Statistics for specific domains

Adam Boult (www.bou.lt)

February 10, 2023

Contents

Preface

This is a live document, and is full of gaps, mistakes, typos etc.

Part I

Computer vision

Chapter 1

Classifying written characters

1.1 Character recognition

1.1.1 MNIST

Chapter 2

Classifying images

2.1 Image recognition

2.1.1 CIFAR-10

2.1.2 ImageNet

Chapter 3

Facial recognition

3.1 Facial recognition

3.1.1 FERET

Chapter 4

Computer vision

4.1 Camera vision

4.1.1 Camera inputs

4.2 Classifying images

4.3 Semantic image segmentation

4.4 Building 3D models

4.4.1 Multi-view CNNs

4.4.2 Volumetric models

4.4.3 Point clouds

4.4.4 Polygon mesh

4.4.5 Generative Query Network (GQN)

4.4.6 Primitive-based CAD

4.4.7 3D ShapeNets

4.4.8 Building 3D models from scans

4.5 LIDAR

4.5.1 LIDAR

4.5.2 Classification with voxels

4.5.3 Absolute risk aversion

Correspondence (2 models of humans in different poses; understand where each part of one relates to a specific part of the other)

4.5.4 Parsing

Mulitple objects in scene, objects have parts segmentation.

Part II

Applications for time series models

Chapter 5

Data cleaning: Text

5.1 Cleaning categorial data

5.1.1 One Hot Encoding

5.2 Cleaning text data

5.2.1 Bag-of-words

5.2.2 N-grams

Introduction

We can add start and end of sentence markets. * and STOP

Generally remove punctuation

5.2.3 Feature hashing

Chapter 6

Text prediction

6.1 Text

Chapter 7

Text translation

7.1 Text

Chapter 8

Natural Language Processing (NLP)

8.1 Other

8.1.1 Probabilistic language models

Introduction

Probabilistic language models can predict future words given a history of words.

This can be used for predictive text. For example if a user types "Did you call your" we may want to estimate the probability that the next word is "child".

We can state this problem:

$P(child|did\ you\ call\ your)$

By definition this is:

$$P(child|did\ you\ call\ your) = \frac{P(did\ you\ call\ your\ child)}{P(did\ you\ call\ your)}$$

We can estimate each of these:

$$P(did\ you\ call\ your\ child) = \frac{|did\ you\ call\ your\ child|}{|5\ word\ sentences|}$$

$$P(did\ you\ call\ your) = \frac{|did\ you\ call\ your|}{|4\ word\ sentences|}$$

Data requirements

This needs a large corpus, which may not be practical.

Additionallly, the words must be indexed, and not simply stored as a bag of words.

Decomposition

We can decompose the probabilities using the chain rule.

$P(did\ you\ call\ your\ child) = P(did)P(you|did)...P(child|did\ you\ call\ your)$

$P(w_1, ..., w_k) = \prod_k p(w_k|w_1, ..., w_{k-1})$

N-grams

We can simplify the decomposition using the Markov assumption:

$P(w_k|w_1, ..., w_{k-1}) = P(w_k|w_{k-1})$

This is a 1-gram.

We can do this for n words back. This is an n-gram.

Smoothing

We can use smoothing to address small corpuses.

$$P(did\ you\ call\ your\ child) = \frac{|did\ you\ call\ your\ child| + 1}{|5\ word\ sentences| + V}$$

$$P(did\ you\ call\ your) = \frac{|did\ you\ call\ your| + 1}{|4\ word\ sentences| + V}$$

For some value V.

Perplexity

We can compare probabilistic language models using perplexity.

We can then choose the model with the lowest perplexity.

$$perplexity(w_1, w_2, ..., w_n) = P(w_1, w_2, ..., w_n)^{-\frac{1}{n}}$$

We can expand this:

$$perplexity(w_1, w_2, ..., w_n) = \prod_i P(w_i|w_1, ..., w_{i-1})^{-\frac{1}{n}}$$

Depending on which n-gram we use we can then simplify this.

8.1.2 Word2vec

8.1.3 Latent Semantic Analysis

8.2 Machine translation

8.2.1 Machine translation

Part III

Audio and Natural Language Processing

Chapter 9

Voice recognition

9.1 Audio

Chapter 10

Song recognition

10.1 Audio

Chapter 11

Audio-to-text / text-to-audio

11.1 Audio

Part IV

Linguistics

Chapter 12

Comparative method

Chapter 13

Internal reconstruction

Chapter 14

Universal grammar

Part V

Communication

Chapter 15

Cryptographic hashes

15.1 Adversaries

15.1.1 Brute force attacks

15.1.2 Pre-image attacks

Given hash value h, can we find message m?

15.1.3 Defence from pre-image attacks

15.1.4 Second pre-image attacks

Given m_1, can we find m_2 with same hash?

