mãi chẳng thấy code base đâu (chắc do lĩnh vực mới nên không có kinh nghiệm). May quá lại có bạn frankydotid có project về nhận diện tiếng Indonesia ở github. Trong README.md bạn dẫy bảo là phải cần đọc HTK Book. Tốt quá đang cần cơ bản.

Trong hệ thống nhận dạng tiếng nói, tín hiệu âm thanh được thu thập như những mẫu phù hợp cho quá trình xử lý của máy tính và được đưa vào quá trình nhận diện. Đầu ra của hệ thống là một câu phụ đề của câu nói.

Nhận dạng tiếng nói là một nhiệm vụ phức tạp và hệ thống tốt nhất trong nhận dạng tiếng nói rất phức tạp. Có rất nhiều cách tiếp cận cho mỗi thành phần. Trong phần này, người viết chỉ muốn đưa ra một cái nhìn tổng thể về nhận dạng tiếng nói, các khó khăn chính, các thành phần cơ bản, chức năng và tương tác của chúng trong một hệ thống nhận dạng tiếng nói.

7.1 Các thành phần của hệ thống nhận dạng tiếng nói

Trong bước thứ nhất, trích rút thông tin *Feature Extraction*, các mẫu tín hiệu được tham số hóa. Mục tiêu là trích xuất ra một tập các tham số (đặc trưng) từ tín hiệu có nhiều thông tin hữu ích nhất cho quá trình phân loại. Các đặc trưng chính được trích xuất với điều kiện *thích nghi* với các sự thay đổi của âm thanh và *nhạy cảm* với các nội dung ngôn ngữ.

mô hình âm học

Trong module phân loại, các vector đặc trưng được ánh xạ với các pattern, được gọi là **mô hình âm học** (acoustic model). Mô hình học thường là HMM được train với toàn bộ từ, hay âm như là một đơn vị ngôn ngữ.

từ điển phát âm

Từ điển phát âm (pronunciation dictionary) định nghĩa cách kết hợp âm cho các ký tự. Nó có thể chứa cách phát âm khác nhau cho cùng một từ. Bảng 1 hiển thị chính xác một từ điển. Từ (graphme) ở cột bên trái ứng với cách phát âm (các âm) ở cột bên phải (các ký tự âm trong bảng được dùng phổ biến đối với tiếng Anh)

Ví dụ một phần của từ điển âm học tiếng Anh trong thực tế

word	pronunciation
INCREASE	ih n k r iy s
INCREASED	ih n k r iy s t
INCREASES	ih n k r iy s ah z
INCREASING	ih n k r iy s ih ng
INCREASINGLY	ih n k r iy s ih ng l iy
INCREDIBLE	ih n k r eh d ah b ah l

mô hình ngôn ngữ

Mô hình ngôn ngữ (language model) chứa các thông tin về cú pháp. Mục tiêu để dự đoán khả năng một từ xuất hiện sau các từ khác trong một ngôn ngữ. Nói cách khác, xác xuất để một từ k xảy ra sau khi $k-1$ từ trước đó được định nghĩa bởi $P(w_k|w_{k-1}, w_{k-2}, ..., w_1)$

Mô hình hóa sub-word với HMMs

Trong các hệ thống ASR, HMMs được dùng để biểu diễn các đơn vị dưới từ (ví dụ như âm). Với ngôn ngữ, thông thường có 40 âm. Số lượng âm phụ thuộc vào từ điển được sử dụng. Số lượng âm phụ thuộc vào từ điển được sử dụng. Mô hình từ có thể được xây dựng bằng cách kết hợp các mô hình dưới từ.

Trong thực tế, khi nhận dạng một âm phụ thuộc rất nhiều vào các âm bên cạnh. Do đó, mô hình âm phụ thuộc ngữ cảnh (*context dependence*) được sử dụng rất phổ biến. Mô hình *biphone* chú ý đến âm bên trái hoặc âm bên phải, mô hình *triphone* chú ý đến cả hai phía, với một âm, các mô hình khác nhau được sử dụng trong ngữ cảnh khác nhau. Hình dưới thể hiện các mô hình monophone, biphone và triphone của từ *bat* (b ae t)

7.2 Quá trình huấn luyện

Huấn luyện các mô hình monophone

Một mô hình monophone là một mô hình âm học, trong đó không chứa thông tin ngữ cảnh về các âm trước và sau. Nó được sử dụng như thành phần cơ bản cho các mô hình triphone - mô hình sử dụng những thông tin về ngữ cảnh.

Việc huấn luyện sử dụng framework Gaussian Mixture Model/Hidden Markov Model.

Dóng hàng âm thanh trong mô hình âm học

Các tham số trong mô hình âm học được tính toán trong quá trình huấn luyện; tuy nhiên, quá trình này có thể được tối ưu hóa bởi việc lặp lại quá trình huấn luyện và dòng hàng. Còn lại là huấn luyện Viterbi (liên quan đến phương pháp này, nhưng dùng nhiều khối lượng tính toán hơn là thuật toán Forward-Backward và Expectation Maximization). Bằng cách dóng hàng âm thanh - phụ đề với mô hình âm học hiện tại, các thuật toán huấn luyện có thể sử dụng kết quả này để cải thiện và hiệu chỉnh tham số của mô hình. Do đó, mỗi quá trình huấn luyện sẽ theo bởi một bước dóng hàng trong đó âm thanh và văn bản được dóng hàng lại.

Huấn luyện các mô hình triphone

Trong khi các mô hình monophone đơn giản biểu diễn các đặc trưng âm thanh như một đơn âm, trong khi các âm vị sẽ thay đổi đáng kể phụ thuộc vào

ngữ cảnh. Mô hình triphone thể hiện một âm trong ngữ cảnh với hai âm bên cạnh.

Đến đây, một vấn đề là không phải tất cả các đơn vị triphone được thể hiện trong dữ liệu huấn luyên. Có tất cả $(\#ofphonemes)^3$ triphone, nhưng chỉ có một tập thực sự tồn tại trong dữ liệu. Hơn nữa, các đơn vị xảy ra nhiều lần trong dữ liệu đưa ra kết quả thống kê tốt hơn trong dữ liệu. Một nhóm cây quyết định phân chia các triphones vào các nhóm, mục đích giảm thiểu tham số và đưa ra quyết định tốt hơn.

Dóng hàng các mô hình âm học và huấn luyện lại các mô hình triphone

Lặp lại các bước dòng hàng âm thanh và huấn luyện các mô hình triphone với các thuật toán huấn luyện để hiệu chỉnh mô hình. Các phương pháp phổ biến là delta+delta-delta, LDA-MLLT và SAT. Các giải thuật dóng hàng bao gồm dóng hàng cho từng người nói và FMLLR.

Các thuật toán huấn luyện

Huấn luyện delta+delta-delta tính các đặc trưng delta và double-delta, hay các hệ số động, để thêm vào các đặc trưng MFCC. Delta và delta-delta là các đặc trưng số học, tính các đạo hàm bậc 1 và 2 của tín hiệu. Do đó, phép tính toán này thường được thực hiện trên một window của các đặc trưng vector. Trong khi một window của hai đặc trưng vector có thể hiệu quả, nó là các xấp xỉ thô (giống như delta-diffrence là một xấp xỉ thô của đạo hàm). Đặc trưng delta được tính toán trong các window của các đặc trưng cơ bản, trong khi delta-delta được tính toán trong các window của đặc trựng delta.

LDA-MLLT viết tắt của Linear Discriminant Analysis - Maximum Likelihood Linear Transform. Linear Discriminant Analysis lấy các đặc trưng vector và xây dựng các trạng thái HMM, nhưng giảm thiểu không gian vector. Maximum Likelihood Linear Transfrom lấy các đặc trưng được giảm từ LDA, và thực hiện các biến đổi đối với từng người nói. MLLT sau đó thực hiện một bước chuẩn hóa, để giảm sự khác biệt giữa các người nói.

SAT viết tắt của Speaker Adaptive Training. SAT cũng thực hiện các chuẩn hóa đối với người nói bằng cách thực hiện biến đổi trên mỗi người nói. Kết quả của quá trình này đồng nhất và chuẩn hóa hơn, cho phép mô hình có thể sử dụng những tham số này để giảm thiểu sự biến đổi của âm, đối với từng người nói hoặc môi trường thu.

Các thuật toán dóng hàng

Thuật toán dòng hàng luôn luôn cố định, trong đó các kịch bản chấp nhận các loại đầu vào âm học khác nhau. Dòng hàng đối với từng người nói, sẽ tách biệt thông tin giữa các người nói trong quá trình dóng hàng.

fMLLR viết tắt của Feature Space Maximum Likelihood Linear Regression. Sau quá trình huấn luyện SAT, các mô hình âm học không huấn luyện trên các đặc trưng ban đầu, mà đối với các đặc trưng chuẩn hóa theo người nói. Với quá trình dóng hàng, xóa bỏ sự khác biệt giữa người nói (bằng cách nghịch đạo ma trận fMLLR), sau đó loại bỏ nó khỏi mô hình *bằng cách nhân ma trận nghịch đảo với đặc trưng vector). Mô hình âm học quasi-speaker-independent có thể sử dụng trong quá trình dóng hàng.

Dóng hàng (Forced Alignment)

Hệ thống nhận dạng tiếng nói sử dụng một máy tìm kiếm bên cạnh mô hình âm học và ngôn ngữ trong đó chứa tập các từ, âm và tập dữ liệu để đối chiếu với dữ liệu âm thanh cho câu nói. Máy tìm kiếm này sử dụng các đặc trưng

được trích xuất bởi dữ liệu âm thanh để xác định sự xuất hiện của từ, âm và đưa ra kết quả.

![](https://www.isip.piconepress.com/projects/speech/software/tutorials/production/fundamentals/v1.0/

Quá trình dòng hàng cũng tương tự như vậy, nhưng khác ở một điểm quan trong. Thay vì đưa vào tập các từ có thể để tìm kiếm, máy tìm kiếm đưa vào đoạn phụ đề tương ứng với câu nói. Hệ thống sau đó dóng hàng dữ liệu văn bản với dữ liệu âm thanh, xác định đoạn nào trong âm thanh tương ứng với từ cụ thể nào trong dữ liệu văn bản.

![](https://www.isip.piconepress.com/projects/speech/software/tutorials/production/fundamentals/v1.0/

Dóng hàng có thể sử dụng để dóng âm trong dữ liệu với bản với dữ liệu âm thanh, giống như hình dưới đây, các âm được xác định trong từng đoạn của âm thanh.

