
CS 224N: Assignment 2

PETER888@STANFORD.EDU

SUNDAY 28TH JANUARY, 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
[<i>ROOT</i>]	[<i>I, parsed, this, sentence, correctly</i>]		Initial Configuration
[<i>ROOT, I</i>]	[<i>parsed, this, sentence, correctly</i>]		SHIFT
[<i>ROOT, I, parsed</i>]	[<i>this, sentence, correctly</i>]		SHIFT
[<i>ROOT, parsed</i>]	[<i>this, sentence, correctly</i>]	<i>parsed</i> → <i>I</i>	LEFT-ARC
[<i>ROOT, parsed, this</i>]	[<i>sentence, correctly</i>]		SHIFT
[<i>ROOT, parsed, this, sentence</i>]	[<i>correctly</i>]		SHIFT
[<i>ROOT, parsed, sentence</i>]	[<i>correctly</i>]	<i>sentence</i> → <i>this</i>	LEFT-ARC
[<i>ROOT, parsed</i>]	[<i>correctly</i>]	<i>parsed</i> → <i>sentence</i>	RIGHT-ARC
[<i>ROOT, parsed, correctly</i>]	[]		SHIFT
[<i>ROOT, parsed</i>]	[]	<i>parsed</i> → <i>correctly</i>	RIGHT-ARC
[<i>ROOT</i>]	[]	<i>ROOT</i> → <i>parsed</i>	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 \mathbf{d} set entries in \mathbf{h} to zero at probability p_{drop} , let's say \mathbf{h} has full expectation value $\mathbb{E}[\mathbf{h}] = 1$, so $\mathbb{E}[\mathbf{d} \circ \mathbf{h}] = (1 - p_{drop})$, this because p_{drop} of \mathbf{h} values are become zero.

$$\text{So, } 1 = \frac{\mathbb{E}[\mathbf{d} \circ \mathbf{h}]}{(1 - p_{drop})} \Rightarrow 1 = \mathbb{E}[\mathbf{h}_{drop}] = \frac{1}{(1 - p_{drop})} \mathbb{E}[\mathbf{d} \circ \mathbf{h}] \Rightarrow \gamma = \frac{1}{1 - p_{drop}}$$

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
List of predicted labels

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

Answer:

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:

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:

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

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

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

$$\frac{J_{\text{skip-gram}}(\text{word}_{c-m\dots c+m})}{\partial \mathbf{v}_c} = \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial F(\mathbf{w}_{c+j}, \mathbf{v}_c)}{\partial \mathbf{v}_c}$$

$$\frac{J_{\text{skip-gram}}(\text{word}_{c-m\dots c+m})}{\partial \mathbf{v}_j} = 0, \forall j \neq c$$

$$\frac{J_{\text{skip-gram}}(\text{word}_{c-m\dots c+m})}{\partial \mathbf{U}} = \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial F(\mathbf{w}_{c+j}, \mathbf{v}_c)}{\partial \mathbf{U}}$$

Derivatives for the CBOW model

$$\frac{J_{\text{CBOW}}(\text{word}_{c-m\dots c+m})}{\partial \mathbf{v}_j} = \frac{\partial F(\mathbf{w}_c, \hat{\mathbf{v}})}{\partial \hat{\mathbf{v}}}, \forall (j \neq c) \in \{c-m\dots c+m\}$$

$$\frac{J_{\text{CBOW}}(\text{word}_{c-m\dots c+m})}{\partial \mathbf{v}_j} = 0, \forall (j \neq c) \notin \{c-m\dots c+m\}$$

$$\frac{J_{\text{CBOW}}(\text{word}_{c-m\dots c+m})}{\partial \mathbf{U}} = \frac{\partial F(\mathbf{w}_c, \hat{\mathbf{v}})}{\partial \mathbf{U}}$$

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.