CS 224N: Assignment 2

PETER888@STANFORD.EDU SUNDAY 4TH 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:
Placeholder variables and feed dictionary make it possible to feed external data (such as training data for network) into Tensorflow's computational graph.

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:
When the model's $train\_op$ is called. Computation going forward through the computation graph with Placeholder's input and the init/start values in Variables. The Tensorflow automatic differentiation will start the backward propagation, it will automatically calculate the gradients and differentiations, and update the Variables back through the computation graph.

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:

The rolling average, m*, can reduce the variance in updates, by down-weighting the impact of the current minibatch's gradient. For instance, if* β *is 0.5, when updating* $\boldsymbol{\theta}$*, it will use half of past rolling average* m*, and half of current minibatch's gradient. Consider if the current minibatch's gradient varies too larger or too smaller than the last rolling average* m*, the momentum will reduce the larger or smaller part of the gradient to just half to get new* m*, and use the reduced new* m *to do update.*

2.7.2 ii) Adaptive Learning Rates

Answer:

The parameters with smaller gradients (especial rolling average smaller) will get larger updates. For parameters been changing flatly, the adaptive learning rate will help to these parameter get larger updates, and helping these parameter move off plateaus. This can increase learning speed.

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

Answer:

See code: ∼/`q2_parser_model.py`.

Report the best UAS

The best dev UAS: 88.11, and it's test UAS: 88.31

List of predicted labels, see file $q2\_test.predicted.pkl$

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

Answer:

Implemented L2 regularization. The best dev UAS: 87.49, and it's test UAS: 87.86. There is no significant improvement by using L2 regularization. This could be the overfit not due to parameters values, need to tune other hyper-parameters such as hidden layer size, or dropout rate.

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, assume the k-th element $y_k^{(t)}$ is 1

so,

$$\boldsymbol{J}^{(t)}(\boldsymbol{\theta}) = -log(\hat{y}_k^{(t)}) = log(\frac{1}{\hat{y}_k^{(t)}}) \tag{3.1}$$

and we also have

$$PP^{(t)}(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = \frac{1}{\hat{y}_k^{(t)}} \tag{3.2}$$

thus, we have

$$\boldsymbol{J}^{(t)}(\boldsymbol{\theta}) = log(PP^{(t)}(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)})) \tag{3.3}$$

3.1.2 ii) Equivalent (1 point)

Answer:
We know $min\{log(f(x)\} = min\{f(x)\}, f(x) > 0$

then we have,

$$min\{\boldsymbol{J}^{(t)}(\boldsymbol{\theta}) = log(PP(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)})) : \boldsymbol{\theta} > 0\} = min\{PP^{(t)}(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)})\} \forall (t \in [1..T])$$

from convex theory not hard to know minimizing the geometric mean equivalent to minimizing the arithmetic mean if the related function have same minimizing equivalent.

$min\{(\prod_{j=1}^{T} f_j(x))^{\frac{1}{T}}\} = min\{\frac{1}{T}\sum_{i=1}^{T}(g_i(x))\}$*, when* $min\{\boldsymbol{f}(x)\} = min\{\boldsymbol{g}(x)\}$*, for* $\boldsymbol{f}$*,*$\boldsymbol{g}$ *are positive functions*

Finally, we can get the minimizing geometric mean perplexity equivalent to minimizing the arithmetic mean cross-entropy loss

$$min\{(\prod_{t=1}^{T} PP(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}))^{\frac{1}{T}}\} = min\{\frac{1}{T}\sum_{t=1}^{T} CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)})\}$$

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

Answer:
for given word $\boldsymbol{\omega}_j$,

$$\bar{P}\left(\boldsymbol{x}^{(t+1)} = \boldsymbol{\omega}_j | \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)}\right) = \frac{1}{|V|}$$

so the perplexity for that single word $\boldsymbol{\omega}_j$, is $|V|$

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

because, $\boldsymbol{J}^{(t)}(\boldsymbol{\theta}) = log(PP^{(t)}(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}))$, when $|V|$=10000

$$\boldsymbol{J}^{(t)}(\boldsymbol{\theta}) = log(|V|) \approx 9.213$$

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

Answer:

$$\boldsymbol{\delta}_{\mathbf{1}}^{(\boldsymbol{t})} = \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\theta}^{(t)}} = -\mathbf{y} + \hat{\mathbf{y}} \tag{3.4}$$