![](https://www.isip.piconepress.com/projects/speech/software/tutorials/production/fundamentals/v1.0/

7.3 Hidden Markov Model

Hidden Markov Model (HMM) là mô hình trọng số với các trọng số ở cung, chỉ khả năng xuất hiện của cung.

Một trong những ứng dụng của HMM, là phán đoán chuỗi các trạng thái thay đổi, dựa vào chuỗi các quan sát

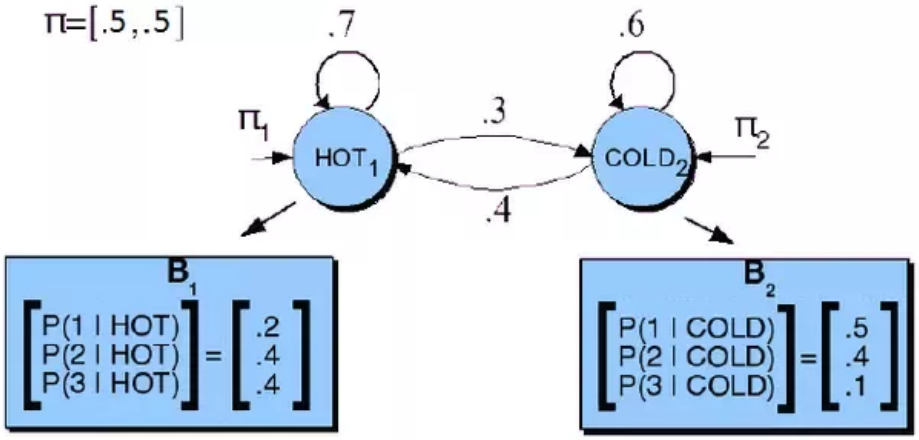
Các trọng số trong trạng thái gọi là observation likelihood, các trọng số ở cung gọi là transition likelihood.

Sau đây là một ví dụ:

dữ liệu ăn đá

Trong Jurafsky u.a. (2009), tác giả giới thiệu dữ liệu ăn đá mô tả số lượng đá tiêu thụ trong các trạng thái thời tiết khác nhau. Đây là mô hình rất hay mô phỏng HMM

- Thời tiết trong một ngày có thể là NÓNG hoặc LẠNH
- Khi trời NÓNG, 20% bạn dùng 1 viên đá, 40% bạn dùng 2 viên, 40% cho 3 viên.
- Khi trời LẠNH, 50% bạn dùng 1 viên, 40% bạn dùng 2 viên, 10% bạn dùng 3 viên. Đây là các khả năng trong quá trình quan sát (observation likelihood)
- Khi trời NÓNG, 30% nó sẽ chuyển sang LẠNH, 70% giữ nguyên. Khi trời LẠNH, 40% nó sẽ chuyển sang NÓNG, 60% giữ nguyên. Đây là khả năng dịch chuyển (trainsition likelihood)



Giờ, giả sử chung ta quan sát trong 3 ngày, bạn dùng $1, 2, 3$ viên đá. Thời tiết có khả năng diễn ra như thế nào?

Đến đây chúng ta dùng thuật toán Viterbi. Về cơ bản, nó là dynamic programming với hai chiều $[state, position\_in\_sequence]$

Gọi S là trạng thái hiện tại $HOT, COLD$ trong quan sát i, S' là trạng thái trước đó, và A là lượng đá tiêu thụ $1, 2, 3$ trong quan sát i

$$Viterbi[S, i] = Viterbi[S', i-1] * p(S|S') * p(A|S)$$

$$V[S, i] = V[S', i-1] * transition\_likelihood * observation\_likelihood$$

HMM được sử dụng trong các hệ thống thỏa mãn

1. Có hữu hạn các trạng thái nội tại (internal state), là nguyên nhân của các sự kiện (external events) (các quan sát)
2. Trạng thái nội tại không quan sát được (hidden)
3. Trạng thái hiện tại chỉ phụ thuộc vào trạng thái trước đó (qúa trình Markov)

Wow! George nhanh chóng liên hệ vụ của anh đấy với mô hình HMM. George nhận ra rằng CCTV footage từ các cập có thể coi như là chuỗi quan sát được, anh đấy có thể dùng mô hình và sử dụng nó để phát hiện hành vị ẩn mà Bob và William hoạt động.

3 vấn đề cơ bản được Jack Ferguson giới thiệu trong những năm 1960

1. (Likelihood): Cho một HMM $\lambda = (A, B)$ và một chuỗi quan sát O, xác định likelihood $P(O|\lambda)$
2. (Decoding): Cho một chuỗi quan sát O, và một HMM $\lambda = (A, B)$, xác định chuỗi ẩn Q tốt nhất
3. (Learning): Cho một chuỗi quan sát O, một tập các trạng thái trong HMM, học các tham số A và B

7.4 Likelihood Computation

Vấn đề đầu tiên là tính xác suất xảy ra của một chuỗi quan sát. Ví dụ, trong bài toán ăn đá ở hình 9.3, xác suất xảy ra chuỗi *3 1 3* là bao nhiêu?

Tính toán Likelihood: Chuỗi một HMM $\lambda = (A, B)$, và mỗi chuỗi quan sát O, xác định likelihood $P(O|\lambda)$

Thuật toán Forward, nếu sử dụng Bayes rule, để tính likelihood, cần khối lượng tính toán N^T với N là số trạng thái có thể có và T là chiều dài chuỗi quan sát. Ví dụ trong bài toán gán nhãn có N=10 nhãn, chiều dài của chuỗi trung bình là 28, thì cần 10^{28} bước tính toán. Một giải thuật với hiệu quả $O(N^2T)$ được đề xuất với tên gọi **forward algorithm**

Tài liệu tham khảo

* http://www.igi.tugraz.at/lehre/CI/SS08/tutorials/ASR/node1.html * https://www.isip.piconepress.com
$http://www.igi.tugraz.at/lehre/CI/SS08/tutorials/ASR/node1.html*https://www.isip.piconepress.com/projects/speech/software/tutorials/production/fundamentals/v1.0/section$
$https://www.quora.com/What-is-a-simple-explanation-of-the-Hidden-Markov-Model-algorithm$

Chương 8

Tổng hợp tiếng nói

Tóm lại thì việc sinh ra tiếng nói từ text gồm 4 giai đoạn

1. Sinh ra features từ file wav sử dụng tool sptk
2. Tạo một lab, trong đó có dữ liệu huấn luyện (những đặc trưng của âm thanh được trích xuất từ bước 1), text đầu vào
3. Sử dụng htk để train dữ liệu từ thư mục lab, đầu ra là một model
4. Sử dụng model để sinh ra output với text đầu vào, dùng hts_engine để decode, kết quả được wav files.

Phù. 4 bước đơn giản thế này thôi mà không biết. Lục cả internet ra mãi chẳng hiểu, cuối cùng file phân tích file 'train.sh' của bạn Truong Do mới hiểu. Ahihi

20/12/2017: Text to speech 100. Cảm ơn project rất hay của bạn Truong Do ở vais, nếu không có project này chắc mình phải mất rất nhiều thời gian mới có được phiên bản text to speech đầu tiên.

Chương 9

Phân loại văn bản

<h3>Naive Bayes Classifier</h3> Tham khảo thư viện <a href="http://scikit-learn.org/stable/modules/naive$_b$$ayes.html"$target = "$_b$$lank"$rel = "$noopener" > $Scikit - learn < /a >$

Xét bài toán classification với C classes 1,2,. . . ,C. Tính xác suất để 1 điểm dữ liệu rơi vào class C ta có công thức: $latexP(\frac{c}{x})$. Tức tính xác suất để đầu ra là class C biết rằng đầu vào là vector x. Việc xác định class của điểm dữ liệu đó bằng cách chọn ra class có xác suất cao nhất: c = argmax($latexP(\frac{c}{x})$) với c 1,. . . ,C Sử dụng quy tắc Bayes: c = argmax($latexP(\frac{c}{x})$) = argmax($latexP(\frac{P(\frac{c}{x})P(x)}{P(x)})$ = argmax($latexP(\frac{P(\frac{c}{x})}{P(c)})$)

<h4>Các phân phối thường dùng</h4> Gaussian Naive Bayes Mô hình này được sử dụng chủ yếu trong loại dữ liệu mà các thành phần là các biến liên tục. Multinomial Naive Bayes Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà feature vectors được tính bằng Bags of Words. Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản đó. Khi đó, $latexP(\frac{x_i}{c})$ tỉ lệ với tần suất từ thứ i xuất hiện trong các văn bản của class c: $latexP(\frac{x_i}{c})$ = $latex\frac{Nx_i}{Nc}$ Trong đó: $latexNx_i$ là tổng số lần từ thứ i xuất hiện trong các văn bản của class c, nó được tính là tổng của tất cả các thành phần thứ i của các feature vectors ứng với class c. $latexNc$ là tổng số từ (kể cả lặp) xuất hiện trong class c. Hay bằng tổng độ dài của toàn bộ các văn bản thuộc vào class c. Nếu có một từ mới chưa bao giờ xuất hiện trong class c thì biểu thức trên sẽ bằng 0, điều này dẫn đến vế phải của c bằng 0. Bernoulli Naive Bayes Mô hình này được áp dụng cho các loại dữ liệu mà mỗi thành phần là một giá trị binary. Ví dụ: cũng với loại văn bản nhưng thay vì đếm tổng số lần xuất hiện của 1 từ trong văn bản, ta chỉ cần quan tâm từ đó có xuất hiện hay không. Khi đó: $latexP(\frac{x_i}{c})$ = $latexP(\frac{i}{c})x_i$ + (1 $latexP(\frac{i}{c})$(1 $latexx_i$)) Với $latexP(\frac{i}{c})$ là xác suất từ thứ i xuất hiện trong các văn bản của class c.

Chương 10

Pytorch

Bí kíp luyện công

(cập nhật 08/12/2017): cảm giác [talk](http://videolectures.net/deeplearning2017$_{c}hintala_{t}orch/)caanhSc

Sau khi nghe bài này thì hâm mộ luôn anh Soumith Chintala, tìm loạt bài anh trình bày luôn

* [PyTorch: Fast Differentiable Dynamic Graphs in Python with a Tensor JIT](https://www.youtube.com/watch?v=DBVLcgq2Eg0amp;t=2s), Strange Loop Sep 2017 * [Keynote: PyTorch: Framework for fast, dynamic deep learning and scientific computing](https://www.youtube.com/watch?v=LAMwEJZqesUamp;t=66s), EuroSciPy Aug 2017

So sánh giữa Tensorflow và Pytorch?

