Name - Mudit Jain
Id - muditjai@
Doc - CS224n, HW1 | Late Day Used - 1
Collaborator - Marcello Hasegawa

1 (a) RHS = softmax$(x + c)$ where x is a vector

$$\text{softmax}(x+c)_i = \frac{e^{x_i + c}}{\sum_j e^{x_j + c}}$$

$$= \frac{e^{c} e^{x_i}}{\sum_j e^{x_j} e^{c}}$$

$$= \frac{\cancel{e^{c}} e^{x_i}}{\cancel{e^{c}} \sum_j e^{x_j}} = \text{softmax}(x)_i$$

$\therefore \forall i \quad \text{softmax}(x+c)_i = \text{softmax}(x)_i$

$\Rightarrow \underline{\underline{\text{softmax}(x+c) = \text{softmax}(x)}}$

2 (a) $\sigma(x) = \dfrac{1}{1+e^{-x}}$

$$\frac{d\sigma}{dx} = \frac{d\left(1+e^{-x}\right)^{-1}}{dx} = \frac{(-1)(-1)\,e^{-x}}{\left(1+e^{-x}\right)^{2}}$$

$$= \frac{e^{-x}}{\left(1+e^{-x}\right)} \frac{1}{\left(1+e^{-x}\right)}$$

$$= \frac{\left(1+e^{-x}-1\right)}{\left(1+e^{-x}\right)} \sigma(x)$$

$$= \left(1 - \frac{1}{1+e^{-x}}\right) \sigma(x)$$

$$= \left(1-\sigma(x)\right)\left(\sigma(x)\right)$$

$$\therefore \quad \underline{\underline{\sigma' = \sigma(1-\sigma)}}$$

2 (b) $CE(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i)$

$= -y_K \log(\hat{y}_K)$

$= -\log(\hat{y}_K)$ since y is a 1-hot vector with $y_K = 1$

Let $\theta = [\theta_1, \ldots, \theta_N]$

then $y_K = \dfrac{e^{\theta_K}}{\sum_{i=1}^{N} e^{\theta_i}}$

$$CE(y, \hat{y}) = -\log \frac{e^{\theta_K}}{\sum_{i=1}^{N} e^{\theta_i}} = -\theta_K + \log \sum_i e^{\theta_i}$$

$$\frac{\partial CE}{\partial \theta} = \left[\frac{\partial CE}{\partial \theta_1}, \ldots, \frac{\partial CE}{\partial \theta_K}, \ldots, \frac{\partial CE}{\partial \theta_N}\right]$$

$$\frac{\partial CE}{\partial \theta_K} = \frac{\partial\left(-\theta_K + \log \sum_i e^{\theta_i}\right)}{\partial \theta_K} = -1 + \frac{1 \cdot e^{\theta_i}}{\sum_i e^{\theta_i}}$$

$$= \underline{\underline{-1 + \hat{y}_K}}$$

$\forall j \; 1 \leq j \leq N, j \neq K, \; \dfrac{\partial CE}{\partial \theta_j} = 0 + \dfrac{e^{\theta_j}}{\sum_i e^{\theta_i}} = \underline{\underline{\hat{y}_j}}$

$$\therefore \nabla_\theta CE = \underline{\underline{[\hat{y}_1, \ldots, (-1+\hat{y}_K), \ldots, \hat{y}_N]}} = \hat{y} - y$$

2(c) Let $h_1 = \sigma(xW_1 + b_1)$

$$h_2 = h_1 W_2 + b_2$$
$$\hat{y} = \text{softmax}(h_2)$$
$$J = CE(\hat{y}, y)$$

From 2(b) $\frac{\partial J}{\partial h_2} = (\hat{y} - y)$

From 2(a) $\frac{\partial \sigma(x)}{\partial (x)} = \sigma' = \sigma(1-\sigma)$

Now $\frac{\partial h_2}{\partial h_1}$ is a matrix. Let's look at its ij^{th} element

$$\left(\frac{\partial h_2}{\partial h_1}\right)_{ij} = \frac{\partial h_2^i}{\partial h_1^j} = \frac{\partial \sum_t h_1^t W_2^{ti}}{\partial h_1^j} = W_2^{ji}$$

$$\Rightarrow \frac{\partial h_2}{\partial h_1} = W_2^T$$

similarly $\frac{\partial h_1}{\partial x} = \frac{\partial \sigma(xW_1 + b_1)}{\partial x} = \sigma' W_1^T$

$$= \sigma(1-\sigma) W_1^T$$

Applying Chain rule ~~on~~ for all terms

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial x} = \underline{\underline{(\hat{y} - y) W_2^T \sigma(1-\sigma) W_1^T}}$$

