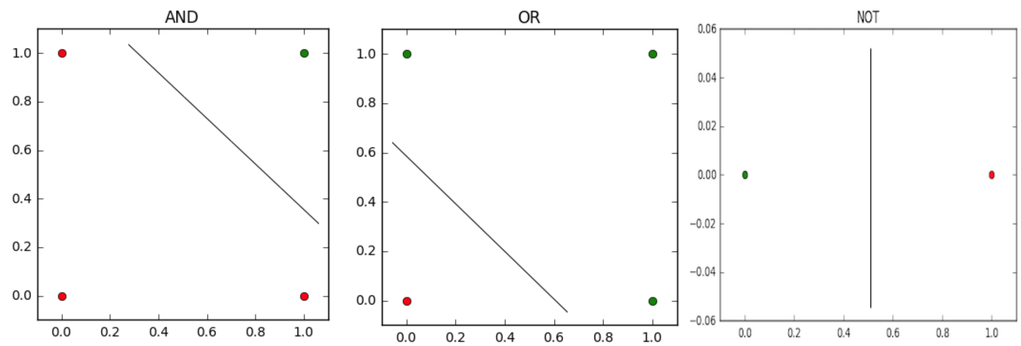
Deep Learning for NLP (236601) - HW2

Segev Arbiv

0 Warmup: Boolean Logic

0.1 a

$$
\begin{aligned}
A \oplus B &= (A \wedge \neg B) \vee (\neg A \wedge B) && (0.1) \\
&= (A \vee B) \wedge \neg (A \wedge B) && (0.2)
\end{aligned}
$$



(a) (b) (c)

Figure 0.1: Single layer classifier for boolean operatos

0.2 b

We will use the hint given for adjusting our weights and bias for the given step activation function $h_i(x, y) = \theta(w_{i1}x + w_{i2}y + b_i)$ in the following way:

NOT: $w_{i1} = -1, b_i = 0.5$

AND: $w_{i1} = w_{i2} = 1, b_i = -1.5$

OR: $w_{i1} = w_{i2} = 1, b_i = -0.5$

1 Deep Networks for Named Entity Recognition

(a)

We will use the fact that in HW1 we already computed a lot of this derivation, let us recall that:

$$\frac{\partial J(\theta)}{\partial \theta} = \hat{y} - y$$

And let us denote the following: $z_1 = Wx^{(t)} + b_1$, $z_2 = Uh + b_2$, calculating the derivative simply becomes:

$$\begin{aligned}
\frac{\partial J(\theta)}{\partial U} &= \frac{\partial J(\theta)}{\partial z_2}\frac{\partial z_2}{\partial U} &(1.1)\\
&= (\hat{y} - y)h^T &(1.2)\\
&= \delta^2 h^T \in R^{5\times 100} &(1.3)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial J(\theta)}{\partial b_2} &= \frac{\partial J(\theta)}{\partial z_2}\frac{\partial z_2}{\partial b_2} &(1.4)\\
&= \delta^2 \in R^5 &(1.5)
\end{aligned}$$

By recalling the definition of backpropagation that: $\frac{\partial J}{\partial z_i} = \delta^i$, and the recursive formula of the error $\delta^i = ((W^i)^T \delta^{i+1}) \circ f'(z^i)$, we get:

$$\begin{aligned}
\frac{\partial J(\theta)}{\partial W} &= \frac{\partial J(\theta)}{\partial z_1}\frac{\partial z_1}{\partial W} &(1.6)\\
&= \delta^1 (x^{(t)})^T &(1.7)\\
&= \left((U^T\delta^2) \circ (\tanh(z_1))'\right)(x^{(t)})^T &(1.8)\\
&= \left((U^T\delta^2) \circ (1 - \tanh^2(z_1))\right)(x^{(t)})^T \in R^{100\times 150} &(1.9)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial J(\theta)}{\partial b_1} &= \frac{\partial J(\theta)}{\partial z_1}\frac{\partial z_1}{\partial b_1} &(1.10)\\
&= \delta^1 \in R^{100} &(1.11)
\end{aligned}$$

While L_i is the input of the net, as in HW1:

$$\frac{\partial J(\theta)}{\partial L_i} = W^T \delta^1 \in R^{50} \tag{1.12}$$

(b)

Now, we wish to compute derivatives to the problem with the regularization term. As the regularization term only depends on W and U, the derivatives w.r.t b_1, b_2, L_i are simply 0.

