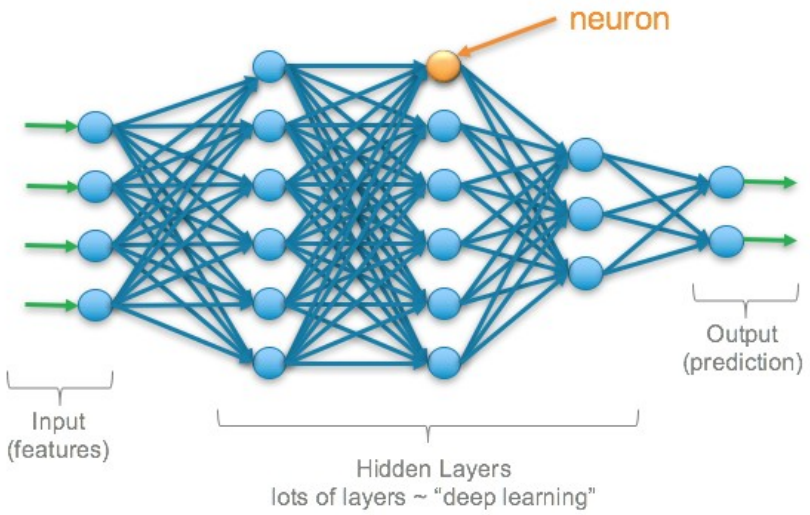
סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Deep Learning and NLP



הקדמה

בינה מלאכותית: תחום רחב מאוד במדעי המחשב.
בתוך התחום הזה נמצאים התחומים הבאים:

- Machine learning: למידת מכונה, שהרעיון הכללי בו הוא לבנות מודל שיכול לחזות תוצאות בהתבסס על מידע. ככל שיש יותר מידע, התוצאות יותר טובות.
- deep learning: למידה עמוקה, תחחום בתוך למידת מכונה. זוהי למידה היררכית. בדרך כלל שאומרים למידה עמוקה מתכוונים לרשת נויירונים.
- NLP: עיבוד שפות טבעיות, ישנם דרכים לעבד שפות טבעיות עם DL ויש בלי.

*TensorFlow: ספרייה בפייתון שמשתמשים בה ל deep learning והיא open source.

דרישות כדי ש DL יעבוד כמו שצריך:
1. הרבה מידע.
2. כח חישוב.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

NLP Introduction

חשוב להבין שמחשבים לא מבינים שפות טבעיות. הם לומדים לפתור משימות ספציפיות או סט של משימות בהינתן DATASET מתאים.
הבנת שפות טביות יכולה להיות מבלבלת. לדוגמה:

- "I saw Alice with my telescope."
- "The trophy would not fit in the brown suitcase because it was too big"

האתגר העיקרי שאנו מנסים לפתור הוא הרב משמעותיות.

Natural Language Tool-Kit (NLTK)

NLTK- כלי עבודה עם שפות טבעיות: tokenization, stemming, POS, preprocessing in general...
להתקנה: pip install nltk

Tokenization

כאשר אנו מעבדים משפט, הפעולה הכי בסיסית שאנו עושים היא טוקניזישן: הפרדת המשפט לפי טוקנז (מילים).

```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
my_text = "Where is St. Paul located?  I don't seem to find it. It isn't in my map."
# print split the sentence by backspace
print(my_text.split(" "))
# print the sentence after word tokenize
print(word_tokenize(my_text))
# print the sentence after sentence tokenize
print(sent_tokenize(my_text))
```

```
['Where', 'is', 'St.', 'Paul', 'located?', 'I', "don't", 'seem', 'to', 'find', 'it.', 'It', "isn't", 'in', 'my', 'map.']
['Where', 'is 'St.', 'Paul', 'located', '?', 'I', 'do', "n't", 'seem', 'to', 'find', 'it', '.', 'It', 'is', "n't", 'in', 'my', 'map', '.']
['Where is St. Paul located?', "I don't seem to find it.", "It isn't in my map."]
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Stemming

התייחסות זהה למילים דומות.

פעמים רבות אנו רוצים להתייחס למילים קרובות כזהות, למשל המילים:

walking, walk, walks

אנו יכולים להתיחס אליהם כאל walk.

תחילה עלינו לבצע TOKENIZE של המשפט לאחר מכן אנו יכולים לעשות STEM על המילים.

```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import PorterStemmer
# create PorterStemmer object
ps = PorterStemmer()
my_text = "Whoever eats many cookies is regretting doing so"
stemmed_sentence = []
for word in word_tokenize(my_text):
    stemmed_sentence.append(ps.stem(word))
print(stemmed_sentence)
```

['whoever', 'eat', 'mani', 'cooki', 'is', 'regret', 'do', 'so']

Part of speech tagging(POS)

חלקי דיבור הוא מודל הנותן לכל מילה במשפט פירוש תחבירי לחלקי הדיבור הנכונים במשפט.

למשל: פועל, שם עצם, שם תואר ועוד.

Noun: boy, John, birthday

Verb: went, ate, is

Pronoun: it, she, ours

Adjective: big, smart, five

Adverb: well, quickly

כאשר אנו מעבדים שפה טבעית, לדעת חלקי דיבר יכול להיות דבר מאוד מועיל. אם למשל אנחנו מחפשים פקודות הנאמרות לאפליקציה שלנו, אז אנחנו נחפש פעלים ואם אנחנו מחפשים שמות (כגון: אנשים, כתובות, ארגונים, וכדומה) החיפוש יהיה לפי מילים שהן שמות עצם.

I saw the **show**

Show me where to go

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
my_text = "Whoever eats many cookies is regretting doing so"
my_tokenize_text = word_tokenize(my_text)
print(nltk.pos_tag(my_tokenize_text))
# for pprint all the tagset
# nltk.help.upenn_tagset()
```

[('Whoever', 'NNP'), ('eats', 'VBZ'), ('many', 'JJ'), ('cookies', 'NNS'), ('is', 'VBZ'), ('regretting', 'VBG'), ('doing', 'VBG'), ('so', 'RB')]

Lemmatization (קיבוץ צורות נטייה שונות למילת יסוד אחת)

זהו כלי פיענוח שלוקח יותר זמן כי אין לו איזשהו כלל שהוא עובד לפיו.

הוא חייב לקבל כקלט את ה POS.

Lemmatization מאוד דומה ל stemming אבל היא יותר מורכבת (איטית יותר, אך בעלת תוצאות טובות יותר).

דוגמה לליימטיזציה של מילה:

"going" and "went" are actually lemitized to the same lemma: "go"

הלמיטיזר חייב לדעת את ה POS של המילים על מנת לבצע את עבודתו.

ה- WordNetLemmatizer ב- NLTK משתמש ברשימה קצרה יותר של חלק מתגי הדיבור מאשר Penn TreeBank, (לכן אנו משתמשים בפונקציה קצרה כדי לבצע את ההמרה).

```
from nltk.corpus import wordnet as wn
def is_noun(tag):
    return tag in ['NN', 'NNS', 'NNP', 'NNPS']
def is_verb(tag):
    return tag in ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']
def is_adverb(tag):
    return tag in ['RB', 'RBR', 'RBS']
def is_adjective(tag):
    return tag in ['JJ', 'JJR', 'JJS']
def penn2wn(tag):
    if is_adjective(tag):
        return wn.ADJ
    elif is_noun(tag):
        return wn.NOUN
    elif is_adverb(tag):
        return wn.ADV
    elif is_verb(tag):
        return wn.VERB
    return wn.NOUN
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

```
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
# create WordNetLemmatizer object
lzr = WordNetLemmatizer()
my_text = "Whoever eats many cookies is regretting doing so"
lemed = []
for (word, pos) in nltk.pos_tag(word_tokenize(my_text)):
  lemed.append(lzr.lemmatize(word, penn2wn(pos)))
print(lemed)
```

['Whoever', 'eat', 'many', 'cooky', 'be', 'regret', 'do', 'so']

Chunking

שיטה לחתוך משפט או כמה משפטים לנתחים או קבוצות שונות. צ'אנקינג משתמש בביטויים רגולריים בחלקי הדיבור.

השיטה: אנחנו צרכים להשתמש בביטוי רגולרי עם הסימנים הבאים:

סימן שאלה: 0 מופעים או מופע 1.

כוכבית: 0 מופעים או יותר.

אם לא כתוב כלום מופע אחד בדיוק.

פלוס(+): מופע אחד או יותר.

דוגמה לביטוי רגולרי:

NP: {<DT>?<JJ>*<NN>}

<DT>: מילה כלשהי, לדוגמה a או the יופיע 0 פעמים או פעם אחת.

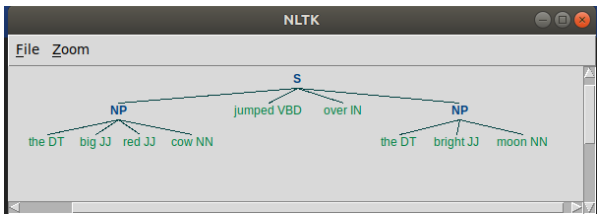
<JJ>: תואר שיופיע 0 פעמים או יותר.

<NN>: שם עצם שיופיע בדיוק פעם אחת.

הדוגמה הזאת מתארת את המשפט: the nice big boy.

```
import nltk
from nltk import tkinter
my_text = "the big red cow jumped over the bright moon"
tagged = nltk.pos_tag(nltk.tokenize.word_tokenize(my_text))
grammar = "NP: {<DT>?<JJ>*<NN>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(tagged)
print(result)
result.draw()
```

```
(S
  (NP the/DT big/JJ red/JJ cow/NN)
  jumped/VBD
  over/IN
  (NP the/DT bright/JJ moon/NN))
```

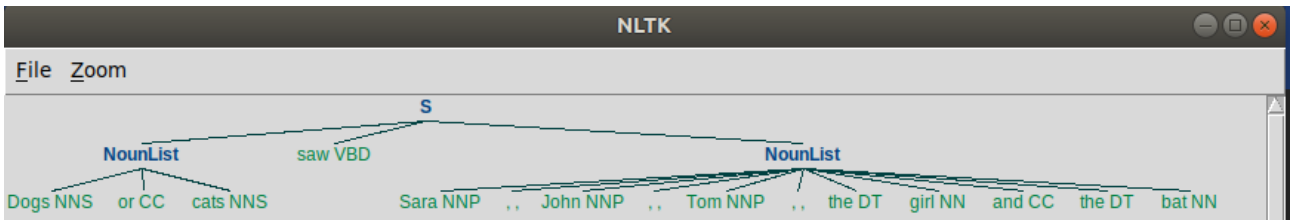


סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

```
# my_text = "the big red cow jumped over the bright moon"
my_text = "Dogs or cats saw Sara, John, Tom, the girl and the bat"
tagged = nltk.pos_tag(nltk.tokenize.word_tokenize(my_text))
# grammar = "NP: {<DT>?<JJ>*<NN>}"
grammar = "NounList: {(<DT>?<NN.?><,>?)+ <CC><DT>?<NN.?>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(tagged)
print(result)
result.draw()
```



Bi-grams

בעיבוד שפות טבעיות, אנו הרבה פעמים מתענינים בצמדי מילים. לדוגמה:

"I did it you did it you did it"

We get the following bi-grams count:

[((I, did), 1), ((did, it), 3), ((it, you), 2), ((you, did), 2)]

דבר זה יכול להיות שימושי בחילוץ תכונות, יצירת סיפור וכדומה.

Tri-grams הם כמו בי-גרמס גם עם שלשות של מילים. באופן כללי, אנו אומרים n-grams.

```
import nltk
text = "It is a simple text this, this is a simple text, is it simple?"
my_list = list(nltk.ngrams(nltk.word_tokenize(text), 3))
print(my_list)
```

[('It', 'is', 'a'), ('is', 'a', 'simple'), ('a', 'simple', 'text'), ('simple', 'text', 'this'), ('text', 'this', ','), ('this', ',', 'this'), (',', 'this', 'is'), ('this', 'is', 'a'), ('is', 'a', 'simple'), ('a', 'simple', 'text'), ('simple', 'text', ','), ('text', ',', 'is'), (',', 'is', 'it'), ('is', 'it', 'simple'), ('it', 'simple', '?')]

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

N-grams – Text Generation

בהינתן טקסט, אנו רוצים לייצר טקסט חדש הדומה לטקסט שניתן.

אנו הולכים להשתמש באלגוריתם הבא:

- נפריד את המילים ל tri-grams.
- נתחיל עם שני מילים, לאחר מכן כל פעם נדגום את המילה הבאה, לפי שתי המילים הקודמות, על פי פילוג הנתונים.

```
import nltk
import urllib
from random import randint
paragraph_len = 100
# read https://s3.amazonaws.com/text-datasets/nietzsche.txt text to all_text
all_text = urllib.request.urlopen("https://s3.amazonaws.com/text-datasets/nietzsche.txt").read().decode("utf-8")
# split the text by tokens
tokens = nltk.word_tokenize(all_text)
# split the tokens into tri-grams
my_grams = list(nltk.ngrams(tokens, 3))
# starts with to words
sentence = ["It", "is"]
# run 100 times
for i in range(paragraph_len):
    options = []
    # go over each tri-gram
    for trig in my_grams:
        # if the words in position one and two equal to the last to word in the sentence add to third word to options
        if trig[0].lower() == sentence[len(sentence)-2].lower() and trig[1].lower() == sentence[len(sentence)-1].lower():
            options.append(trig[2])
    # if options list include at least one word
    if len(options) > 0:
        # add to sentence random word from options
        sentence.append(options[randint(0, len(options)-1)])
# join -the opposite action to word_tokenize
print(" ".join(sentence))
```

It is desired to obtain pleasure or avoid pain . In the highest altitudes of his successful executive instruments , and are seldom at peace -- such must pass over to the _operari_ -- in these later French skeptics , as a power over others and nothing for the tenets of his fatherland his highest end is attained : so that at first , in particular as he now , in the case of tooth ache occasions ! Therefore when injury is not to mention Italy , which , as among the learned jargon is : because someone truly and openly is

אם נתעמק בסיפור שקיבלנו נשים לב שהוא לא כל כך הגיוני. בהמשך נלמד כיצד לשפר את התוצאה הנ"ל באמצעות רשת נוירונים.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Context Free-Grammar (CFG)

כפי שלמדנו באוטומטים ושפות פורמליות. CFG מורכב מארבעה רכיבים:

T: terminal vocabulary (the words of the language being defined)

V: non-terminal vocabulary

P: a set of productions of the form a->b, (a is a non-terminal and b is a sequence of one or more symbols from T U V)

S: the start symbol (member of V)

```
import nltk
# create a grammar
grammar1 = nltk.CFG.fromstring("""
S -> NP VP
VP -> V NP | V NP PP
PP -> P NP
V -> "saw" | "ate" | "walked"
NP -> "John" | "Mary" | "Bob" | Det N | Det N PP
Det -> "a" | "an" | "the" | "my"
N -> "man" | "dog" | "cat" | "telescope" | "park"
P -> "in" | "on" | "by" | "with" """)
my_text = "Mary saw Bob"
tokens = nltk.word_tokenize(my_text)
rd_parser = nltk.RecursiveDescentParser(grammar1)
print(list(rd_parser.parse(tokens))[0])
```

(S (NP Mary) (VP (V saw) (NP Bob)))

Ambiguity with CFG

ניתן לפתור דו משמעות:

```
import nltk
groucho_grammar = nltk.CFG.fromstring("""
 S -> NP VP
 PP -> P NP
 NP -> Det N | Det N PP | 'I'
 VP -> V NP | VP PP
 Det -> 'an' | 'my'
 N -> 'elephant' | 'pajamas'
 V -> 'shot'
 P -> 'in'
 """)
sent = ['I', 'shot', 'an', 'elephant', 'in', 'my', 'pajamas']
parser = nltk.ChartParser(groucho_grammar)
for tree in parser.parse(sent):


    print(tree) #tree.draw()
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Probabilistic Context Free Grammar (*PCFG*)

ניתן להוסיף לכל משתנה גזירה את ההסתברות שהוא הוא ייגזר

```
import nltk
grammar = nltk.PCFG.fromstring("""
    S    -> NP VP            [1.0]
    VP   -> TV NP            [0.4]
    VP   -> IV               [0.3]
    VP   -> DatV NP NP       [0.3]
    TV   -> 'saw'            [1.0]
    IV   -> 'ate'            [1.0]
    DatV -> 'gave'           [1.0]
    NP   -> 'telescopes'     [0.8]
    NP   -> 'Jack'           [0.2] """)
viterbi_parser = nltk.ViterbiParser(grammar)
for tree in viterbi_parser.parse(['Jack', 'saw', 'telescopes']):
    print(tree)
```

```
(S (NP Jack) (VP (TV saw) (NP telescopes))) (p= 0.064)
```

CoreNLP

היא ספרייה הנכתבה על ידי סטנפורד בשפת ג'אבה אך ניתן להשתמש בה בשפות נוספות כגון פייתון. זוהי ספרייה יותר גדולה מ NLTK, הכוללת למשל, דקדוק מובנה גדול.

קישור לגיטאהב: https://github.com/dasmith/stanford-corenlp-python

Sentiment Analysis

המון פעמים אנו רוצים לדעת האם משפט מסוים הוא שלילי או חיובי. לדוגמה: יש לנו המון תגובות על מוצר כלשהו, ואנו רוצים לחלק אותן לתגובות חיוביות ושליליות.