Có 2 điều cần phải nói khi mọi người luôn luôn so sánh giữa Tensorflow và Pytorch. (1) Tensorflow khiến mọi người "không thoải mái" (2) Pytorch thực sự là một đối thủ trên bàn cân. Một trong những câu trả lời hay nhất mình tìm được là của anh Hieu Pham (Google Brain) [trả lời trên quora (25/11/2017)](https://www.quora.com/What-are-your-reviews-between-PyTorch-and-TensorFlow/answer/Hieu-Pham-20?srid=5O2u). Điều quan trọng nhất trong câu trả lời này là *"Dùng Pytorch rất sướng cho nghiên cứu, nhưng scale lên mức business thì Tensorflow là lựa chọn tốt hơn"*

Behind The Scene

(15/11/2017) Hôm nay bắt đầu thử nghiệm pytorch với project thần thánh classification sử dụng cnn https://github.com/Shawn1993/cnn-text-classification-pytorch

Cảm giác đầu tiên là make it run khá đơn giản

“‘ conda create -n test-torch python=3.5 pip install http://download.pytorch.org/whl/cu80/torch-0.2.0.post3-cp35-cp35m-manylinux1$_{x}86_{6}4.whlpipinstalltorchvisionpipinstalltorchtext“‘

Thế là ‘main.py‘ chạy! Hay thật. Còn phải vọc để bạn này chạy với CUDA nữa.

Cài đặt CUDA trong ubuntu 16.04

Kiểm tra VGA

“‘ $lspci|grepVGA01:00.0VGAcompatiblecontroller:NVIDIACorporationGM204[GeForceGTX980]($

Kiểm tra CUDA đã cài đặt trong Ubuntu [1]

“‘ $nvcc--versionnvcc:NVIDIA(R)CudacompilerdriverCopyright(c)2005-2016NVIDIACorporationBuiltonSun_Sep_42$2:14:01$_{C}DT_{2}016Cudacompilationtools,release8.0,V8.0.44“‘

Kiểm tra pytorch chạy với cuda ‘test$_{c}uda.py$‘

“‘python import torch print("Cuda:", torch.cuda.is$_{a}vailable()$)“‘

“‘ $pythontest_cuda.pyCUDA : True$“‘

Chỉ cần cài đặt thành công CUDA là pytorch tự work luôn. Ngon thật!

Ngày X

Chẳng hiểu sao update system kiểu nào mà hôm nay lại không sử dụng được CUDA ‘torch.cuda.is$_a vailable() = False$‘.$Saukhidnglnh$‘$torch.Tensor().cuda()$‘$thgpli$

“‘ AssertionError: The NVIDIA driver on your system is too old (found version 8000). Please update your GPU driver by downloading and installing a new version from the URL: http://www.nvidia.com/Download/index.aspx Alternatively, go to: https://pytorch.org/binaries to install a PyTorch version that has been compiled with your version of the CUDA driver. “‘

Kiểm tra lại thì mình đang dùng nvidia-361, làm thử theo [link này](http://www.linuxandubuntu.com/hom to-install-latest-nvidia-drivers-in-linux) để update NVIDIA, chưa biết kết quả ra sao?

May quá, sau khi update lên nvida-387 là ok. Haha

Ngày 2

Hôm qua đã bắt đầu implement một nn với pytorch rồi. Hướng dẫn ở [Deep Learning with PyTorch: A 60 Minute Blitz](http://pytorch.org/tutorials/beginner/deep$_l earning_6 0min_b litz.h$

Hướng dẫn implement các mạng neural với pytorch rất hay tại [PyTorch-Tutorial](https://github.com/MorvanZhou/PyTorch-Tutorial)

(lượm lặt) Trang này [Awesome-pytorch-list](https://github.com/bharathgs/Awesome-pytorch-list) chứa rất nhiều link hay về pytorch như tập hợp các thư viện liên quan, các hướng dẫn và ví dụ sau đó là các cài đặt của các paper sử dụng pytorch.

(lượm lặt) Loạt video hướng dẫn pytorch [PyTorchZeroToAll](https://www.youtube.com/watch?v=SKq-pmkekTkamp;list=PLlMkM4tgfjnJ3I-dbhO9JTw7gNty6o$_2 m)catgiSungKimtrnyoutube.$

Bước tiếp theo là visualize loss và graph trong tensorboard, sử dụng [tensorboard$_l ogger$](*https* : *//github.com/TeamHG − Memex/tensorboard$_l$ogger*)*khhay.*

“‘ pip install tensorboard$_l oggerpipinstalltensorboard$“‘

Chạy tensorboard server

“‘ tensorboard –log-dir=runs “‘

Ngày 3: Vấn đề kỹ thuật

Hôm qua cố gắng implement một phần thuật toán CNN cho bài toán phân lớp văn bản. Vấn đề đầu tiên là biểu diễn sentence thế nào. Cảm giác load word vector vào khá chậm. Mà thằng tách từ của underthesea cũng chậm kinh khủng.

Một vài link tham khảo về bài toán CNN: [Implementing a CNN for Text Classification in TensorFlow](http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/), [Text classification using CNN : Example](https://agarnitin86.github.io/blog/2016/12/23/text-classification-cnn)

[1] : *https* : *//askubuntu.com/questions/*799184/*how − can − i − install − cuda − on − ubuntu* − 16 − 04

Chương 11

Big Data

View online http://magizbox.com/training/bigdata/site/

Big Data QA 1. What is "Big Data"? 1 https://www.youtube.com/watch?v=TzxmjbL-i4Y

2. How big is big data? 2

3. How much data is "Big Data"? 3

4. What are characteristics of "Big Data"? 4

5. What is big data ecosystem? 5

6. What is big data landscape 6

7. What are benefits of big data? 7

https://www.youtube.com/watch?v=TzxmjbL-i4Y

http://scoop.intel.com/what-happens-in-an-internet-minute/

http://www.quora.com/How-much-data-is-Big-Data

https://en.wikipedia.org/wiki/Big$_d ataCharacteristics$

http://www.clearpeaks.com/blog/big-data/big-data-ecosystem-spark-and-tableau

https://vladimerbotsvadze.wordpress.com/2015/01/28/the-big-data-landscape-technology-businessintelligence-analytics/

http://blog.galaxyweblinks.com/big-data-with-bigger-benefits/

11.1 Distribution Storage

11.1.1 HDFS

The Hadoop Distributed File System (HDFS) — a subproject of the Apache Hadoop project—is a distributed, highly fault-tolerant file system designed to run on low-cost commodity hardware. HDFS provides high-throughput access to application data and is suitable for applications with large data sets. This article explores the primary features of HDFS and provides a high-level view of the HDFS architecture. : sequenceiq/hadoop-docker

Big Data Stack: HDFS, Kibana, ElasticSearch, Neo4J, Apache Spark

11.1.2 HBase

Apache HBase^{TM} is the Hadoop database, a distributed, scalable, big data store. Download Apache HBase^{TM} Click here to download Apache HBase^{TM}.

1. When Would I Use Apache HBase? 1 HBase isn't suitable for every problem.

First, make sure you have enough data. If you have hundreds of millions or billions of rows, then HBase is a good candidate. If you only have a few thousand/million rows, then using a traditional RDBMS might be a better choice due to the fact that all of your data might wind up on a single node (or two) and the rest of the cluster may be sitting idle.

Second, make sure you can live without all the extra features that an RDBMS provides (e.g., typed columns, secondary indexes, transactions, advanced query languages, etc.) An application built against an RDBMS cannot be "ported" to HBase by simply changing a JDBC driver, for example. Consider moving from an RDBMS to HBase as a complete redesign as opposed to a port.

Third, make sure you have enough hardware. Even HDFS doesn't do well with anything less than 5 DataNodes (due to things such as HDFS block replication which has a default of 3), plus a NameNode.

HBase can run quite well stand-alone on a laptop - but this should be considered a development configuration only.

2. Features 2 Linear and modular scalability. Strictly consistent reads and writes. Automatic and configurable sharding of tables Automatic failover support between RegionServers. Convenient base classes for backing Hadoop MapReduce jobs with Apache HBase tables. Easy to use Java API for client access. Block cache and Bloom Filters for real-time queries. Query predicate push down via server side Filters Thrift gateway and a REST-ful Web service that supports XML, Protobuf, and binary data encoding options Extensible jruby-based (JIRB) shell Support for exporting metrics via the Hadoop metrics subsystem to files or Ganglia; or via JMX 3. Architecture

HBase Shell [code lang="shell"]

list all table list [/code]

Up Running 1. Download HBase 0.94.27 (HBase 0.98 won't work)

[code lang="shell"] wget https://www.apache.org/dist/hbase/hbase-0.94.27/hbase-0.94.27.tar.gz tar -xzf hbase-0.94.27.tar.gz [/code]

2. Setup 1. edit $HBASE_ROOT/conf/hbase-site.xmlandadd$

[code lang="xml"] hbase.rootdir file:///full/path/to/where/the/data/should/be/stored hbase.cluster.distributed false [/code]

3. Verify Go to http://localhost:60010 to see if HBase is running.

When Should I Use HBase? HBase Config HBase Remote 1. Change /etc/hosts [code] 127.0.0.1 [username] [server$_ip$]*hbase.io*[/*code*]

Example

[code] 127.0.0.1 crawler 192.168.0.151 hbase.io [/code]

2. Change hostname [code] hostname hbase.io [/code]

3. Change region servers Edit $HBASE_ROOT/conf/regionservers$

[code] hbase.io [/code]

4. Change $HABSE_ROOT/conf/hbase-site.xml[codelang = "xml"title =$ "$hbase-site.xml$"] <?$xml-stylesheettype = "text/xsl"href = "configuration.xsl"? >$ $hbase.rootdirfile : ///home/username/Downloads/hbase/datahbase.cluster.distributedfalsehbase.zookeep$ $unsecurehbase.rpc.timeout2592000000$[/*code*]

Docker HBase 0.94

Image: https://github.com/Banno/docker-hbase-standalone

[code] docker run -d -p 2181:2181 -p 60000:60000 -p 60010:60010 -p 60020:60020 -p 60030:60030 banno/hbase-standalone [/code]