As for the other variables we can see that:

$$\frac{\partial J_{reg}}{\partial W} = \lambda W \tag{1.13}$$

$$\frac{\partial J_{reg}}{\partial U} = \lambda U \tag{1.14}$$

Finally, all we nee to do is add the derivatives from 1 to achieve (for some variable X):

$$\frac{\partial J_{full}(\theta)}{\partial X} = \frac{\partial J(\theta)}{\partial X} + \frac{\partial J_{reg}(\theta)}{\partial X}$$

(c)

the function *random weight matrix(m,n)* in misc.py was implemented.

(d)

Notice, Ive changed the *softmax* function in the math.py file so it can work with matrices (same implementation as in HW1).
part1-NER.ipynb was done, NER model created.

As it states that no brute-force search is needed, I played with the parameters until 80.71% $F1$ accuracy achieved on the dev data. The parameters are listed at table 1.

Param	Value
Regularization	0.001
dimensions	[100,50]
Learning rate - α	0.01 + annhealing every 20k steps
SGD batch size - k	15

Table 1: Model optimal hyperparameters

(e)

In 1.1 we can see a plot of the learning curve for the best model.

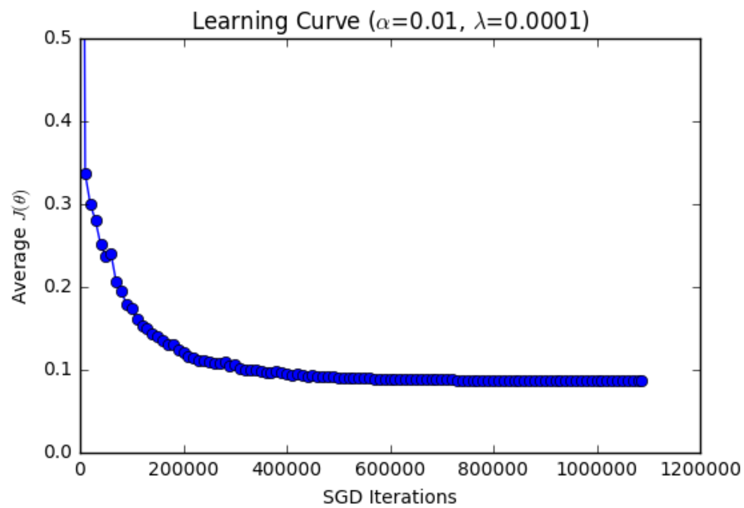


Figure 1.1: Best model that was trained

In 1.2 there is a plot comparing $\alpha = 0.01$ to $\alpha = 0.1$.

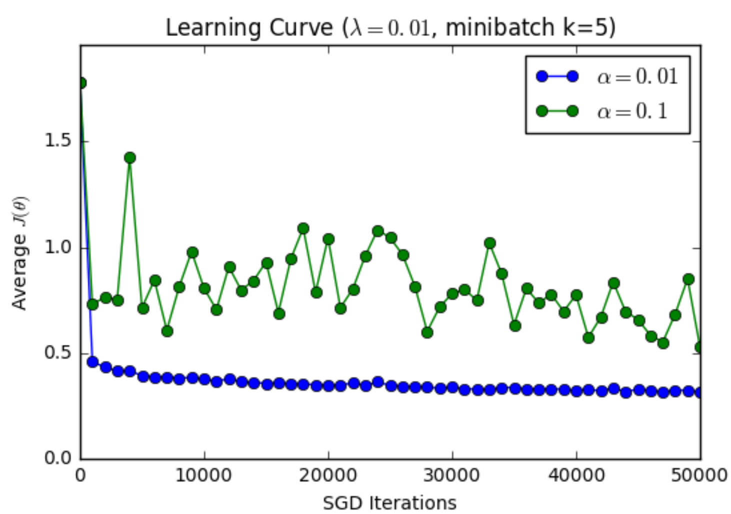


Figure 1.2: Comparing learning rates

We can clearly see at 1.2 that if the model tries to learn to fast (i.e $\alpha = 0.1$), it might skip/miss a local optimum that may also be a global one.