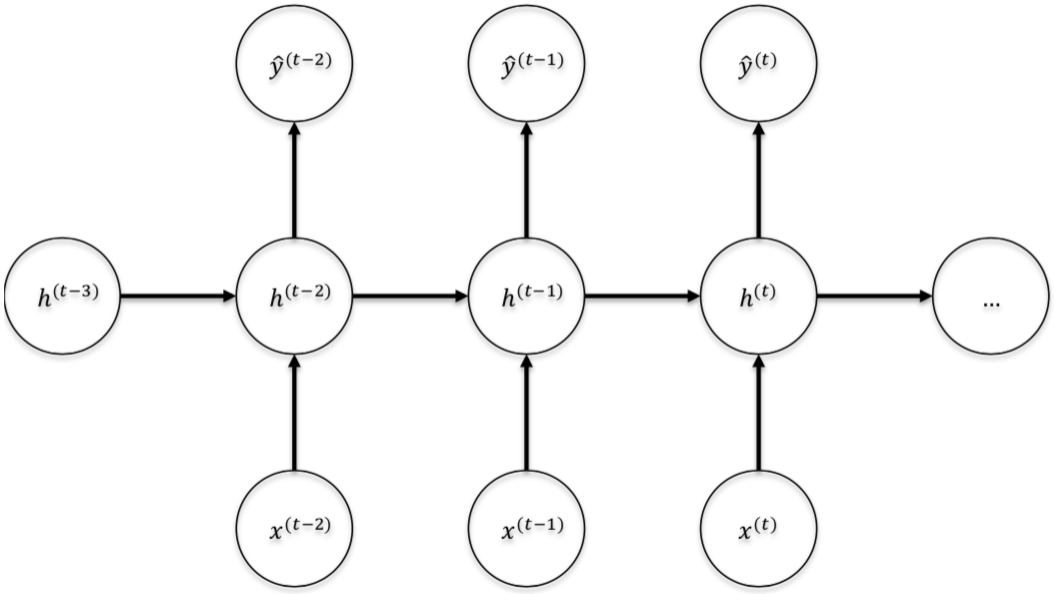
Write sigmoid(x) as $\sigma(x)$, and it's derivative as $\sigma^{'}(x)$

$$\begin{aligned}
\boldsymbol{\delta}_{\mathbf{2}}^{(\boldsymbol{t})} &= \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{z}^{(t)}} = \boldsymbol{\delta}_{\mathbf{1}}^{(\boldsymbol{t})} \frac{\partial \boldsymbol{\theta}^{(t)}}{\partial \boldsymbol{h}^{(t)}} \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{z}^{(t)}} \\
&= \boldsymbol{U}^T \cdot \boldsymbol{\delta}_{\mathbf{1}}^{(\boldsymbol{t})} \circ \sigma^{'}(\boldsymbol{z}^{(t)}) \\
&= \boldsymbol{U}^T \cdot \boldsymbol{\delta}_{\mathbf{1}}^{(\boldsymbol{t})} \circ \boldsymbol{h}^{(t)} \circ (1 - \boldsymbol{h}^{(t)}) \qquad (3.5) \\
\frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{U}} &= \frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{\theta}^{(t)}} \frac{\partial \boldsymbol{\theta}^{(t)}}{\partial \boldsymbol{U}} = \boldsymbol{\delta}_{\mathbf{1}}^{(\boldsymbol{t})} (\boldsymbol{h}^{(t)})^T \qquad (3.6) \\
\frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{e}^{(t)}} &= \frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{z}^{(t)}} \frac{\partial \boldsymbol{z}^{(t)}}{\partial \boldsymbol{e}^{(t)}} = (\boldsymbol{W}_e|_{(t)})^T \boldsymbol{\delta}_{\mathbf{2}}^{(\boldsymbol{t})} \qquad (3.7) \\
\left.\frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{W}_{\boldsymbol{e}}}\right|_{(t)} &= \frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{z}^{(t)}} \left.\frac{\partial \boldsymbol{z}^{(t)}}{\partial \boldsymbol{W}_{\boldsymbol{e}}}\right|_{(t)} = \boldsymbol{\delta}_{\mathbf{2}}^{(\boldsymbol{t})} (\boldsymbol{e}^{(t)})^T \qquad (3.8) \\
\left.\frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{W}_h}\right|_{(t)} &= \frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{z}^{(t)}} \left.\frac{\partial \boldsymbol{z}^{(t)}}{\partial \boldsymbol{W}_h}\right|_{(t)} = \boldsymbol{\delta}_{\mathbf{2}}^{(\boldsymbol{t})} (\boldsymbol{h}^{(t-1)})^T \qquad (3.9) \\
\frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}} &= \frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{z}^{(t)}} \frac{\partial \boldsymbol{z}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}} = (\boldsymbol{W}_h|_{(t)})^T \boldsymbol{\delta}_{\mathbf{2}}^{(\boldsymbol{t})} \qquad (3.10)
\end{aligned}$$

