Logic - Stanford CS 224N Winter 2018

Rohit Apte

2018-01-16

1 Problem 1: Softmax

a Prove that softmax is invariant to constant osets in the input, that is, for any input vector x and any constant c, $softmax(x) = softmax(x_c)$

We know that
$softmax(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$
Then for $softmax(x + c)$ we have
$softmax(x + c)_i = \frac{e^{x_i+c}}{\sum_j e^{x_j+c}} = \frac{e^{x_i}e^c}{\sum_j e^{x_j}e^c} = \frac{e^{x_i}e^c}{e^c\sum_j e^{x_j}} = \frac{e^{x_i}}{\sum_j e^{x_j}} = softmax(x)_i$
Therefore $softmax(x) = softmax(x + c)$.

2 Problem 2: Neural Network Basics

a 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 $\phi(x)$, but not x, is present). Assume that the input x is a scalar for this question. Recall, the sigmoid function is $\phi(x) = \frac{1}{1+e^{-x}}$.

$\phi(x) = \frac{1}{1+e^{-x}} = (1 + e^{-x})^{-1}$
$\therefore \frac{d\phi(x)}{dx} = -1(1 + e^{-x})^{-2} \times -e^{-x} = \frac{e^{-x}}{(1+e^{-x})^{-2}} = \frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}}$
$= \frac{1}{1+e^{-x}}\frac{1+e^{-x}-1}{1+e^{-x}} = \frac{1}{1+e^{-x}}(1 - \frac{1}{1+e^{-x}}) = \phi(x)(1 - \phi(x))$

$\therefore \frac{d\phi(x)}{dx} = \phi(x)(1 - \phi(x))$

b Derive the gradient with regard to the inputs of a softmax function when cross entropy loss is used for evaluation, i.e., nd the gradients with respect to the softmax input vector θ, when the prediction is made by $\hat{y} = softmax(\theta)$. Remember the cross entropy function is
$CE(y, \hat{y}) = -\sum_i y_i log(\hat{y_i})$
where y is the one-hot label vector, and $\hat{y}$ is the predicted probability vector for all classes. (Hint: you might want to consider the fact many elements of y are zeros, and assume that only the $k-th$ dimension of y is one.)

We know that
$CE(y, \hat{y}) = -\sum_i y_i log(\hat{y_i})$
$\therefore \frac{\partial CE}{\partial \theta_i} = -\sum_k y_k \frac{\partial log(\hat{y}_k)}{\partial \theta_i} = -\sum_k y_k \frac{\partial log(\hat{y}_k)}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial \theta_i} = -\sum_k y_k \frac{1}{\hat{y}_k} \frac{\partial \hat{y}_k}{\partial \theta_i}$
$\frac{\partial \hat{y}_k}{\partial \theta_i}$ is the derivative of the softmax function. Lets expand that term next
We know that $\hat{y}_i = softmax(\theta)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$
$\therefore \frac{\partial \hat{y}}{\theta_j} = \frac{\partial \frac{e^{\theta_i}}{\sum_k e^{\theta_k}}}{\partial \theta_j}$
We know that $(\frac{f}{g})' = \frac{f'g - fg'}{g^2}$
In our case $f(\theta) = e^{\theta_i}$ and $g(\theta) = \sum_k e^{\theta_k}$
Note that $\frac{\partial g(\theta)}{\partial \theta_j} = e^{\theta_j}$ but $\frac{\partial f(\theta)}{\partial \theta_j} = e^{\theta_i}$ if $i = j$ and 0 if $i \neq j$.
Therefore we need to consider 2 cases

$i = j$
$\frac{\partial \hat{y}}{\theta_j} = \frac{e^{\theta_i}\sum_k e^{\theta_k} - e^{\theta_i}e^{\theta_j}}{(\sum_k e^{\theta_k})^2} = \frac{e^{\theta_i}}{\sum_k e^{\theta_k}} \frac{\sum_k e^{\theta_k} - e^{\theta_j}}{\sum_k e^{\theta_k}} = \frac{e^{\theta_i}}{\sum_k e^{\theta_k}}(1 - \frac{e^{\theta_j}}{\sum_k e^{\theta_k}}) = \hat{y}_i(1 - \hat{y}_j) =$
$\hat{y}_i(1 - \hat{y}_i)$