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
# nltk.download('vader_lexicon')
sna = SentimentIntensityAnalyzer()
print(sna.polarity_scores("The movie was great!"))
print(sna.polarity_scores("I liked the book, especially the ending."))
print(sna.polarity_scores("The staff were nice, but the food was terrible.")
```

```
{'neg': 0.0, 'neu': 0.406, 'pos': 0.594, 'compound': 0.6588}
{'neg': 0.0, 'neu': 0.641, 'pos': 0.359, 'compound': 0.4215}
{'neg': 0.318, 'neu': 0.536, 'pos': 0.146, 'compound': -0.5023}
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Tensor-Flow (1.x)

constants

יצירת קבוע בטנסור:

```
import tensorflow as tf
# creating tensorflow constants
a = tf.constant(3)
print(a)
b = tf.constant(4)
print(b)
c = a*b
print(c)
# to get the real value in the constants need to run a session
sess = tf.Session()
print(sess.run(a))
print(sess.run(c))
```

```
Tensor("Const:0", shape= (), dtype= int32)
Tensor("Const_1:0", shape= (), dtype= int32)
Tensor("mul:0", shape= (), dtype= int32)
3
12
```

Variables

במהלך הסשן טנסטר פלאו יכול לשנות את ערכי המשתנים, בניגוד לקבועים.

בנוסף בגלל שמדובר במשתנים, לפני שמגדירים אותם ב Session צריך לאתחל אותם על ידי הפקודה: sess.run(tf.global_variables_initializer())

יצירת משתנה בטנסור:

```
import tensorflow as tf
# creating tensorflow variables
var1 = tf.Variable(3)
print(var1)
var2 = tf.Variable(4)
c2 = var1 * var2
print(c2)
# to get the real value in the variables need to run a session
sess = tf.Session()
# the global variables need to initialize in the session
sess.run(tf.global_variables_initializer())
print(sess.run(var1))
print(sess.run(c2))
```

```
<tf.Variable 'Variable:0' shape= () dtype= int32_ref>
Tensor("mul:0", shape= (), dtype= int32)
3
12
```

*הפקודה Session נועדה למעשה לאפשר לנו לרוץ על הגרף שנוצר.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

simple counting program

```
import tensorflow as tf
# creating tensorflow variable
x =  tf.Variable(1)
# creating tensorflow constant
step =  tf.constant(2)
update =  tf.assign(x, x+ step)
sess =  tf.Session()
sess.run(tf.global_variables_initializer())
for i in range(4):
    print(sess.run(update))
print(sess.run(update))
print(sess.run(x))
print(sess.run(x))
sess.run(tf.global_variables_initializer())
print(sess.run(x))
```

```
3
5
7
9
11
11
11
1
```

Data, Model, Loss, Algorithm

- data: קבוצה של ערכי X והערכים שלהם Y.

לדוגמה: דאטה של משפטים שמתוכם אנו רוצים לדעת אילו משפטים חיוביים ואילו שליליים אזי X יהיו המשפטים ו Y יהיה חיובי או שלילי.

- model function: הפונקציה שבה אנו משתמשים על מנת לקבל את ה-Y מה-X. (אנו נשתמש בפונקציה על מנת לחזות את ערכי ה-Y, כאשר אין לנו אותם, או בשלב בחינת המודל).

לדוגמה: בהינתן משפט הפונקציה תנחש אם המשפט חיובי או שלילי.

- loss function: הפונקציה שקובעת את השגיאה שברצוננו למזער.
- לדוגמה: אם אנחנו רוצים לנחש מחיר של בית בהינתן הגודל שלו, הפונקציה של ה model תחשב מה המחיר והפונקציה loss תחזיר את הקשר בין המחיר האמיתי למחיר שניבאנו.
- algorithm: האלגוריתם שאנו נשתמש בו על מנת למצוא את השגיאה המינימלית.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות
נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

תוכנית שבודקת בכמה מספר הוכפל

- Data:
  - x's: [[7.01], [3.02], [4.99], [8.]]
  - y's: [[14.01], [6.01], [10.], [16.04]
- Model / hypothesis:
  - h(x) = wx (w is a weight we would like to learn)
- Loss: Mean squared error
  - $\frac{1}{m}\sum_{i=1}^{m}(h(x)-y)^2$
- Algorithm: gradient descent
  - Gradient of loss is: $\frac{2}{m}\sum_{i=1}^{m}(wx-y)x$
  - We will use alpha of 0.001

```
import tensorflow as tf
# data
x = tf.constant([7.01, 3.02, 4.99, 8.])
y_ = tf.constant([14.01, 6.01, 10., 16.04])
# creating tensorflow variables
w = tf.Variable(0.)  # note the dot
# model function
y = w*x
# loss function
loss = tf.reduce_mean(tf.pow(y - y_, 2))
# algorithm
update = tf.train.GradientDescentOptimizer(0.0001).minimize(loss)
# run a session
sess = tf.Session()
# initialize global variables
sess.run(tf.global_variables_initializer())
# runs 1000 iterations
for _ in range(0, 1000):
    sess.run(update)
# print the results
print(sess.run(w))
```

2.0005188

*הערה: בדרך כלל נעדיף שלא לשים את הדאטה במשתנים קבועים שאי אפשר לשנות ונשים אותם ב placeholder. שאומר במילים אחרות בהמשך נשים שם את הדאטה (שומר מקום).

שמירת מודל

לאמן datasets גדולים לוקח המון זמן, נרצה שנוכל להשתמש במשקלים שלמדנו כבר: על מנת לחזות דברים חדשים, או לחדש את האימונים.
שיטה טובה היא לשמור בכל עדכוני X.
על מנת לשמור:

```
saver = tf.train.Saver()
saver.save(sess, filename)
saver.restore(sess, filename)
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות
נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

TensorBoard
כלי גרפי המספק ויזואליזציה וכלי עבודה הדרושים ללמידת מכונה: מעקב והמחשה של מדדים כגון loss and accuracy.

```
import tensorflow as tf
x = tf.placeholder(tf.float32, [None, 1])
y_ = tf.placeholder(tf.float32, [None, 1])
w = tf.Variable(0.)
y = w * x
loss = tf.reduce_mean(tf.pow(y - y_, 2))
update = tf.train.GradientDescentOptimizer(0.0001).minimize(loss)
msum = tf.summary.scalar('m', w)
losssum = tf.summary.scalar('loss', loss)
merged = tf.summary.merge_all()
sess = tf.Session()
# the path to the log file
file_writer = tf.summary.FileWriter('./my_graph', sess.graph)
sess.run(tf.global_variables_initializer())
data_dict = {x: [[7.01], [3.02], [4.99], [8.]], y_: [[14.01], [6.01], [10.], [16.04]]}
for i in range(0, 1000):
    [_, curr_sammary] = sess.run([update, merged], feed_dict= data_dict)
    file_writer.add_summary(curr_sammary, i)
file_writer.close()
print(sess.run(w))
```

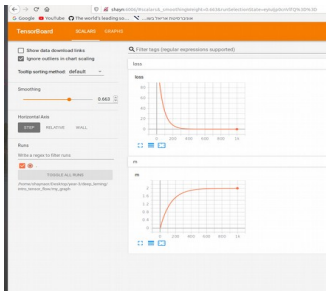
```
# sess.graph contains the graph definition; that enables the Graph Visualizer.

file_writer = tf.summary.FileWriter('/path/to/logs', sess.graph)

tensorboard --logdir path/to/logs
```

לדוגמה:

```
tensorboard --logdir ~/Desktop/year-3/deep_lerning/intro_tensor_flow/my_graph
```



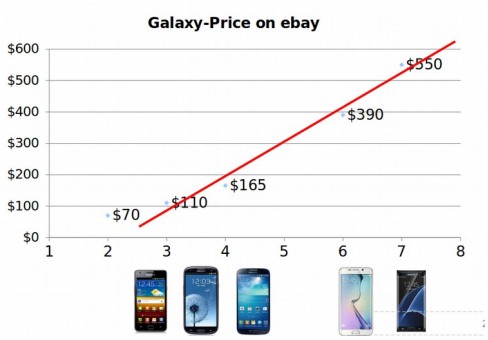
סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Linear Regression

רגרסיה לינארית היא הסוג הפשוט ביותר של supervised learning. מטרתה של רגרסיה לינארית היא למצוא את הקשר בין הקלט (הפיצ'רים) לבין ערך היעד ולתת לנו שערוך, ערך רציף כלשהו.



חיזוי ערך של גלאקסי לפי שם

פנקציית המודל:

y= wx + b

הניבוי שלנו יהיה h(x) = wx + b

הדאטה:

Y = {y1, y2,....,ym}

X = {x1, x2, ..., xm}

פונקצייה ההפסד שלנו: $J(w,b) = \frac{1}{m}\sum_{i=1}^{m}|wx_i + b - y_i|$ או: $J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(h(x_i) - y_i)^2$

המטרה שלנו היא למצוא w, b שעבורם פונקציית ההפסד היא מינימאלית.

מוטיבציה לפונקציית ההפסד:

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(wx_i + b - y_i)^2$$

Maximum Likelihood

אנחנו מניחים כי הדאטה שקיבלנו הוא הכי סביר שניתן לקבל. מכאן אנחנו מניחים שהוא יעבוד טוב גם על דאטה אחר, שהוא לא ראה לפני.

מטרה: נרצה למצוא את הניבוי הכי סביר (Y) בהינתן X.

Assume I.I.D. (independently and identically distributed) Gaussian noise with 0 mean and σ^2 variance:

$$y_i = h(x_i) + \epsilon_i = wx_i + b + \epsilon_i; \epsilon_i \sim N(0, \sigma^2)$$

$$p(y_i|x_i; w, b) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y_i-(wx_i+b))^2}{2\sigma^2}} \qquad p(\epsilon_i) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(\epsilon_i)^2}{2\sigma^2}}$$

$$p(y_i|x_i; w, b) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y_i-(wx_i+b))^2}{2\sigma^2}}$$

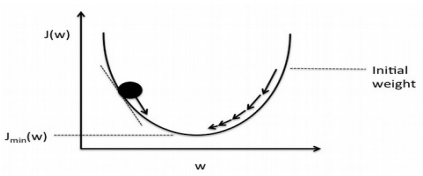
$$p(y|x; w, b) = \prod_i p(y_i|x_i; w, b) = \prod_i \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y_i-(wx_i+b))^2}{2\sigma^2}}$$

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות
נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

למצוא את המקסימום זה קשה, לכן נעזר ב log (נוכל כי היא פונקציה מונוטונית עולה) :

$$log(p(y|x;w,b)) = \sum_i log(\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y_i-(wx_i+b))^2}{2\sigma^2}})$$
$$= mlog(\frac{1}{\sqrt{2\pi}\sigma}) + \sum_i log(e^{-\frac{(y_i-(wx_i+b))^2}{2\sigma^2}})$$
$$= mlog(\frac{1}{\sqrt{2\pi}\sigma}) - \sum_i \frac{(y_i-(wx_i+b))^2}{2\sigma^2}$$

Gradient descent
נשים לב כי הפונקציה שאנו מנסים למצוא לה מינימום היא פונקציה רבת משתנים זה קשה למצוא ערכים המאפסים את הנגזרת.



Schematic of gradient descent.

מה זה גרדיאנט ?
גרדיאנט הוא וקטור המייצג את הנגזרת של פונקציה שיש לה מספר משתנים.
כל ערך הוא נגזרת הפונקציה ביחס לאחד הפרמטרים.

הגרדיאנט עבור רגרסיה לינארית

Our loss function: $J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(wx_i+b-y_i)^2$

$$\frac{\partial J}{\partial w} = \frac{1}{2m}\sum_{i=1}^{m} 2((wx_i+b-y_i)x_i)$$
$$= \frac{1}{m}\sum_{i=1}^{m}(wx_i+b-y_i)x_i$$
$$\frac{\partial J}{\partial b} = \frac{1}{m}\sum_{i=1}^{m}(wx_i+b-y_i)$$

$$\nabla(J) = (\frac{1}{m}\sum_{i=1}^{m}(wx_i+b-y_i)x_i, \frac{1}{m}\sum_{i=1}^{m}(wx_i+b-y_i))$$

ישנו אלגוריתם למציאת המודל שלנו:
- בחר ערך רנדומלי עבור w, b.
- בחר learning rate אלפא לדוגמה 0.01
- חזור עד להתכנסות:

– Update w to w-α$\frac{1}{m}\sum_{i=0}^{m} x_i(h(x_i)-y_i)$

– Update b to b-α$\frac{1}{m}\sum_{i=0}^{m} 1\cdot(h(x_i)-y_i)$

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

תוכנית לחיזוי ערך מכשיר גלאקסי

```
import numpy as np
# galaxy dataset (galaxy2 cost 70, galaxy3 cost 110 ...)
galaxy_data = np.array([[2, 70], [3, 110], [4, 165], [6, 390], [7, 550]])
# set w,b to 0
w = 0
b = 0
# learning rate
alpha = 0.01
for iteration in range(10000):
    # calculate the gradient for w, b
    gradient_b = np.mean(1 * ((w * galaxy_data[:, 0] + b) - galaxy_data[:, 1]))
    gradient_w = np.mean(galaxy_data[:, 0] * ((w * galaxy_data[:, 0] + b) - galaxy_data[:, 1]))
    # update w, b
    b -= alpha * gradient_b
    w -= alpha * gradient_w
    if iteration % 400 == 0:
        print("it: {}, grad_w: {}, grad_b: {}, w: {}, b: {}".format(iteration, gradient_w, gradient_b, w, b))
print("Estimated price for Galaxy S5: ", w * 5 + b)
print("Estimated price for Galaxy S1: ", w * 1 + b)
```

```
it: 9000, grad_w: -1.0487349891263876e-05, grad_b: 5.3996920536292235e-05, w:
96.86039310667573, b: -169.18567575097853
it: 9200, grad_w: -7.838952842575964e-06, grad_b: 4.036093972672461e-05, w:
96.86041129144546, b: -169.18576938011572
it: 9400, grad_w: -5.859362158844306e-06, grad_b: 3.0168488115123183e-05, w:
96.86042488396866, b: -169.1858393648469
it: 9600, grad_w: -4.37968244568765e-06, grad_b: 2.254996247756935e-05, w: 96.86043504393714,
b: -169.1858916761545
it: 9800, grad_w: -3.273670108683291e-06, grad_b: 1.6855362650858298e-05, w:
96.8604426381827, b: -169.18593077715337
Estimated price for Galaxy S5: 315.1162815729334
Estimated price for Galaxy S1: -72.32551158772418
```

קיבלנו ערך שלילי עבור גלאקסי 1. מה ניתן לעשות ?

אחנו יכולים להוסיף פיצ'רים נוספים:

- גודל מסך.
- מספר מעבדים.
- מהירות מעבד.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

גרדיאנט דיסנט עם מספר פיצ'רים

- ∇ is: $\frac{1}{n}\sum_{i=0}^{n} \overleftarrow{x_i}\,(h(\overleftarrow{x_i}) - y_i)$
- Initialize W, b
- Repeat:
  - Update: W := W - α $\frac{1}{m}\sum_{i=0}^{m} \overleftarrow{x_i}\,(h(\overleftarrow{x_i}) - y_i)$
  - Update: b := b - α $\frac{1}{m}\sum_{i=0}^{m} (h(\overleftarrow{x_i}) - y_i)$

הוספת פיצ'רים ריבועיים

אנחנו יכולים להוסיף פיצ'רים שהם פשוט ריבוע של הפיצ'ר הראשון.

נקבל 2 משקלים וביאס אחד: $w_1 x + w_2 x^2 + b$.

בדרך זו במקום להתאים קו לינארי נתאים פרבולה.

```
import tensorflow as tf
import numpy as np
# the number of features
features = 2
x = tf.placeholder(tf.float32, [None, features])
y_ = tf.placeholder(tf.float32, [None, 1])
W = tf.Variable(tf.zeros([features, 1]))
b = tf.Variable(tf.zeros([1]))
y = tf.matmul(x, W) + b
loss = tf.reduce_mean(tf.pow(y - y_, 2))
update = tf.train.GradientDescentOptimizer(0.001).minimize(loss)
data_x = np.array([[2, 4], [3, 9], [4, 16], [6, 36], [7, 49]])
data_y = np.array([[70], [110], [165], [390], [550]])
sess = tf.Session()
sess.run(tf.global_variables_initializer())
for i in range(0, 100000):
    sess.run(update, feed_dict= {x: data_x, y_: data_y})
    if i % 10000 = = 0:
        print('Iteration:', i, ' W:', sess.run(W), ' b:', sess.run(b), ' loss:',
              loss.eval(session= sess, feed_dict= {x: data_x, y_: data_y}))
# (actual: $250)
print('Prediction for Galaxy S5:', np.matmul(np.array([5, 25]), sess.run(W)) + sess.run(b))
# (actual: $30)
print('Prediction for Galaxy S1:', np.matmul(np.array([1, 1]), sess.run(W)) + sess.run(b))
```

```
Iteration: 70000  W: [[-39.583035]
 [ 15.037192]]  b: [88.113594]  loss: 13.24863
Iteration: 80000  W: [[-41.34598 ]
 [ 15.223395]]  b: [91.67418]  loss: 11.654491
Iteration: 90000  W: [[-42.684937]
 [ 15.364843]]  b: [94.377625]  loss: 10.735012
Prediction for Galaxy S5: [264.72846889]
Prediction for Galaxy S1: [68.20209408]
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

over fitting

היא בעיה שבה המודל מותאם יתר על המידה לאוסף מסויים של נתונים ועל כן מצליח פחות בביצוע התחזיות בטסט. התאמת יתר מתרחשת כאשר המודל נקבע על ידי יותר מידי פיצ'רים. עודף הפיצ'רים מאפשר למודל ללמוד את הרעש הסטטיסטי כאילו הוא מייצג התנהגות אמיתית.

