MTAT.03.311
Loomuliku keele töötlus *Python*is
(6 EAP)

Karl-Oskar Masing

03.11.2015

Klasterdamine

Sarnaste objektide automaatne grupeerimine.

Klasterduse hindamine

- Visualiseerimine
  - eksperdi hinnang gruppidele
- Automaatne
  - tõelise klasterduse ning leitud klasterduse võrdlemine (*external validity index*)
    - eeldab tõelise klasterduse teadmist
  - leitud klasterduse struktuuri hindamine (*internal validity index*)
    - sageli klastrisisene vs -väline kaugus

Automaatne klasterduse hindamine struktuuri läbi

- silueti koefitsient
  - vahemikus [-1,1]
    - -1 = halb
    - 1 = hea

Silueti koefitsient

a - keskmine kaugus fikseeritud punkti ja kõikide teiste samasse klastrisse kuuluvate punktide vahel

b - keskmine kaugus fikseeritud punkti ja kõikide punktide vahel, mis kuuluvad teise lähimasse klastrisse

Ühe punkti jaoks: (b - a) / max(b - a)

Klasterduse jaoks: kõikide punktide keskmine

Silueti koefitsient Pythonis

```
>>> import numpy as np
>>> from sklearn import metrics
>>> from sklearn.cluster import KMeans
>>> X = np.random.rand(20,5)
>>> model = KMeans(n_clusters=4, random_state=14).fit(X)
>>> metrics.silhouette_score(X, model.labels_, metric='cosine')
0.2325130498004003
```

Keelemudelid

Motivatsioon

- Süsteemid leiavad palju erinevaid tekstilisi kandidaate
  - kõnetuvastus
  - käekirjatuvastus
  - masintõlge

Kuidas leida keeleliselt parim tekst?

Kuidas hinnata teksti?

Keelemudel

Sõnajärjendite tõenäosusjaotus.

Tõenäosusjaotus

- Seab igale võimalikule tulemusele tõenäosuse
- Tõenäosuste summa on 1

Keelemudel

“Isa Goriot’ makaronidele riivi juustu peale.”

vs

“Lisa Goriot makaronidele, riivi juustu peale.”

P(“Isa Goriot’ makaronidele riivi juustu peale.”)

>

P(“Lisa Goriot makaronidele, riivi juustu peale.”)

Keelemudel

- Aluseks sõnad
- Sõnad sõltuvad teineteisest ning järjekorrast
- Laused on erinevate pikkustega
- Õpitud treeningandmestikul

P(“Isa Goriot’ makaronidele riivi juustu peale.”) =

= P(“Isa”) x P(“Goriot” | “Isa”) x

x P(“makaronidele” | “Isa Goriot”) x

x P(“riivi” | “Isa Goriot makaronidele”) x ...

x P(“.” | “Isa Goriot makaronidele riivi juustu peale”)

Keelemudel

- Teoreetiliselt OK
- Õpiks üle
- Arvutuslikult mõeldamatu
  - keelemudel peab praktikas
    - olema kiire
    - suutma väärtustada suvalise tingliku tõenäosuse
      - mida teha 10-, 50-, 100-sõnaliste lausetega?

Keelemudel

- Teoreetiliselt OK
- Õpiks üle
- Arvutuslikult mõeldamatu
  - keelemudel peab praktikas
    - olema kiire
    - suutma väärtustada suvalise tingliku tõenäosuse
      - mida teha 10-, 50-, 100-sõnaliste lausetega?

Peame probleemi lihtsustama.

n-gram

“Lisa Goriot makaronidele, riivi juustu peale.”

- 1-gram
  - (“Lisa”), (“Goriot”), (“makaronidele”), (“,”), (“riivi”), …
- 2-gram
  - (“Lisa” ,”Goriot”), (“Goriot”, “makaronidele”), (“makaronidele”, ”,”), …
- 3-gram
  - (“Lisa”, “Goriot”, “makaronidele“), (“Goriot”, “makaronidele”, “,”), ...

n-grammi mudel

- Eeldame, et suvaline sõna sõltub vaid (n-1) eelnevast sõnast
- Piisab vaid n-grammide ja (n-1)-grammide sageduste talletamisest
- Teoreetiliselt ebaefektiivne, sageli töötab piisavalt hästi

n = 3

$P(w_4 \mid w_1,w_2,w_3) = C(w_1,w_2,w_3,w_4) / C(w_1,w_2,w_3)$

ilus koer ja kass .

