CS 224N: Assignment 2

PETER888@STANFORD.EDU FRIDAY 2ND FEBRUARY, 2018

Problem 1: Tensorflow Softmax (25 points)

1.1 (a) Implement Softmax use Tensorflow (5 points, coding)

Answer:
See code: ∼/`q1_softmax.py`.

1.2 (b) Implement Cross-Entropy use Tensorflow (5 points, coding)

Answer:

See code: ∼/`q1_softmax.py`.

1.3 (c) Tensorflow Placeholder and Feed Dictionary (5 points, coding/written)

Answer:

See code: ∼/`q1_classifier.py`.

Explanation:

1.4 (d) Implement Classifier (5 points, coding)

Answer:
See code: ~/q1_classifier.py.

1.5 (e) Implement Model (5 points, coding/written)

Answer:
See code: ~/q1_classifier.py.

Explanation:

Problem 2: Neural Transition-Based Dependency Parsing (50 points + 2 bonus points)

2.1 (a) Dependency Parsing (6 points, written)

Answer:

stack	**buffer**	**new dependency**	**transition**
$[ROOT]$	$[I, parsed, this, sentence, correctly]$		Initial Configuration
$[ROOT, I]$	$[parsed, this, sentence, correctly]$		`SHIFT`
$[ROOT, I, parsed]$	$[this, sentence, correctly]$		`SHIFT`
$[ROOT, parsed]$	$[this, sentence, correctly]$	parsed → I	`LEFT-ARC`
$[ROOT, parsed, this]$	$[sentence, correctly]$		`SHIFT`
$[ROOT, parsed, this, sentence]$	$[correctly]$		`SHIFT`
$[ROOT, parsed, sentence]$	$[correctly]$	sentence → this	`LEFT-ARC`
$[ROOT, parsed]$	$[correctly]$	parsed → sentence	`RIGHT-ARC`
$[ROOT, parsed, correctly]$	[]		`SHIFT`
$[ROOT, parsed]$	[]	parsed → correctly	`RIGHT-ARC`
$[ROOT]$	[]	ROOT → parsed	`RIGHT-ARC`

2.2 (b) How many steps (2 points, written)

Answer:

$2n$ *parse steps. Because each word take exactly one shift transition from buffer to stack, take exactly one *-ARC (either LEFT-ARC or RIGHT-ARC) transition move out from stack, and every transition either add or remove word in the stack.*

2.3 (c) Parser Step (6 points, coding)

See code: ~/`q2_parser_transitions.py`.

2.4 (d) Parser Transitions (6 points, coding)

See code: ~/`q2_parser_transitions.py`.

2.5 (e) Xavier Initialization (4 points, coding)

See code: ~/`q2_initialization.py`.

2.6 (f) Dropout (2 points, written)

Answer:

$$\gamma = \frac{1}{1 - p_{drop}}$$

The mask vector $\boldsymbol{d}$ *set entries in* $\boldsymbol{h}$ *to zero at probability* p_{drop}*, let's say* $\boldsymbol{h}$ *has full expectation value* $\mathbb{E}[\boldsymbol{h}]$*=1, so* $\mathbb{E}[\boldsymbol{d} \circ \boldsymbol{h}]$*=(1-*$p_{drop}$*), this because* p_{drop} *of* $\boldsymbol{h}$ *values are become zero.*
So, $1 = \frac{\mathbb{E}[\boldsymbol{d} \circ \boldsymbol{h}]}{(1-p_{drop})} \Rightarrow 1 = \mathbb{E}[\boldsymbol{h}_{drop}] = \frac{1}{(1-p_{drop})}\mathbb{E}[\boldsymbol{d} \circ \boldsymbol{h}] \Rightarrow \gamma = \frac{1}{1-p_{drop}}$

2.7 (g) Adam Optimizer (4 points, written)

2.7.1 i) Momentum

Answer:

2.7.2 ii) Adaptive Learning Rates

Answer:

2.8 (h) Parser Model (20 points, coding/written)

Answer:
See code: ~/`q2_parser_model.py`. *Report the best UAS*
The dev UAS: 87.83, the test UAS: 88.09
List of predicted labels, see file $q2_test.predicted.pkl$

2.9 (i) Bonus (2 points, coding/written)

Answer: Implemented L2 regularization. And attempt

Problem 3: Recurrent Neural Networks (25 points + 1 bonus point)

3.1 (a) Perplexity (4 points, written)

3.1.1 i) Derive Perplexity (2 points)

Answer:

As $\boldsymbol{y}^{(t)}$ is an one-hot vector, suppose the k-th element $y_k^{(t)}$ is 1

so,

$\boldsymbol{J}^{(t)} = -log(\hat{y}_k^{(t)}) = log(\frac{1}{\hat{y}_k^{(t)}})$

and we also have

$PP^{(t)}(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = \frac{1}{\hat{y}_k^{(t)}}$

thus, we have

$\boldsymbol{J}^{(t)} = log(PP^{(t)}(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}))$