$i \neq j$
$\frac{\partial \hat{y}}{\theta_j} = \frac{0 - e^{\theta_i}e^{\theta_j}}{(\sum_k e^{\theta_k})^2} = -\frac{e^{\theta_i}}{\sum_k e^{\theta_k}} \frac{e^{\theta_j}}{\sum_k e^{\theta_k}} = -\hat{y}_i\hat{y}_j$
Then
$\frac{\partial CE}{\partial \theta_i} = -\sum_k y_k \frac{1}{\hat{y}_k} \frac{\partial \hat{y}_k}{\partial \theta_i} = (-y_i \frac{1}{\hat{y}_i} \frac{\partial \hat{y}_i}{\partial \theta_i}) - \sum_{k \neq i} y_k \frac{1}{\hat{y}_k} \frac{\partial \hat{y}_k}{\partial \theta_k}$
$= (-y_i \frac{1}{\hat{y}_i} \hat{y}_i(1 - \hat{y}_i)) - (\sum_{k \neq i} y_k \frac{1}{\hat{y}_k} - \hat{y}_i\hat{y}_k)$
$= (-y_i + y_i\hat{y}_i) + \sum_{k \neq i} y_k\hat{y}_i$
$= -y_i + \sum_k y_k\hat{y}_i = -y_i + \hat{y}_i \sum_k y_k$
Since y_k is a one hot vector the sum is 1.
Therefore
$\frac{\partial CE}{\partial \theta_i} = \hat{y}_i - y_i$

c Derive the gradients with respect to the inputs x to an one-hidden-layer neural network (that is, nd $\frac{\partial J}{\partial x}$ where $J = CE(y, \hat{y})$ is the cost function for the neural network). The neural network employs 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 $\sigma(x)$ as the shorthand for sigmoid gradient, and feel free to dene any variables whenever you see t.)
Recall that the forward propagation is as follows
$h = sigmoid(xW_1 + b_1)$
$\hat{y} = softmax(hW_2 + b_2)$
Note that here were assuming that the input vector (thus the hidden variables and output probabilities) is a row vector to be consistent with the programming assignment. When we apply the sigmoid function to a vector, we are applying it to each of the elements of that vector. W_i and b_i $(i = 1, 2)$ are the weights and biases, respectively, of the two layers.

Let $A_1 = xW_1 + b_1$, $h = \sigma(A_1)$, $A_2 = hW_2 + b_2$, and $\hat{y} = softmax(A_2)$.
Then $\frac{\partial CE}{\partial A_2} = \hat{y} - y$.
$\frac{\partial A_2}{\partial h} = W_2^T$
$\frac{\partial A_1}{\partial x} = W_1^T$
$\therefore \frac{\partial CE}{\partial x} = \frac{\partial CE}{\partial A_2}\frac{\partial A_2}{\partial h}\frac{\partial h}{\partial A_1}\frac{\partial A_1}{\partial x} = (\hat{y} - y)W_2^T \odot \sigma'(h)W_1^T$

d How many parameters are there in this neural network, assuming the input is D_x-dimensional, the output is D_y-dimensional, and there are H hidden units?

W_1 is $D_x \times H$. b_1 is H. W_2 is $H \times D_y$ and b_2 is D_y. Therefore there are $D_x \times H + H + H \times D_y + D_y$ parameters, i.e.
$(D_x + 1)H + (H + 1)D_y$

3 Assume you are given a "predicted" word vector v_c corresponding to the center word c for Skip-Gram, and word prediction is made with the softmax function found in word2vec models

$\hat{y}_o = p(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w=1}^{V} exp(u_w^T v_c)}$
where $u_w (w = 1, \ldots, V)$ are the "output" word vectors for all words in the vocabulary. Assuming cross entropy cost is applied to this prediction and word o is the expected word (the o-th element of the one-hot label vector is one), derive the gradients with respect to v_c.
Hint: It will be helpful to use notation from question 2. For instance, letting $\hat{y}$ be the vector of softmax predictions for every word, y as the expected word vector, and the loss function
$J_{softmaxCE}(o, v_c, U) = CE(y, \hat{y})$
where $U = [u_1, u_2, \ldots, u_V]$ is the matrix of all the output vectors. Make sure you state the orientation of your vectors and matrices.

We know that
$\hat{y}_o = p(o|c) = \frac{e^{u_o^T v_c}}{\sum_{w=1}^{V} e^{u_w^T v_c}}$ y and $\hat{y}$ are column vectors (dimension $\mid V \mid \times 1$).
We will assume u_i is a column vector (dimension $D \times 1$ where D=number of embeddings. Then $U = D \times \mid V \mid$.
Then $CE(y, \hat{y}) = -\sum_i y_i log(\hat{y}_i)$
But since y is a one hot vector, only 1 element is 1. Lets say the j^{th} element is 1. Then
$CE = -y_j log(\hat{y}_j) = -y_j log(\frac{e^{u_j^T v_c}}{\sum_{w=1}^{V} e^{u_w^T v_c}}) = -y_j(e^{u_j^T} - log \sum_{w=1}^{V} e^{u_w^T v_c}) = -y_j(e^{u_j^T} - log \sum_{w=1}^{V} e^{u_w^T v_c})$
$\therefore \frac{\partial CE}{\partial v_c} = -y_k(u_k - \sum_{w=1}^{V} \hat{y}_w)$