Regularization

באמצעות רגולריזציה אנו מגבילים את הכוח של המודל שלנו, נאלץ את המודל לשלם על כל פיצ'ר שהוא משתמש בו. (עלול להפחית את השונות של דגמים מורכבים).

דוגמאות:

Lasso: לאסו הוא אגרסיבי יותר כלפי משקלים קטנים, ולכן רבים ממשקלים אלה מתאפסים בסופו של דבר.

$$J(w,b) = \left(\frac{1}{2m}\sum_{i=1}^{m}(WX_i + b - y_i)^2\right) + \lambda\|W\|_1$$

$$\|W\|_p = \left(\sum_{j=1}^{k}|w_j|^p\right)^{\frac{1}{p}}$$

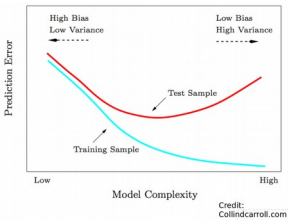
Ridge regression: הרבה משקלים יהיו בסוף התהליך קטנים, אבל לא אפסים.

$$J(w,b) = \left(\frac{1}{2m}\sum_{i=1}^{m}(WX_i + b - y_i)^2\right) + \frac{\lambda}{2}\|W\|_2^2$$

הערה: בשני השיטות יכולות להיות משקולות שליליות. נגיד אם אנחנו מדברים על טלפונים ואנחנו רוצים לנבא את המחיר של הטלפון אז לפיצ'ר כמות הבאגים אמורה להיות משולת שלילית.

Early stopping

אם ניתן למודל שלנו להתאמן יותר מידי אנחנו עלולים להגיע למצב של overfitting.



כל פעם אנחנו נשמור את המודל, נרצה לקחת את המודל שבו ה validation הכי נמוך.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

BGD, SGD, MB-GD

- (BGD) Batch gradiend descent: משתמש בכל ה DATASET על מנת לחשב את הגרדיאנט. עשוי להיות גדול מידי ולקחת יותר מידי זמן במידה וה DATASET ממש גדול.
- (SGD) Stochastic gradient descent: משתמש בכל פעם בדוגמה אחת בלבד. יותר איטרציות, בעוד שכל איטרציה פחות מדוייקת.
- (MB-GD) Mini batch gradient decent: משתמש בכל פעם בתת קבוצה של ה DATASET (למשל 50 דוגמאות).
  - פשרה המנצלת את הווקטוריזציה.
  - הכי שכיח בעבודה עם DATASET גדול.
  - נקרא לפעמים SGD.

נרמול -Normalization

שיטת הנרמול היא: $x = \frac{x-\mu}{\sigma}$ (לחסר בממוצע ולחלק בסטיית התקן)

לעיתים קרובות זה מאוד מועיל לנרמל את כל הנתונים סביב $[0,0,0,.....,0]$ עם סטיית תקן כלשהי.
במידה ומנרמלים גם את label, צריך לזכור להכפיל את ה LOSS שנוצר עם השונות כדי להשיג את ה LOSS בפועל על הנתנים המקוריים.
ניתן לנרמל את כל הנתונים בהתחלה, או להשתמש ב batch-normalization, ולנרמל בכל פעם את התת קבוצה הנתונה.

איך נדע האם הריגרסור שלנו טוב?

האם שגיאה של 280.3 היא טובה?
שיטה (נאיבית) לבדיקה: חשב את הממוצע של כל הנתונים וחשב את ההפסד הממוצע, ודא שהתוצאה שקיבלת נמוכה מהממוצע.

Validation

אחרי שאימנו את המודל (או במהלך אימון), ולפני שאנו מריצים את הטסט נעשה את כל הבדיקות והחיזוי על ה validation מתוך ההנחה שה test וה validation דומים.
תהליך הלמידה על ה validation: תחילה נבדוק את ה validation על מודל לא כל כך מורכב ואז ה validation עשוי להיות גבוה. ככל שהמודל נעשה יותר מורכב כך ה validation צפוי לרדת. הנקודה בה אנו רוצים לתפוס את המודל שלנו היא נקודת המינימום של ה validation שבה אנו משערים שגם ה test יהיה מינימאלי.
איך נעשה זאת? ניתן להריץ את ה traning וה validation, ומידי פעם לעצור לראות אם ה validation עלה, אם כן הולכים צעד אחורה ולוקחים את המודל מנקודת המינימום.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

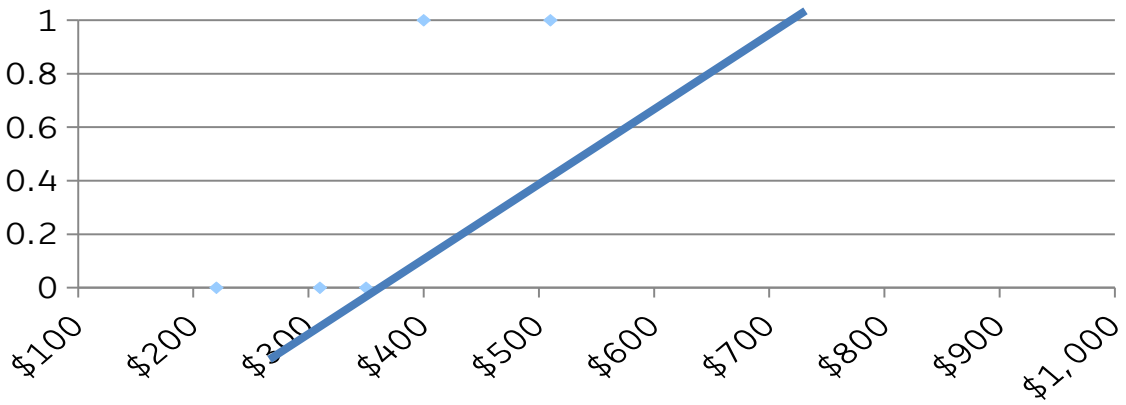
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Classification

לאחר שדיברנו על פתרון פשוט לבעיות של רגרסיה הבעיה הזמן לדבר על פתרון לבעיות של סיווג.
לשם פתרון של בעיות סיווג נשתמש ב Logistic Regression.

What is Logistic Regression ?

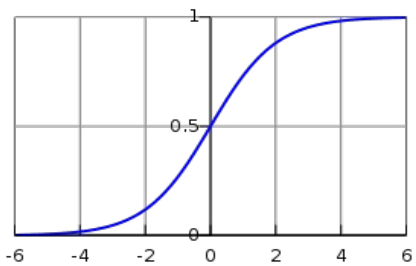
זה אלגוריתם סיווג, נשתמש בו כאשר ערך ההחזרה הוא קטגורי. הרעיון של רגרסיה לוגיסטית הוא למצוא קשר בין תכונות והסתברות לתוצאה מסוימת.
כלומר, היא עונה על שאלות של כן או לא.
נניח שאנו רוצים לסווג טלפון בתור חדש(1)/ישן(0), באמצעות המחיר כפיצ'ר בודד.
האם נוכל להשתמש ברגרסיה לינארית ?



זהי בעיה שאינה מתאימה לרגרסיה לינארית, באמצעות רגרסיה לינראית נפתור בעיות של חיזוי ערך רציף, כמו מחיר דירה וכדומה. לבעיות של סיווג נשתמש ב Logistic regression.

Logistic regression

- הפונקציה בה נשתמש היא פונקציית סיגמואיד:



$$g(z) = \frac{1}{1+e^{-z}}$$

- פונקציית המודל שלנו תהיה:

$$h(x) = \frac{1}{1+e^{-(xW+b)}}$$

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

- חיזוי של הערך 1, אומר שאני בטוחים כי הערך הוא ממחלקה 1. באופן כללי:

$$p(y_i = 1|x_i; w, b) = h(x_i)$$
$$p(y_i = 0|x_i; w, b) = 1 - h(x_i)$$

- ניתן לאחד אותם ל:

$$p(y_i|x_i; w, b) = h(x_i)^y(1 - h(x_i))^{1-y}$$

Loss function

גם כאן אני רוצים למצוא את W,b שימזערו את ההסתברות של y's בהינתן ה x's.

$$p(y|x; w, b) = \prod_i p(y_i|x_i; w, b) = \prod_i h(x_i)^y(1 - h(x_i))^{1-y}$$

$$log(p(y|x; w, b)) = \sum_{i=1}^m log(h(x_i)^y(1 - h(x_i))^{1-y})$$

$$= \sum_{i=1}^m (ylog(h(x_i)) + (1 - y)(log(1 - h(x_i)))$$

$$J(w, b) = -\frac{1}{m}\sum_{i=1}^m (y_i(log(h(x_i))) + (1 - y_i)log(1 - h(x_i)))$$

Gradient of loss

$$h(x) = \frac{1}{1+e^{-(wx+b)}}$$

תחילה נחשב את הנגזרת של פונקציית ה LOGISTIC:

$$g(z) = \frac{1}{1+e^{-z}} = (1 + e^{-z})^{-1}$$
$$g'(z) = \frac{1}{(1+e^{-z})^2}e^{-z}$$
$$= \frac{1}{1+e^{-z}}\frac{e^{-z}}{1+e^{-z}}$$
$$= \frac{1}{1+e^{-z}}\frac{1+e^{-z}-1}{1+e^{-z}}$$
$$= \frac{1}{1+e^{-z}}(1 - \frac{1}{1+e^{-z}})$$
$$= g(z)(1 - g(z))$$

$$J(w, b) = -\frac{1}{m}\sum_{i=1}^m (y_i(log(h(x_i))) + (1 - y_i)log(1 - h(x_i)))$$
$$h(x) = g(Wx + b)$$

$$\frac{\partial J(w,b)}{\partial w_j} = -\frac{1}{m}\sum_i(y_i\frac{1}{g(Wx_i+b)} - (1 - y_i)\frac{1}{1-g(Wx_i+b))})\frac{\partial}{\partial w_j}g(Wx_i + b)$$

$$-\frac{1}{m}\sum_i(\frac{y_i(1-g(Wx_i+b))-(1-y_i)g(Wx_i+b)}{g(Wx_i+b)(1-g(Wx_i+b))})(g(Wx_i+b)(1-g(Wx_i+b))\frac{\partial}{\partial w_j}(Wx_i+b)$$

$$-\frac{1}{m}\sum_i (y_i(1 - g(Wx_i + b)) - (1 - y_i)g(Wx_i + b))x_{i,j}$$

$$-\frac{1}{m}\sum_i (y_i - y_ig(wx_i + b) - g(wx_i + b) + y_ig(wx_i + b))x_{i,j}$$

$$-\frac{1}{m}\sum_i (y_i - h(x_i))x_{i,j}$$

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

לבסוף קיבלנו את אותו גרדיאנט שהיה לנו ברגרסיה לינארית:

- בחר ערך רנדומלי עבור w, b.
- בחר learning rate אלפא לדוגמה 0.01
- חזור עד להתכנסות:

– Update w to w-$\alpha\frac{1}{m}\sum_{i=0}^{m} x_i(h(x_i) - y_i)$

– Update b to b-$\alpha\frac{1}{m}\sum_{i=0}^{m} 1 \cdot (h(x_i) - y_i)$

*נשים לב כי האלגוריתם שונה מכיוון שפונקציית הניבוי שונה.

Employed or not ?

יש לנו מסד נתונים עם משתמשים, ואנו רוצים לשלוח הצעות עבודה. לשם זה, עלינו לדעת אילו מהמשתמשים הם מובטלים. היינו רוצים לבנות מסווג שיקבע עבור כל משתמש האם הוא מובטל או לא, לפי הגיל, מגדר, שנות ניסיון של המשתמשים.

ה DATASET:

- Employed users:
  - Female, 28 years old, 4 years of experience
  - Female, 60 years old, 34 years of experience
  - Female, 25 years old, 3 year of experience
  - Male, 54 years old, 20 years of experience
  - Male, 24 years old, 2 years of experience
  - Male, 39 years old, 12 years of experience
  - Male, 30 years old, 4 years of experience

- Unemployed users:
  - Female, 36 years old 10 years of experience
  - Female, 26 years old 1 year of experience
  - Male, 44 years old, 9 years of experien2ce

```
import numpy as np
# [gender,years old, experience]
data_x = np.array(
    [[1, 28, 4], [1, 60, 34], [1, 25, 3], [0, 54, 20], [0, 24, 2], [0, 39, 12], [0, 30, 4], [1, 36, 10], [1, 26,
1],
     [0, 44, 9]])
# 1-employed, 0-not employed
data_y = np.array([1, 1, 1, 1, 1, 1, 1, 0, 0, 0])
# model function
def h(x, w, b):
    return 1 / (1 + np.exp(-(np.dot(x, w) + b)))
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

```
w = np.array([0., 0, 0])
b = 0
alpha = 0.001
for iteration in range(100000):
    gradient_b = np.mean(1 * ((h(data_x, w, b)) - data_y))
    gradient_w = np.dot((h(data_x, w, b) - data_y), data_x) * 1 / len(data_y)
    b -= alpha * gradient_b
    w -= alpha * gradient_w
print("User [1, 49, 8] prob of working: ", h(np.array([[1, 49, 8]]), w, b))
print("User [0, 29, 3] prob of working: ", h(np.array([[0, 29, 3]]), w, b))
print("User [1, 29, 3] prob of working: ", h(np.array([[1, 29, 3]]), w, b))
```