koer oli ilus .

kass vaatas koera .

n = 1 (unigram - kontekstita)

P(“koer oli kass .”) = P(“koer”) * P(“oli”) * P(“kass”) * P(“.”) =

= C(“koer”) / C() * C(“oli”) / C() * C(“kass”) / C() * C(“.”) / C() =

= 2 / 13 * 1 / 13 * 2 / 13 * 3 / 13 =

= $12 / 13^4$

<s> ilus koer ja kass . </s>

<s> koer oli ilus . </s>

<s> kass vaatas koera . </s>

n = 2 (bigram)

P(“koer oli kass .”) = P(“koer” | <s>) * P(“oli” | “koer”) * P(“kass” | “oli”) * P(“.” | “kass”) * P(</s>, “.”) =

= C(<s>,“koer”) / C(<s>) * C(“koer”, “oli”) / C(“koer”) * C(“oli”, “kass”) / C(“oli”) *

* C(“kass”, “.”) / C(“kass”) * C(“.”,”</s>”) / C(“.”) =

= 1 / 3 * 1 / 2 * 0 / 1 * 1 / 2 * 3 / 3 =

= 0

Andmete hõredus

- Asendame harvaesinevad tüübid metasümbolitega
  - nt
    - numbrid -> <num>
    - harvemini esinevad kirjavahemärgid -> <punct>
- Agregeerimine OK
- Andmete kaotamine (nagu tunnuste juures) POLE OK

Andmete hõredus

- Silume

- Treeningandmetes mitteesinevad sõnad
  - OOV - *out of vocabulary*
  - <unk> (*unkown*)
  - treenides nt iga sõna esimene esinemine talletada kui <unk>

Silumine

Smoothing

Silumine

- Eesmärk
  - kaotada mudelist nullid
- Viis
  - jagada tõenäosusmassi 0 sagedusega sõnade järjenditele

Laplace’i silumine

- $P(w_2 \mid w_1) = [\ C(w_1,w_2) + 1\ ] / [\ C(w_1) + V\ ]$
  - *add-one*
  - jagab liiga palju tõenäosusmassi õigetelt (treeningandmetes nähtud) sõnade järjenditelt
    - hõredad andmed, enamus sagedustest 0
- $P(w_2 \mid w_1) = [\ C(w_1,w_2) + \alpha\ ] / [\ C(w_1) + \alpha * V\ ]$
  - $0 \leq \alpha \leq 1$
  - üldistatud

Sageduste sättimine uute sõnade korral

- Good-Turing
  - kasutame n sagedusega sõnade tegeliku sageduse hindamiseks n+1 sagedusega sõnade sagedust
  - liigutame tõenäosusmassi sujuvamalt
  - $c^* = (c+1)*N_{c+1} / N_c$
    - c^*- vaadeldava sõna uus absoluutne sagedus
    - c - vaadeldava sõna absoluutne sagedus
    - N_c - c sagedusega sõnade sagedus
- $P(x) = c^* / N$

Mudeli evalueerimine

- väline hindamine
  - parim
  - mudelid integreeritakse rakendusse ja hinnatakse, milline on parim (täpsus jms)
- sisemine hindamine
  - mõõdame kvaliteeti testandmestikul
  - nõutus (*perplexity*)

Nõutus (perplexity)

- Võrreldavad mudelid samal testandmestikul ja sama sõnavaraga
- Defineeritud üle ühe lause/teksti
- Mida suurem tõenäosus, seda väiksem nõutus

$$W = w_1\ w_2\ \ldots\ w_N$$

$$PP(W) = P(w_1\ w_2\ \ldots\ w_N)\ **\ (-1/N)$$

Tõenäosuste hinnangute ühendamine

- Seni oleme tegelenud ühe mudeliga
- Oleme ignoreerinud rohkema info eraldamist andmetest
- Saame kasutada mitme n-grammi mudeli tarkust

Lineaarne interpolatsioon

- Lineaarkombinatsioon tõenäosuste hinnangutest

n = 2

$$P(w_i \mid w_{i-2}, w_{i-1}) = \lambda_1 P(w_i \mid w_{i-2}, w_{i-1}) + \lambda_2 P(w_i \mid w_{i-1}) + \lambda_3 P(w_i) \quad , \quad \lambda_1 + \lambda_2 + \lambda_3 = 1$$