2(d) Total params

$$= sz(W_1) + sz(b_1) + sz(W_2) + sz(b_2)$$

where sz = size of matrix or vector

$$= \underline{\underline{D_x H + H + H D_y + D_y}}$$

3(a) $J = CE(y, \hat{y}) = -\log(\hat{y}_o) = -\log \dfrac{e^{U_o^T V_c}}{\sum_w e^{U_w^T V_c}}$

$$= -U_o^T V_c + \log \sum_w e^{U_w^T V_c}$$

$$\frac{\partial J}{\partial V_c} = -U_o + \frac{1}{\sum_w e^{U_w^T V_c}} \left(\sum_w e^{U_w^T V_c} \frac{\partial U_w^T V_c}{\partial V_c} \right)$$

$$= \boxed{-U_o + \frac{\sum_w U_w e^{U_w^T V_c}}{\sum_w e^{U_w^T V_c}}}$$

(Using ~~vec~~ vetor calculus rule $\dfrac{\partial a^T b}{\partial b} = a$)

3(b) Lets derive separately for U_o & other U_w

$$\frac{\partial J}{\partial U_o} = \frac{\partial}{\partial U_o}\left(-U_o^T V_c + \log \sum_w e^{U_w^T V_c}\right)$$

$$= -V_c + \frac{1}{\sum_w e^{U_w^T V_c}} e^{U_o^T V_c} V_c$$

$$= \boxed{-V_c + \frac{V_c e^{U_o^T V_c}}{\sum_w e^{U_w^T V_c}}}$$

$\forall\, \cancel{w}\, i \; \cancel{w}\, i \neq 0 \quad \dfrac{\partial J}{\partial U_i} = \dfrac{\partial \left(-U_o^T V_c + \log \sum_w e^{U_w^T V_c} \right)}{\partial U_i}$

$$= 0 + \frac{1}{\sum_w e^{U_w^T V_c}} \cdot e^{U_i^T V_c}\, V_c$$

$$= \boxed{\frac{V_c\, e^{U_i^T V_c}}{\sum_w e^{U_w^T V_c}}}$$

3(c) $J_{neg\ sample} = -\log(\sigma(U_o^T V_c)) - \sum_{k=1}^{K} \log(\sigma(-U_k^T V_c))$

$$= -\log(\sigma(U_o^T V_c)) - \sum_{k} \log(1-\sigma(U_k^T V_c))$$

Using $\sigma(-x) = \frac{1}{1+e^{+x}} = \frac{e^{-x}}{1+e^{-x}} = \frac{e^{-x}+1-1}{1+e^{-x}} = 1-\frac{1}{1+e^{-x}} = (1-\sigma)$

$$\frac{\partial J}{\partial V_c} = \frac{(-1)\cancel{(\sigma(U_o^T V_c))}(1-\sigma(U_o^T V_c))\, U_o}{\cancel{(\sigma(U_o^T V_c))}}$$

$$- \sum_{k} \frac{1}{\cancel{(1-\sigma(U_k^T V_c))}} (-1)\,\sigma(U_k^T V_c)\cancel{(1-\sigma(U_k^T V_c))}\, U_k$$

(using $\sigma' = (1-\sigma)\sigma$ & $\frac{\partial a^T b}{\partial b} = a$)

$$\frac{\partial J}{\partial V_c} = \cancel{(1-\sigma)} \boxed{-(1-\sigma(U_o^T V_c))\,U_o + \sum_{k} \sigma(U_k^T V_c)\, U_k}$$

