1. Softmax

(a).(5 points) Prove that softmax is invariant to constant offsets in the input, that is, for any input vector x and any constant c, softmax(x) = softmax(x + c),where x + c means adding the constant c to every dimension of x.

Answer: Adding a constant offsets in the input doesn't change the value of softmax because

Softmax(x+c)i= $\frac{e^{x_i+c}}{\sum_j e^{x_j+c}}$ = $\frac{e^{x_i}\cdot e^c}{\sum_j e^{x_j}\cdot e^c}$ sicne e^c is a constant so the result will be $\frac{e^{x_i}}{\sum_j e^{x_j}}$ which is equals to Softmax(x)i .

(b). See q1_softmax.py

2. Neural Network Basics

(a)(3 points) Derive the gradients of the sigmoid function and show that it can be rewritten as a function of the function value (i.e., in some expression where only σ(x), but not x, is present). Assume that the input x is a scalar for this question. Recall, the sigmoid function is

Answer:

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

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

(b) (3 points) Derive the gradient with regard to the inputs of a softmax function when cross entropy loss is used for evaluation, i.e., find the gradients with respect to the softmax input vector θ, when the prediction is made by $\hat{y}$ = softmax(θ). Remember the cross entropy function is: $CE(y,\hat{y}) = -y_i \log(\hat{y}_i)$

Answer:

Since only kth element of vector yi is 1 the others are 0 so the gradient with regard to the inputs of softmax is:

$$\frac{dCE(y,\hat{y})}{d\theta} = \frac{dCE(y,\hat{y})}{d\hat{y}} \cdot \frac{d\hat{y}}{d\theta} = - \frac{1}{Softmax(\theta)_k} \cdot Softmax(\theta)_i \cdot (1 - Softmax(\theta)_j)$$

(c).(6 points) Derive the gradients with respect to the inputs x to an one-hidden-layer neural network (that is,find∂J whereJ=CE(y,yˆ) is the cost function for the neural network).The neural network employs

∂x

sigmoid activation function for the hidden layer, and softmax for the output layer. Assume the

one-hot label vector is y, and cross entropy cost is used. (Feel free to use σ′(x) as the shorthand for sigmoid gradient, and feel free to define any variables whenever you see fit.)

Answer: The result of $\frac{\partial J}{\partial x}$ = - $\frac{1}{\hat{y}} \cdot S_i \cdot (1 - S_i) \cdot W_2 \cdot \sigma(xW_1 + b_1)$
$\cdot (1 - \sigma(xW_1 + b_1)) \cdot W_1$

(d). (2 points) How many parameters are there in this neural network, assuming the input is Dx-dimensional, the output is Dy-dimensional, and there are H hidden units?
Answer: The total number of parameters of the network is $D_x \cdot H + H + H \cdot D_y + D_y$

(e) see q2 sigmoid.py.
(f) see q2 gradcheck.py
(g) see q2 neural.py

3. Word2Vec

(a). Answer:
The gradients with respect to v_c is U_o * ($\frac{exp(U_O * V_c)}{\sum_{1}^{w} exp(U_w * V_c)}$ - 1)

(b). Answer:
The gradients with respect to U_k is V_c * ($\frac{exp(U_k * V_c)}{\sum_{1}^{w} exp(U_w * V_c)}$ - δ), δ = 1 if k = o, and δ = 0 if k $\neq$ o.

(c). Answer:
For negative sampling the gradients with respect to v_c is

$$U_0^T(\sigma(U_0^T V_c) - 1) - \sum_{i=1}^{k} U_k^T(\sigma(-U_k^T V_c) - 1)$$

The gradients with respect to U_k is :
If k = o:

$$V_c\,(\sigma(U_0^T V_c) - 1)$$

If k $\neq$ o:

$$-V_c\,(\sigma(-U_k^T V_c) - 1)$$

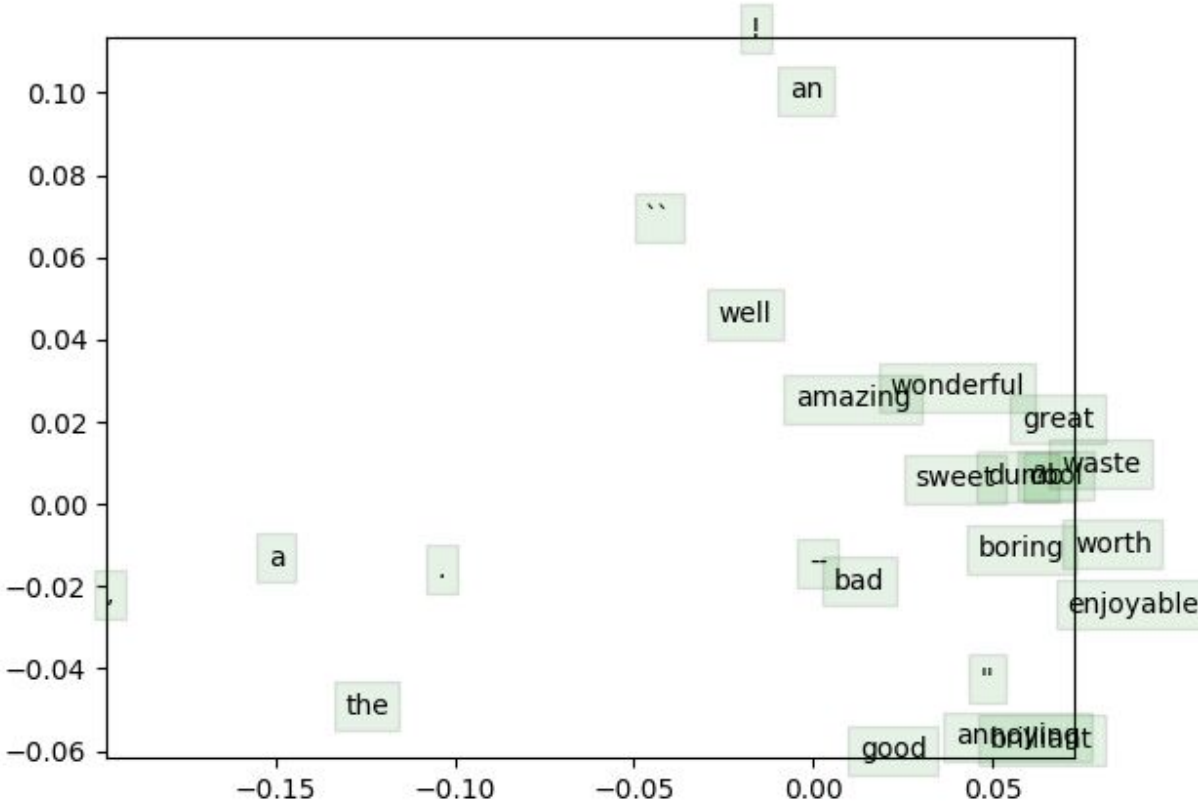
negative sampling is more efficient than softmax-crossentropy, because we don't need to calculate the exponential of all word vectors in our vocabulary , instead we only need to calculate the word vectors we sampled in this question (k+1). So the fraction of runtime ratio is :
R(softmax-cross entropy) / R(negative sampling) = vocabulary size / (k+1)

(b). Answer:

For skip gram model the gradient with regards to w_t is $\sum_{-m \le j \le m} \frac{\sigma F(W_{t+j}, V_c)}{\sigma W_{t+j}}$ the gradient with regards to W_{t+j} is $\frac{\sigma F(W_{t+j}, V_c)}{\sigma W_c}$ (for j $\neq 0$).

For CBOW model the gradient with regards to wt is $\frac{\sigma F(W_t, \widehat{V})}{\sigma \widehat{V}}$, the gradient with regards to W_{t+j} is $\frac{\sigma F(W_t, \widehat{V})}{\sigma W_t}$.