```
User [1, 49, 8] prob of working:  [0.21107079]
User [0, 29, 3] prob of working:  [0.66430518]
User [1, 29, 3] prob of working:  [0.43735087]
```

Text example bag of words model

סיווג הודעות טקסט לדחוף/לא דחוף, על סמך תוכן ההודעה.

למשל:

"Where are you? I'm trying to reach you for half an hour already, contact me ASAP I need to leave now!"

bag of words model:

זהו מודל המשמש למען ייצוג משפט. במודל זה הטקסט מיוצג על ידי ווקטור של מלותיו, תוך התעלמות מדקדוק ומסדר המילים, אך ישנו שמירה על ריבוי. מודל זה משמש בדרך כלל כשיטה לסיווג מסמכים בהן התדירות של כל מילה משמשת כתכונה מכרעת.

$x_1 = \{0, 0, 0, 0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots, \ldots, 1, \ldots\}$

במקום להשתמש בכל המילים, ניתן להשתמש רק במילים המופיעות במערך האימונים או במילים ה-X הנפוצות ביותר תוך סימון כל שאר המילים כ-UNK.

משמעות התוצאה של Logistic Regression

התוצאה הניתנת על ידי רגרסיה לוגיסטית מתייחסת להסתברות שהאיבר שייך למחלקה 1 כלומר, $p(y=1 \mid X)$. רגרסיה לוגיסטית הוא מודל מפלה מכיוון שהוא מנסה למדל ישירות את $p(y \mid X)$.

נבדיל בין 2 מודלים

generative model, naive bayes:מודל יצרני, לומד מה הסיכוי לכל אחד מהאופציות ואז משתמש בהם כדי לבדוק $p(y|x)$.

discriminative model: logistic regression:בודק מה ה $p(y|x)$ ישירות.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

מהו מסווג טוב?

נניח כי הסיכוי לחלות בסרטן הוא 1%. כלומר, ניתן לבנות מודל הכי פשוט שיש המסווג את כולם כבריאים, למודל זה יהיה 99% הצלחה. אבל האם מודל שכזה אכן יכול להחשב כמודל טוב?

Confusion Matrix	Classified as Positive	Classified as Negative
Really Positive	True Positive	False Negative
Really Negative	False Positive	True Negative

- Accuracy:
  Trues/All
  (True Positive + True Negative) / (True Positive + True Negative + False Negative + False Positive)
- Recall:
  What fraction of positives did we actually find?
  True Positive / Really Positive
  True Positive / (True Positive + False Negative)
- Precision:
  If we say positive, how precise are we?
  True Positive / Classified as Positive
  True Positive / (True Positive + False Positive)

F-Measure (F_1-Score, F-Score)

קומבינציה של precision and recall (הממוצע ההרמוני של השניים):

2*precision*recall/(precision + recall)

Imbalanced Data

פתרונות:

- under-sampling:
  - לוותר על חלק מה DATA (לא רצוי).
  - בכל תור נדגום באופן אקראי אחיד נתונים חדשים מהמחלקה הגדולה.
- Over-sampling: משתמש באותו DATA המון פעמים.
- Loss function interventions:

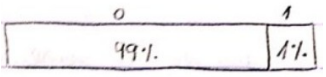
Weighted loss: multiply loss term of each class (outside of the log) by total_examples/examples_in_class

Use tf.nn.weighted_cross_entropy_with_logits(), and provide total_examples/#examples_in_positive_class

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות
נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

- Raise (or reduce) prediction threshold:
נוכל להגדיר למשל את הסף על 0.4 (במקום 0.5) מה שיוביל לחיזויים נוספים עבור מחלקה 1.

נניח וה data שלנו מהצורה הבאה:



ישנם כמה אופציות להתמודד עם דאטה לא מאוזן:
1. נקח כל פעם 1% מכל מחחלקה (under-sampling).
2. נשכפל את ה 1% המון פעמים ואז נאמן על הכל (over-sampling).
3. נשנה את ה loss function על ידי כך שנכפיל בחלק היחסי שלו בדאטה (השיטה הכי מומלצת):

$$-\frac{1}{m}\sum \underbrace{y}_{\frac{total\_example}{positive\_exa}}\log(h(x)) + \underbrace{(1-y)}_{\frac{total\_example}{negative\_exm}}\log(1-h(x))$$

4. שינוי ערך הסף- במקרה כזה ה recall עשוי לרדת על חשבון ה precision ולהפך.

Multiple Classes
עד עכשיו למדנו לסווג בעיות בכן או לא, כלומר: האם בן אדם חולה או לא, האם הודעה מסויימת היא דחופה או לא. פעמים רבות הסיווג יהיה בין מספר labels.
לדוגמה: לסווג תמונה לאובייקט: cat/dog/airplane/sea
אנו נשתמש בייצוג של one-hot vector עבור ה labels שלנו.
לדוגמה: cat= $[1,0,0,0]$, dog = $[0,1,0,0]$
שכבת ההפעלה של ה softmax: כאילו אנו מבצעים רגרסיה לוגיסטית עבור כל פלט לבד ואז:

$$h(y=i|x) = \frac{e^{x^T W_i + b_i}}{\sum_{j=1}^{k} e^{x^T W_j + b_j}}$$

הערכים של $W_j x + b$ נקראים ה logits.
Softmax is a soft version of the maximum

דוגמה:
בהינתן הערכים הבאים של $W_j x + b$: $[5,2,-1,3]$
אם נעלה אותם בחזקת e נקבל: $[e^5, e^2, e^{-1}, e^3] = [148.4,\ 7.4,\ 0.4,\ 20.1]$
נפעיל את הנרמול:
$e^5 + e^2 + e^{-1} + e^3 = 176.3$
$[20.1, 0.4, 7.4, 148.4] / 176.3 = [0.842, 0.042, 0.002, 0.114]$

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

כלומר בהינתן הדוגמה הזאת (X) נקבל ש $h(y=1|x)=0.84$, $h(y=2|x)=0.042$

נשים לב ש softmax הוא הכללה של רגרסיה לוגיסטית עבור מספר קלאסים.
לדוגמה: נניח ויש לנו 2 מחלקות 0,1 כמו שיש לנו ברגרסיה לוגיסטית נקבל:

$$h(y=1|x)=\frac{e^{x^T W_1+b_1}}{e^{x^T W_0+b_0}+e^{x^T W_1+b_1}}$$

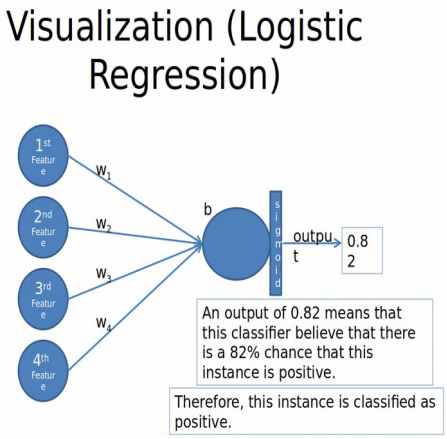
$$=\frac{e^{x^T W_1+b_1}}{e^{x^T W_1+b_1}\left(1+e^{x^T(W_0-W_1)+(b_0-b_1)}\right)}$$

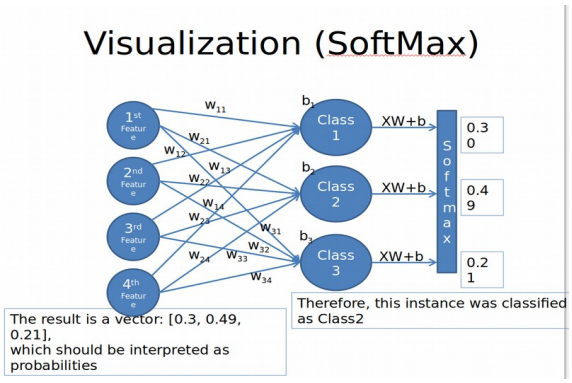
נסמן $w = w0-w1$, $b= b0-b1$
ונקבל:

$$=\frac{1}{1+e^{-(x^T W+b)}}$$

פונקציית ההפסד שלנו תהיה:
loss = −tf.reduce_mean(y_*tf.log(y))

המחשה לרגרסיה לוגיסטית
התוצאה שקיבלנו היא 0.8 כלומר
$h(y=1|x)=0.8$





סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Numerical Issues

כאשר אנו מחשבים softmax אנו מחשבים: $h(y=i|x) = \frac{e^{x^T W_i + b_i}}{\sum_{j=1}^{k} e^{x^T W_j + b_j}}$

וכאשר אנו מחשבים את ה cross entropy אנו מחשבים: tf.reduce_mean(y_*tf.log(y))−
אם h(y|x) נהייה 0 או 1, זה עלול לגרום לבעיה אפילו אם אנחנו צודקים בתחזית שלנו. נקבל אינסוף או מינוס אינסוף.
ולכן אנחנו צרכים להשתמש בפונקציה המובנית של טנסורפלאו שם בעיה זו נלקחת בחשבון:
tf.nn.softmax_cross_entropy_with_logits_v2()
נשים לב שעבור ה inference אנחנו עדיין נצטרך להשתהש בפונקציית ה softmax.
tf.nn.softmax_cross_entropy_with_logits_v2() מחשב שני רמות יחד:

$$loss = -\frac{1}{k}\sum_i y_i(log(\frac{e^{(x^T W_i + b_i)}}{\sum_{j=1}^{k} e^{(x^T W_j + b_j)}})) \quad \text{let } l_i = x^T W_i + b_i, L = max_i(l_i)$$

$$= -\frac{1}{k} y_i \sum_i (log(e^{(l_i)}) - log(\sum_{j=1}^{k} e^{l_j})))$$

$$= -\frac{1}{k} y_i \sum_i (l_i - log(\sum_{j=1}^{k} e^{l_j})))$$

$$log(\sum_{j=1}^{k} e^{l_j})) = log(\sum_{j=1}^{k} e^{L} e^{l_j - L})) = L + log(\sum_{j=1}^{k} e^{l_j - L}))$$

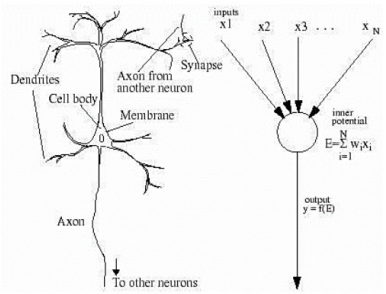
$$= \frac{1}{k} y_i \sum_i (L - l_i + log(\sum_{j=1}^{k} e^{l_j - L})))$$

Numerically stable (and can be computed only once, not for every class)

Neural Networks

מה זה בכלל רשת נויירונים?

זוהי רשת של נויירונים המשמשת לעיבוד מידע. על מנת ליצור רשתות אלה, מדענים בדקו את מכונת עיבוד הנתונים המרשימה ביותר – המוח. המוח שלנו מעבד מידע באמצעות רשתות נויירונים. הנויירונים מקבלים קלט, מעבדים אותו ובהתאם פולטים אותות חשמליים לנויירונים אליהם הם מחוברים. בעזרת חיקוי תהליך זה, הצלחנו ליישם את הארכיטקטורה של מוחנו כדי לקדם את תחום הבינה המלאכותית. למעשה, רשת נויירונים מלאכותית משחזרת את המבנה של נויירונים אנושיים כדי לעבד מידע וכתוצאה מכך מניבה תוצאות מדוייקות בהרבה מאשר מודלים של רגרסייה.



סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

המחשת צורך

נניח ויש לנו את ה training set הבא:

$$X = [[2,32],\ [25,1.2],\ [5,25.2],\ [23,2],\ [56,8.5],\ [60,60],\ [3,3],\ [46,53],\ [3.5,2]]$$

$$Y = [[1],\ [1],\ [1],\ [1],\ [1],\ [0],\ [0],\ [0],\ [0]]$$

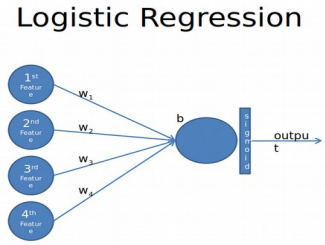
איזה ערך ננבא ל $[44,3]$? $[6,5]$?

נשים לב, שכאשר ההפרש בין הפיצ'רים גדול ערכו של Y הוא 1, כאשר הוא קטן ערכו של Y הוא 0.

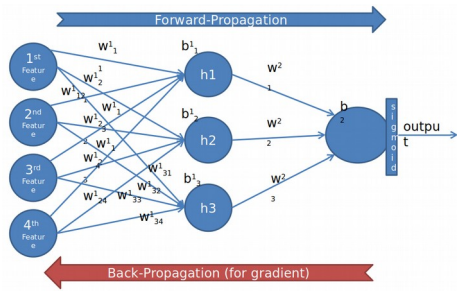
לא נוכל לפתור בעיה זו בעזרת הכלים שלמדנו עד כה, למשל עם logistic regression מכיוון שיש תלות גדולה בין הפיצ'רים. אנחנו צרים מודל מורכב יותר.

אופציה נוספת היא להוסיף פיצ'ר נוסף $|x_{i1}-x_{i2}|$ זה נכון למקרה זה אבל במקרים מסובכים יותר לא תמיד נוכל להסתכל על הדאטה ולזהות את התבניתיות.

ככה זה היה נראה ב logistic regression:



נוסיף hidden layer:



מבנה רשת נויירונים

רשת נויירונים בנוייה מ 3 חלקים עיקריים:

- input layer: השכבה המזינה את רשת הנויירונים במידע. כל פיצ'ר מייצג תכונה שמוזנת לרשת. תכונה זו עשוייה להיות כל דבר– ערך של פיקסל, מחיר דירה וכדומה.
- Hidden layers: השכבה שנמצאת בין ה input layer לבין ה output layer ומכאן שמה. אלו הן השכבות שעושות את כל עיבוד הנתונים עבור הרשת. אנו יכולים לקבוע כרצוננו את מספר וגודל שכבות אלו. באופן כללי, ככל ששכבה יותר גדולה וככל שיש יותר שכבות כאלה, רמת האבסטרקציה עולה, ולמודל שלנו יהיה יותר כח – ונקבל דיוק רב יותר. כל שכבה מורכבת מצמתים המחקים את עצבי המוח שלנו. צמתים אלה מקבלים מידע מצמתי השכבה הקודמת, ומעבירים כפלט נתונים לשכבה הבאה בתור.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

- Output layer: שכבה זו מאגדת את המידע מה hidden layer האחרון של הרשת ומחזירה את הפלט הנדרש המתוכנית.

MultiLayer Perceptron(MLP)

התפיסה הכללית באה מנוירונים במוח, והאופן בו המוח מעביר מידע.

שאלה: מה היה קורה אם הנוירונים באמצע (hidden layers) היו רק קומבינציה לינארית ?

תשובה: המודל כולו היה נותר קומבינציה לינארית. ולכן אנו צרכים activation layers.

שאלה: מה יקרה אם נאתחל את כל ערכי המשתנים ל 0 ?

תשובה: עלינו לאתחל את המודל בערכים רנדומאליים, אחרת נגמור עם identical neurons.

התהליך של ה foward propagation הרבה יותר אינטואטיבי מהתהליך של ה back propagation (נלמד בסוף הסמסטר) שם הדברים יותר מסובכים.

Activation layers

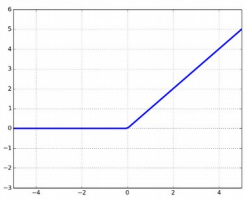
פונקציית הפעלה היא פשוט פונקציה מתמטית המיושמת על ההפעלה(y= Wx+ b). בין כל שתי שכבות לינאריות עלינו לשים activation layer (שכבת הפעלה), ובכך נכניס אי-לינאריות למודל שלנו. רשת נוירונים היא למעשה שילוב בין פונקציות לינאריות מרובות ופונקציות לא לינאריות ביניהם. הפונקציה: y= Wx+ b היא לינארית, במידה ולא היינו מוסיפים את פונקציות ההפעלה היה ניתן להסתכל על הרשת כולה כצירוף לינארי, רשת שכזו לא תוכל ללמוד הרבה.

דוגמאות לשכבות הפעלה:

Logistic function, Step function, Tanh function, ReLU, Leaky ReLU, ELU, (Swish, SELU)

$$ReLU(x) = \begin{cases} x & if\ x > 0 \\ 0 & otherwise \end{cases}$$

What is the Gradient?



ReLU (Rectified linear Unit):

אנחנו צרכים activation layer בין שני נוירונים. במטרה לפשט את חישוב הגרדיאנט דיסנט עבור TF נשתמש הרבה פעמים ב ReLU שהוא פשוט.

חישוב הגרדיאנט נשעה פשוט אם X>0 אז 1 אחרת 0.

Name	Plot	Equation	Derivative (with respect to x)	Range	Order of continuity	Monotonic	Derivative Monotonic	Approximates identity near the origin
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$	C^∞	Yes	Yes	Yes
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$	C^{-1}	Yes	No	No
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$	C^∞	Yes	No	No
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	C^∞	Yes	No	Yes
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$	C^∞	Yes	No	Yes
Softsign [7][8]		$f(x) = \frac{x}{1+\lvert x\rvert}$	$f'(x) = \frac{1}{(1+\lvert x\rvert)^2}$	$(-1, 1)$	C^1	Yes	No	Yes
Inverse square root unit (ISRU)[9]		$f(x) = \frac{x}{\sqrt{1+\alpha x^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3$	$\left(-\frac{1}{\sqrt{\alpha}}, \frac{1}{\sqrt{\alpha}}\right)$	C^∞	Yes	No	Yes
Rectified linear unit (ReLU)[10]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$	C^0	Yes	Yes	No
Leaky rectified linear unit (Leaky ReLU)[11]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0	Yes	Yes	No
Parameteric rectified linear unit (PReLU)[12]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0	Yes iff $\alpha \geq 0$	Yes	Yes iff $\alpha = 1$
Randomized leaky rectified linear unit (RReLU)[13]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ [1]	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0	Yes	Yes	No
Exponential linear unit (ELU)[14]		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\alpha, \infty)$	$\begin{cases} C_1 & \text{when } \alpha = 1 \\ C_0 & \text{otherwise} \end{cases}$	Yes iff $\alpha \geq 0$	Yes iff $0 \leq \alpha \leq 1$	Yes iff $\alpha = 1$

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

MLP in TF

```
import tensorflow as tf
import numpy as np
features = 2
# in the hidden layer gonna be 10 neurons
hidden_layer_nodes = 10
# x- matrix [sizeof(batches)][#features]
x = tf.placeholder(tf.float32, [None, features])
# y_ - vector [sizeof(batches)]
y_ = tf.placeholder(tf.float32, [None, 1])
# W1 matrix of randomized values [#features][#hidden_layer_nodes]
W1 = tf.Variable(tf.truncated_normal([features, hidden_layer_nodes], stddev= 0.1))
# b1 - vector[hidden_layer_nodes]
b1 = tf.Variable(tf.constant(0.1, shape= [hidden_layer_nodes]))
# z1 = x*W1+ b
z1 = tf.add(tf.matmul(x, W1), b1)
# a1 - activation layer
a1 = tf.nn.relu(z1)
# W2 - vector[hidden_layer_nodes]
W2 = tf.Variable(tf.truncated_normal([hidden_layer_nodes, 1], stddev= 0.1))
# b2 is a scalar
b2 = tf.Variable(0.)
# z2 = a1 * W2 + b2
z2 = tf.matmul(a1, W2) + b2
# y - sigmoid function
y = 1 / (1.0 + tf.exp(-z2))
loss = tf.reduce_mean(-(y_ * tf.log(y) + (1 - y_) * tf.log(1 - y)))
update = tf.train.GradientDescentOptimizer(0.0001).minimize(loss)
data_x = np.array([[2, 32], [25, 1.2], [5, 25.2], [23, 2], [56, 8.5], [60, 60], [3, 3], [46, 53], [3.5, 2]])
data_y = np.array([[1], [1], [1], [1], [1], [0], [0], [0], [0]])
sess = tf.Session()
sess.run(tf.global_variables_initializer())
for i in range(0, 30000):
    sess.run(update, feed_dict= {x: data_x, y_: data_y})
print('prediction: ', y.eval(session= sess, feed_dict= {x: [[13, 12], [0, 33], [40, 3], [1, 1], [50, 50]]}))
```

```
prediction:  [[0.1728706 ]
 [0.99723023]
 [0.9955304 ]
 [0.43772173]
 [0.00346024]]
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

The MNIST Dataset

דאטה–סט של ספרות הכתובות בכתב יד.

הדאטה–סט מכיל 60,000 תמונות:

50 אלף עבור train.

10 אלף עבור test.

נשתמש ב MNIST על מנת לפתור בעיית סיווג, בהינתן תמונה של מספר מסויים מהדאטה סט צריך לנבא נכון את המספר שבתמונה.

נתחיל בלהראות דוגמה פשוטה ב SoftMax ולאחר מכן דוגמה יותר מורכבת עם hidden layers.