For U_i we separately calculate U_o, U_i $i \in \{1 \ldots k\}$, U_i $i \in$ remaining

$$\frac{\partial J}{\partial U_o} = \frac{(-1)\cancel{\sigma(U_o^T V_c)}(1-\sigma(U_o^T V_c))\,V_c}{\cancel{\sigma(U_o^T V_c)}} + 0$$

since Σ doesn't have U_o terms

$$= \boxed{-(1-\sigma(U_o^T V_c))\,V_c}$$

$$\frac{\partial J}{\partial U_i}\Big|_{i \in \{1 \ldots k\}} = 0 + \sum_{k} \frac{1}{\cancel{(1-\sigma)}}\,\sigma\cancel{(1-\sigma)}\,V_c = \boxed{\sum_{i \in \{1..k\}} \sigma(U_i^T V_c)\,V_c}$$

Not a fn of U_i

wherever ~~k=i~~ i is in K samples

For U_i
$i \notin \{1 \ldots K\}$

$$\frac{\partial J}{\partial U_i} = 0 + 0 = \boxed{0}$$

there's no U_i in either side

Negative sampling is efficient since it ~~not~~ needs to look at K samples only, whereas softmax looks at W (vocab.size) words.

Hence speedup is $O\left(\frac{W}{K}\right)$.

3(d) For skipgram

$$J_{skipgram} = \sum_{\substack{-m \le j \le m \\ j \ne 0}} F(w_{c+j}, v_c)$$

$$\frac{\partial J}{\partial v_c} = \frac{\partial \sum_j F(w_{c+j}, v_c)}{\partial v_c}$$

$$= \boxed{\sum_{\substack{-m \le j \le m \\ j \ne 0}} \frac{\partial F(w_{c+j}, v_c)}{\partial v_c}}$$

Similarly for U_i

$$\frac{\partial J}{\partial U_i} = \boxed{\sum_{\substack{-m \le j \le m \\ j \ne 0}} \frac{\partial F(w_{c+j}, v_c)}{\partial U_i}}$$

$\therefore$ Both $\frac{\partial J}{\partial v_c}$ & $\frac{\partial J}{\partial U_i}$ are simple summations of the pair-wise form before.

For CBOW

$$J_{CBOW} = F(w_c, \hat{v}) \quad \text{where } \hat{v} = \sum_{\substack{-m \le j \le m \\ j \ne 0}} v_{c+j}$$

Here ~~only~~ we calculate $\frac{\partial J_{CBOW}}{\partial \hat{v}}$ then propagate to v_{c+j} through chain rule.

$$\frac{\partial J_{CBOW}}{\partial \hat{v}} = \frac{\partial F(w_c, \hat{v})}{\partial \hat{v}}$$

Using chain rule if $a+b=c$ & $J=f(c)$ then $\frac{\partial J}{\partial c} = \frac{\partial J}{\partial a} = \frac{\partial J}{\partial b}$

$$\therefore \frac{\partial J}{\partial \hat{v}} = \frac{\partial J}{\partial v_{c+j}} \quad \forall -m \le j \le m, \; j \ne 0$$

$$\frac{\partial J}{\partial v_{c+j}} = \boxed{\sum_{\substack{\hat{v} \text{ contains} \\ v_{c+j}}} \frac{\partial F(w_c, \hat{v})}{\partial \hat{v}}}$$

This form is same as $\frac{\partial F(o, v_c)}{\partial v_c}$ before & can be calculated similarly.