We also know $-log(x)$ is a strict convex function about x, and $\frac{1}{x}$ also a strict convex function (given $x > 0$) about x

so, below two optimal problem will be equal

$min{-}log(\hat{y}_k^{(t)})$

$min\frac{1}{\hat{y}_k^{(t)}}$

3.1.2 ii) Equivalent (1 point)

Answer:

3.1.3 iii) Perplexity for a single word (1 point)

Answer:

3.2 (b) Gradients on Single Point (7 points, written)

Answer:

$$
\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{v}_c} &= \frac{\partial}{\partial \boldsymbol{v}_c}((-log(\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c))) - \sum_{k=1}^{K} log(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c))) \\
&= -\frac{\partial}{\partial \boldsymbol{v}_c}(log(\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c))) - \sum_{k=1}^{K} \frac{\partial}{\partial \boldsymbol{v}_c} log(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)) \\
&= \frac{-1}{\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c)}(\frac{\partial}{\partial \boldsymbol{v}_c}\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c)) - \sum_{k=1}^{K} \frac{1}{\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)}\frac{\partial}{\partial \boldsymbol{v}_c}\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c) \\
&= -(1-\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c))\boldsymbol{u}_o - \sum_{k=1}^{K}(1-\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c))(-\boldsymbol{u}_k) \\
&= (\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c)-1)\boldsymbol{u}_o - \sum_{k=1}^{K}(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)-1)\boldsymbol{u}_k
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{u}_o} &= \frac{\partial}{\partial \boldsymbol{u}_o}((-log(\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c))) - \sum_{k=1}^{K} log(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c))) \\
&= \frac{-1}{\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c)}(\frac{\partial}{\partial \boldsymbol{u}_o}\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c)) - 0 \\
&= -(1-\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c))\frac{\partial}{\partial \boldsymbol{u}_o}(\boldsymbol{u}_o^T\boldsymbol{v}_c) \\
&= (\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c)-1)\boldsymbol{u}_o \\
\frac{\partial J}{\partial \boldsymbol{u}_k} &= \frac{\partial}{\partial \boldsymbol{u}_k}((-log(\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c))) - \sum_{k=1}^{K} log(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c))) \\
&= 0 - \frac{\partial}{\partial \boldsymbol{u}_k} log(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)) \\
&= -\frac{1}{\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)}\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)(1-\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c))\frac{\partial}{\partial \boldsymbol{u}_k}(-\boldsymbol{u}_k^T\boldsymbol{v}_c) \\
&= -(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)-1)\boldsymbol{v}_c) \\
&= (\sigma(\boldsymbol{u}_k^T\boldsymbol{v}_c)\boldsymbol{v}_c, \text{ for all } k \neq o
\end{aligned}
$$

Negative sampling is faster than softmax-CE loss at speed up ratio $\frac{V}{K}$, where V is all words in dictionary count, and K is the sampling size.

3.3 (c) Gradients (7 points, written)

Answer:

3.4 (d) How Many Operations for Single Timestep (3 points, written)

Answer:

Derivatives for the skip-gram model

$$\frac{J_{skip-gram}(word_{c-m\ldots c+m})}{\partial \boldsymbol{v}_c} = \sum_{-m\leqslant j\leqslant m, j\neq 0} \frac{\partial F(\boldsymbol{w}_{c+j}, \boldsymbol{v}_c)}{\partial \boldsymbol{v}_c}$$
$$\frac{J_{skip-gram}(word_{c-m\ldots c+m})}{\partial \boldsymbol{v}_j} = 0, \forall j \neq c$$
$$\frac{J_{skip-gram}(word_{c-m\ldots c+m})}{\partial \boldsymbol{U}} = \sum_{-m\leqslant j\leqslant m, j\neq 0} \frac{\partial F(\boldsymbol{w}_{c+j}, \boldsymbol{v}_c)}{\partial \boldsymbol{U}}$$

Derivatives for the CBOW model

$$\frac{J_{CBOW}(word_{c-m\ldots c+m})}{\partial \boldsymbol{v}_j} = \frac{\partial F(\boldsymbol{w}_c, \hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}}, \forall (j \neq c) \in \{c-m\ldots c+m\}$$
$$\frac{J_{CBOW}(word_{c-m\ldots c+m})}{\partial \boldsymbol{v}_j} = 0, \forall (j \neq c) \notin \{c-m\ldots c+m\}$$
$$\frac{J_{CBOW}(word_{c-m\ldots c+m})}{\partial \boldsymbol{U}} = \frac{\partial F(\boldsymbol{w}_c, \hat{\boldsymbol{v}})}{\partial \boldsymbol{U}}$$

3.5 (e) How Many Operations for Entire Sequence (3 points, written)

Answer: See code: ∼/`q3_word2vec.py`.

3.6 (f) Which largest? Term RNN? (1 point, written)

Answer: See code: ∼/`q3_sgd.py`.

3.7 (g) Bonus (1 point, written)

Answer:

Explain: In the Word Vectors image, words clustered at similarity, such as the emotion words ”amazing”, ”wonderful” and ”great” are very close to each other. The word ”well” a little further but still close to ”amazing” , the connection characters and words ”the” ”a” ”,” etc are spread around alone.