Programming Assignment 2- NLP

Xiaocong Li

# 3.1 Softmax

#### q1\_softmax.py

Running basic tests...

[0.26894142 0.73105858]

[[0.26894142 0.73105858]

[0.26894142 0.73105858]]

[[0.73105858 0.26894142]]

You should verify these results!

# 3.2 Neural Network Basics

#### 1.q2\_sigmoid.py

Running basic tests...

[[0.73105858 0.88079708]

[0.26894142 0.11920292]]

[[0.19661193 0.10499359]

[0.19661193 0.10499359]]

You should verify these results!

#### 2.q2\_gradcheck.py

Running sanity checks...

Gradient check passed!

Gradient check passed!

Gradient check passed!

#### 3.q2\_neural.py

Running sanity check...

Gradient check passed!

# 3.3 Word2Vec

#### 1.q3\_word2vec.py

Testing normalizeRows...

[[0.6 0.8 ]

[0.4472136 0.89442719]]

==== Gradient check for skip-gram ====

Gradient check passed!

Gradient check passed!

==== Gradient check for CBOW ====

Gradient check passed!

Gradient check passed!

=== Results ===

(11.16610900153398, array([[ 0. , 0. , 0. ],

[ 0. , 0. , 0. ],

[-1.26947339, -1.36873189, 2.45158957],

[ 0. , 0. , 0. ],

[ 0. , 0. , 0. ]]), array([[-0.41045956, 0.18834851, 1.43272264],

[ 0.38202831, -0.17530219, -1.33348241],

[ 0.07009355, -0.03216399, -0.24466386],

[ 0.09472154, -0.04346509, -0.33062865],

[-0.13638384, 0.06258276, 0.47605228]]))

(6.3296858833189935, array([[ 0. , 0. , 0. ],

[ 0. , 0. , 0. ],

[-1.46912698, -1.55354006, 0.24305369],

[ 0. , 0. , 0. ],

[ 0. , 0. , 0. ]]), array([[-0.11265089, 0.05169237, 0.39321163],

[ 0.07307589, -0.0335325 , -0.25507379],

[-0.11382109, 0.05222934, 0.39729628],

[-0.21068407, 0.09667707, 0.73539969],

[-0.32248118, 0.14797767, 1.1256312 ]]))

(0.798995801090665, array([[ 0.23330542, -0.51643128, -0.8281311 ],

[ 0.11665271, -0.25821564, -0.41406555],

[ 0.11665271, -0.25821564, -0.41406555],

[ 0. , 0. , 0. ],

[ 0. , 0. , 0. ]]), array([[ 0.80954933, 0.21962514, -0.54095764],

[-0.03556575, -0.00964874, 0.02376577],

[-0.13016109, -0.0353118 , 0.08697634],

[-0.1650812 , -0.04478539, 0.11031068],

[-0.47874129, -0.1298792 , 0.31990485]]))

(2.944667018896025, array([[-0.95433581, -1.83919201, -0.53425666],

[-0.47716791, -0.919596 , -0.26712833],

[-0.47716791, -0.919596 , -0.26712833],

[ 0. , 0. , 0. ],

[ 0. , 0. , 0. ]]), array([[ 0.21992784, 0.0596649 , -0.14696034],

[-0.34456262, -0.09347746, 0.23024388],

[ 0. , 0. , 0. ],

[-0.86318467, -0.23417603, 0.57679789],

[-1.18374504, -0.32114184, 0.79100296]]))

#### 2.q3\_sgd.py

Running sanity checks...