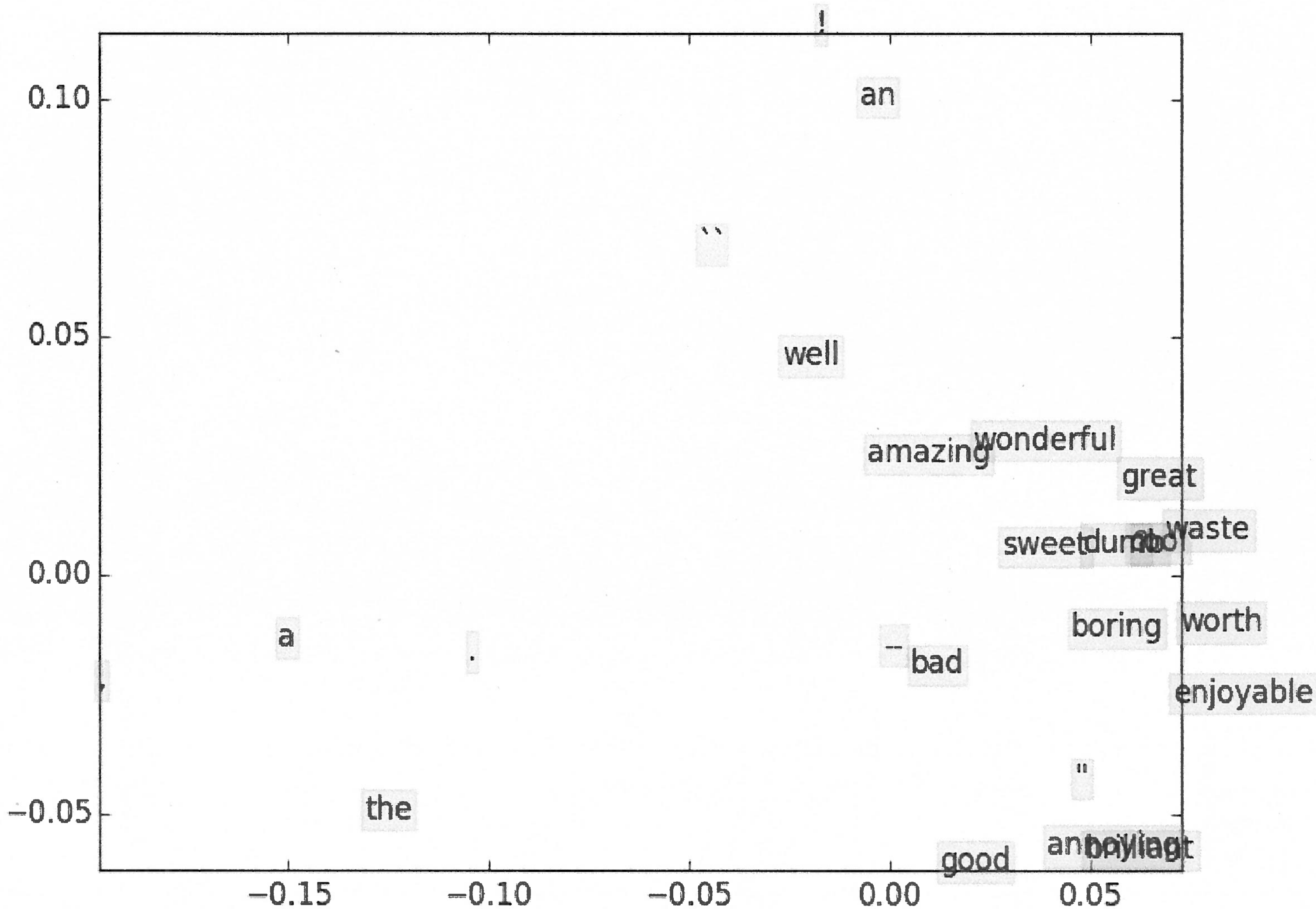
For other v_i $\boxed{\frac{\partial J}{\partial v_i} = 0}$

$$\boxed{\frac{\partial J}{\partial U_i} = \frac{\partial F(w_c, \hat{v})}{\partial U_i}}$$

This again is similar form as before.

Means summation for all occurances of v_{c+j} in $\hat{v}$.

3(g) Plot observations 1. We ~~plot~~ project along the 1st 2 dims of U. Adjective words are close to each other in a cluster. Punctuation words are far away. Adjective words are not well separated based on meaning ie +ve or -ve.



4(b) Regularization ~~so~~ is needed to prevent overfitting to training set & allow better generalization ie higher performance on test set. It does so by restricting the param values from going too high.