(e). See q3_word2vec.py
(f). See q3_sgd.py
(g). Answer:



(h). See q3 word2vec.py

4. Sentiment Analysis

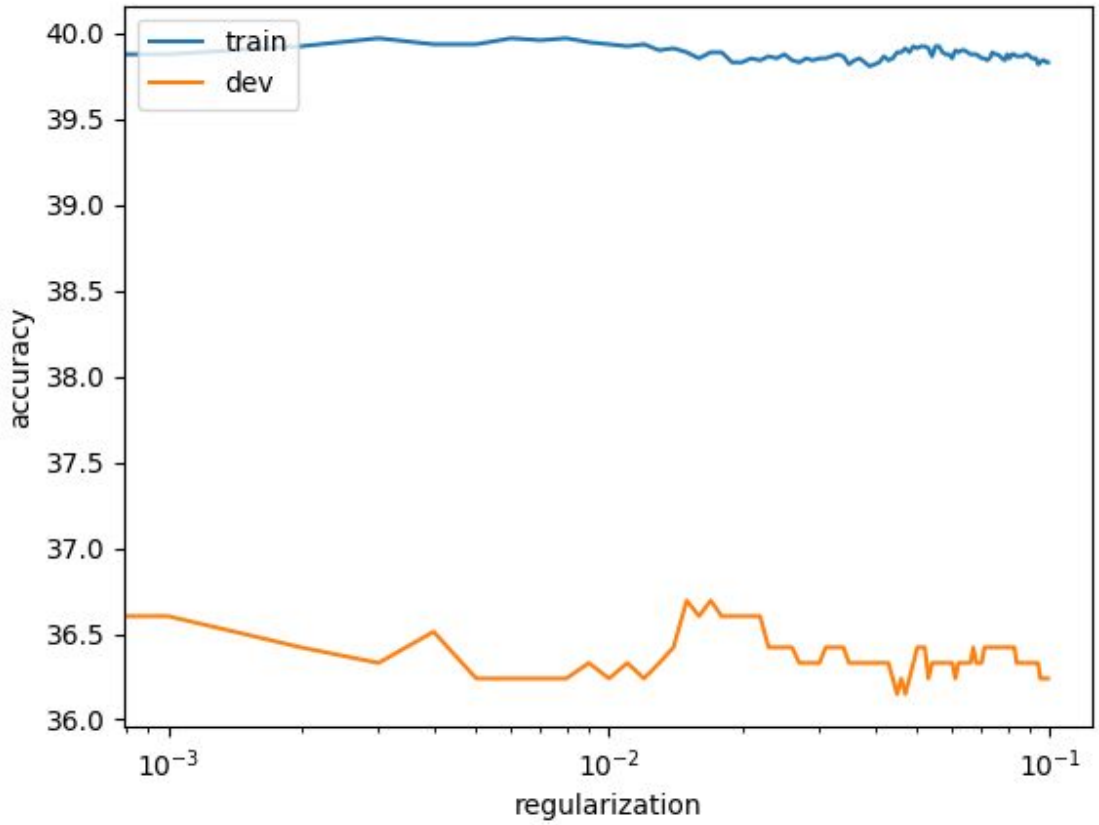
(a). See q3_sgd.py

(b). Regularization can discourage learning a more complex or flexible model by penalizing the parameter in your model in order not to learn noise in your training set, so as to avoid overfitting.