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
# reads the MNIST dataset
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
# flat the picture 28*28=784 – x is a matrix [batch size][num of pixels(784)]
x = tf.placeholder(tf.float32, [None, 784])
# 10 because has ten digits between 0–9 – y_ is a matrix [batch size][num of classes]
y_ = tf.placeholder(tf.float32, [None, 10])
# W is a matrix [num of pixels][num of classes], init to zeros
W = tf.Variable(tf.zeros([784, 10]))
# b is a vector of zeroes in len 10 (number of classes)
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.add(tf.matmul(x, W), b))
# loss function
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
for i in range(50000):
    batch_xs, batch_ys = mnist.train.next_batch(100)  # MB-GD
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))
```

0.9215

קיבלנו 0.9215 זוהי תוצאה סבירה אבל מייד נראה שנוכל לשפר זאת באמצעות הוספת שני hidden layers

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
# reads the MNIST dataset
mnist = input_data.read_data_sets("MNIST_data/", one_hot= True)
# two layers of hidden nodes the firs 100 nodes and the second 50
(hidden1_size, hidden2_size) = (100, 50)
# flat the picture 28*28= 784 – x is a matrix [batch size][num of pixels(784)]
x = tf.placeholder(tf.float32, [None, 784])
# 10 because has ten digits between 0–9 – y_ is a matrix [batch size][num of classes]
y_ = tf.placeholder(tf.float32, [None, 10])
# W1 is a matrix [num of pixels][number of nodes in the first hidden layer], init with randomized values
W1 = tf.Variable(tf.truncated_normal([784, hidden1_size], stddev= 0.1))
# b is a vector of 0.1 in len of the number of nodes in the first hidden layer
b1 = tf.Variable(tf.constant(0.1, shape= [hidden1_size]))
z1 = tf.nn.relu(tf.add(tf.matmul(x, W1), b1))
W2 = tf.Variable(tf.truncated_normal([hidden1_size, hidden2_size], stddev= 0.1))
b2 = tf.Variable(tf.constant(0.1, shape= [hidden2_size]))
z2 = tf.nn.relu(tf.matmul(z1, W2) + b2)
W3 = tf.Variable(tf.truncated_normal([hidden2_size, 10], stddev= 0.1))
b3 = tf.Variable(tf.constant(0.1, shape= [10]))
y = tf.nn.softmax(tf.matmul(z2, W3) + b3)
cross_entropy = tf.reduce_mean(–tf.reduce_sum(y_ * tf.log(y), reduction_indices= [1]))
train_step = tf.train.GradientDescentOptimizer(0.001).minimize(cross_entropy)
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
for i in range(1000):
    for _ in range(1000):
        batch_xs, batch_ys = mnist.train.next_batch(100)
        sess.run(train_step, feed_dict= {x: batch_xs, y_: batch_ys})
    print(i, sess.run(accuracy, feed_dict= {x: mnist.test.images, y_: mnist.test.labels}))
```

```
0 0.3079
1 0.5591
2 0.6767
3 0.7368
4 0.7784
.....
498 0.9725
499 0.9724
500 0.9722
.....
997 0.9748
998 0.9744
999 0.9748
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות
נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

לאחר כחצי שעה של אימון קיבלנו 0.9748 999 שזה הרבה יותר טוב. נראה בהמשך כיצד ניתן לשפר גם את התוצאה הזאת באמצעות CNN

בעיות אפשריות

אם הטעות ב train גדולה מידי:

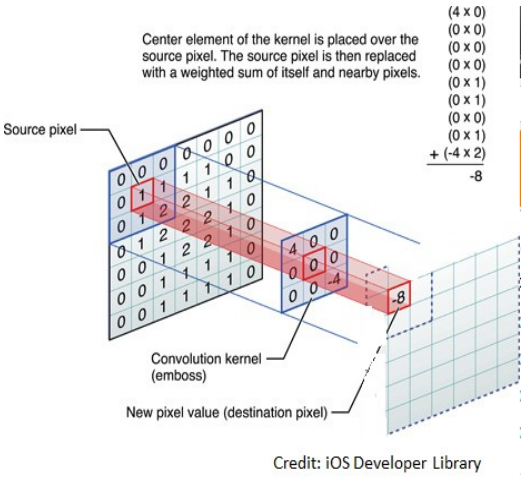
- יתכן כי יש באג בקוד.
- האימון לא ארוך מספיק (להגדיל את האיטרציות).
- ניתן לעשות מודל מופשט יותר, מורכב יותר, להוסיף עוד שכבות, עוד פיצ'רים.
- learning rate לא מתאים.
- ניתן להשתמש ב optimization algoritems: SGD, Adagrad, Adam

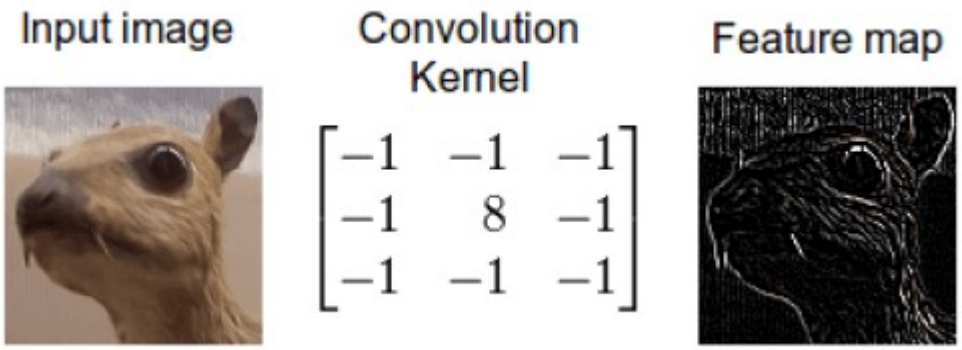
אם הטעות ב train בסדר אך הטעות ב test גבוהה מידי:

- הוספת דאטה.
- early stopping.
- regularization: dropout, Ridge, LASSO

<u>convolutional neural network (CNN)</u>

בלמידה עמוקה CNN מיושמת לרוב לניתוח דברים חזותיים. יש ל CNN יישומים רבים בזיהוי תמונות ווידאו, ניתוח תמונות רפואיות ועיבוד שפות טבעיות.





Credit: timdettmers.com

הרעיון המרכזי הגיע מהצורה שבה המוח מעבד תמונה.
Convolution יעילה מאוד בעיבוד תמונה, אך יש לה שימוש גם בתחומים שונים.

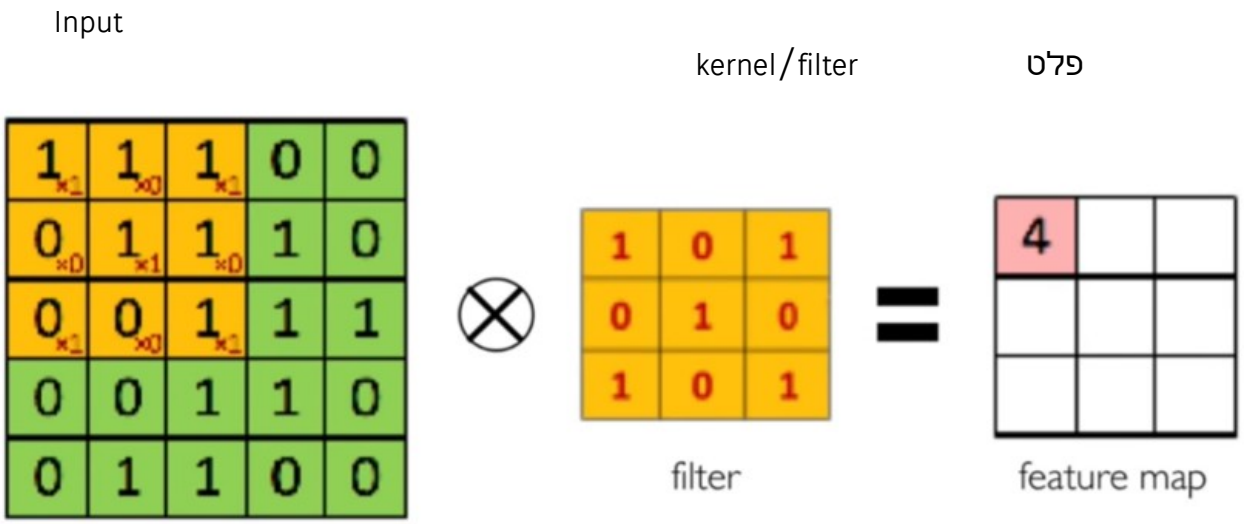
Convolution layers

$$h_{ij}^{l} = \sum_{a=0}^{k_n} \sum_{b=0}^{k_m} w_{ab} h_{(i+a)(j+b)}^{l-1}$$

המשקלים נקראים kernels (size: k_n x k_m).

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות
נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

הרעיון הוא לעשות weight sharing, כלומר במקום להשתמש במשקולות חדשות עבור כל נוירון, אנו משתמשים באותם משקולות שוב ושוב. מה שהופך את הדגם למורכב יותר.
Convolution מנצלת את המבנה של הקלט ואת היחסים האפשריים בין פיקסלים סמוכים.
בפועל אנחנו שמים בערימה רבים מהפילטרים הללו זה על גבי זה, ומקבלים תוצאות רבות שאנו ממשיכים לעבד באמצעות הרשת שלנו. כל פילטר בוחן היבטים שונים על התמונה.
היתרון הוא שכאשר נוספת שכבה נוספת אז המודל יהיה מורכב יותר מצד אחד אך מצד שני מוגבל, כתוצאה מזה אנו מקבלים איזון.

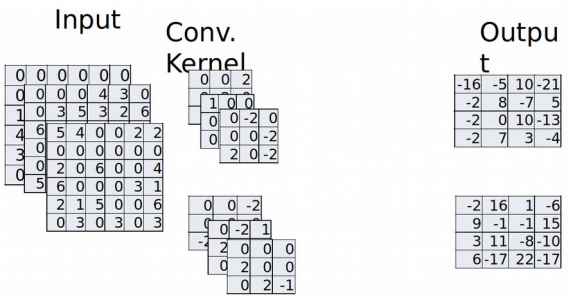


kernel size = [H, W, DIM, num of kernels]
בדוגמה הנ"ל [3, 3, 1, 1]

4D Convolution weights
בפועל הרבה פעמים הקלט ל convolution layer הוא שלושה מימדים:

- תמונה צבעונית, תמונה שיש בה ערוצי RGB.
- פלט של convolution layer קודם.

במקרה כזה בכל קבוצת KERNELS כל KERNEL מכפיל מימד אחר ולאחר מכן סוכמים את התוצאות.



Convolution layers parameters
גודל הפילטר, לדוגמה 3X3.
מספר הפילטרים (depth), לדוגמה 3X3X15.
Stride: כמה צעדים בכל מימד: בדרך כלל נשתמש ב [1, 1, 1, 1].

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

padding: הוספת אפסים.

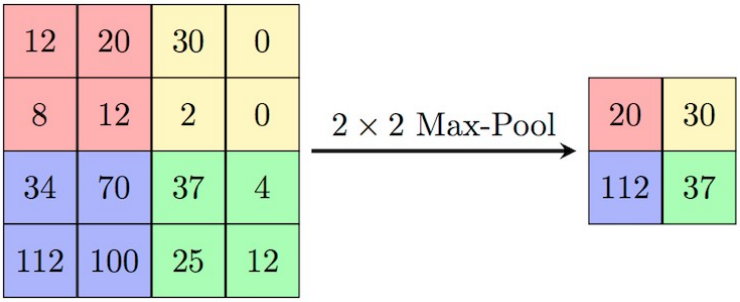
- valid: ללא הוספת אפסים (כמו בדוגמאות שלמעלה). התוצאה שנקבל היא לדוגמה אם גודל התמונה הוא 96X96 עם פילטר בגודל 3X3 הפלט יהיה בגודל 1 +3−96 כלומר בגודל 94X94
- same: הוספת אפסים, לדוגמה בהינתן תמונה בגודל 96X96 עם פילטר בגודל 5X5, לפני הקונבולוציה נוסיף אפסים מסביב למטריצה ונהפוך את הגודל שלה ל 100X100. מכאן שהתוצאה שנקבל היא 1 +5−100 כלומר 96X96 כמו מטריצת הקלט.

Bias and activation layer

אנחנו בדרך כלל נוסיף ביאס לכל אחד ממטריצות הפלט.

בדרך כלל לאחר שכבת הקונבולוציה, תבוא שכבת אקטיבציה למשל Relu בדיוק כמו שעשינו ב fully connected network.

Max pooling



המגבלה של ה feature map היא שהם נותנים את המיקום המדוייק של התכונות בקלט.

משמעות הדבר שתנועות קטנות במיקום הפיצ'ר בתמונות יביאו למפת פיצ'רים אחרת.זה יכול לקרות גם במידה ועושים augmentation לתמונות. גישה נפוצה לטיפול בבעיה זו נקראת down sampling. כאן נוצרת גרסת רזולוציה נמוכה יותר של אותו קלט שעדיין מכילה את האלמנטים החשובים, ללא המיקום המדוייק שלו שאולי לא מועיל למשימה.

ניתן להשיג down sampling באמצעות הגדלת ה stride, או גישה שכיחה יותר היא להשתמש ב pooling layer. שתי פונקציות נפוצות המשמשות ל pooling:

- max pooling: לקיחת הערך המקסימלי בכל תת מטריצה.
- average pooling: חישוב הממוצע של תת המטריצה.

הערה: נשים לב כי בשכבת ה max pooling אין כלל משקולות!

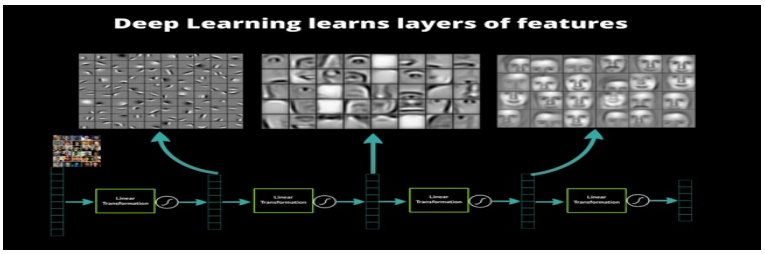
למעשה נוירונים בשכבות הראשוניות לומדים לזהות דברים פשוטים (מושגים בסיסיים) וככל שנעלה בשכבות ונעמיק לתוך המודל הנוירונים לומדים לזהות דבריסם מורכבים יותר ויותר.

לדוגמה: בשכבות הראשונות הנוירונים יזהו קצוות, בשכבות הבאות הם יזהו איברים ובשכבות הגבוהות הם כבר יזהו פרצופים ואלמנטים מורכבים יותר.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה



Credit: http://www.cc.gatech.edu/~hays/compvision/results/proj6/yyeh32/index.html

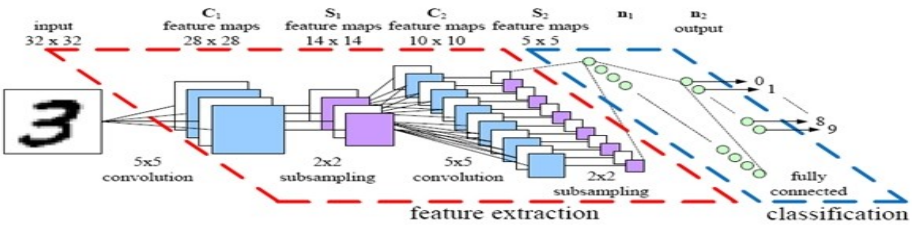
לדוגמה, שכבות המודל עשויות להראות כך:

Input Image

Convolutional Layer

Nonlinearity

Pooling Layer

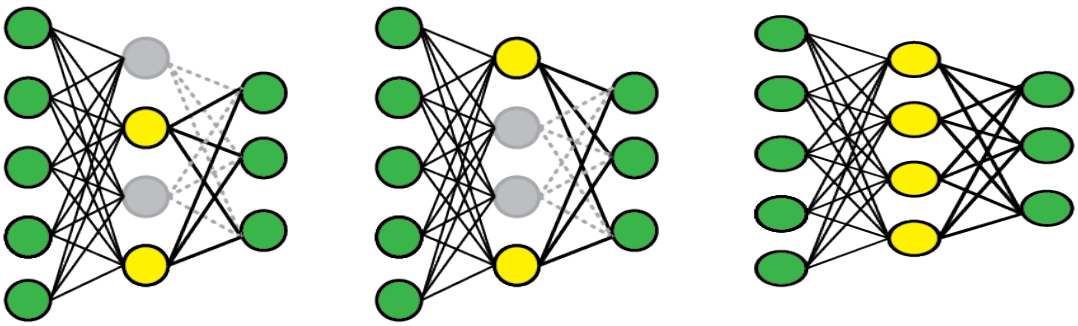


כמה משקולות יש לנו בדוגמה הנ"ל?

(5*5*4+ 4) + (4*5*5*12+ 12) + (5*5*12*1000+ 1000) + (1000*10+ 10)

Dropout(Regularization)

dropout הופכת את הלמידה לבריאה יותר, מונעת מצב של overffiting.



Credit: Matt Krause

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

בשיטה זו אנחנו "נמחק" באופן אקראי נוירונים מסויימים.
יש כמה הסברים מדוע שכבות dropput עוזרות:

- מקשה על הרשת להגיע למצב של ovefit, מכיוון שבכל פעם השכבה הבאה רואה את זרימת המידע בצורה שונה.
- מאלצת את הרשת להצליח גם כשאר ישנם נוירונים חסרים, ולכן היא נאלצת לא "לשים את כל הביצים שלה באותו סל", וכל הנוירונים חייבים לעבוד. זה עשוי לשפר את הביצויים הכוללים.
- ניתן לראות ב dropout כמעין מכלול גדול של NN שלכל אחת קשרים שונים.
- במידה ואין לנו הרבה דאטה, מאפשר לנו להוסיף רעש ל דאטה קיים. גם אם נעבור מספר פעמים על הדאטה היא כל פעם תגיע בצורה שונה מכיוון שהנוירונים "נזרקים" באופן אקראי.

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
# reads the MNIST dataset
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
# the image comes flatten
x = tf.placeholder(tf.float32, shape=[None, 784])
# 10 because has ten digits between 0-9 - y_ is a matrix [batch size][num of features]
y_ = tf.placeholder(tf.float32, shape=[None, 10])
# reshape the image to 28X28
x_image = tf.reshape(x, [-1, 28, 28, 1])  # if we had RGB, we would have 3 channels
# -------------first convolution layer ----------
#  kernel size = [H,W,DIM,num of kernels] - 32, filters size 5X5
W_conv1 = tf.Variable(tf.truncated_normal([5, 5, 1, 32], stddev=0.1))
# for each kernel apply bias
b_conv1 = tf.Variable(tf.constant(0.1, shape=[32]))
# padding='SAME'-add zeroes before convo such that HxW stay as before. and apply ReLU activation function
h_conv1 = tf.nn.relu(tf.nn.conv2d(x_image, W_conv1, strides=[1, 1, 1, 1], padding='SAME') + b_conv1)
# max-pooling, divided the feature map into 2X2 matrices and takes the max value of each
h_pool1 = tf.nn.max_pool(h_conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
# -------------  second convolution layer  ----------
#  kernel size = [H,W,DIM,num of kernels] - 32 filters from the last convo layer, filters size 5X5
W_conv2 = tf.Variable(tf.truncated_normal([5, 5, 32, 64], stddev=0.1))
# for each kernel apply bias
b_conv2 = tf.Variable(tf.constant(0.1, shape=[64]))
h_conv2 = tf.nn.relu(tf.nn.conv2d(h_pool1, W_conv2, strides=[1, 1, 1, 1], padding='SAME') + b_conv2)
h_pool2 = tf.nn.max_pool(h_conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
# -------------  fully connected  ----------
# flatten 64 filters in size 7X7
h_pool2_flat = tf.reshape(h_pool2, [-1, 7 * 7 * 64])
W_fc1 = tf.Variable(tf.truncated_normal([7 * 7 * 64, 1024], stddev=0.1))
b_fc1 = tf.Variable(tf.constant(0.1, shape=[1024]))
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
# regularization- dropout layer
keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