4(c)

```python
def chooseBestModel(results):
    """Choose the best model based on parameter tuning on the dev set

    Arguments:
    results -- A list of python dictionaries of the following format:
        {
            "reg": regularization,
            "clf": classifier,
            "train": trainAccuracy,
            "dev": devAccuracy,
            "test": testAccuracy
        }

    Returns:
    Your chosen result dictionary.
    """
    bestResult = {}

    ### YOUR CODE HERE
    bestResult["dev"] = -np.inf
    for result in results:
        if result["dev"] > bestResult["dev"]:
            bestResult = result
    ### END YOUR CODE

    return bestResult
```

q4(d)

--yourvectors

=== Recap ===

Reg	Train	Dev	Test
1.00E-04	31.051	32.516	30.498
1.00E-03	31.145	32.516	30.452
1.00E-02	30.946	32.334	29.910
1.00E-01	30.290	31.880	29.819
2.00E-01	29.846	32.153	29.502
5.00E-01	29.260	31.244	28.326
1.00E+00	28.897	29.609	27.149
2.00E+00	27.914	26.794	25.204
5.00E+00	27.353	25.522	23.213
1.00E+01	27.247	25.522	23.077
2.00E+01	27.235	25.522	23.032
4.00E+01	27.247	25.522	23.032
5.00E+01	27.247	25.522	23.032
6.00E+01	27.247	25.522	23.032
1.00E+02	27.247	25.522	23.032
2.00E+02	27.247	25.522	23.032
5.00E+02	27.247	25.522	23.032
1.00E+03	27.247	25.522	23.032
1.00E+04	27.247	25.522	23.032
2.00E+04	27.247	25.522	23.032
5.00E+04	27.247	25.522	23.032
1.00E+05	27.247	25.522	23.032

Best regularization value: 1.00E-04

Test accuracy (%): 30.497738

--pretrained vectors

=== Recap ===

Reg	Train	Dev	Test
1.00E-04	39.923	36.512	37.014
1.00E-03	39.911	36.421	37.014
1.00E-02	39.934	36.331	37.195
1.00E-01	39.794	36.240	37.149
2.00E-01	39.759	36.421	37.330
5.00E-01	39.618	36.149	37.376
1.00E+00	39.525	36.603	37.330
2.00E+00	39.490	36.876	37.240
5.00E+00	39.010	36.785	37.376
1.00E+01	38.624	36.876	37.692

2.00E+01	38.237	36.694	37.014
4.00E+01	37.441	36.603	36.335
5.00E+01	37.114	36.421	36.199
6.00E+01	36.880	36.058	36.154
1.00E+02	36.330	35.059	35.701
2.00E+02	35.042	34.060	34.434
5.00E+02	33.509	32.425	33.213
1.00E+03	32.163	31.153	30.588
1.00E+04	27.271	25.613	23.122
2.00E+04	27.247	25.522	23.032
5.00E+04	27.247	25.522	23.032
1.00E+05	27.247	25.522	23.032

Best regularization value: 2.00E+00
Test accuracy (%): 37.239819

Best accuracies of trained modd & glove are highlighted.