Compose

[code] hbase.vmware: build: ./docker-hbase-standalone/. command: "/opt/hbase/hbase-0.94.15-cdh4.7.0/bin/hbase master start" hostname: hbase.vmware ports: - 2181:2181 - 60000:60000 - 60010:60010 - 60020:60020 - 60030:60030 volumes: - ./docker-hbase-standalone/hbase-0.94.15-cdh4.7.0:/opt/hbase/hbase-0.94.15-cdh4.7.0 - ./data/hbase:/tmp/hbase-root/hbase /code]

11.2 Distribution Computing

11.2.1 Apache Spark

Apache Spark is an open-source cluster computing framework originally developed in the AMPLab at UC Berkeley. In contrast to Hadoop's two-stage disk-based MapReduce paradigm, Spark's in-memory primitives provide performance up to 100 times faster for certain applications. By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well-suited to machine learning algorithms. Installation Requirements: Hadoop, YARN

Install Hadoop

Insatll YARN

Install Java

Verification Tutorial From Pandas to Apache Spark's DataFrame

Big Data Stack: HDFS, Kibana, ElasticSearch, Neo4J, Apache Spark

Apache Spark: Tutorials Beginners Guide: Apache Spark Machine Learning with Large Data

Spark and Spark Streaming Unit Testing Recipes for Running Spark Streaming Applications in Production- Databricks

Spark Streaming

Spark and Spark Streaming Unit Testing Recipes for Running Spark Streaming Applications in Production- Databricks

11.3 Components

11.3.1 Ambari

The Apache Ambari project is aimed at making Hadoop management simpler by developing software for provisioning, managing, and monitoring Apache Hadoop clusters. Ambari provides an intuitive, easy-to-use Hadoop management web UI backed by its RESTful APIs.

Ambari enables System Administrators to:

Provision a Hadoop Cluster

Ambari provides a step-by-step wizard for installing Hadoop services across any number of hosts. Ambari handles configuration of Hadoop services for the cluster. Manage a Hadoop Cluster

Ambari provides central management for starting, stopping, and reconfiguring Hadoop services across the entire cluster. Monitor a Hadoop Cluster

Ambari provides a dashboard for monitoring health and status of the Hadoop cluster. Ambari leverages Ambari Metrics System for metrics collection. Ambari leverages Ambari Alert Framework for system alerting and will notify you when

your attention is needed (e.g., a node goes down, remaining disk space is low, etc). Ambari enables Application Developers and System Integrators to:

Easily integrate Hadoop provisioning, management, and monitoring capabilities to their own applications with the Ambari REST APIs. Docker

Receipts:

Image: sequenceiq/ambari (git) Multinode cluster with Ambari 1.7.0 1 Get the docker images

[code] docker pull sequenceiq/ambari:1.7.0 [/code]

Get ambari-functions [code] curl -Lo .amb j.mp/docker-ambari-170 . .amb [/code]

Create your cluster – automated

[code] amb-deploy-cluster 3 [/code]

Multinode cluster with Ambari 1.7.0

11.3.2 Kibana

Kibana is an open source data visualization plugin for Elasticsearch. It provides visualization capabilities on top of the content indexed on an Elasticsearch cluster. Users can create bar, line and scatter plots, or pie charts and maps on top of large volumes of data.

11.3.3 Logstash

https://www.digitalocean.com/community/tutorials/how-to-use-logstash-and-kibana-to-centralize-logs-on-centos-6

11.3.4 Elasticsearch

Elasticsearch is a search server based on Lucene. It provides a distributed, multitenant-capable full-text search engine with a RESTful web interface and schema-free JSON documents. Elasticsearch is developed in Java and is released as open source under the terms of the Apache License. Elasticsearch is the second most popular enterprise search engine 1. Basic Concenpts Relational Database Elasticsearch Database Index Table Type Row Document Column Field Schema Mapping 2. Index Query Get all indices $/_stats Search API 1 Search All/bank/_search?q = *hits.hits ˘ actual array of search results (default sto first 10 documents)$

Query Language elasticsearch provides a full Query DSL based on JSON to define queries.

curl -XPOST /bank/$_search//match all, limit 10 offset 10"query" : "match_all" : ,"from" : 10,"size" : 10

// select fields "query": "match$_all$" : ,$_s$ $ource$: ["$account_number$","$balance$"]"$size$" : 10

// where account equals 20 "query": "match": "account$_number$" : 20$Filter$

curl -XPOST elastic:9200/index/type/$_search - d'"query" : "filtered" : "query" : "term" : "feature" : 1,'

curl -XPOST elastic:9200/index/type/$_search - d'"query" : "filtered" : "query" : "term" : "feature" : 1,'

curl -XPOST localhost:9200/test "mappings" : "$_default$" : "$_timestamp$" : "enabled" : $true$,"store" : $true$

Easy, fast, performant No need for special queries Only applicable when one-to-one relationships are maintained Nested

Nested docs are stored in the same Lucene block as each other, which helps read/query performance. Reading a nested doc is faster than the equivalent parent/child. Updating a single field in a nested document (parent or nested

children) forces ES to reindex the entire nested document. This can be very expensive for large nested docs "Cross referencing" nested documents is impossible Best suited for data that does not change frequently Parent/Child

Updating a child doc does not affect the parent or any other children, which can potentially save a lot of indexing on large docs Children are stored separately from the parent, but are routed to the same shard. So parent/children are slightly less performance on read/query than nested Parent/child mappings have a bit extra memory overhead, since ES maintains a "join" list in memory Sorting/scoring can be difficult with Parent/Child since the Has Child/Has Parent operations can be opaque at times Denormalization

You get to manage all the relations yourself! Most flexible, most administrative overhead May be more or less performant depending on your setup 4. Backup Elastic Dump 5 Tools for moving and saving indicies.

bin/elasticdump –input=http://localhost:9200/index$_1$ $--output = http : //localhost : 9200/index_{1b}ackup--type = data--scrollTime = 100Alias6curl-XPOST'http : //localhost : 9200/_aliases'-d'quot; actionsquot; : [quot; removequot; : quot; indexquot; : quot; EnginesRankingofSearchEngines$

The Search API http://stackoverflow.com/a/17146144/772391 http://stackoverflow.com/a/23407367/77 https://www.elastic.co/guide/en/elasticsearch/guide/current/modeling-your-data.html https://github.com/taskrabbit/elasticsearch-dump https://www.elastic.co/guide/en/elasticsearch/reference/ aliases.html https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-scripting.html Elasticsearch tutorial series 1: Metric Aggregations with Social Network Data Table of content

Avg, Max, Min, Sum Aggregation Cardinality Aggregation Stats Aggregation Extended Stats Aggregation Percentile Aggregation Percentile Ranks Aggregation Top hits Aggregation Avg, Max, Min, Sum, Count Aggregation Doc: Avg Aggregation, Doc: Max Aggregation, Doc: Min Aggregation

Get max, min, avg, sum, count about number of likes, shares, comments

Request

POST /facebook$_crawler/post/_search"aggs" : "sum_like" : "sum" : "field" : "num_like","min_like" : "mi$

"aggregations": "avg$_comment$" : "value" : 75.23860589812332,"min_like" : "value" : 0,"avg_like" : "value" : 1761974365266098.2,"sum_like" : "value" : 3238508883359088600,"max_sha "value" : 30407,"$max_comment$" : "value" : 11000,"sum_share" : "value" : 117844,"max_like" : "value" : 2751488761761411000,"avg_share" : "value" : 250.19957537154988,"$sum_comment$" : "value" : 28064,"$min_comment$" : "value" : 2,"min_share" : "value" : 1CardinalityAggregationCardinality

Get total of users

Request

POST /facebook$_crawler/post/_search"aggs" : "num_authors" : "cardinality" : "field" : "from.fb_id"Res

"aggregations": "num$_authors$" : "value" : 7385$StatsAggregationDoc : StatsAggregation$

Basic Stats of like, share comment

Request

POST /facebook$_crawler/post/_search"aggs" : "shares" : "stats" : "field" : "num_share","likes" : "stats$

"aggregations": "shares": "count": 471, "min": 1, "max": 30407, "avg": 250.19957537154988, "sum": 117844 , "comments": "count": 373, "min": 2, "max": 11000, "avg": 75.23860589812332, "sum": 28064 , "likes": "count": 1838, "min": 0, "max": 2751488761761411000, "avg": 1761974365266098.2, "sum": 3238508883359088600 Extended Stats Aggregation Extended Stats Aggregation

Stats of like, share comment with more metrics, such as sum, std$_deviation$, $std_deviation_bounds$, $variance$

Request

POST /facebook$_{c}rawler/post/_{s}earch"aggs" : "like_{s}tats" : "extended_{s}tats" : "field" : "num_{l}ike","shar
"aggregations": "like$_{s}tats$" : "$count$" : 1838,"min" : 0,"max" : 2751488761761411000,"avg" : 176197436
"$count$" : 471,"min" : 1,"max" : 30407,"avg" : 250.19957537154988,"sum" : 117844,"$sum_{o}f_{s}quares$" : 176
"$count$" : 373,"min" : 2,"max" : 11000,"avg" : 75.23860589812332,"sum" : 28064,"$sum_{o}f_{s}quares$" : 13153
$PercentilesAggregation$

Comment, Like, Share Percentiles

Request

POST /facebook$_{c}rawler/post/_{s}earch"aggs" : "like_{p}ercentiles" : "percentiles" : "field" : "num_{l}ike","s
"aggregations": "like$_{p}ercentiles$" : "$values$" : "1.0" : 0,"5.0" : 0,"25.0" : 4,"50.0" : 18.35,"75.0" : 72.5357
"$values$" : "1.0" : 2,"5.0" : 2,"25.0" : 5,"50.0" : 10,"75.0" : 26,"95.0" : 139.39999999999998,"99.0" : 1000,"s
"$values$" : "1.0" : 1,"5.0" : 1,"25.0" : 1,"50.0" : 4,"75.0" : 25,"95.0" : 251.5,"99.0" : 5560.3$LikePercentiles$

Request

POST /facebook$_{c}rawler/post/_{s}earch"aggs" : "share_{p}ercentiles" : "percentiles" : "field" : "num_{s}hare"
"aggregations": "share$_{p}ercentiles$" : "$values$" : "0.0" : 1,"10.0" : 1,"80.0" : 37.33333333333333,"90.0" : 9
$PercentileRanksAggregation$