```
W_fc2 = tf.Variable(tf.truncated_normal([1024, 10], stddev=0.1))
b_fc2 = tf.Variable(tf.constant(0.1, shape=[10]))
y_conv = tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y_conv), reduction_indices=[1]))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)  # uses moving averages momentum
correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
sess = tf.InteractiveSession()
sess.run(tf.global_variables_initializer())
for i in range(20000):
    batch = mnist.train.next_batch(50)
    if i % 100 == 0:
        train_accuracy = accuracy.eval(feed_dict={x: batch[0], y_: batch[1], keep_prob: 1.0})
        print("step %d, training accuracy %g" % (i, train_accuracy))
    train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})
print("test accuracy %g" % accuracy.eval(feed_dict={x: mnist.test.images, y_: mnist.test.labels, keep_prob:
1.0}))
```

Recurrent Neural Networks (RNN)

RNN היא הכללה של NN בעלת זיכרון פנימי. RNN הוא מחזורי מכיוון שהוא מבצע אותה פונקציה עבור כל קלט נתונים ואילו הפלט של הקלט הנוכחי תלוי בחישוב האחרון. לאחר הפקת הפלט, הוא מועתק ונשלח חזרה לרשת. על מנת לקבל החלטה מסויימת, הרשת מחשיבה את הקלט הנוכחי ואת הפלט שלמד מהקלט הקודם. שלא כמו NN, RNN יכול להשתמש במצב הפנימי (זיכרון) על מנת לעבד רצפים. זה הופך אותו ליישומי למשימות כמו זיהוי כתב יד, זיהוי דיבור וכדומה.

Sequences

כאשר משתמשים ב bag of word אנו מאבדים את היחסים שיש בין המילים.

לשני המשפטים הבאים יהיה ייצוג זהה ב bag of word:

He likes bananas but not apples

He likes apples but not bananas

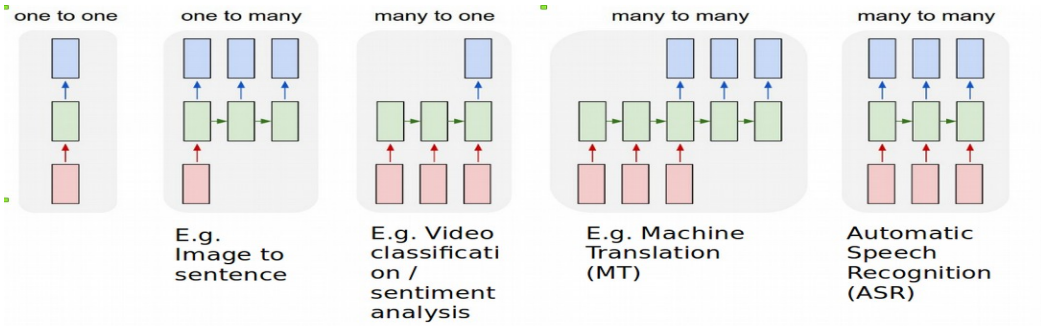
למשפטים הללו תוכן שונה לגמרי ולכן לא נרצה מצב כמו פה שייוצגו באותו צורה.

על מנת לפתור זאת, לעיתים נשתמש ב CNN גם עבור טקסט. אבל בדרך כלל עבור רצפים/סדרות נשתמש ב RNN.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

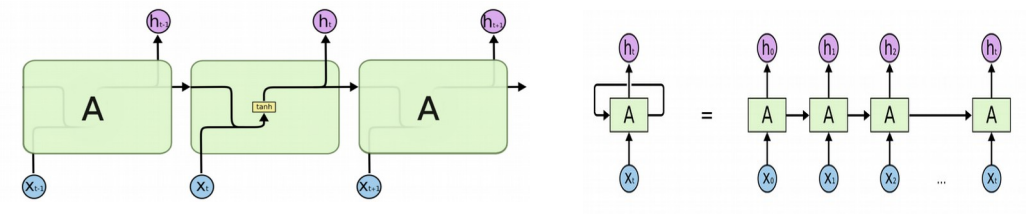
נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

סוגי רצפים:



Fully connected layer for sequences

נניח שבנינו רשת שיש לה hidden layer שבו על הנוירונים המחוברים לכל אחד מהפיצ'רים של כל אחד מהאיברים ברצף. (עלינו לקבוע אורך רצף מקסימאלי). זה ישמור על סדר הרצף, אך יהיה מאוד לא יעיל (הרבה יותר מידי משקולות, למודל יהיה יותר מידי כח, חשש ל overfitting).
משקולות אלה ילמדו דפוסים מיותרים בין היתר.
אנחנו שוב זקוקים ל weight sharing כמו ב CNN.



תיאור התמונה: ראשית, הרשת מקבלת את ה X_0 מרצף הקלט ופולטת את h_0 שיחד עם X_1 הם הקלט לשלב הבא, וכך הלאה. בצורה זו אנו למעשה מכריחים את המודל להתייחס לקלט כרצף.
הנוסחה עבור המצב הנוכחי היא:

$$h_t = f(h_{t-1}, x_t)$$

נפעיל את ה activation function:

$$h_t = \tanh(w_{hh}h_{t-1} + w_{xh}x_t)$$

W הוא משקל, h הוא הוקטור הנסתר היחיד, w_{hh} הוא המשקל במצב הקודם, w_{xh} הוא המשקל עבור הקלט הנוכחי, tanh היא פונקציית ההפעלה, המיישמת את האי ליניאריות הפלט שלה הוא מספר בטווח $[-1,1]$.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות
נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

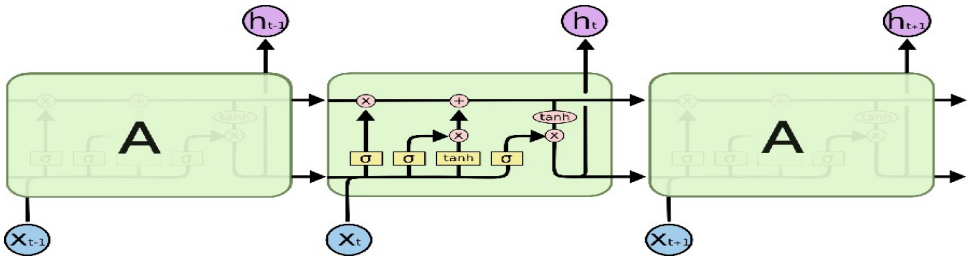
יתרונות של RNN:
1. יכול למדל רצפים של מידע כך שניתן להניח כי כל דוגמה יכולה להיות תלויה גם בדוגמאות הקודמות.
2. ניתן להשתמש בייחד עם CNN כדי להרחיב את שכנות הפיקסלים היעילה.

חסרונות של RNN:
1. עלול להיות מצב של Gradient vanishing and exploding.
2. אימון RNN – משימה קשה.
3. לא ניתן לעבד רצפים ארוכים מאוד עם משתמשים ב tanh או relu כ activation finction.

על מנת לשנות את המשקולות אנו צרכים לחשב כמה כל משקולת משפיעה על ה output לאורך כל המודל. במצב שבו במודל יש הברה רמות (מספר הרמות = כמספר המילים שאנו מנתחים) השינוי ישפיע על הכל, שהרי כל המילים עובדות על אותן משקולות.

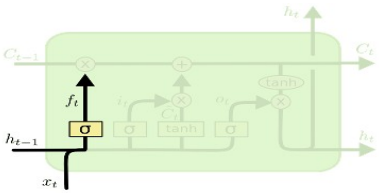
Gradient vanishing
בעיקרון, כל נוירון שהשתתף בחישוב הפלט, מקושר לפונקציית הלוס, ועלינו לעדכן את המשקולות על מנת למזער את השגיאה. העניין עם RNNs זה שלא רק הנוירונים שנמצאים ממש מתחת לשכבת הפלט הזו הם שתרמו אלא גם כל הנוירונים הנמצאים הרחק אחורה בזמן. לכן עלינו לשנות משקולות גם לנוירונים האלה. הבעיה היא שהמשקל המשמש לחיבור השכבות הנסתרות בעצמו נוסף לחישוב של הרמה הבאה. אנו מכפילים אותו משקל מספר פעמים, וכאן מתעוררת הבעיה: כשאתה מכפיל משהו במספר קטן הערך שלך יורד מהר מאוד.

Long Short-Term Memory (LSTM)
LSTM הוא גרסה שונה של RNN, שמקלה על זכירת נתוני העבר בזיכרון. בעיית ה vanishing gradien נפתרת כאן. LSTM מתאים לבעיות סיווג, לעבד ולחזות סדרות זמן.



לרשת LSTM שלושה שערים:

1. forget gate:
מגלה אילו פרטים ברצוננו למחוק, כלומר האם ברצוננו למחוק את הזיכרון. זה נקבע על ידי פונקציית ה sigmoid. השער מקבל את



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

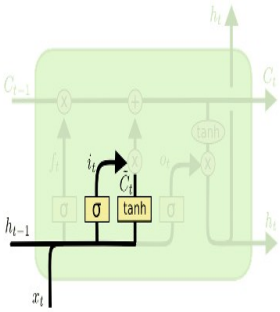
סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

המצב הקודם h_{t-1} וכניסת

התוכן x_t ומוציא מספר בין 0 (לשכוח) לבין 1 (לשמור)



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

2. input gate:

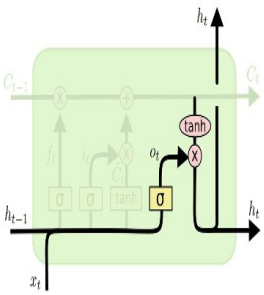
מגלה עבור איזה ערך מהקלט הזיכרון משתנה.
פונקציית ה sigmoid מחליטה אילו ערכים להעביר(0,1) ופונקציית ב tanh נותנת משקל לערכים המועברים, ומחליטה את רמת החשיבות שלהם (−1,1).

נשאל את עצמנו שתי שאלות:

א. האם נרצה לזכור? החלק של הסיגמואיד כאשר פלט של אחד אומר שברצוננו לזכור את הפלט הנוכחי פלט של 0 אומר שברצוננו לשכוח אותו.

ב. מה הרצה לזכור? או עד כמה שהוא חשוב? ופונקציית ב tanh נותנת משקל לערכים המועברים, ומחליטה את רמת החשיבות שלהם (−1,1).

3.output gate:



$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

הקלט והזיכרון משמשים לקביעת הפלט.
פונקציית ה sigmoid מחליטה אילו ערכים להעביר 0,1. ופונקציית ה tanh מעניקה משקל לערכים המועברים ומחליטה את רמת החשיבות שלהם, ומוכפלים עם הפלט של פונקציית ה sigmoid.

בהינתן LSTM כמו בתמונה שלמעלה עם:

Input dimension: k
LSTM bandwidth (memory size): d
Maximum sequence length: t
Mini−batch size: m

כמה משקלים (פרמטרים) יש לנו?

(k+ d+ 1)*4d

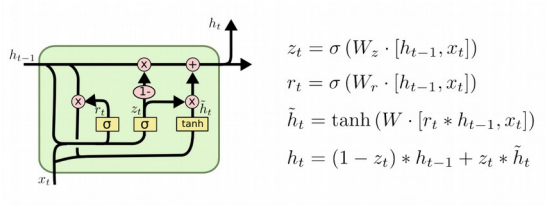
סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

GRU

מודל שקול אך חסכוני יותר. במודל זה ה input וה forget מחוברים.



Generating text

ברצוננו לכתוב תוכנית שתייצר טקסט. עשינו זאת בעבר באמצעות n-grams. כעת נעשה זאת באמצעות LSTM, אך פעם נעשה זאת ברמת התו! כלומר בהינתן מספר תווים נחזה את התו הבא.

(הקוד המלא במחשב)

```
cellsize = 30
x = tf.placeholder(tf.float32, [None, num_past_characters, possible_chars])
y = tf.placeholder(tf.float32, [None, possible_chars])
lstm_cell = tf.nn.rnn_cell.BasicLSTMCell(cellsize, forget_bias= 0.0)
output, _ = tf.nn.dynamic_rnn(lstm_cell, x, dtype= tf.float32)
output = tf.transpose(output, [1, 0, 2])
last = output[-1]
W = tf.Variable(tf.truncated_normal([cellsize, possible_chars], stddev= 0.1))
b = tf.Variable(tf.constant(0.1, shape= [possible_chars]))
z = tf.matmul(last, W) + b
res = tf.nn.softmax(z)
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y * tf.log(res), reduction_indices= [1]))
train_step = tf.train.GradientDescentOptimizer(0.1).minimize(cross_entropy)
sess = tf.InteractiveSession()
sess.run(tf.global_variables_initializer())
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(res, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
num_of_epochs = 100
for ephoch in range(num_of_epochs):
    acc = 0
    curr_batch = 0
    while True:
        batch_xs, batch_ys = next_batch()
        if batch_xs is None:
            break
        else:
            sess.run(train_step, feed_dict= {x: batch_xs, y: batch_ys})
            acc + = accuracy.eval(feed_dict= {x: batch_xs, y: batch_ys})
    print("step %d, training accuracy %g" % (ephoch, acc / curr_batch))
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

```
def text2arr(source, start):
    src_as_num = [ord(x) for x in source]
    ret_arr = numpy.zeros(shape=(1, num_past_characters, possible_chars))
    for inc in range(num_past_characters):
        ret_arr[0][inc][inverse_char_map[src_as_num[start + inc]]] = 1
    return ret_arr
requested_length = 200
text = list("hello world")
for i in range(requested_length):
    predicted_letter = back_to_text(res.eval(feed_dict={x: text2arr(text, len(text) -
num_past_characters)}))
    text += predicted_letter
print(''.join(text))
```

tf.nn.rnn_cell.BasicLSTMCell.__init__(num_units, forget_bias=1.0, state_is_tuple=True, activation=tanh)

הפרמטרים:

num_units–מספר היחידות בתא LSTM.

forget_bias– הביאס שמתווסף ל forgate gate, ברירת מחדל היא 1 על מנת להפחית את השכחה בתחילת האימון.

state_is_tuple– אם true, מקבלים ומחזירים שני טופלים של c_state ו m_state. אם false, הם קשורים לאורך הציר.

Activation– פונקציית ההפעלה של המצבים הפנימיים.

סיכום

חיזוי של סדרות נתונים מציב אתגר גדול בפני למידת מכונה בגלל שעל המחשב לזכור את התוצאות שחושבו בשלבים הקודמים של תהליך הלמידה, דבר שלא מתאפשר ברשת נוירונים רגילה. על מנת לטפל בבעיה זו ניתן להשתמש ב RNN.

רשתות LSTM משתמשות ביחידות (במקום בנוירונים) כשכל יחידה מכילה 3 סוגי שערים:

1. forget gate– מחליט האם להשתמש בנתונים מהשלב הקודם.
2. שער הקלט– מחליט איזה מידע שמגיע מהקלט יוזן לתוך הרשת.
3. שער הפלט– מחליט איזה מידע להעביר לשכבה הבאה בתור.

Deep Reinforcement Learning(RL)

RL הוא תת נושא ב ML המלמד סוכן כיצד לבחור בפעולה מתוך מרחב פעולות נתון, בתוך סביבה מסויימת, על מנת למקסם את התגמולים לאורך זמן.

ל RL יש 4 יסודות חיוניים:

1. סוכן agent: שאותו אנו מאמנים במטרה לבצע משימה מסויימת.
2. סביבה environment: העולם, האמיתי או הוירטואלי, בו הסוכן מבצע את הפעולות.
3. פעולה action: מהלך שנעשה על ידי הסוכן הגורם לשינוי סטטוס בסביבה.
4. תגמולים rewards: הערכת פעולה, שיכולה להיות חיובית או שלילית.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Open AI Gym

סביבת קוד פתוח המאפשרת לסוכנים אוטונומיים לשחק משחקים. ישנם משחקים פשוטים מאוד, ומשחקים מתוחכמים יותר כמו פאקמן ודום. ניתן להשתמש בספרייה באמצעות פייתון:

Pip install gym

MDP (S,A,R,T)

S= states. המצבים האפשריים של העולם. מצבים מסויימים יכולים להיות מצביי סיום.

A= actions. הפעולות שהסוכן יכול לבצע.

R= reward function. פונקציית התגמול בדרך כלל קשור למצב ספציפי R(S) או מצב ופעולה R(s, a)

T= transition function. פונקציית מעבר. T(s, a, s′). ההסתברות שהסוכן יגיע למצב s′ בהתחשב בעובדה שהוא היה במצב s וביצע פעולה a.