(c) See q4 sentiment.py

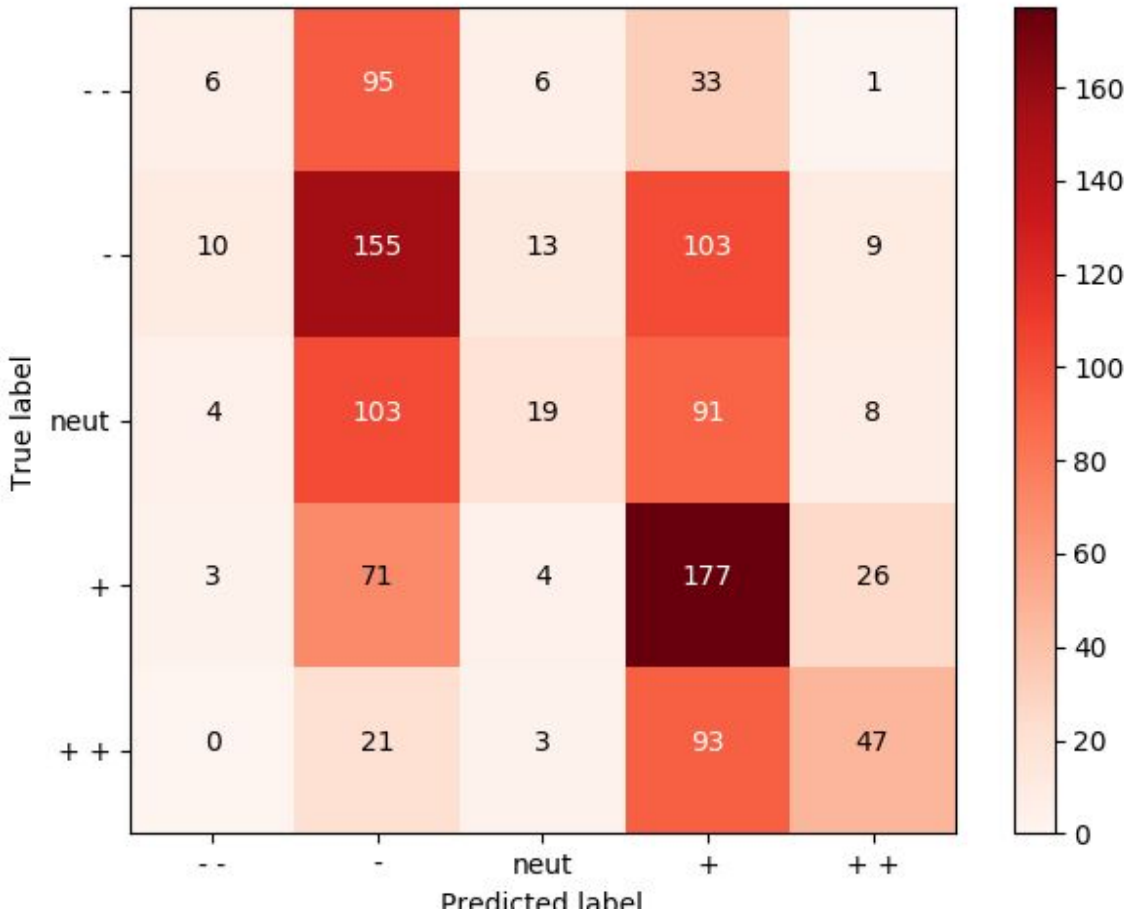
(d). Answer: Word vector generated by Glove has 50 dimension, whereas word vector generated by word2vec only has 10 dimension, so the word vector generated by Glove will keep more information of the context and relationship between each word; Glove considers global word counts and do dimension reduction on their Co-occurrence matrix, which will capture global context information rather than window size context; The pretrained Glove model was trained on powerful machine with state of the art optimization method, my vector only trained for 40000 iteration, which is not accurate enough.

(e).



The accuracy on training set is higher than development set; and with the change of regularization the accuracy of development set is more fluctuate than the accuracy of training set; with the increasing of regularization term, the accuracy of development set first increase and then decrease.

(f).



Answer: Model can distinguish between classes with large differences.

(g). Answer:

Sentence	Predicted	True	Explanation
“and if you 're not nearly moved to tears by a couple of scenes , you 've got ice water in your veins .”	1	3	The glove model destroy the order of word and can not capture the handle negation. So when it sees not moved, it will treat it as negative.
“ultimately feels empty and unsatisfying , like swallowing a communion wafer without the wine .”	0	3	The model can’t get the meaning of metaphor.

Ahhhh…..revenge is sweet.	3	2	It is even hard for human to distinguish the two classes.