Best trained model accuracy - 30.49%

Best Glove modll accuracy - 37.23%

Reasons for glove performance -

1. Trained on bigger dataset ie Wiki vs Stanford data

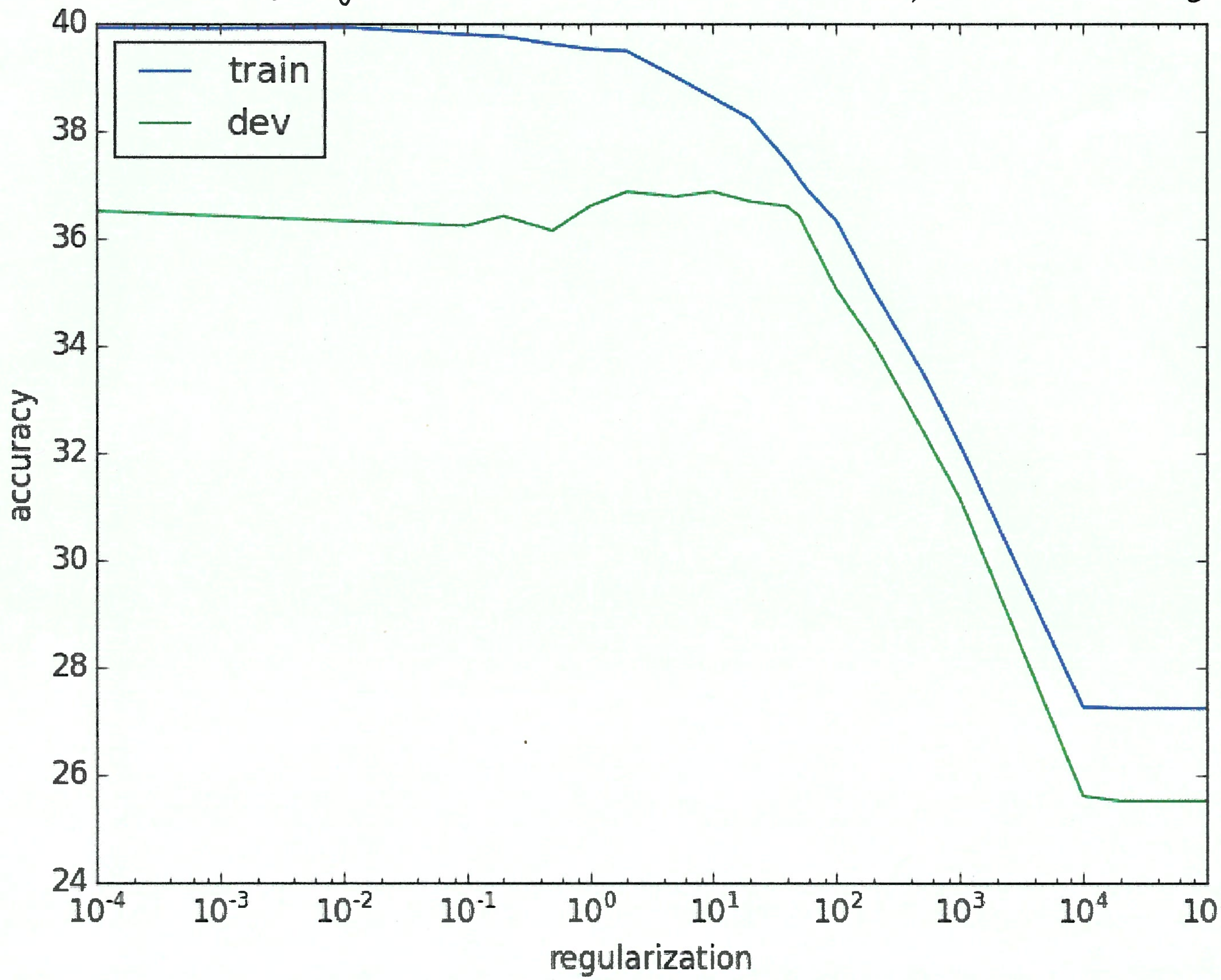
2. Uses a better cost fn to incorporate a global view

3. Perhaps better hyperparam tuning eg num ~~iter~~ iterations, regularization, feature dimensions etc.

4(e) 1. Training accuracy is higher than dev accuracy since we trained on it.

