

# Sentiment Classification

## Multi-Task-Learning und $l_1/l_2$ -Regularisierung

Mirko Hering, Julia Kreutzer, Jasmin Schröck

21. Juli 2013

# Inhaltsverzeichnis

Aufgabenstellung und Lösungsansatz

Umsetzung

Evaluation

Demo

Fazit

Referenzen

# 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  $l_1/l_2$ -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  $\eta^t$
- ▶ 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
<b>Input splits:</b> key: category name value: review	<b>Input splits:</b> key: feature value: weight vector value
<b>Mapper:</b> creates one perceptron per category parses review trains perceptron on review	<b>Mapper:</b> returns feature-value pairs to reducer
<b>Input:</b> key: category name value: weight vector	<b>Input:</b> key: feature value: weight vector value
<b>Reducer:</b> gets all weight vectors of one category calculates average weight vector	<b>Reducer:</b> 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  $\eta^t$ :  $10, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, \frac{1}{t}, exp, dec$

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  $y_i \langle x_i, w_z, t \rangle \leq 1$ )
- ▶ 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  $\eta^t$ : 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

	<b>independent</b>	<b>small</b>	<b>all</b>
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

	<b>shards = tasks</b>	<b>random shards</b>
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