(f)

The performance of our model on the dev set (as output by eval performance() can be seen at 1.3.

	precision	recall	f1-score	support
O	0.97	0.99	0.98	42759
LOC	0.89	0.84	0.86	2094
MISC	0.88	0.71	0.79	1268
ORG	0.74	0.64	0.69	2092
PER	0.89	0.83	0.86	3149
avg / total	0.95	0.95	0.95	51362

```
=== Performance (omitting 'O' class) ===
Mean precision:  84.98%
Mean recall:     76.96%
Mean F1:         80.71%
```

Figure 1.3: Results on dev set

The list of predicted labels for the test set are attached in the file test.predicted.

1.1 Deep Networks: Probing Neuron Responses

(a)

Here we examined the "responses" of certain neurons on the hidden layer, to the input words. The top-10 word lists for the centre word, on 5 chosen hidden layer neurons are listed in at table 2

Neuron	Words
1	starts, before, leaving, least, how, "nt", nothing, very, too, ¡/s¿
20	appearing, zealand, spite, philippines, france, spain, italy, le, netherlands, referred
40	angeles, governments, studios, &, who, agencies, boards, districts, kong, francisco
60	league, norwegian, communist, german, foreign, socialist, african, asian, coalition, broadcast
80	estimate, would, how, [, missed, example, estimated, why, instance, ¡/s¿

Table 2: Top-10 word lists for the center word

Interesting to see that specific neurons relate to a narrow range of word/concepts. For example, neuron 20 relates to countries, neuron 60 relates to nationality.

(b)

The top-10 word lists for the centre word, on model output for PER, ORG, LOC, and MISC are listed in the table 3

NER	Words
LOC	malaysia, austria, mexico, russia, france, italy, china, spain, indonesia, japan
MISC	turkish, austrian, german, ottoman, brazilian, swedish, iranian, belgian, italian, danish
ORG	engineering, college, computing, library, institute, arts, magnet, commons, department, campus
PER	martin, daniel, thompson, adam, gazing, sarah, dejected, wept, trembling, innocence

Table 3: Top-10 word lists for the center word

(c)

The top-10 word lists for the first word (preceding the centre word), on model output for PER, ORG, LOC, and MISC are listed in the table 4

We can see a clear connection between the centre word and the preceding word. For example, we can see relations of *(direction, state)*, *(firstname, surename)*, *(middlename,*

NER	Words
LOC	governed, born, southeast, san, near, los, visiting, attended, at, southwest
MISC	17th, medieval, ¡/s¿, 20th, resemble, 16th, century, nineteenth, twentieth, comprises
ORG	sphere, disney, bay, corporation, cemetery, forum, transit, liberty, behalf, v
PER	thomas, pat, sarah, uncle, anthony, joseph, e., ray, samuel, aunt

Table 4: The top-10 word lists for the first word

surename), *(family relation, name)* etc. Furthermore, preceding words for locations may be "visited","near", specific direction etc.

2 Recurrent Neural Networks: Language Modeling

(a)

As the *exp* function is monotonic, optimizing $J(\theta)$ is equivalent to optimizing $2^{J(\theta)}$. Hence:

$$\begin{align}
2^{J(\theta)} &= 2^{-\sum_{j=1}^{|V|} y_j \log \hat{y}_j} \tag{2.1}\\
&= \frac{1}{2^{\sum_{j=1}^{|V|} y_j \log \hat{y}_j}} \tag{2.2}\\
&= \frac{1}{\prod_{j=1}^{|V|} 2^{\log \hat{y}_j^{y_j}}} \tag{2.3}\\
&= \frac{1}{\prod_{j=1}^{|V|} \hat{y}_j^{y_j}} \tag{2.4}\\
&\underbrace{=}_{y_j=1\text{hot}} \frac{1}{\sum_{j=1}^{|V|} y_j \cdot \hat{y}_j} \tag{2.5}\\
&= PP\left(\hat{y}, y\right) \tag{2.6}
\end{align}$$

Summing the above over all the samples, results in **multiplying** the corresponding PP terms, which means that minimizing the arithmetic mean of the CE loss is equivalent of minimizng the geometric mean of the PP.

Completely random predictions results in assigning $\frac{1}{|V|}$ value for each of the elements in $\hat{y}_j$, which results in PP of $\frac{1}{\sum_{j=1}^{|V|} \frac{1}{|V|}} = 1$. As for the cross-entropy:

$$J(\theta) = -\sum_{j=1}^{|V|} \log \frac{1}{|V|} = -|V| \log \frac{1}{|V|} = |V| \log |V|$$

Which yields 6602 and 4000 for $|V|$=2000 and $|V|$=10000, respectively.