Discount factor

אנחנו מניחים גורם הנמחה כלשהו Y על תגמולים עתידיים. לדוגמה, אם Y= 0.8, והסוכן מקבל 2 על התור הנוכחי, 5 על התור הבא, 25 על הבא הבא. הערך הכולל שיקבל הסוכן הוא :

2+ Y5+ YY25 = 2+ 0.8*5+ 0.8*0.8*25 = 22

אינטואיציה לגבי discount factor: הרעיון הוא שכאשר אנו מקבלים תגמול עכשיו, זה יותר שווה לנו מאשר לקבל את התגמול בהמשך.

Model-Free MDP Reinforcement Learning Assumptions

אנחנו מניחים שניתן לנו המצב המדוייק.

אנו מניחים למידה ללא מודל, כלומר אין לנו מידע על מצבי העבר, או על פונקציית התגמול.

אנו מניחים שאנו מכירים את כל הפעולות שאנו יכולים לבצע.

Q-Values

אנו מגדירים ערך Q_π של הצמד (state, action), להיות התועלת/ערך ההחזרה הצפוי עבור הסוכן ב state s שנוקט בפעולה a (ופועל במסגרת policy π).

לדוגמה, אם במצב s ננקוט בפעולה a יביא לתגמול של 5 וייקח את הסוכן למצב סיום, אזי:

$Q_\pi = (s, a) = 5$

ה policy של סוכן תהיה פשוטה: עקוב תמיד אחר הפעולות המעניקות את ערך ה -Q הגבוהה ביותר מבין הפעולות האפשריות.

כלומר, לנקוט בפעולה: $\arg\max_{a' \in A} Q(s, a')$

Q-learning

נניח שהתחלנו ב state s, לקחנו action a, קיבלנו reward r, והגענו למצב s′ שונה מ s.

(נניח שכאשר אנו מבצעים את פעולה a במצב s אנו תמיד מגיעים למצב s′ כלומר: T(s, a, s′)= 1.)

נניח שיש לנו את ערכי ה Q האופטימליים המדוייקים של כל הזוגות action-state למעט s, a.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

מה יהיה $Q(s,a)$? ?

$Q(s,a) = r * \gamma * \max_{a'} Q(s',a')$

לאחר ביצוע פעולה, קבלת התגמול והתבוננות במצב החדש ה Q-learner מעדכן את ה Q-values.
עדכון זה נעשה תוך כדי שמיוש ב learning rate, אלפא (כמו ב SGD). Q-learner מוסיף את ההפרש בין הערך הישן לערך החדש הצפוי.

זה מה שאנו הינו משיגים אם היינו מבצעים gradient descent, ומגדירים את ה loss כ:

$$Q(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t,a_t)}_{\text{old value}} \right)$$

$$((r + \gamma \max_{a'} Q(s',a')) - Q(s,a))^2$$

Exploration / Explotation

אנחנו איננו מקבלים את ערכי ה Q אלא מגלים אותם תוך כדי למידה.
Exploration חקר: נקיטת פעולה שעשויה להיות לא הכי אופטימלית אך עשויה ללמד את הסוכן משהו חדש. למשל, הולכים למסעדה חדשה.
Explotation ניצול: ניצול המדניות(policy) שנלמדה מפעולות עבר, ונקיטת הפעולה בעלת התגמול הגדול ביותר. למשל, אני הולך למסעדה הכי טובה שאני מכיר.

Q-learning

אלגוריתם RL לומד ערך Q עבור כל זוג (מצב, פעולה).
בדרך כלל בוחר פעולה אקראית בהסתברות אפסילון (חקר), ובוחר את הפעולה הטובה ביותר לפי ערכי Q הנוכחיים בהסתברות אחד מינוס אפסילון.
בהנחה שבפעם הבאה הסוכן יבצע את הפעולה האופטימלית לפי ערכי ה Q.

```
import gym
import numpy as np
import random
epsilon = 0.1  # Exploration
gamma = 0.95  # discount factor
alpha = 0.1  # learning rate
env = gym.make('FrozenLake-v0')  # create FrozenLake environment
env.reset()
# Q values, INIT with zeroes len(num of states, num of actions)
Q = np.zeros([env.observation_space.n, env.action_space.n])
for i in range(10000):  # number of games
    s = env.reset()
    done = False
    while not done:  # until game over, or max number of steps
        if random.random() < epsilon:
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

```
        a = env.action_space.sample()  # in prob of 0.1 take a random move
    else:
        a = np.argmax(Q[s, :])  # take the best action
    s_n, r, done, _ = env.step(a)  # do the best action
    Q[s, a] = Q[s, a] + alpha * (r + gamma * np.max(Q[s_n, :]) - Q[s, a])  # update the Q values
    s = s_n  # next step
```

תזכורת:

argmax –יחזיר את הפעולה הכי טובה (הארגומנט שממקסם)

max– יחזיר את הערך הכי טוב.

What Happens When |S| is very large?

נניח שיש לנו רשת שבהינתן מצב מחזירה את ערכי ה Q האפשריים עבור כל אחת מהפעולות.

היינו נותנים לה את המצב הנוכחי והיא היתה מחשבת את ערכי ה Q עבור כל פעולה אפשרית.

לאחר מכן היינו בוחרים בפעולה הממקסת את ערך ה Q הזה.

אבל איך נקבל רשת כזו?

אנו נהפוך את ערכי ה Q לבעיית Regression?

כלומר, אנו מנסים לחזות את ערכי ה Q (לכל פעולה) בהינתן מצב מסויים. אבל מאיפה משיגים את ערכי ה Q האמיתיים לאימונים?(y_)

סדר הפעולות:

- בצע את הפעולה (הממקסמת את ערכי ה Q).
- קבל את הפרס (r).
- קבל את המצב החדש (s_n).
- חשב את ערכי ה Q של המצב החדש, עבור כל אחת מהפעולות. q_n יציין את ה Q המרבי.

כעת ערך ה Q "האמיתי" (עבור צמד הפעולות הקודם הוא פשוט:

$$r + \gamma * q\_n$$

(ערכי ה Q האחרים, עבור הפעולות שלא בצענו ישארו ללא שינוי).

חישוב ה LOSS וכדומה.

The Pong game!

יש לנו כמות ענקית של מצבים אפשריים.

ו 6 פעולות אפשריות.

להתקנה: pip install gym[atari]

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

:Random player

```
import gym
import time
env = gym.make('Pong-v0')  # creates Pong game environment
env.reset()
for _ in range(1000):
    env.render()
    env.step(env.action_space.sample())  # random action
    time.sleep(0.03)
```

:linear regression

```
import gym
import tensorflow as tf
import random
import numpy as np
env = gym.make('Pong-v0')  # creates Pong game environment
epsilon = 0.1  # Exploration
gamma = 0.999  # discount factor
num_of_games = 100
# state in size [batch size = 1, Screen dimensions are 210x160, RGB]
state = tf.placeholder(tf.float32, shape=[1, 210, 160, 3])
# because we are doing linear regression we flatten the image
state_vec = tf.reshape(state, [1, 210 * 160 * 3])
W1 = tf.Variable(tf.truncated_normal([210 * 160 * 3, env.action_space.n], stddev=1e-5))
b1 = tf.Variable(tf.constant(0.0, shape=[env.action_space.n]))
Q4actions = tf.add(tf.matmul(state_vec, W1), b1)
Q_n = tf.placeholder(tf.float32, shape=[1, env.action_space.n])
loss = tf.pow(Q4actions - Q_n, 2)
update = tf.train.GradientDescentOptimizer(1e-10).minimize(loss)
sess = tf.Session()
sess.run(tf.global_variables_initializer())
for i in range(num_of_games):
    env.reset()
    s, _, done, _ = env.step(0)  # first move doesn't matter anyway, just get the state
    while not done:
        all_Qs = sess.run(Q4actions, feed_dict={state: [s]})
        if random.random() < epsilon:
            next_action = env.action_space.sample()
        else:
            next_action = np.argmax(all_Qs)
        s_n, r, done, _ = env.step(next_action)
        Q_corrected = np.copy(all_Qs)
        next_Q = sess.run(Q4actions, feed_dict={state: [s_n]})
        Q_corrected[0][next_action] = r + gamma * np.max(next_Q)
        sess.run(update, feed_dict={state: [s], Q_n: Q_corrected})
        s = s_n  # move to the next state
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

התוצאות:

לא ממש עובד.

הקלט הוא גדול מידי 210*160*3. אנו יכולים לעשות down-sample ולהמיר ל gray-scale ולקבל 80*150.

המודל שלנו פשוט מידי! אנחנו צרכים מודל מורכב יותר!

Deep CNN Regression

אנו נמיר את הרגרסיה הלינארית Qlearner ל CNN regression Q-learner. נבנה 2 convolution layers ו 2 fully connected layers. אנו נשמור על פריימים קודמים (כדי לדעת את כיווני התנועה). אזי המצב עבור ה CNN מורכב כעת מ 4 מצבים קודמים.

שיפורים נוספים:

- נתחילים לשחק באופן רנדומלי, את ה X המשחקים הראשונים.
- ההסתברות למהלך אקראי פוחתת עם הזמן.
- על מנת לזרז את הלמידה, אנו מאחסנים את ההיסטוריה של המהלכים, ומחשבים את ההפסד עבור סט של פריימים .(mini batch GD).
- הזנת חלק אחר מההיסטוריה בכל פעם (בחירה אקראית). ניתן להזין פריט מספר פעמים.
- יצירת 2 NN ראשי ויעד: ה NN הראשי ישמש למציאת פעולה, כמו כן גם את הפעולה הבאה המניבה את המקסימום. ה NN מטרה, משמש למציאת ה Qvalues "האמיתיים".

On-Policy RL

Q-learning היא שיטת למידה off-policy, כלומר היא אינה משתמשת במדיניות שלה בפועל כדי לחשב את ערכי ה- Q שלה.

ב Q-learning, ערכי Q אינם לוקחים בחשבון את ε (פעולות אקראיות).

(State Action Reward State Action) SARSA היא גרסה on-policy של Q-learning. ההבדל היחיד הוא בכך שהוא בוחר תחילה את הצעד הבא שלו (באמצעות ε) ורק לאחר מכן מחושב העדכון.

$$Q(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t,a_t)}_{\text{old value}} \right)$$

Value / Policy Based Methods

הן Q-Learning והן SARSA הן שיטות מבוססות ערך, כלומר הן מעריכות את הערך של כל מצב, וה policy מתקבל בעקיפין בלבד.

שיטות אלה סובלות לעיתים מהתכנסות לקויה.

ישנם מספר בעיות שלא ניתנות לפתרון באמצעות מדיניות דטרמיניסטית.

שיטות מבוססות מדיניות מנסות ללמוד ישירות את המדיניות ולנסות לשפר את המדיניות שלהן בזמן שהיא פועלת.

שיטות אלה מתכנסות פעמים רבות למקסימום מקומי.

ניתן להשתמש בשיטות מבוססות מדיניות לפעולות רצופות. (למשל מדיניות גאוסיינית).

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Softmax for Determining the Action

נניח שאנחנו משתמשים בסופטמקס פשוט כדי לקבוע איזו פעולה עלינו לנקוט בהמשך. אנו מחשבים שילוב ליניארי על הפיצ'רים שחולצו מהמצב (באמצעות משקלים נפרדים לכל פעולה), כדי להשיג את ה logits. לאחר מכן אנו מחשבים softmax ונבצע פעולה לפי ההסתברויות. באופן כללי נרצה להגדיל את ההסתברות של הפעולות המובילות לתשואה גבוהה ולהקטין את ההסתברות של הפעולות המובילות לתשואה נמוכה.

REINFORCE (Monte-Carlo Policy Gradient)

- Update parameters by stochastic gradient ascent
- Using policy gradient theorem
- Using return v_t as an unbiased sample of $Q^{\pi_\theta}(s_t, a_t)$

$$\Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$$

```
function REINFORCE
    Initialise θ arbitrarily
    for each episode {s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T} ~ π_θ do
        for t = 1 to T − 1 do
            θ ← θ + α∇_θ log π_θ(s_t, a_t) v_t
        end for
    end for
    return θ
end function
```

Policy Gradient Theorem

עבור MDP כללי, כל שעלינו לעשות הוא להחליף r בערך Q (בהתאם למדיניות).

$$\nabla J(\theta) = \mathbf{E}_{\pi_\theta}[\nabla log(\pi_\theta(s,a))r]$$

$$\nabla J(\theta) = \mathbf{E}_{\pi_\theta}[\nabla log(\pi_\theta(s,a))Q(s,a)]$$

Actor-critic

actor-critic היא שיטת on-policy RL שמנסה ללמוד את ה policy הטובה ביותר באופן ישיר (policy-based), אך גם ללמוד את הערך של כל מצב (כך שהיא גם מבוססת-ערך). לשחקן יש policy שהוא פועל לפיו. המבקר, מעריך את ערכה של כל מצב ומספק "critic" לשחקן, כך שהשחקן יוכל לעדכן את ה policy שלו (using gradient ascent). המבקר מעדכן את ערכה של המצב הישן לפי הפעולה שנקט השחקן.

Actor-Critic vs REINFORCE

Actor-Critic פועל רק על צעד אחד, בעוד ש REINFORCE זקוק לפרק מלא.

כלל העדכון של REINFORCE:

$$\theta \leftarrow \theta + \alpha \nabla_\theta log \pi_\theta(s_t, a_t) v_t$$

Actor-Critic משתמש באומדן של Q:

$$\theta \leftarrow \theta + \alpha \nabla_\theta log \pi_\theta(s_t, a_t) Q_W(s,a)$$

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Actor-Critic אלגוריתם:

- Using linear value fn approx. $Q_w(s,a) = \phi(s,a)^\top w$
  - Critic Updates w by linear TD(0)
  - Actor Updates θ by policy gradient

function QAC
  Initialise s, θ
  Sample $a \sim \pi_\theta$
  for each step **do**
    Sample reward $r = \mathcal{R}_s^a$; sample transition $s' \sim \mathcal{P}_{s,\cdot}^a$
    Sample action $a' \sim \pi_\theta(s', a')$
    $\delta = r + \gamma Q_w(s', a') - Q_w(s,a)$
    $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s,a) Q_w(s,a)$
    $w \leftarrow w + \beta \delta \phi(s,a)$
    $a \leftarrow a', s \leftarrow s'$
  end for
end function

סיכום

מדיניות (פוליסי) המצויינת על ידי π: פוליסי היא ההסתברות שפעולה a ננקטת במצב s.

יש לנו שני סוגי למידה:

1. evaluate Q(s,a): חוזה את סכום התגמולים בעתיד.(מבוססות ערך).
2. find π(s,a): שמניב תגמול מרבי.(מבוססות פוליסי).

On-policy and off-policy learning קשורות לסוג הלמידה הראשון, evaluate Q.

ההבדל ביניהם הוא:

- ב on-policy learning ה Q(s,a), הפונקציה נלמדת מפעולות שאנו עושים באמצעות הפוליסי הנוכחי שלנו π.
- ב off-policy learning ה $Q(s,a)$, הפונקציה נלמדת מפעולות שונות. אנחנו לא משתמשים בפוליסי בכלל.

עדכון הפונקציה עבור on-policy SARSA:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

כאשר a' היא פעולה שננקטה לפי הפוליסי π.

עדכון הפונקציה עבור off-policy Q-learning:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma * \max_{a'} Q(s',a') - Q(s,a))$$

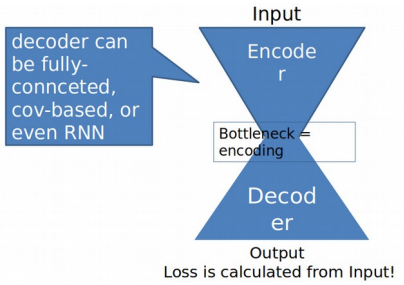
כאשר a' הן כל הפעולות במצב s'.

Softmax policy, REINFORCE, Actor critic קשורות לסוג הלמידה השני כלומר, פעולות מבוססות פוליסי.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות
נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Auto-Encoders

מה זה autoencoder ?
כאשר אנו קונים פריטים או שירותים באינטרנט, אנו מוודאים שהאתר מאובטח על ידי שימוש בפרוטוקול https. אנו מכניסים את פרטי כרטיס האשראי שלנו לשם הרכישה. פרטי כרטיסי האשראי שלנו מקודדים ברשת באמצעות אלגוריתם קידוד כלשהו. פרטי כרטיס האשראי המקודד מפוענחים כדי ליצור את מספר כרטיס האשראי המקורי לצורך אימות.
בדוגמה של כרטיס האשראי, לקחנו את פרטי כרטיס האשראי, קודדנו אותו באמצעות פונקציה כלשהי. מאוחר יותר פיענחו אותו באמצעות פונקציה אחרת כדי לשחזר את הפלט הזהה לקלט.
כך עובדים autoencoders.



Autoencoders משתמשים בקלט בתור ה labal שלהם, ולכן הם אינם דורשים תיוג כלשהו של הדאטה.
בעיה שאינה מצריכה תיוג נקראת unsupervised (בניגוד ללמידה supervised וזה מה שעשינו עד כה).
נתונים מתויגים הם בדרך כלל מה שיקר (בדרך כלל מתויג ידנית) ולכן בדרך כלל קל הרבה יותר לקבל נתונים ללא תווית.