How like, share, comment distribute

Request

POST /facebook$_{c}rawler/post/_{s}earch"aggs" : "like_{p}ercentile_{r}anks" : "percentile_{r}anks" : "field" : "nur
"aggregations": "share$_{p}ercentile_{r}anks$" : "$values$" : "10.0" : 60.438782731776364,"100.0" : 89.915074309
"$values$" : "10.0" : 39.281828073993466,"100.0" : 79.39530545624125,"1000.0" : 90.98349676683587,"10000.0
"$values$" : "10.0" : 49.865951742627345,"100.0" : 92.18395545473294,"1000.0" : 98.92761394101876,"10000.0

Top hits Aggregation Doc: Top hits Aggregation

Example

Request

Response

An Aggregation Doc: Link

Config elasticsearch.yml

discovery.zen.minimum$_{m}aster_{n}odes$: 1$discovery.zen.ping.multicast.enabled$:
$falsediscovery.zen.ping.unicast.hosts$: ["$localhost$"]

network.host: 0.0.0.0 http.cors.enabled: true http.cors.allow-origin: '*' script.inline:
on script.indexed: on Docker Image

https://hub.docker.com/r/$_{e}lasticsearch/$

Run

docker run -d -v "$PWD/esdata$" : $/usr/share/elasticsearch/dataelasticsearchDockerFolder$

elasticsearch/ config elasticsearch.yml Dockerfile Dockerfile

FROM elasticsearch:2.2.0

ADD config/elasticsearch.yml /elasticsearch/config/elasticsearch.yml Com-
pose

elasticsearch: build: ./elasticsearch/. ports: - 9200:9200 - 9300:9300 volumes:
- ./data/elasticsearch:/usr/share/elasticsearch/data Elasticsearch: Search Ig-
nore Accents The ICU 1 2 analysis plug-in for Elasticsearch uses the Inter-
national Components for Unicode (ICU) libraries to provide a rich set of tools for
dealing with Unicode. These include the icu$_{t}okenizer, which is particularly useful for Asian languages, and an$

Step 1: Install ICU-Plugin 3 cd /usr/share/elasticsearch sudo bin/plugin in-
stall analysis-icu Step 2: Create an analyzer setting: "settings": "analysis": "an-
alyzer": "vnanalysis": "tokenizer": "icu$_{t}okenizer$","$filter$" : ["$icu_{f}olding$","$icu_{n}ormalizer$"]$Step3$:
$Createyourindex, createafieldwithtypestringandanalyzerisvnanalysisyouhavecreated$"$key$" :
"$type$" : "$string$","$analyzer$" : "$vnanalysis$"$Step4 : SearchwithsensePOST/your_{i}ndex/your_{d}oc_{t}ype/_{s}earc
ImportCSVtoElasticsearchhttps : //gist.github.com/clemsos/8668698$

Install lastest Elasticdump with NVM As a matter of best practice we'll update our packages:

apt-get update The build-essential package should already be installed, however, we're going still going to include it in our command for installation:

apt-get install build-essential libssl-dev To install or update nvm, you can use the install script using cURL:

curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.31.0/install.sh | bash if you have below problem or after you type nvm ls-remote command it result N/A: curl: (77) error setting certificate verify locations: CAfile: /etc/pki/tls/certs/ca-bundle.crt CApath: none

head to this 1:

or Wget:

wget -qO- https://raw.githubusercontent.com/creationix/nvm/v0.31.0/install.sh | bash Don't forget to restart your terminal

Then you use the following command to list available versions of nodejs

nvm ls-remote To download, compile, and install the latest v5.0.x release of node, do this:

nvm install 5.0 And then in any new shell just use the installed version:

nvm use 5.0 Or you can just run it:

nvm run 5.0 –version Or, you can run any arbitrary command in a subshell with the desired version of node:

nvm exec 4.2 node –version You can also get the path to the executable to where it was installed:

nvm which 5.0 Node Version Manager

how to solve https problem

ICU plug-in Github

Installing the ICU plug-in

11.3.5 Neo4J

version: 2.3.1

Neo4j is an open-source graph database, implemented in Java. The developers describe Neo4j as "embedded, disk-based, fully transactional Java persistence engine that stores data structured in graphs rather than in tables". Neo4j is the most popular graph database.

Installation

Docker Docker Image: https://hub.docker.com/r/library/neo4j/

Run these below command to open neo4j

clone datahub project git clone https://github.com/magizbox/datahub.git

change folder to datahub directory cd datahub

set your config in docker-compose.yml

run docker docker-compose up

Cypher

Schema Discovery List all nodes label, list all relation type

> START n=node(*) RETURN distinct labels(n)

> match n-[r]-() return distinct type(r) UI Way: Click to Overtab in Neo4j Browser

Sample 10 entities > MATCH (n:Entity) RETURN n, rand() as random ORDER BY random LIMIT 10 Group By http://www.markhneedham.com/blog/2013/02/17/neo4jcypher-sql-style-group-by-functionality/

Graph Algorithms

shortestPath, dijkstra

POST http://localhost:7474/db/data/node/72/paths

Headers Accept: application/json Authorization: Basic bmVvNGo6cGFzc3dk

Body "to" : "http://localhost:7474/db/data/node/77", "max$_{d}epth$" : 5, "$relationships$" : "$type$" : "$FRIEND$", "$direction$" : "out", "$algorithm$" : "$shortestPath$" $GraphAnalystic$

pagerank, closeness$_{c}entrality$, $betweenness_{c}entrality$, $triangle_{c}ount$, $connected_{c}omponents$, $strongly_{c}onn$

Client

In this article you will know how to connect to neo4j database from python.

Python Client We can use Py2neo to connect to neo4j from python.

Py2neo is a client library and comprehensive toolkit for working with Neo4j from within Python applications and from the command line. The core library has no external dependencies and has been carefully designed to be easy and intuitive to use.

Snippets to connect, create, add nodes, add relationship and update property

from py2neo import authenticate, Graph, Node, Relationship connect to graph authenticate("localhost:7474", "neo4j", "passwd") graph = Graph("http://localhost:7474/db/data/")

create unique graph.schema.create$_{u}niqueness_{c}onstraint('Person','name')$

add nodes graph.create(Node.cast('Person', "name": "Alice")) graph.create(Node.cast('Person', "name": "Bob"))

add relationship source = graph.merge$_{o}ne$("$Person$", "$name$", "$Alice$")$target$ = $graph.merge_{o}ne$("$Person$", "$name$", "Bob")$graph.create_{u}nique(Relationship(source, "FRIEND", target))$

update property alice = graph.merge$_{o}ne$("$Person$", "$name$", "$Alice$")$alice$["age"] = 30$alice.push()$

11.4 Web Crawling

11.4.1 Introduction

Web Crawler Static Crawler

Apache Nutch Dynamic Crawler

nutch-selenium Intelligent Extractor

boilerpipe Web Content Extraction Through Machine Learning Priority Crawler, Social Crawler

Features a crawler must provide We list the desiderata for web crawlers in two categories: features that web crawlers must provide, followed by features they should provide.

Robustness:

The Web contains servers that create spider traps, which are generators of web pages that mislead crawlers into getting stuck fetching an infinite number of pages in a particular domain. Crawlers must be designed to be resilient to such traps. Not all such traps are malicious; some are the inadvertent side-effect of faulty website development.

Politeness:

Web servers have both implicit and explicit policies regulating the rate at which a crawler can visit them. These politeness policies must be respected.

Features a crawler should provide Distributed The crawler should have the ability to execute in a distributed fashion across multiple machines.

Scalable

The crawler architecture should permit scaling up the crawl rate by adding extra machines and bandwidth.

Performance and efficiency

The crawl system should make efficient use of various system resources including processor, storage and network bandwidth.

Quality

Given that a significant fraction of all web pages are of poor utility for serving user query needs, the crawler should be biased towards fetching "useful" pages first.

Freshness

In many applications, the crawler should operate in continuous mode: it should obtain fresh copies of previously fetched pages. A search engine crawler, for instance, can thus ensure that the search engine's index contains a fairly current representation of each indexed web page. For such continuous crawling, a crawler should be able to crawl a page with a frequency that approximates the rate of change of that page.

Extensible

Crawlers should be designed to be extensible in many ways - to cope with new data formats, new fetch protocols, and so on. This demands that the crawler architecture be modular.

Crawling The basic operation of any hypertext crawler (whether for the Web, an intranet or other hypertext document collection) is as follows.

The crawler begins with one or more URLs that constitute a seed set. It picks a URL from this seed set, then fetches the web page at that URL. The fetched page is then parsed, to extract both the text and the links from the page (each of which points to another URL). The extracted text is fed to a text indexer. The extracted links (URLs) are then added to a URL frontier, which at all times consists of URLs whose corresponding pages have yet to be fetched by the crawler. Initially, the URL frontier contains the seed set; as pages are fetched, the corresponding URLs are deleted from the URL frontier. The entire process may be viewed as traversing the web graph. In continuous crawling, the URL of a fetched page is added back to the frontier for fetching again in the future. This seemingly simple recursive traversal of the web graph is complicated by the many demands on a practical web crawling system: the crawler has to be distributed, scalable, efficient, polite, robust and extensible while fetching pages of high quality. We examine the effects of each of these issues. Our treatment follows the design of the Mercator crawler that has formed the basis of a number of research and commercial crawlers. As a reference point, fetching a billion pages (a small fraction of the static Web at present) in a month-long crawl requires fetching several hundred pages each second. We will see how to use a multi-threaded design to address several bottlenecks in the overall crawler system in order to attain this fetch rate.

Before proceeding to this detailed description, we reiterate for readers who may attempt to build crawlers of some basic properties any non-professional crawler should satisfy:

Only one connection should be open to any given host at a time. A waiting time of a few seconds should occur between successive requests to a host. Politeness restrictions should be obeyed.

A New Approach to Dynamic Crawler Build a crawler system for dynamic websites is not easy task. While you can use a web browser automator (like

selenium), or event when you can integrate selenium with nutch (by using nutch-selenium). These solutions are still hard to develop, hard to test and hard to manage sessions because we still "translate" our process to languages (such as java or python)

I suppose a new approach for this problem. Instead of using a web browser automator, we can inject native javascript codes into browser (via extension or add-on).The advantages of this approach is we can easily inject third party libraries (like jquery (for dom selector), Run.js (for complicated process) and APIs that supported by browsers). And we can take advance of debugging tool and testing framework in javascript world.

If you want to know about more details, feel free to contact me.