(b)

Similarly to 1, let us denote the following terms that will help us with the gradient notations:

$$z_1^{(t)} = Hh^{(t-1)} + Lx^{(t)}$$

$$z_2^{(t)} = Uh^{(t)}$$

$$\delta_i^{(t)} = \frac{\partial J^{(t)}}{\partial z_1^{(i)}}$$

Note that the current error $\delta_i^{(t)}$ depends (and propogates) to former times $i \leq t$ (if subscript is omitted, then its referring to the same time (t)).
Let us start with the "simplest" derivative:

$$\frac{\partial J^{(t)}}{\partial U} = \frac{\partial J^{(t)}}{\partial z_2^{(t)}} \frac{\partial z_2^{(t)}}{\partial U} \tag{2.7}$$

$$= (\hat{y}^{(t)} - y^{(t)}) h^{(t)T} \tag{2.8}$$

$$\tag{2.9}$$

Once the error is defined as above, we can assist the known backpropagation rule (example)

$$\delta_i^{(t)} = (H^T \delta_i^{t+1}) \circ f'(z_1^{(t)})$$

Hence,

$$\frac{\partial J^{(t)}}{\partial L} = \frac{\partial J^{(t)}}{\partial z_1^{(t)}} \frac{\partial z_1^{(t)}}{\partial L} \tag{2.10}$$

$$= \delta^{(t)} x^{(t)} \tag{2.11}$$

As currently we're not required of summing the gradients for H, we can calculate:

$$\frac{\partial J^{(t)}}{\partial H}|_{(t)} = \frac{\partial J^{(t)}}{\partial z_1^{(t)}} \frac{\partial z_1^{(t)}}{\partial H} \tag{2.12}$$

$$= \delta^{(t)} h^{(t-1)T} \tag{2.13}$$

(c)

Using the backpropogtion of the error term δ, we can write:

$$\frac{\partial J^{(t)}}{\partial L_{x^{(t-1)}}} = \frac{\partial J^{(t)}}{\partial z_1^{(t-1)}} \frac{\partial z_1^{(t-1)}}{\partial L_{x^{(t-1)}}} \tag{2.14}$$

$$= \delta_{t-1}^{(t)} x^{(t-1)} \tag{2.15}$$

$$\frac{\partial J^{(t)}}{\partial H}|_{(t-1)} = \frac{\partial J^{(t)}}{\partial z_1^{(t-1)}} \frac{\partial z_1^{(t-1)}}{\partial H} \tag{2.16}$$

$$= \delta_{t-1}^{(t)} h^{(t-2)T} \tag{2.17}$$

(d)

In the forward propagation part, we are executing the following multiplications:

$$Hh \in O(D_h^2)\ ,\ Lx \in O(D_h|V|)$$

So under the assumption that $|V| >> D_h$, the total forwards propagartion takes us $O(D_h|V|)$

In the backward propagation, theres the derivative calculation (assuming $O(1)$), and the multiplication of the error δ with H, U, which takes us:

$$O(D_h|V| + D_h^2) = O(D_h|V|)$$

Back propagating for τ times results in many more $H\delta$ multiplications:

$$O(D_h|V| + \tau D_h^2) = O(D_h|V|)$$

We can see that the bottleneck here comes from the size of the vocabulary $|V|$.

(e)

rnnlm.py was implemented.

(f)

As the training sessions takes a while, the tuning was a bit infeasible. I used the following hyper parameters which resulted in a cost of 4.712 on the train set. The parameters are listed at 5

Param	Value
backprop timesteps	3
Learning rate - α	0.1
SGD batch size - k	10
PP on dev set	21.484

Table 5: Model optimal hyperparameters

(g)

Unigram-filling was implemented, and here are few of the sentences that been generated:

- *¡s¿ " the labor oil commission of its dollar l. represents mr. coal all-out reopened loss of the financial reported the dollar 's idiots of the prefers and homes mr. stadiums and reported last year . ¡/s¿*

- *¡s¿ variables , a sardonic accepted itself and flying norwegian , how he says it have sneakers internal industry also think never been either from both ¡/s¿*

- *¡s¿ but the month are the only home , the good well to confiscated agencies magazine hbo is deaths their young in ranging judge as an ghost , chairman of acquired for date park , maria some of them , and asset news . ¡/s¿*

We can see that there is little sense in the above, yet the syntax seems to be alright (commas, dots etc.)