3.3 (c) Gradients (7 points, written)

Answer:

Figure 3.1: Unrolled RNN



We know:

$$\boldsymbol{z}^{(t-1)} = \boldsymbol{W}_h|_{(t-1)}\boldsymbol{h}^{(t-2)} + \boldsymbol{W}_e|_{(t-1)}\boldsymbol{e}^{(t-1)} + \boldsymbol{b}_1|_{(t-1)} \tag{3.11}$$

$$\boldsymbol{h}^{(t-1)} = sigmoid(\boldsymbol{z}^{(t-1)}) = \sigma(\boldsymbol{z}^{(t-1)}) \tag{3.12}$$

$$\boldsymbol{\gamma}^{(t-1)} = \frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}} \tag{3.13}$$

Then,

$$\begin{aligned}\boldsymbol{\delta}_{\boldsymbol{2}}^{(\boldsymbol{t-1})} &= \frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{z}^{(t-1)}} = \frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}}\frac{\partial \boldsymbol{h}^{(t-1)}}{\partial \boldsymbol{z}^{(t-1)}} \\ &= \boldsymbol{\gamma}^{(t-1)} \circ \sigma^{'}(\boldsymbol{z}^{(t-1)}) \\ &= \boldsymbol{\gamma}^{(t-1)} \circ \boldsymbol{h}^{(t-1)} \circ (1 - \boldsymbol{h}^{(t-1)})\end{aligned} \tag{3.14}$$

$$\frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{e}^{(t-1)}} = \frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{z}^{(t-1)}}\frac{\partial \boldsymbol{z}^{(t-1)}}{\partial \boldsymbol{e}^{(t-1)}} = (\boldsymbol{W}_e|_{(t-1)})^T \boldsymbol{\delta}_{\boldsymbol{2}}^{(\boldsymbol{t-1})} \tag{3.15}$$

$$\left.\frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{W}_{\boldsymbol{e}}}\right|_{(t-1)} = \left.\frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{z}^{(t-1)}}\frac{\partial \boldsymbol{z}^{(t-1)}}{\partial \boldsymbol{W}_{\boldsymbol{e}}}\right|_{(t-1)} = \boldsymbol{\delta}_{\boldsymbol{2}}^{(\boldsymbol{t-1})}(\boldsymbol{e}^{(t-1)})^T \tag{3.16}$$

$$\left.\frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{W}_h}\right|_{(t-1)} = \left.\frac{\partial \boldsymbol{J}^{(t)}}{\partial \boldsymbol{z}^{(t-1)}}\frac{\partial \boldsymbol{z}^{(t-1)}}{\partial \boldsymbol{W}_h}\right|_{(t-1)} = \boldsymbol{\delta}_{\boldsymbol{2}}^{(\boldsymbol{t-1})}(\boldsymbol{h}^{(t-2)})^T \tag{3.17}$$

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

Answer:

Go through each equations in part (b), those backpropagation's operations are $O(|V| \times D_h)$, $O(d \times D_h)$ and $O(D_h \times D_h)$, adding up all those operations, for a single step the operation count is $\boldsymbol{O}(|V| \times D_h + d \times D_h + D_h \times D_h)$

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

Answer:

Go through each backpropagation equations operation in part (c), the operation count also $\boldsymbol{O}(|V| \times D_h + d \times D_h + D_h \times D_h)$. So for T words, the total operation count is $\boldsymbol{O}(T \times (|V| \times D_h + d \times D_h + D_h \times D_h))$

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

Answer:

$\boldsymbol{O}(|V| \times D_h)$ term is the largest. Word dictionary $|V|$is normally large, and $D_h << |V|$. It happens when backpropagate from output layer to hidden layer.

3.7 (g) Bonus (1 point, written)

Answer:

If we can change the operation count from $\boldsymbol{O}(T \times (|V| \times D_h + d \times D_h + D_h \times D_h))$ to $\boldsymbol{O}(T \times (C \times D_h + d \times D_h + D_h \times D_h))$, by mapping words to classes, and C is the number of classes, because $C < |V|$, this will speed up the calculation. And this approach suggested by the paper "Externsions of recurrent neural network language model" (Mikolov, et al. 2013)