Auto-Encoders' Code (embedding)
בשיטה זו יש לנו הרבה data ואנו נרצה ללמוד על המידע ללא labels. אופן חישוב פונצקיית ה loss היא הפער בין ה output ל input. נרצה שה output יהיה כמה שיותר דומה ל input. מה שמכריח את המודל לשמור את המידע הכי חשוב על הקלט על מנת שיוכל בהמשך לשחזר את הקידוד לפלט שיהיה כמה שיותר דומה לקלט (תלוי במטרה של המודל, נניח אם המטרה של המודל היא השלמת תמונות אז מה שנכניס למודל יהיה התמונה עם הפגם, ה loss יחושב בין פלט המודל לתמונה המקורית, ללא הפגם).
ניתן להשתמש ב Autoencoders לתמונות, אודיו, וידאו ועוד.
לדוגמה:
תמונות של ספרות (mnist): הקוד עשוי לכלול את הספרה, אך גם את הסוג שלה.
הקידוד עשוי לכלול מידע נוסף: רוחב ספרות, מיקום ספרות, ועוד.
בעלי חיים: הקוד עשוי לכלול את החיה, הגודל, המין וכדומה.
מוזיקה: הקוד עשוי לכלול סוג מוזיקה, עוצמת קול וכדומה.
הקוד מהמקודד באופן אוטומטי צריך להכיל את מהות הנתונים. ניתן להשתמש ב Autoencoders כדחיסה של דברים.
הקידוד עשוי להיות שימושי גם עבור חיפוש תמונות, ללא צורך ב labeling אנושי.
אנו עשויים לחשוב על קידוד זה כשיבוץ ממרחב תכונות אחד(פיקסלים) למשנהו (הקוד).
ניתן לחשוב על הקידוד כאל וקטור במרחב Hilbert ממימד n. שתי תמונות של אותו חתול בזווית שונה, צריכות להיות בעלות קוד דומה. באופן דומה, שתי תמונות של ספלים בעלי תכונות זהות פרט לצבעם צריכות גם להיות ממפות לווקטורים קרובים.

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Auto-Encoder for Mnist

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
#  -------  encoder: exactly as we've seen previously  -------
x = tf.placeholder(tf.float32, shape=[None, 784])
# y_ = tf.placeholder(tf.float32, shape=[None, 10]) #we don't provide the labeles!
W_conv1 = tf.Variable(tf.truncated_normal([5, 5, 1, 32], stddev=0.1))
b_conv1 = tf.Variable(tf.constant(0.1, shape=[32]))
x_image = tf.reshape(x, [-1, 28, 28, 1])  # if we had RGB, we would have 3 channels
h_conv1 = tf.nn.relu(tf.nn.conv2d(x_image, W_conv1, strides=[1, 2, 2, 1], padding='SAME') + b_conv1)
W_conv2 = tf.Variable(tf.truncated_normal([5, 5, 32, 64], stddev=0.1))
b_conv2 = tf.Variable(tf.constant(0.1, shape=[64]))
h_conv2 = tf.nn.relu(tf.nn.conv2d(h_conv1, W_conv2, strides=[1, 2, 2, 1], padding='SAME') + b_conv2)
W_fc1 = tf.Variable(tf.truncated_normal([7 * 7 * 64, 1024], stddev=0.1))
b_fc1 = tf.Variable(tf.constant(0.1, shape=[1024]))
h_pool2_flat = tf.reshape(h_conv2, [-1, 7 * 7 * 64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
W_fc2 = tf.Variable(tf.truncated_normal([1024, 10], stddev=0.1))
b_fc2 = tf.Variable(tf.constant(0.1, shape=[10]))
code = tf.nn.softmax(tf.matmul(h_fc1, W_fc2) + b_fc2)
#  --------  decoder: fully connected ----------
feed_code = tf.placeholder(tf.float32, shape=[None, 10])
W_dfc2 = tf.Variable(tf.truncated_normal([10, 1024], stddev=0.1))
b_dfc2 = tf.Variable(tf.constant(0.1, shape=[1024]))
h_dfc2 = tf.nn.relu(tf.matmul(code, W_dfc2) + b_dfc2)
h_dfc2_feed = tf.nn.relu(tf.matmul(feed_code, W_dfc2) + b_dfc2)
W_dfc1 = tf.Variable(tf.truncated_normal([1024, 28 * 28], stddev=0.1))
b_dfc1 = tf.Variable(tf.constant(0.1, shape=[28 * 28]))
x_dimage = tf.nn.relu(tf.matmul(h_dfc2, W_dfc1) + b_dfc1)
x_dimage_feed = tf.reshape(tf.nn.relu(tf.matmul(h_dfc2_feed, W_dfc1) + b_dfc1), [-1, 28, 28])
#  -------------  training: same as before but no labels  ------------
loss = tf.reduce_mean(tf.pow(x - x_dimage, 2))
update = tf.train.AdamOptimizer(1e-4).minimize(loss)
sess = tf.InteractiveSession()
sess.run(tf.global_variables_initializer())
for i in range(10000):
    batch = mnist.train.next_batch(50)
    if i % 100 == 0:
        curr_loss = loss.eval(feed_dict={x: batch[0]})
        print("step %d, loss %g" % (i, curr_loss))
    update.run(feed_dict={x: batch[0]})
```

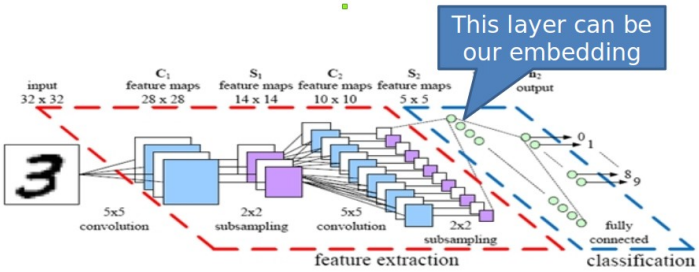
סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Embedding Obtained From a Classifier

אנו יכולים להשתמש באחת השכבות של neural network כקידוד (בדרך כלל זוהי השורה שלפני השכבה האחרונה). השימוש המקורי של ה neural network יכול להיות למטרה של סיווג או רגסיה.

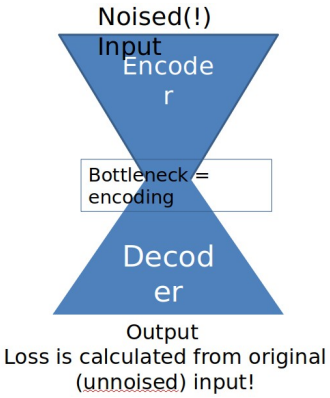


Denoised Auto-Encoder

מתייחס להוספה מכוונת של רעש לקלט לפני מתן הקלט לרשת.

זה עוזר להימנע מהעתקת הקלט לפלט מבלי ללמוד תכונות על הנתונים.

הוספת הרעש לקלט יכולה להיעשות באופן אקראי על ידי הפיכת חלק מהקלט לאפס. Denoised Auto-Encoder חייבים להסיר את הרעש על מנת ליצור פלט הדומה לקלט. הפלט משווה לקלט ללא הרעש.



סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור
מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Word embedding

מהם word embeddings ? למה משתמשים בהם ? לפי שנכנס לפרטים נתחיל מדוגמה קטנה:
ישנם אתרים רבים המבקשים מאיתנו לתת ביקורות או משוב על המוצר שיש לנו בעת השימוש בהם. כמו: אמזון, IMDB.
אנו משתמשים גם ב– Google כדי לחפש על ידי צירוף של כמה מילים ולקבל תוצאות שקשורות להם.
אז איך הם עושים את זה. למעשה הדברים האלה הם יישום של עיבוד טקסטים. אנו משתמשים בטקסט לניתוח רגשות, קיבוץ מילים דומות, סיווג מסמכים ותיוג.
איך אנו גורמים למחשבים לספר לכם על כדורגל או על רונאלדו כשאתם מחפשים את מסי ?
למשימות כמו זיהוי אובייקטים או דיבור אנו יודעים שכל המידע הנדרש לביצוע המשימה בהצלחה מקודד בנתונים. עם זאת, מערכות עיבוד שפות טבעיות מתייחסות באופן מסורתי למלים כסמלים אטומיים נפרדים, ולכן 'חתול' עשוי להיות מיוצג כ– קידוד אחד ו'כלב 'בקידוד שונה לגמרי. קידודים אלה הם שרירותיים ואינם מספקים מידע שימושי למערכת בנוגע לקשרים העשויים להתקיים בין הסמלים האישיים.
כאן נכנסת word embeddings. word embeddings אינן אלא ייצוג מספרי של טקסטים.
נחשוב על המילים כווקטורים. האם החתול יהיה קרוב לחתלתול ?
בדומה לקוד Auto-encoders.
מה זה: Girl – Boy + King ?
Queen

Word Embedding Applications
Word Embedding יכול להחליף את ה one-hot encoding הרגיל למילים. ניכר שהוא מניב תוצאות טובות יותר.
למילים בעלות משמעות קרובה צריכות להיות וקטורים קרובים.
מאפשר להפעיל CNN עבור משפטי טקסט.
ניתן להשתמש בחיפוש בסיס של מילות מפתח: האלגוריתם ההתאמה יכול לשקול גם מילים קרובות (מילים נרדפות).

Very Basic Word Embedding
כל מילה מיוצגת על ידי מספר שלם (אינדקס).
אנו נלמד מטריצה: שורה (ווקטור) לכל מילה.
אנו מגדירים משקולות (ו biases) ומשתמשים ברגרסיה softmax פשוטה כדי לעבור embedding להסתברויות שכל מילה תתרחש יחד עם מילים אחרות.
אנו ממזערים את הטעות שלנו. (נשתמש ב cross entropy).

Data:
(Royal:) queen king prince princess
(Plain:) woman man girl boy
(Peafowl:) peacock peahen peachick
(Adults:) queen king woman man peacock peahen

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

(Females:) queen princess woman girl peahen
(Males:) king prince man boy peacock
(Yong:) prince princess girl boy peachick

עבור התווית הנכונה (y_) נשתמש בשבר שייצג את מספר הפעמים שכל מילה מתרחשת יחד עם כל אחת מהמילים האחרות. למשל:

0 (queen) [3 2 1 2 2 1 1 0 1 2 0] / 15

Basic Word Embedding Implementation

```
import tensorflow as tf
import numpy as np
vocabulary_size = 11
embedding_size = 3
embeddings = tf.Variable(tf.random_uniform([vocabulary_size, embedding_size], -1.0, 1.0))
W1 = tf.Variable(tf.truncated_normal([embedding_size, vocabulary_size], stddev=0.1))
b1 = tf.Variable(tf.zeros([vocabulary_size]))
x = tf.placeholder(tf.int32, shape=[None])
y_ = tf.placeholder(tf.float32, shape=[None, vocabulary_size])
y = tf.nn.softmax(tf.matmul(tf.nn.embedding_lookup(embeddings, x), W1) + b1)
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))
update = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
x_data = np.array(range(vocabulary_size))
y_data = conv_data_to_labels(all_data)
sess = tf.Session()
sess.run(tf.global_variables_initializer())
for i in range(1000):
    sess.run(update, feed_dict={x: x_data, y_: y_data})
```

Where Does the Data Come From?

ה word embedding הפשוט שלנו הניח כי יש לנו קבוצות של מילים. בפועל נסמוך על קורפוס גדול של מסמכים. המודל skip-gram word2vec מנסה לחזות את תדירות התרחשות המילים בסביבת מילה נתונה.

בהינתן מילה, ננסה לנחש את המילים שבסביבתה.

Word2Vec המקורי אינו סופר תחילה את כל המופעים, אלא מחשב עבור כל חלון בנפרד.

במקום לנסות לחזות נכון כל מילה נכונה, אנו יכולים להוסיף מדגם של מילות רעש אקראיות ולנסות לחזות את המילה הנכונה מהסט.

ניתן להגדיר את הלוס בדרכים רבות ומגוונות, למשל ב- SoftMax השתמשנו ב cross entropy.

SpaCy

SpaCy היא ספריית שפות טבעיות בפיתון (בדומה ל- NLTK). יש לו Word2Vec מובנה.

```
>pip install spacy
>python -m spacy.en.download all
```

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

Back Propagation

Why Back-Propagation? Our Goal.

אנו רוצים לאמן את הרשת שלנו. כלומר, אנו רוצים לעדכן את המשקולות והביאס כך שפלט הרשת שלנו יהיה קרוב ככל האפשר לתוויות הנתונות.

כלומר, אנו רוצים למזער את ההפסד הנתון. למשל: $\frac{1}{2m}(y\_ - y)^2$, $-(y\_*\log(y)+(1-y\_)*\log(1-y))$

אנו משתמשים ב gradient descent.

What Do We Need For GD ?

אנחנו צרכים את הגרדיאנט של הרשת שלנו. כלומר, כיצד כל משתנה (weight or bias) משפיע על ה loss/error, שאותו כאמור אנו מנסים למזער.

בהינתן הגרדיאנט, נוכל לעבור על הדאטה ולעדכן: $w := w - \alpha \frac{\partial Error}{\partial w}$

(באופן דומה עבור ה biases).

עלינו לדעת: $\frac{\partial Error}{\partial w}$ and $\frac{\partial Error}{\partial b}$

עבור כל weight and bias ברשת שלנו.

סימונים

W^l_{jk}: weight at level l from node k (at level l-1) to node j (at level l)

b^l_j: bias of node j in level l

a^l_j: output of node j in level l (after activation) $a^l_j = \sigma(z^l_j)$

Z^l_j: input to activation: $z^l_j = \left(\sum_i a^{l-1}_i \cdot w^l_{ji}\right) + b^l_j$

Delta Error

השגיאה האחרונה (E) היא פשוט ההפסד שלנו. לדוגמה:

E= $\frac{1}{2m}(y\_ - y)^2$, $-(y\_*\log(y)+(1-y\_)*\log(1-y))$

נגדיר את δ^l_j בתור כמה הפלט לפני האקטיבציה של הצומת j שנמצא ברמה l משפיע על השגיאה הסופית E. כלומר, כמה כל שינוי של Z ישפיע על E.

השגיאה מוגדרת כך: $\delta^l_j = \frac{\partial E}{\partial z^l_j}$

נניח שיש לנו את δ^l_j עבור כל צומת ברשת, אזי נוכל לחשב את $\frac{\partial Error}{\partial w}$ and $\frac{\partial Error}{\partial b}$ באמצעות כלל השרשרת:

$$\frac{\partial E}{\partial b^l_j} = \frac{\partial E}{\partial z^l_j}\frac{\partial z^l_j}{\partial b^l_j} = \delta^l_j \frac{\partial(\sum_i a^{l-1}_i \cdot w^l_{ji}) + b^l_j}{\partial b^l_j} = \delta^l_j$$

$$\frac{\partial E}{\partial w^l_{jk}} = \frac{\partial E}{\partial z^l_j}\frac{\partial z^l_j}{\partial w^l_{jk}} = \delta^l_j \frac{\partial(\sum_i a^{l-1}_i \cdot w^l_{ji}) + b^l_j}{\partial w^l_{jk}} = \delta^l_j \cdot a^{l-1}_k$$

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

איך נשיג אבל את δ_j^l ??

נוכל לחשב את δ_j^l על סמך הרמה ה l+1:

$$\delta_j^l = \frac{\partial E}{\partial z_j^l} = \sum_i \frac{\partial E}{\partial z_i^{l+1}} \frac{\partial z_i^{l+1}}{\partial z_j^l} = \sum_i \delta_i^{l+1} \frac{\partial z_i^{l+1}}{\partial z_j^l}$$

$$= \sum_i \delta_i^{l+1} \frac{\partial \sum_k a_k^l \cdot w_{ik}^{l+1} + b_j^{l+1}}{\partial z_j^l}$$

$$= \sum_i \delta_i^{l+1} \frac{\partial a_j^l \cdot w_{ij}^{l+1}}{\partial z_j^l} = \sum_i \delta_i^{l+1} \sigma'(z_j^l) \cdot w_{ij}^{l+1}$$

Operations on Matrices

ניתן לחשב את כל הפעולות עבור כל רמה בבת אחת.

זה יעיל בהרבה הן מבחינה אלגוריתמית והן מבחינה חומרה עבור מעבדים נוכחיים ובעיקר עבור GPUs.

ציין על ידי δ^l את כל δ של רמה l (כלומר δ^l_1, δ^l_2 וכו ').

אנו יכולים (וצריכים) להשתמש ב mini-batches ולחשב את הממוצע על כל הדוגמאות ורק לאחר מכן לבצע שלב GD. זה מאפשר יתרונות נוספים באופטימזציה על מטריצות.

האלגוריתם

חזור עד להתכנסות:

- עבור כל דוגמה בדאטה (או יותר טוב עבור כל mini-batch):
  - בצע forward-propagation וקבל את כל ה Z,a.
  - חשב את E (loss)
  - חשב את δ^L
  - חשב את כל ה δ^l
  - חשב עבור כל W ו b: $\frac{\partial Error}{\partial w}$ and $\frac{\partial Error}{\partial b}$
  - בצע GD עדכון עבור כל W,b: $w := w - \alpha \frac{\partial Error}{\partial w}$

    $b := b - \alpha \frac{\partial Error}{\partial b}$

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה

1) Suppose an agent was in state s and took action a, received a reward of 6 and arrived at state s'. Assume the discount factor (gamma) is 0.9.

Suppose Q(s, a) = 17, $\arg\max_{a' \in A} Q(s', a') = a^*$ and Q(s', a*) = 10.

Assume we use quadratic loss (squared errors, loss = $(y-y\_)^2$), how much would the loss be in for Q(s,a)?

2) When training a Q-value network, is it possible to use more than a single example in every batch? (explain why it is not possible or how it is)

סיכום קורס למידה עמוקה ועיבוד שפות טבעיות

נכתב על ידי: שי נאור

מבוסס על ההרצאות והמצגות של דר' עמוס עזריה