11.4.2 Scrapy

Scrapy An open source and collaborative framework for extracting the data you need from websites. In a fast, simple, yet extensible way.

Build and run your web spiders $pipinstallscrapy$ cat > myspider.py «EOF import scrapy

class BlogSpider(scrapy.Spider): name = 'blogspider' $start_urls = ['https://blog.scrapinghub.com']$

def parse(self, response): for title in response.css('h2.entry-title'): yield 'title': title.css('a ::text').$extract_first()$

$next_page = response.css('div.prev-post > a :: attr(href)').extract_first() if next_page : yield scrapy.Request(response.urljoin(next_page), callback = self.parse) EOF$
scrapy runspider myspider.py Deploy them to Scrapy Cloud $shub login Insert your Scrapinghub API Key :< API_KEY >$

Deploy the spider to Scrapy Cloud

$shub deploy$

Schedule the spider for execution shub schedule blogspider Spider blogspider scheduled, watch it running here: https://app.scrapinghub.com/p/26731/job/1/8

Retrieve the scraped data $shub items 26731/1/8 "title" : "Improved Frontera : Web Crawling at Scale with$

11.4.3 Apache Nutch

Highly extensible, highly scalable Web crawler 1 Nutch is a well matured, production ready Web crawler. Nutch 1.x enables fine grained configuration, relying on Apache Hadoop^{TM} data structures, which are great for batch processing.

History

Usecases

1. Features 1 1. Transparency Nutch is open source, so anyone can see how the ranking algorithms work. With commercial search engines, the precise details of the algorithms are secret so you can never know why a particular search result is ranked as it is. Furthermore, some search engines allow rankings to be based on payments, rather than on the relevance of the site's contents. Nutch is a good fit for academic and government organizations, where the perception of fairness of rankings may be more important.

2. Understanding We don't have the source code to Google, so Nutch is probably the best we have. It's interesting to see how a large search engine works. Nutch has been built using ideas from academia and industry: for instance, core parts of Nutch are currently being re-implemented to use the MapReduce.

Map Reduce distributed processing model, which emerged from Google Labs last year. And Nutch is attractive for researchers who want to try out new search algorithms, since it is so easy to extend.

3. Extensibility Don't like the way other search engines display their results? Write your own search engine–using Nutch! Nutch is very flexible: it can be customized and incorporated into your application. For developers, Nutch is a great platform for adding search to heterogeneous collections of information, and being able to customize the search interface, or extend the out-of-the-box functionality through the plugin mechanism. For example, you can integrate it into your site to add a search capability.

Process 5 0. initialize CrawlDb, inject seed URLs Repeat generate-fetch-update cycle n times:

1. The Injector takes all the URLs of the nutch.txt file and adds them to the CrawlDB. As a central part of Nutch, the CrawlDB maintains information on all known URLs (fetch schedule, fetch status, metadata, ...).

2. Based on the data of CrawlDB, the Generator creates a fetchlist and places it in a newly created Segment directory.

3. Next, the Fetcher gets the content of the URLs on the fetchlist and writes it back to the Segment directory. This step usually is the most time-consuming one.

4. Now the Parser processes the content of each web page and for example omits all html tags. If the crawl functions as an update or an extension to an already existing one (e.g. depth of 3), the Updater would add the new data to the CrawlDB as a next step.

5. Before indexing, all the links need to be inverted by Link Inverter, which takes into account that not the number of outgoing links of a web page is of interest, but rather the number of inbound links. This is quite similar to how Google PageRank works and is important for the scoring function. The inverted links are saved in the Linkdb.

6-7. Using data from all possible sources (CrawlDB, LinkDB and Segments), the Indexer creates an index and saves it within the Solr directory. For indexing, the popular Lucene library is used. Now, the user can search for information regarding the crawled web pages via Solr.

Installation Requirements

1. OpenJDK 7

2. Nutch 2.3 RC (yes, you need 2.3, 2.2 will not work)

wget https://archive.apache.org/dist/nutch/2.3/apache-nutch-2.3-src.tar.gz
tar -xzf apache-nutch-2.3-src.tar.gz 3. HBase 0.94.27 (HBase 0.98 won't work)

wget https://www.apache.org/dist/hbase/hbase-0.94.27/hbase-0.94.27.tar.gz
tar -xzf hbase-0.94.27.tar.gz 4. ElasticSearch 1.7

wget https://download.elastic.co/elasticsearch/elasticsearch/elasticsearch-1.7.0.tar.gz
tar -xzf elasticsearch-1.7.0.tar.gz Other Options: nutch-2.3, hbase-0.94.26, ElasticSearch 1.4

Setup HBase 1. edit $HBASE_ROOT/conf/hbase-site.xml and add

<configuration> <property> <name>hbase.rootdir</name> <value>file:///full/path/to/where/the/da
</property> <property> <name>hbase.cluster.distributed</name> <value>false</value>
</property> </configuration> 2. edit $HBASE_ROOT/conf/hbase-env.sh and enable JAVA_HOME and set it

- export $\text{JAVA}_{HOME} = /usr/java/jdk1.6.0/ + export JAVA_HOME =$
$/usr/lib/jvm/java-7-openjdk-amd64/This step might seem redundant, but even with JAVA_HOME being set$

3. kick off HBase:

HBASE$_R OOT/bin/start-hbase.shConfigureNutch1.EnabletheHBasedependencyinNUTCH_ROOT/iv$
<dependency org="org.apache.gora" name="gora-hbase" rev="0.5" conf="*-
>default" /> 2. Configure the HBase adapter by editing the $NUTCH_ROOT/conf/gora.properties$
-gora.datastore.default=org.apache.gora.mock.store.MockDataStore +gora.datastore.default=org.apache.
3. Build Nutch
cdNUTCH$_R OOTantcleanantruntimeThiscantakeawhileandcreates$NUTCH$_R OOT/runtime/local.$
4. configure Nutch by editing $NUTCH_ROOT/runtime/local/conf/nutch-site.xml$
<configuration> <property> <name>http.agent.name</name> <value>mycrawlername</value>
<!– this can be changed to something more sane if you like –> </property>
<property> <name>http.robots.agents</name> <value>mycrawlername</value>
<!– this is the robot name we're looking for in robots.txt files –> </property>
<property> <name>storage.data.store.class</name> <value>org.apache.gora.hbase.store.HBaseStore</va
</property> <property> <name>plugin.includes</name> <!– do NOT en-
able the parse-html plugin, if you want proper HTML parsing. Use something
like parse-tika! –> <value> protocol-httpclient|urlfilter-regex|parse-(text|tika|js)|index-
(basic|anchor)|query-(basic|site|url)|response-(json|xml)|summary-basic|scoring-
opic|urlnormalizer-(pass|regex|basic)|indexer-elastic </value> </property> <prop-
erty> <name>db.ignore.external.links</name> <value>true</value> <!– do
not leave the seeded domains (optional) –> </property> <property> <name>elastic.host</name>
<value>localhost</value> <!– where is ElasticSearch listening –> </prop-
erty> </configuration> or you configure Nutch by editing $NUTCH_ROOT/runtime/local/conf/nutch-site.xml$
<configuration> <property> <name>plugin.includes</name> <!– do ×
NOT enable the parse-html plugin, if you want proper HTML parsing. Use some-
thing like parse-tika! –> <value> protocol-http|protocol-httpclient|urlfilter-
regex| parse-(text|tika|js)|index-(basic|anchor)|query-(basic|site|url)|response-(json|xml)|
summary-basic|scoring-opic|urlnormalizer-(pass|regex|basic)|indexer-elastic| index-
metadata|index-more </value> </property> <property> <name>db.ignore.external.links</name>
<value>true</value> <!– do not leave the seeded domains (optional) –>
</property>
<!– elasticsearch index properties –> <property> <name>elastic.host</name>
<value>localhost</value> <description>The hostname to send documents to
using TransportClient. Either host and port must be defined or cluster. </de-
scription> </property>
<property> <name>elastic.port</name> <value>9300</value> <descrip-
tion> The port to connect to using TransportClient. </description> </prop-
erty> <property> <name>elastic.index</name> <value>nutch</value> <de-
scription> The name of the elasticsearch index. Will normally be autocreated
if it doesn't exist. </description> </property> <!– end index –> </configura-
tion> 5. configure HBase integration by editing $NUTCH_ROOT/runtime/local/conf/hbase-site.xml$
<?xml version="1.0" encoding="UTF-8"?> <configuration> <property>
<name>hbase.rootdir</name> <value>file:///full/path/to/where/the/data/should/be/stored</value>
<!– same path as you've given for HBase above –> </property> <property>
<name>hbase.cluster.distributed</name> <value>false</value> </property>
</configuration> or you configure HBase integration by editing $NUTCH_ROOT/runtime/local/conf/hbase-site.xml$:
<configuration> <property> <name>hbase.rootdir</name> <value>file:///$PATH/database <
$/value >< /property >< property >< name > hbase.cluster.distributed <

$/name >< value > false < /value >< /property >< property >< name >$ $hbase.zookeeper.quorum < /name >< value > hbase.io < /value >< /property >< property >< name > zookeeper.znode.parent < /name >< value > /hbase - unsecure < /value >< /property >< property >< name > hbase.rpc.timeout < /name >< value > 2592000000 < /value >< /property >< /configuration >$ $That's it. Everything is now setup to crawl websites.$

Run Nutch 1. Create an empty directory. Add a textfile containing a list of seed URLs

$mkdir seed$ echo "https://www.website.com" » seed/urls.txt $echo "https://www.another.com" >> seed/urls.txt$ echo "https://www.example.com" » seed/urls.txt Inject them into Nutch by giving a file URL (!)

$NUTCH_ROOT/runtime/local/bin/nutch inject file : ///path/to/seed/2. Generate a new set of URLs to fet$

This is is based on both the injected URLs as well as outdated URLs in the Nutch crawl db.

