Sentiment Classification

Multi-Task-Learning und l1/l2-Regularisierung

Mirko Hering, Julia Kreutzer, Jasmin Schröck

21. Juli 2013

Inhaltsverzeichnis

Aufgabenstellung und Lösungsansatz
- Spezifikation revived

Aufgabenstellung

Ziel

- Gewinnung von Features, die für alle Kategorien der Testdaten aussagekräftig und bei der Klassifizierung nützlich sind
- Lernverfahren auf bewertete Produktrezensionen von Amazon.com anwenden, mit Hilfe der gewonnenen Features in positiv und negativ klassifizieren

Lösungsansatz

- Anwendung von Multitask-Learning mit verteilter l1/l2-Regularisierung zur Feature-Selektion
- Die Produktkategorien (books, dvd, electronics, kitchen) entsprechen den Tasks des Multi-Task-Learnings
- Alternativ zu Produktkategorien: Random Shards

Umsetzung

- Implementierung, Hadoop und Co.

Daten

Multi-Domain Sentiment Dataset (version 2.0)

- Englischsprachige Produktrezensionen von Amazon.com
- 4 Kategorien: Bücher, DVDs, Küchengeräte, Elektronik
- Rezensionen sind positiv und negativ gelabelt
- Preprocessed: Zählung von Unigrammen und Bigrammen
- für jede Kategorie 1000 negative und 1000 positive Rezensionen

Daten

Unsere Aufteilung:

- je 1200 Rezensionen für Training
- und je 400 für Test und Development

Methoden

Parameter:

- Lernrate η^t
- Epochenzahl t

  -diese wird durch die Laufzeit auf Hadoop begrenzt werden
- Gewichtsvektorinitialisierung v_0
- Auswahl der top k Features
- Anzahl/Daten in shards Z

  -4 Kategorien, also 4 Shards

Korpusformat

- Format einer Rezension:
  Kategorie feature:count feature:count (...) #label#:[positive|negative]
- Format des Korpus: eine Rezension pro Zeile

6 Korpora:

- Je ein Korpus mit allen Rezensionen einer Kategorie
- Plus ein Korpus mit allen Rezensionen (pooled - all)
- Plus ein Korpus mit Rezensionen aus allen Kategorien, jedoch nur so groß wie ein Korpus einer einzelnen Kategorie (pooled - small)

Klassenarchitektur

To do!

Klassenarchitektur

To do!

Hadoop - MT Learning

- Aufruf der jar-Datei mit Hadoop
- Angabe der Parameter (top k Features, Epochen, Kategorien)

- Innerhalb einer Epoche: Durchlauf der Phasen 1 & 2
- Abschließend: Selektion der top k Features

Hadoop - MT Learning

PHASE 1	PHASE 2
Input splits: key: category name value: review	**Input splits:** key: feature value: weight vector value
Mapper: creates one perceptron per category parses review trains perceptron on review	**Mapper:** returns feature-value pairs to reducer
Input: key: category name value: weight vector	**Input:** key: feature value: weight vector value
Reducer: gets all weight vectors of one category calculates average weight vector	**Reducer:** gets all values of one feature calculates l2 norm and average value

Hadoop - Random Shards

RANDOM SHARDS

Input splits:
key: category name
value: review

Mapper:
increasing counter;
when counter = max. number of shards:
reset to 0
replace key by counter value

Input:
key: random category name (a number)
value: review

Reducer:
saves each category in its own file

Parameter-Optimierung

Maß: Error-Rate

- Lernrate η^t: $10, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, \frac{1}{t}, exp, dec$

  exp: $1 * 0,85^{\frac{-t}{trainsetSize}}$

  dec: $\frac{1}{1 + \frac{t}{trainsetSize}}$
- Epochenzahl t: 1, 10, 20, 30
- Gewichtsvektorinitialisierung $v_0 = 0$
- Auswahl der top k Features:
  $k = 10, 100, 1.000, 2.000, 5.000, 10.000, 50.000$
- Anzahl der shards Z: 4

Parameter-Optimierung

weitere Experimente:

- Margin Perceptron (update if $y_i \langle x_i, w_z, t \rangle <= 1$)
- Word Net (Synsets für Unigramme)

- Jedoch keine Verbesserung :-(

Hadoop - Random Shards

RANDOM SHARDS

Input splits:
key: category name
value: review

Mapper:
increasing counter;
when counter = max. number of shards:
reset to 0
replace key by counter value

Input:
key: random category name (a number)
value: review

Reducer:
saves each category in its own file

Parameter-Optimierung

Maß: Error-Rate

- Lernrate η^t: $10, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, \frac{1}{t}, exp, dec$
- Epochenzahl t: 1, 10, 20, 30
- Auswahl der top k Features:
  $k = 10, 100, 1.000, 2.000, 5.000, 10.000, 50.000$

Optimale Parameter:

10 Epochen, Lernrate $= 10^{-2}$, top 5000 Features

Evaluationsergebnisse
- sprechende Zahlen

Wie gut klassifiziert unser regularisierter Multi-Task Perceptron...