Üldiselt $\lambda_1 > \lambda_2 > \lambda_3 > 0$

Katzi tagurdamine (*backoff*)

$P\_hat(w_i \mid w_{i-2}, w_{i-1}) =$

- kui $C(w_{i-2}, w_{i-1}, w_i) > 0$, siis
  - $P(w_i \mid w_{i-2}, w_{i-1})$
- kui $C(w_{i-2}, w_{i-1}, w_i) = 0$ ja $C(w_{i-1}, w_i) > 0$, siis
  - $\alpha_1 P(w_i \mid w_{i-1})$
- vastasel juhul
  - $\alpha_2 P(w_i)$

$1 > \alpha_1 > \alpha_2 > 0$, sageli nt $\alpha_2 = \alpha_1^2$

Praktika

- Tõenäosuste korrutise asemel võtame tõenäosustest logaritmid ja liidame
  - väldime *underflow*'id
  - liitmine kiire

$$\log(p_1 \times p_2 \times p_3) = \log(p_1) + \log(p_2) + \log(p_3)$$

Ohud praktikas

- log(0) = defineerimata
  - ~ -∞

- log(x) ∈ (-∞) , x ∈ (0,1)
  - oluline teave mudeli evalueerimisel

Teksti genereerimine

- Kasutame treenitud n-grammi mudelit
  - valime teksti alguse n-grammi vastavalt tõenäosusele
    - algab (n-1) <s> sümboliga
  - kui bigrammide korral osutus valituks
    - <s> Kass
  - siis liidame vastavalt tõenäosusele bigrammi, mis algab sõnaga “Kass”, oletame (“Kass”,”on”)
    - <s> Kass on
- Mitte võtta maksimaalse tõepäraga!

Teksti genereerimine üle mitme korpuse

- Treenime mudeli igal korpusel
  - nt piibel, ajalehe kommentaarid ja “Harry Potter”
- Ühendame alammudelid üheks süsteemiks
  - lihtsaimaks viisiks lineaarne interpolatsioon
    - kasutaja saab valida, kui suure osakaaluga mingist korpusest n-gramme valitakse
- Tõenäosuse arvutamisel kasutatav nimetaja moodustub alammudelite nimetajate summast

Sõnade koti mudel (*bag-of-words*)

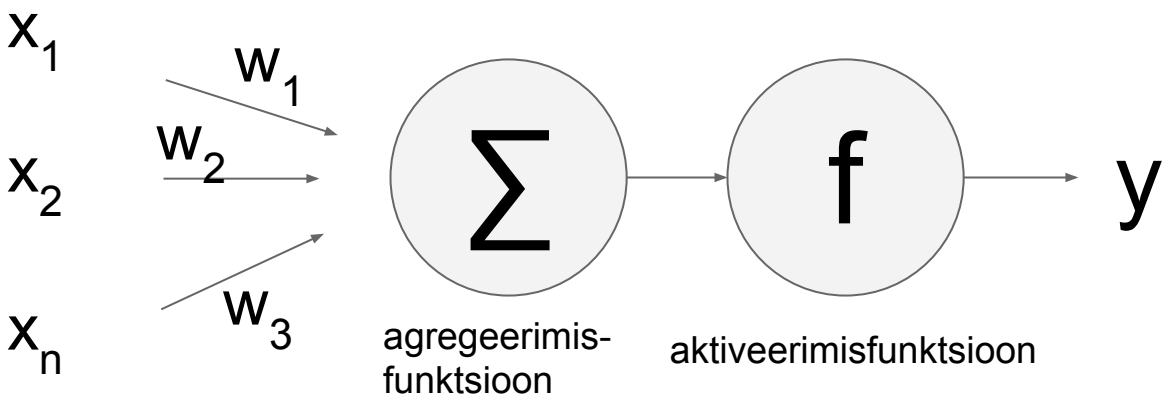
- Ei hooli süntaksist ega sõnade järjekorrast
  - talletab sõnavara esinemiste arvu igas dokumendis
    - oleme tutvunud masinõppe teemade alguses
  - annab tõenäosuse kontekstile