$NUTCH_ROOT/runtime/local/bin/nutch generate -topN 10 The above command will create job batches for 1$

3. Fetch the URLs. We are not clustering, so we can simply fetch all batches:

$NUTCH_ROOT/runtime/local/bin/nutch fetch -all 4. Now we parse all fetched pages :$

$NUTCH_ROOT/runtime/local/bin/nutch parse -all 5. Last step : Update Nutch's internal database :$

$NUTCH_ROOT/runtime/local/bin/nutch updatedb -all On the first run, this will only crawl the injected UR$

6. Putting Documents into ElasticSearch

$NUTCH_ROOT/runtime/local/bin/nutch index -all Configuration Crawl nutch via proxy$

Change $NUTCH_ROOT/runtime/local/conf/nutch - site.xml$

<configuration> <property> <name>http.proxy.host</name> <value>192.168.80.1</value> <description>The proxy hostname. If empty, no proxy is used.</description> </property> <property> <name>http.proxy.port</name> <value>port</value> <description>The proxy port.</description> </property> <property> <name>http.proxy.username</nam <value>username</value> <description>Username for proxy. This will be used by 'protocol-httpclient', if the proxy server requests basic, digest and/or NTLM authentication. To use this, 'protocol-httpclient' must be present in the value of 'plugin.includes' property. NOTE: For NTLM authentication, do not prefix the username with the domain, i.e. 'susam' is correct whereas 'DOMAIN-susam' is incorrect. </description> </property> <property> <name>http.proxy.password</name> <value>password</value> <description>Password for proxy. This will be used by 'protocol-httpclient', if the proxy server requests basic, digest and/or NTLM authentication. To use this, 'protocol-httpclient' must be present in the value of 'plugin.includes' property. </description> </property> </configuration> Nutch Plugins Extension Points In writing a plugin, you're actually providing one or more extensions of the existing extension-points . The core Nutch extension-points are themselves defined in a plugin, the NutchExtensionPoints plugin (they are listed in the NutchExtensionPoints plugin.xml file). Each extension-point defines an interface that must be implemented by the extension. The core extension points are:

Point Description Example IndexWriter Writes crawled data to a specific indexing backends (Solr, ElasticSearch, a CVS file, etc.). IndexingFilter Permits one to add metadata to the indexed fields. All plugins found which implement this extension point are run sequentially on the parse (from javadoc). Parser Parser implementations read through fetched documents in order to extract data to be indexed. This is what you need to implement if you want Nutch to be able to parse a new type of content, or extract more data from currently parseable content. HtmlParseFilter Permits one to add additional metadata to HTML

parses (from javadoc). Protocol Protocol implementations allow Nutch to use different protocols (ftp, http, etc.) to fetch documents. URLFilter URLFilter implementations limit the URLs that Nutch attempts to fetch. The RegexURL-Filter distributed with Nutch provides a great deal of control over what URLs Nutch crawls, however if you have very complicated rules about what URLs you want to crawl, you can write your own implementation. URLNormalizer Interface used to convert URLs to normal form and optionally perform substitutions. ScoringFilter A contract defining behavior of scoring plugins. A scoring filter will manipulate scoring variables in CrawlDatum and in resulting search indexes. Filters can be chained in a specific order, to provide multi-stage scoring adjustments. SegmentMergeFilter Interface used to filter segments during segment merge. It allows filtering on more sophisticated criteria than just URLs. In particular it allows filtering based on metadata collected while parsing page. Getting Nutch to Use a Plugin In order to get Nutch to use a given plugin, you need to edit your conf/nutch-site.xml file and add the name of the plugin to the list of plugin.includes. Additionally we are required to add the various build configurations to build.xml in the plugin directory.

Develop nutch plugins Project structure of a plugin plugin-name plugin.xml build.xml ivy.xml src org apache nutch indexer uml-meta source folder URL-MetaIndexingFilter.java scoring uml-meta source folder URLMetaScoringFilter.java test org apache nutch indexer uml-meta test folder URLMetaIndexing-FilterTest.java scoring uml-meta test folder URLMetaScoringFilterTest.java Follow this link to read develop nutch plugins

Architecture

Architectures

Data Structure The web database is a specialized persistent data structure for mirroring the structure and properties of the web graph being crawled. It persists as long as the web graph that is being crawled (and re-crawled) exists, which may be months or years. The WebDB is used only by the crawler and does not play any role during searching. The WebDB stores two types of entities: pages and links.

A page represents a page on the Web, and is indexed by its URL and the MD5 hash of its contents. Other pertinent information is stored, too, including

the number of links in the page (also called outlinks); fetch information (such as when the page is due to be refetched); the page's score, which is a measure of how important the page is (for example, one measure of importance awards high scores to pages that are linked to from many other pages). A link represents a link from one web page (the source) to another (the target). In the WebDB web graph, the nodes are pages and the edges are links.

A segment is a collection of pages fetched and indexed by the crawler in a single run. The fetchlist for a segment is a list of URLs for the crawler to fetch, and is generated from the WebDB. The fetcher output is the data retrieved from the pages in the fetchlist. The fetcher output for the segment is indexed and the index is stored in the segment. Any given segment has a limited lifespan, since it is obsolete as soon as all of its pages have been re-crawled. The default re-fetch interval is 30 days, so it is usually a good idea to delete segments older than this, particularly as they take up so much disk space. Segments are named by the date and time they were created, so it's easy to tell how old they are.

The index is the inverted index of all of the pages the system has retrieved, and is created by merging all of the individual segment indexes. Nutch uses Lucene for its indexing, so all of the Lucene tools and APIs are available to interact with the generated index. Since this has the potential to cause confusion, it is worth mentioning that the Lucene index format has a concept of segments, too, and these are different from Nutch segments. A Lucene segment is a portion of a Lucene index, whereas a Nutch segment is a fetched and indexed portion of the WebDB.

View gora-hbase-mapping.xml for more details

Config

Config nutch run intellij Copy file

copy all the files in the runtime/conf on out/test/apache-Nutch-2.3 and out/production/apache-Nutch-2.3

add these lines to file $NUTCH_SRC/out/test/nutch - site.xml$

<property> <name>plugin.folders</name> <value><nutch$_s$rc > /build/plugins < /value >< /property > $Runnutchinintellij Run- > EditConfigurations...- > addpathagrs : pathtofilelistlinkscrawlerDevNutchinIntellijReceipts : IntellIJ14, ApacheNutch2.3$

1. Get Nutch source

wget http://www.eu.apache.org/dist/nutch/2.3/apache-nutch-2.3-src.tar.gz tar -xzf apache-nutch-2.3-src.tar.gz 2. Import Nutch source in IntellIJ

[wonderplugin$_s liderid$ = "1"]

3. Get Dependencies by Ant

[wonderplugin$_s liderid$ = "3"]

4. Import Dependencies to IntellIJ

[wonderplugin$_s liderid$ = "4"]

Nutch Dev 1.Intasll java in ubuntu

-Downloads java version .zip

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html -Create folder jvm

sudo mkdir /usr/lib/jvm/ -Cd to folder downloads java version .zip

sudo mv jdk1.7.0$_x$//$usr/lib/jvm/jdk$1.7.0$_x$ - $Runcommandline$

sudo update-alternatives –install /usr/bin/java java /usr/lib/jvm/jdk1.7.0$_x/jre/bin/java$0- $Tetsversionjava$

java -version 2.Intasll ant in ubuntu

-Downloads ant

http://ant.apache.org/manualdownload.cgi -Add path ant vao file environment

sudo nano /etc/environment $ANT_ROOT/bin - Runcommandline$

source /etc/environment ant -version 3.Intasll hbase in ubuntu

-Downloads and extract hbase 0.94.27

https://archive.apache.org/dist/hbase/hbase-0.94.27/ -Edit file $HABSE_ROOT/conf/hbase- site.xml$

<configuration> <property> <name>hbase.rootdir</name> <value>file:///$PATH_DATA_BASE/datal$ /value >< /property >< property >< name > hbase.cluster.distributed < /name >< value > false < /value >< /property >< property >< name > hbase.zookeeper.quorum < /name >< value > hbase.io < /value >< /property >< property >< name > zookeeper.znode.parent < /name >< value > /hbase - unsecure < /value >< /property >< property >< name > hbase.rpc.timeout <

/name >< value > 2592000000 < /value >< /property >< /configuration >
−Editfile$\text{HBASE}_{R}OOT/conf/hbase - env.sh$
export $\text{JAVA}_{H}OME$ =$\text{PATH}_{J}AVA_{H}OME-Editfile\text{HBASE}_{R}OOT/conf/regionservers$
hbase.io.nutch -Edit file hosts in ubuntu
sudo nano /etc/hosts ip hbase.io.nutch -Edit file hostname in ubuntu
sudo nano /etc/hostname hbase.io.nutch -Run and stop hbase in ubuntu
Run hbase : cd $HBASE_{R}OOT/bin./start-hbase.shStophbase$: $cd\text{HBASE}_{R}OOT/bin./stop-hbase.sh * Errorinintasllhbase$
- Error regionserver localhost(Edit file hosts and file host name) - Error client no remote server intasll hbase(Turn off file firewall) 4.Build nutch in ant
-Downloads and extract nutch
http://nutch.apache.org/ -Edit file $NUTCH_{R}OOT/ivy/ivy.xml$
<dependency org="org.apache.gora" name="gora-hbase" rev="0.5" conf="*->default" /> -Edit file $NUTCH_{R}OOT/ivy/ivysettings.xml$
<property name="repo.maven.org" value="http://repo1.maven.org/maven2/" override="false"/>
<property name = "repo.maven.org" value = "http://maven.oschina.net/content/groups/public/" override = "false" /> -Edit file $NUTCH_{R}OOT/conf/nutch - site.xml$
<configuration> <property> <name>plugin.folders</name> <value>$NUTCH_{R}OOT/build/plugins$ < /value >< /property >< property >< name > http.agent.name < /name >< value > mycrawlername < /value ><!−−thiscanbechangedtosomethingmoresaneifyoulike−− >< /property >< property >< name > http.robots.agents < /name >< value > mycrawlername < /value ><!−−thisistherobotnamewe′relookingforinrobots.txtfiles−− >< /property >< property >< name > storage.data.store.class < /name >< value > org.apache.gora.hbase.store.HBaseStore < /value >< /property >< property >< name > plugin.includes < /name ><!−−doNOTenabletheparse−htmlplugin, ifyouwantproperHTMLparsing.Usesomethinglikeparse−tika!−− >< value > protocol − http|protocol − httpclient|urlfilter − regex|parse − (text|tika|js)|index−(basic|anchor)|query−(basic|site|url)|response−(json|xml)|summary−basic|scoring−opic|urlnormalizer−(pass|regex|basic)|indexer−elastic|index−metadata|index − more < /value >< /property >< property >< name > db.ignore.external.links < /name >< value > true < /value ><!−−donotleavetheseededdomains(optiona −− >< /property >
<!– elasticsearch index properties –> <property> <name>elastic.host</name> <value>localhost</value> <description>The hostname to send documents to using TransportClient. Either host and port must be defined or cluster. </description> </property>
<property> <name>elastic.port</name> <value>9300</value> <description> The port to connect to using TransportClient. </description> </property> <property> <name>elastic.index</name> <value>nutch</value> <description> The name of the elasticsearch index. Will normally be autocreated if it doesn't exist. </description> </property> <!– end index –>
<property> <name>http.proxy.host</name> <value>192.168.80.1</value> </property> <property> <name>http.proxy.port</name> <value>8080</value> </property> <property> <name>http.proxy.username</name> <value>user1</value> </property> <property> <name>http.proxy.password</name> <value>user1</value> </property> </configuration> -Edit file file $NUTCH_{R}OOT/conf/gora.property$
gora.datastore.default=org.apache.gora.hbase.store.HBaseStore -Build nucth
ant runtime or ant eclipse -verbose -Create file links
-Run nutch