- ... jede einzelne Kategorie?
- ... im Vergleich zu Single-Task Learning?
- ... im Vergleich zu random sharded Multi-Task Learning?

Baselines:

- Single-Task-Learning auf einzelnen Kategorien
- Single-Task-Learning auf allen Daten
- Multi-Task-Learning mit random shards

Single Task Learning

	independent	**small**	**all**
books	0,2275	0,3000	0,1925
dvd	0,2200	0,2750	0,2000
electronics	0,1825	0,2100	0,1525
kitchen	0,1475	0,2325	0,1425
average	0,1944	0,2544	0,1719

Multi Task Learning

	shards = tasks	**random shards**
books	0,2375	0,2400
dvd	0,2175	0,1900
electronics	0,1550	0,1500
kitchen	0.1450	0,1525
average	0,1888	0,1831

Vergleich

	average error rate	average vector length
ST independent	0,1944	63.213
ST small	0,2544	78.089
ST all	0,1719	194.627
MT shards = tasks	0,1888	5.000
MT random shards	0,1831	5.000

Unseen Corpora

Erstellung zusätzlicher Testcorpora:

- Herunterladen von Amazon-Kundenrezensionen: http://www.esuli.it/fossil/repo/amazonReviewsDownloader/index
- eigenes Python-Skript zum Formatieren

Korpora haben unterschiedliche Längen und sind nicht ausgeglichen:

Kategorie	Rezensionen	Anteil positiv
gardening	1865	0,8
outdoor	916	0,86
snacks	208	0,95
organicfood	97211	0,88

Unseen Corpora

	average error rate on unseen test corpora
ST independent	0,2611
books	0,4164
dvd	0,2997
electornics	0,1873
kitchen	0,1410
ST small	0,2253
ST all	0,1781
MT shards = tasks	0,1616
MT random shards	0,1776

Features

Top 10 Features nach Multi Task Learning mit Random Shards:

- *kagan* : −0.4
- *worst* : −0.8000000000000006
- *poor* : −0.4700000000000001
- *boring* : −0.7000000000000004
- $the_b est$: 0.5200000000000002
- (*pp.* : 0.58
- *excellent* : 0.6800000000000004
- *bad* : −0.38
- *disappointed* : −0.5000000000000002
- *wonderful* : 0.5100000000000002

Demo

- check your sentiment!

Demo

Rezensionen zum Buch „Software Project Survival Guide“:

5 Sterne:

This book is especially helpful to those either: 1) new to managing projects 2) have never been formally trained in managing projects 3) are more on the business side of the fence and need insight into the software development process. I highly recommend this book if you fall into any of the above categories. I also recommend buying this book for any non-technical bosses.

2 Sterne:

Here is änother Project Management Bookör "Do This, Do That Book". It is just about "doing something in order step". It has a wrong name; it is not a survival book. Survival means The continuation of life or existenceäccording to Webster Dictionary and I can assure you that you cannot survive by just following these steps and actions. Your most important ve precious resources are your project team. If you are an experienced project manager, this book is totally a vast of time. There is nothing new inside. If you are new to project management, you may have some idea about agile-driven classical project management, but you should know that it is just a one of many sides of project management. Maybe, I said maybe, you can use this book as a checklist.

Fazit

- our sentiment

Fazit

Ergebnisse:

- Multi-Task Learning besser als Single-Task Learning auf einzelnen Kategorien (independent)
- Random shards besser als taskspezifische shards
- beste Ergebnisse beim Single-Task-Training auf allen Daten (pooled), aber nur um 0.01 besser als Multi-Task-Perceptron
- aber: Größe der Gewichtsvektoren, d.h. Anzahl der Features bei Multi-Task geringer bei vergleichbaren Ergebnissen
- Unseen Corpora: Multi-Task besser als Single-Task

- Feature Selektion & paralleles Lernen sind nützlich, vor allem bei unbekannten Kategorien führt es zu einer besseren Klassifizierung der Daten

Fazit

Verbesserungsvorschläge & weitere Ideen:

- Korpora: größer, andere Tasks, z.B. Filme
- Features: „mehr“ Information (semantisch, syntaktisch, ...)
- Variation der Anzahl der random shards
- Vergleich mit „average“-Single-Task-Perceptron
- Verwendung alternativer MR-Frameworks, die besser für iterative Prozesse geeignet sind, z.B. Spark

Referenzen
- Inspiration und Anleitung

Referenzen

- **Joint Feature Selection in Distributed Stochastic Learning for Large-Scale Discriminative Training in SMT**
  (P.Simianer, S. Riezler, C. Dyer. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)).
- **Domain Adaptation for Sentiment Classification**
  (John Blitzer, Mark Dredze, Fernando Pereira. Biographies, Bollywood, Boom-boxes and Blenders. Association of Computational Linguistics (ACL), 2007).
- **Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty**
  (Yoshimasa Tsuruoka, Jun'ichi Tsujii, Sophia Ananiadou. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL, 2009)).
- **Learning with Kernels - Support Vector Machines, Regularization, Optimization, and Beyond**
  (Bernhard Scholkopf, Alexander J. Smola. The MIT Press, 2002).