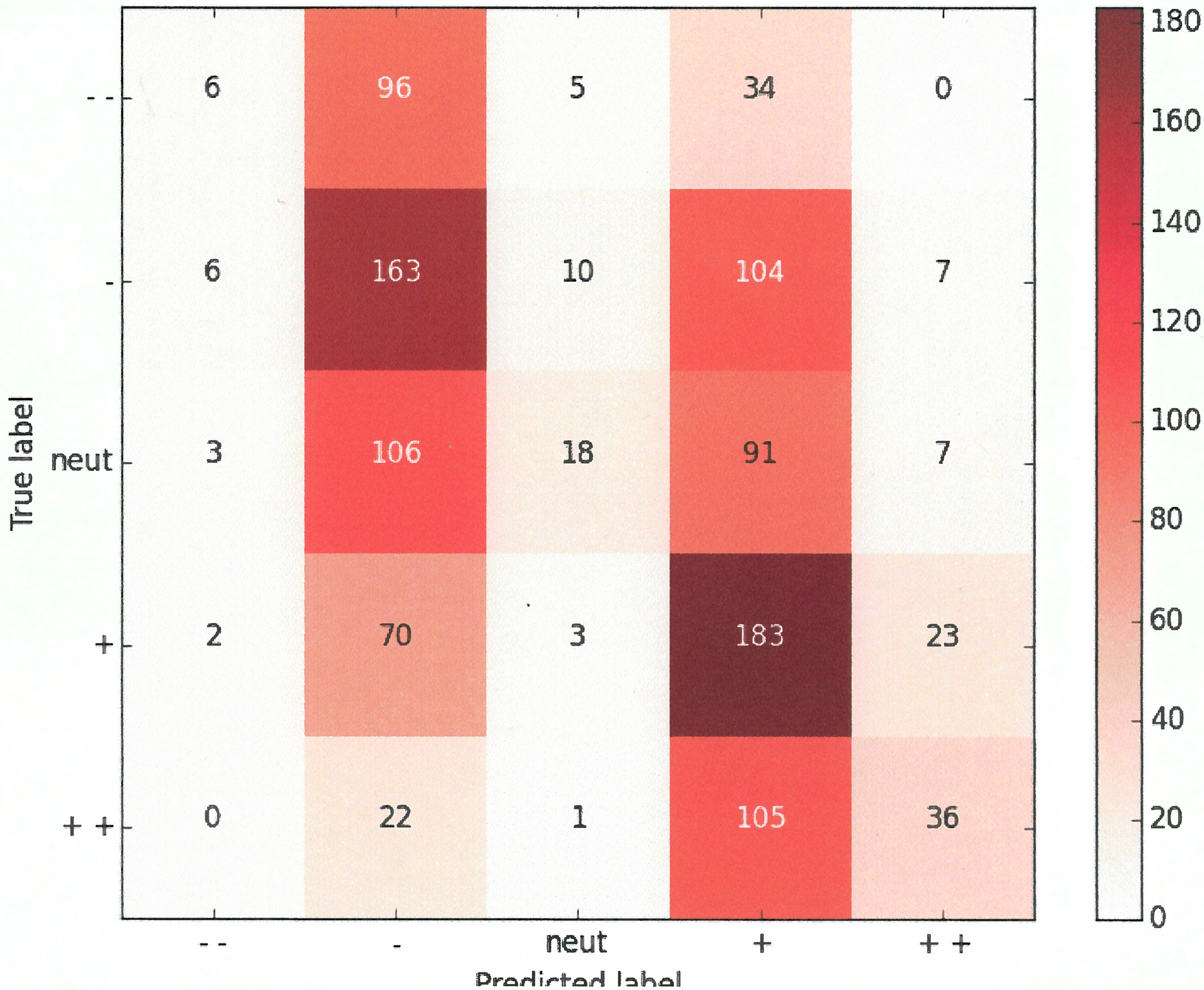
2. The accuracy goes down for train set with inc in regularization.

3. Accuracy goes slightly up & then goes down for dev set with increase in regularization



4(f) 1. There is no extreme disagreement ie (True = -- & Pred = ++) and (True = ++ & Pred = ++) are both 0.

2. For slight negative (-), the classifier is confused betn -ve 163 & +ve 104.



3. For neutral again its confused betn -ve 106 & +ve 91.
& same for +ve. (70, 183)

4(g)

True	Pred	Sentence
0	3	the experience of going to a film festival is a rewarding one ; the experiencing of sampling one through this movie is not .

Perhaps got confused due to "rewarding one" & couldn't detect a ~~se~~ dependence on negation at end.
To fix it a ~~st~~ recurrent or recursive model can help.

True	Pred	Sentence
3	0	hilariously inept and ridiculous .

The classifier got this correctly. This is most likely a judgement error. Can be fixed by better judgement ~~gut~~ verification (eg voting)

True	Pred	Sentence
1	4	while the resident evil games may have set new standards for thrills , suspense , and gore for video games , the movie really only succeeds in the third of these .

Classifier confused by lot of positive words. not understanding the semantic negation at end. Solution is to have a better semantic understanding of sentence & also recursive/recurrent model as before.