Defence from second pre-image attacks

15.1.5 Hash collision

Can i find any two matching messages?

Hash collision attacks

I can get someone to vouch for one of the messages, and then claim they vouched the other.

Hash collision defence

15.2 Passwords

15.2.1 Plaintext databases

15.2.2 Hashed passwords

15.2.3 Rainbow tables

15.2.4 Dictionary attacks

15.2.5 Salting

It is possible to brute force hashes, especially for smaller inputs such as short passwords.

If password hashes for a hashing algorithm were brute forced, then passwords could easily be recovered from another hash table.

To prevent this a salt can be added to the document.

If a password is "apple", then instead the salt "xyz" could be added to create "applexyz". This prevents the previous cracking of "apple" to be used.

The salt would then be stored in plaintext alongside the password hash.

15.3 Examples of cryptographic hash functions

15.3.1 SHA

Chapter 16

Classical encryption

16.1 Introduction

16.1.1 Plaintext and ciphertext

16.1.2 ROT13

Rotate 13. It is its own inverse.

16.1.3 Atbash

Revese the alphabet. It is its own inverse.

16.2 Verifying decryptions

16.2.1 Corpus

verifying solutions when spaces are omitted. can rate fitness using corpus information on popularity

16.3 Caesar

16.3.1 Caesar ciphers

Shift along in alphabet by c.

16.3.2 Affine cipher

page on affine cipher too. like caesar but rather than +c, mx+c

16.3.3 Breaking

For Caesar, only 26 possible keys, can just brute force.

For Affine, can also brute force.

16.4 Monoalphabetic substitution

16.4.1 Monoalphabetic substitution ciphers and keys

(key plus algorithm encrpyts and decrypts)

16.4.2 Breaking monoalphabetic substitution ciphers with frequency analysis

(need to identify algorithm and needs to identify key)

finding substitution cyphers

Search space is larger, $26! = 4 * 10^26$. need alternative to brute force.

Letter popularity. Compare against popularity for corpus. Monogram (ie letters); ngrams(ie n letter in a row frequency); common words.

Single letter words are I or A. More generally. corpus smaller for fewer letters

Can test substitution cypher by matching each word against a corpus

16.5 Polyalphabetic substitution

16.5.1 Polyalphabetic ciphers

Multiple substitution

Vigenere

Rotor machines

The Enigma machine

16.5.2 Breaking polyalphabetic ciphers with the Kasiski examination

16.6 Other

16.6.1 Codebooks

((eg sdrgdr is code for "meet at x on y")

16.6.2 Transposition ciphers

16.6.3 Book cipher

Eg use Bible.

16.6.4 One-time pads

Chapter 17

Modern symmetric encryption

17.1 Methods

17.1.1 Block ciphers

17.1.2 Stream ciphers

17.1.3 Motivation

Increased computer power. How to be secure?

kerckhoff's principle. choose cipher such that secure even if everything but key is known

17.2 Symmetric encryption

17.2.1 Symmetric

We have a document we want to be able to transfer on an insecure medium.

We use a key to encrypt the file, and a key to decrypt the file.

With symmetric encryption these are the same key.

17.3 Options for algorithms

17.3.1 Integer factorisation

Option for algorithm.

17.3.2 Elliptical-curve cryptography

Chapter 18

Modern asymmetric encryption

18.1 Asymmetric encryption

18.1.1 Public keys

18.1.2 RSA

18.1.3 Message signing

18.1.4 Pretty Good Privacy (PGP)

18.1.5 Using public keys to facilitate symmetric encryption

18.1.6 Elliptical-curve cryptography

18.1.7 Asymmetric encryption

Here we use different keys to encrypt and decrypt the file.

Consider two users who wish to send a message securely.

One option would be to use symmetric encrpytion. They would have to meet and share this key securely, however, as transferring it over an insecure network would mean it could be copied.

With public key encryption each user has a public and a private key. The private key is kept secure locally, while the public key can be broadcasted.

In order to encrypt the file, the recipient's public key is used, while both the private and public key are needed to decrypt the file.

As a result anyone can encrypt a file to send to the user, but only the user can read what is sent.

Public-key encryption can be used to facilitate symmetric encryption. If only one party has a public key then the other user can send a symmetric key securely using the public key.

Using this, asymmetric encrpyiton is only used at the start.

This is how HTTPS operates, where the website has a public key, but the client does not.

Each user still needs to trust that the public key is accurate. This could be done by hosting the public key on a secure location.

RSA is an algorithm used for public-key encrpytion, including for HTTPS handshakes and PGP.

18.1.8 Sort

Pages for:

+ Public keys + RSA + Message signing + PGP + Public keys to facilitate symmetric encryption

18.2 Exchanging keys

18.2.1 Diffie-Hellman key exchange

Part VI

Other

Chapter 19

Weather forecasting

19.1 Weather