iter 100: 0.004578

iter 200: 0.004353

iter 300: 0.004136

iter 400: 0.003929

iter 500: 0.003733

iter 600: 0.003546

iter 700: 0.003369

iter 800: 0.003200

iter 900: 0.003040

iter 1000: 0.002888

test 1 result: 8.41483678608e-10

iter 100: 0.000000

iter 200: 0.000000

iter 300: 0.000000

iter 400: 0.000000

iter 500: 0.000000

iter 600: 0.000000

iter 700: 0.000000

iter 800: 0.000000

iter 900: 0.000000

iter 1000: 0.000000

test 2 result: 0.0

iter 100: 0.041205

iter 200: 0.039181

iter 300: 0.037222

iter 400: 0.035361

iter 500: 0.033593

iter 600: 0.031913

iter 700: 0.030318

iter 800: 0.028802

iter 900: 0.027362

iter 1000: 0.025994

test 3 result: -2.52445103582e-09

#### 3.q3\_run,py

iter 10: 23.261994

iter 20: 23.037415

iter 30: 23.025078

iter 40: 22.933648

iter 50: 23.186432

iter 60: 23.413402

iter 70: 23.563869

iter 80: 23.354002

iter 90: 23.322372

iter 100: 23.139831

iter 110: 23.110592

iter 120: 23.028747

iter 130: 23.000204

iter 140: 23.011909

iter 150: 22.991143

iter 160: 22.762088

iter 170: 22.721922

iter 180: 22.701778

iter 190: 22.573827

iter 200: 22.569413

.

.

.

iter 39900: 9.517481

iter 39910: 9.562241

iter 39920: 9.574561

iter 39930: 9.656687

iter 39940: 9.719975

iter 39950: 9.739466

iter 39960: 9.723659

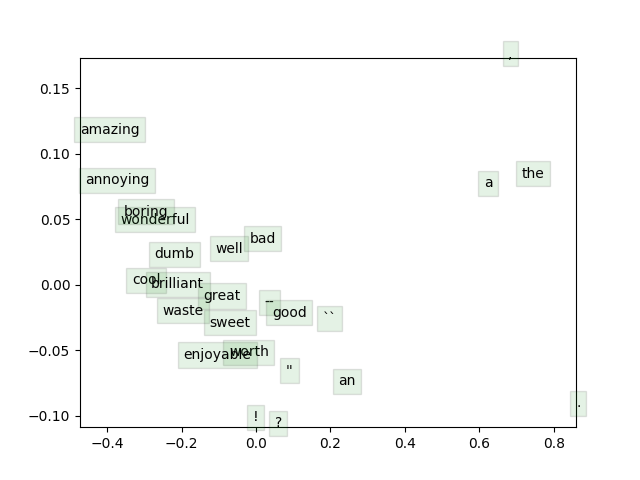
iter 39970: 9.661740

iter 39980: 9.629146

iter 39990: 9.584816

iter 40000: 9.609267

sanity check: cost at convergence should be around or below 10



***This picture shows the relativity between different words, it's obvious that some words of the similar sentiment stick together. For example, “well”, “good” and “great”; however, there are examples that words of opposite sentiments are close to each other, such as “boring” and “wonderful”. Therefore this could be further trained.***

# 3.4 Sentiment Analysis

#### 1.q4\_softmaxreg.py

==== Gradient check for softmax regression ====

Gradient check passed!

=== Results ===

(1.9090602153623348, array([[ 0.14080648, 0.01168914, 0.06119674, 0.14637578, 0.0711719 ],

[-0.10938236, -0.07499234, 0.00852349, 0.09048024, 0.02329987],

[ 0.03037895, -0.07003952, 0.00510437, -0.00256201, -0.0817831 ],

[ 0.11555212, 0.23390651, -0.0305698 , -0.05599756, -0.0317812 ],

[ 0.16376864, 0.07456775, -0.06367415, 0.00590286, -0.24757671],

[ 0.01010793, -0.03069943, 0.28680727, 0.11856594, 0.09281364],

[-0.14668533, -0.16448039, -0.23519681, -0.14252623, 0.08534886],

[ 0.10992796, 0.06278405, -0.03904405, 0.10979171, -0.11455541],

[-0.25308316, 0.1756363 , 0.07432147, 0.09125918, 0.01989027],

[-0.05468889, -0.00213282, 0.10967463, -0.06105177, 0.10419883]]), array([0, 0, 0, 0, 0, 0, 0, 0, 0, 1]))

#### 2.q4\_sentiment.py

=== Recap ===

Reg Train Dev

0.000000E+00 29.705056 30.699364

1.000000E-06 29.658240 30.699364

3.000000E-06 29.541199 30.790191

1.000000E-05 29.623127 30.790191

3.000000E-05 29.494382 30.790191

1.000000E-04 29.108146 30.608538

3.000000E-04 28.675094 28.882834

1.000000E-03 27.352528 25.522252

3.000000E-03 27.235487 25.522252

1.000000E-02 27.247191 25.522252

Best regularization value: 3.000000E-06

Test accuracy (%): 28.235294

***In this case, best regularization value is 1.000000E-05.***