cd $NUTCH_ROOT/runtime/local/bin runinject : ./nutch inject file : ///PATH_LIKN Srun generate : ./nutch generate − topN 10 run fetch : ./nutch fetch − all run parse : ./nutch parse − all run updatedb : ./nutch updatedb − all − Downloads and extract elastic

https://www.elastic.co/downloads/elasticsearch -Run elastic

cd $ELASTIC/bin./elasticsearch − Index data in elastic

cd $NUTCH_ROOT/runtime/bin runindex : ./nutch index − all 5.Run nutch intellij

Change $NUTCH_ROOT/runtime/local/conf/hbase − site.xml

<configuration> <property> <name>hbase.rootdir</name> <value>file:///home/hainv/Downloads/cr
</property> <property> <name>hbase.cluster.distributed</name> <value>false</value>
</property> <property> <name>hbase.zookeeper.quorum</name> <value>hbase.io</value>
</property> <property> <name>zookeeper.znode.parent</name> <value>/hbase-
unsecure</value> </property> <property> <name>hbase.rpc.timeout</name>
<value>2592000000</value> </property> </configuration> Nutch plugin in-
tellij 1.Structure nutch :[1] 2.Run nutch intellij Downloads nucth2.3:http://nutch.apache.org/downloads.html
Editing file $NUTCH_ROOT/ivy/ivysettings.xml

<ivysettings> <property name="oss.sonatype.org" value="http://oss.sonatype.org/content/repositories/
override="false"/> <property name = "repo.maven.org" value = "http://maven.oschina.net/content/groups/
override = "false" /> <property name="repository.apache.org" value="https://repository.apache.org/conten
override="false"/> <property name="maven2.pattern" value="[organisation]/[module]/[revision]/[module]-
[revision]"/> <property name="maven2.pattern.ext" value="maven2.pattern.[ext]"/ >< !−−pullinthelocalrepository−− >< includeurl = "ivy.default.conf.dir/ivyconf-
local.xml"/> <settings defaultResolver="default"/> <resolvers> <ibiblio name="maven2"
root="repo.maven.org"pattern = "maven2.pattern.ext" m2compatible="true"
/> <ibiblio name="apache-snapshot" root="repository.apache.org"changingPattern =
".*−SNAPSHOT"m2compatible = "true"/ >< ibiblioname = "restlet"root =
"http : //maven.restlet.org"pattern = "maven2.pattern.ext" m2compatible="true"
/> <ibiblio name="sonatype" root="oss.sonatype.org"pattern = "maven2.pattern.ext"
m2compatible="true" />

<chain name="default" dual="true"> <resolver ref="local"/> <resolver
ref="maven2"/> <resolver ref="sonatype"/> <resolver ref="apache-snapshot"/>
</chain> <chain name="internal"> <resolver ref="local"/> </chain> <chain
name="external"> <resolver ref="maven2"/> <resolver ref="sonatype"/>
</chain> <chain name="external-and-snapshots"> <resolver ref="maven2"/>
<resolver ref="apache-snapshot"/> <resolver ref="sonatype"/> </chain> <chain
name="restletchain"> <resolver ref="restlet"/> </chain> </resolvers> <mod-
ules> <module organisation="org.apache.nutch" name=".*" resolver="internal"/>
<module organisation="org.restlet" name=".*" resolver="restletchain"/> <mod-
ule organisation="org.restlet.jse" name=".*" resolver="restletchain"/> </mod-
ules> </ivysettings> Editing file $NUTCH_ROOT/ivy/ivy.xml

<dependency org="org.apache.gora" name="gora-hbase" rev="0.5" conf="*-
>default" /> Editing file $NUCTH_ROOT/conf/gora.properties

gora.datastore.default=org.apache.gora.hbase.store.HBaseStore Editing file
$NUTCH_ROOT/conf/nutch_site.xml

<configuration> <property> <name>plugin.folders</name> <value>$NUTCH_ROOT/build/plugins <
/value >< /property >< property >< name > http.agent.name < /name >< value > mycrawlername < /value ><!−−thiscanbechangedtosomethingmoresaneifyoulike−
− >< /property >< property >< name > http.robots.agents < /name >< value > mycrawlername < /value ><!−−thisistherobotnamewe′relookingforinrobots.txtfiles−
− >< /property >< property >< name > storage.data.store.class < /name >< value > org.apache.gora.hbase.store.HBaseStore < /value >< /property ><

property >< name > plugin.includes < /name ><!−−doNOTenabletheparse−htmlplugin, if youwantproperHTMLparsing.Usesomethinglikeparse−tika!−− >< value > protocol−httpclient|urlfilter−regex|parse−(text|tika|js)|index−(basic|anchor)|query−(basic|site|url)|response−(json|xml)|summary−basic|scoring−opic|urlnormalizer−(pass|regex|basic)|indexer−elastic < /value >< /property >< property >< name > db.ignore.external.links < /name >< value > true < /value ><! − −donotleavetheseededdomains(optional) − − >< /property >< property >< name > elastic.host < /name >< value > localhost < /value >< ! − −whereisElasticSearchlistening − − >< /property >

<property> <name>http.proxy.host</name> <value>192.168.80.1</value> <description>The proxy hostname. If empty, no proxy is used.</description> </property> <property> <name>http.proxy.port</name> <value>8080</value> <description>The proxy port.</description> </property> <property> <name>http.proxy.username</nam <value>user1</value> <description>Username for proxy. This will be used by 'protocol-httpclient', if the proxy server requests basic, digest and/or NTLM authentication. To use this, 'protocol-httpclient' must be present in the value of 'plugin.includes' property. NOTE: For NTLM authentication, do not prefix the username with the domain, i.e. 'susam' is correct whereas 'DOMAINsusam' is incorrect. </description> </property> <property> <name>http.proxy.password</name> <value>user1</value> <description>Password for proxy. This will be used by 'protocol-httpclient', if the proxy server requests basic, digest and/or NTLM authentication. To use this, 'protocol-httpclient' must be present in the value of 'plugin.includes' property. </description> </property> </configuration> Editing file $NUTCH_ROOT/conf/hbase − site.xml$

<configuration> <property> <name>hbase.rootdir</name> <value>file:///home/rombk/Downloads/c </property> <property> <name>hbase.cluster.distributed</name> <value>false</value> </property> <property> <name>hbase.zookeeper.quorum</name> <value>hbase.io</value> </property> <property> <name>zookeeper.znode.parent</name> <value>/hbase-unsecure</value> </property> <property> <name>hbase.rpc.timeout</name> <value>2592000000</value> </property> </configuration> Run terminal

ant eclipse -verbose Import nucth intellij

3.Run plugin creativecommons Sample plugins that parse and index Creative Commons medadata.1 Step 1. Create folder creativecommons in path $NUTCH_HOME/out/test/$

Step 2. Create file nutch-site.xml in folder $NUTCH_HOME/out/test/creativecommonsandaddcontent$

<?xml version="1.0"?> <?xml-stylesheet type="text/xsl" href="configuration.xsl"?> <!– Put site-specific property overrides in this file. –> <configuration> <property> <name>plugin.folders</name> <value>*NUTCH_HOME/build/plugins < /value >< /property >< property >< name > http.agent.name < /name >< value > mycrawlername < /value ><!−−thiscanbechangedtosomethingmoresaneifyoulike−− >< /property >< property >< name > http.robots.agents < /name >< value > mycrawlername < /value ><!−−thisistherobotnamewe′relookingforinrobots.txtfiles−− >< /property >< property >< name > storage.data.store.class < /name >< value > org.apache.gora.hbase.store.HBaseStore < /value >< /property >< property >< name > plugin.includes < /name ><!−−doNOTenabletheparse−htmlplugin, if youwantproperHTMLparsing.Usesomethinglikeparse−tika!−− >< value > indexer − elastic|creativecommons|parse − html < /value >< /property >< property >< name > db.ignore.external.links < /name >< value > true < /value ><!−−donotleavetheseededdomains(optional)−− >< /property >< property >< name > elastic.host < /name >< value >*

$localhost < /value >< !--whereisElasticSearchlistening-- >< /property >< !--configproxy-- >< property >< name > http.proxy.host < /name >< value >< hosts >< /value >< description > Theproxyhostname.Ifempty, noproxyisused. < /description >< /property >< property >< name > http.proxy.port < /name >< value >< port >< /value >< description > Theproxyport. < /description >< /property >< property >< name > http.proxy.username < /name >< value >< user1 >< /value >< description > Usernameforproxy.Thiswillbeusedby'protocol-httpclient', iftheproxyserverrequestsbasic, digestand/orNTLMauthentication.Tousethis,' protocol-httpclient'mustbepresentinthevalueof'plugin.includes'property.NOTE : ForNTLMauthentication, donot/description >< /property >< property >< name > http.proxy.password < /name >< value >< user1 >< /value >< description > Passwordforproxy.Thiswillbeusedby'protocol-httpclient', iftheproxyserverrequestsbasic, digestand/orNTLMauthentication.Tousethis,' protocol-httpclient'mustbepresentinthevalueof'plugin.includes'property. < /description >< /property >< /configuration > 2.RunpluginfeedPluginfeedparsingofrssError : ParsingofRSSfeedsfails(tejasp)[2]andreadfile$NUTCH_{R}OOT/CHANFES.txt$

Tài liệu tham khảo

Jurafsky, Daniel / Martin, James H. (2009): *Speech and Language Processing (2nd Edition)*. Upper Saddle River, NJ, USA, Prentice-Hall, Inc.

Chỉ mục

Ghi chú