Tehisneurovõrgud

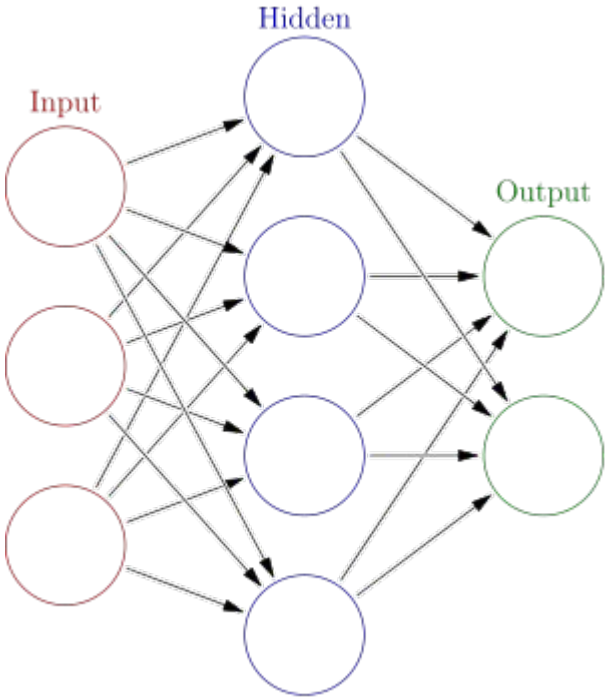
Artificial neural networks

Tehisneurovõrgud

- Imiteerivad bioloogilist närvisüsteemi neuronite tasandil
- Matemaatiline mudel



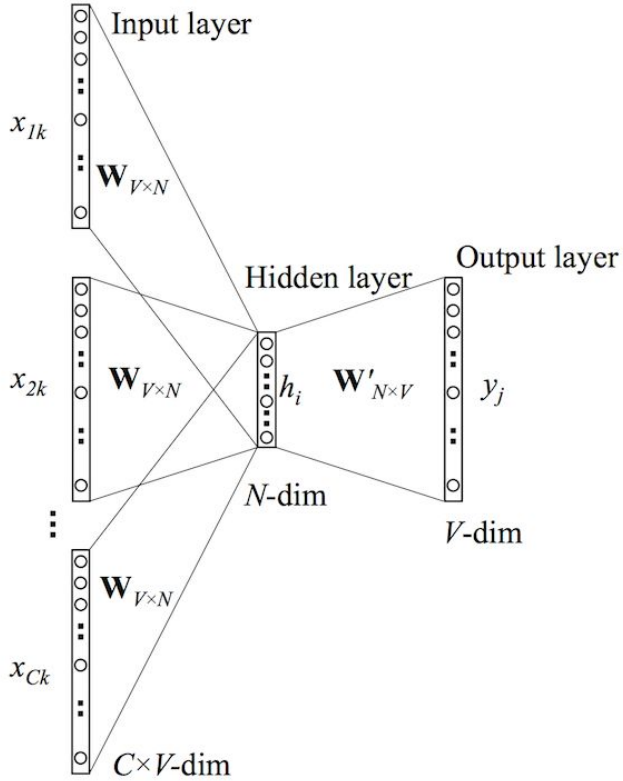
Tehisneurovõrgud



Word2Vec

- Tehisneurovõrgul baseeruv algoritm
- Treenib sõnadele kontekstipõhised vektorkujud
  - sarnase kontekstiga sõnad sarnaste vektorkujudega

Word2Vec arhitektuur



Word2Vec Pythonis

```
>>> from gensim.models.word2vec import Word2Vec
>>> sentences = [['koer', 'oli', 'ilus', '.'],
... ['kass', 'oli', 'ilus', '.'],
... ['koer', 'jalutas', 'toas', '.'],
... ['kass', 'jalutas', 'toas', '.']]
>>> model = Word2Vec(sentences,size=50,min_count=1,workers=1)
>>> model.most_similar("koer",topn=1)
[('jalutas', 0.12807472050189972)]
>>> model.save("minu_mudel.model")
>>> loaded_model = Word2Vec.load("minu_mudel.model")
```

Viiteid

http://www.cs.cornell.edu/courses/cs4740/2014sp/lectures/smoothing+backoff.pdf

https://web.stanford.edu/class/cs124/lec/languagemodeling.pdf