Again, since y is a one hot vector, $u_k = U.y$. Also $\sum_{w=1}^{V} u_w \hat{y}_w = U.\hat{y}$
$\therefore \frac{\partial CE}{\partial v_c} = U(\hat{y} - y)$

a As in the previous part, derive gradients for the output word vectors u_ks (including u_o).

v_c is a $D \times 1$ vector so the gradients for output word vector u_k is $v_c(\hat{y} - y)^T$

b Repeat part (a) and (b) assuming we are using the negative sampling loss for the predicted vector v_c, and the expected output word index is o. Assume that K negative samples (words) are drawn, and they are $1, 2, \times, K$ respectively for simplicity of notation $(o \notin 1, \ldots, K)$. Again, for a given word, o, denote its output vector as u_o. The negative sampling loss function in this case is $J_{neg-sample}(o, v_c, U) = -log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K} log(\sigma(-u_k^T v_c))$ where $\sigma(.)$ is the sigmoid function. After youve done this, describe with one sentence why this cost function is much more effcient to compute than the softmax-CE loss (you could provide a speed-up ratio, i.e., the runtime of the softmax-CE loss divided by the runtime of the negative sampling loss).

$J_{neg-sample}(o, v_c, U) = -log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K} log(\sigma(-u_k^T v_c))$

$\frac{\partial J_{neg-sample}(o,v_c,U)}{\partial v_c} = \frac{\partial[-log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K} log(\sigma(-u_k^T v_c))]}{\partial v_c} = \frac{\partial -log(\sigma(u_o^T v_c))}{\partial v_c} - \frac{\partial \sum_{k=1}^{K} log(\sigma(-u_k^T v_c))}{\partial v_c}$
$\frac{\partial -log(\sigma(u_o^T v_c))}{\partial v_c} = -\frac{1}{\sigma(u_o^T v_c)} \sigma(u_o^T v_c)(1-\sigma(u_o^T v_c))u_o = -(1-\sigma(u_o^T v_c))u_o = (\sigma(u_o^T v_c) - 1)u_o$
$\frac{\partial \sum_{k=1}^{K} log(\sigma(-u_k^T v_c))}{\partial v_c} = \sum_{k=1}^{K} \frac{1}{\sigma(-u_k^T v_c)} \sigma(-u_k^T v_c)(1-\sigma(-u_k^T v_c))(-u_k) = \sum_{k=1}^{K} (\sigma(-u_k^T v_c) - 1)u_k$

$\therefore \frac{\partial J_{neg-sample}(o,v_c,U)}{\partial v_c} = (\sigma(u_o^T v_c) - 1)u_o - \sum_{k=1}^{K} (\sigma(-u_k^T v_c) - 1)u_k$

$\frac{\partial J_{neg-sample}(o,v_c,U)}{\partial u_o} = -\frac{1}{\sigma(u_o^T v_c)} \sigma(u_o^T v_c)(1 - \sigma(u_o^T v_c))v_c - 0 = (\sigma(u_o^T v_c) - 1)v_c$
$\frac{\partial J_{neg-sample}(o,v_c,U)}{\partial u_k} = 0 - \frac{1}{\sigma(-u_k^T v_c)} \sigma(-u_k^T v_c)(1-\sigma(-u_k^T v_c))(-v_c) = -(\sigma(-u_k^T v_c) - 1)v_c$

The cost function is more efficient since its less computationally intensive to calculate the sigmoid vs softmax and because we are calculating for a much smaller set (K) instead of the entire vocabulary.

c Suppose the center word is $c = w_t$ and the context words are $[w_{tm}, \dots, w_{t1}, w_t, w_{t+1}, \dots, w_{t+m}]$, where m is the context size. Derive gradients for all of the word vectors for Skip-Gram and CBOW given the previous parts. Hint: feel free to use $F(o, v_c)$ (where o is the expected word) as a placeholder for the $JsoftmaxCE(o, v_c, \dots)$ or $Jnegsample(o, v_c, \dots)$ cost functions in this part you'll see that this is a useful abstraction for the coding part. That is, your solution may contain terms of the form $\frac{\partial F(o,v_c)}{\partial \dots}$. Recall that for skip-gram, the cost for a context centered around c is
$Jskip - gram(w_{tm \dots t+m}) = \sum_{m \leq j \leq m, j \neq 0} F(w_{t+j}, v_c)$
where w_{t+j} refers to the word at the j^{th} index from the center.
CBOW is slightly different. Instead of using v_c as the predicted vector, we use $\hat{v}$ dened below. For (a simpler variant of) CBOW, we sum up the input word vectors in the context
$J_{CBOW}(w_{c-m \dots c+m}) = F(w_t, \hat{v})$

For the skip gram model
$\frac{\partial J}{\partial U} = \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial F(w_{c+j}, v_c)}{\partial U}$
$\frac{\partial J}{\partial v_c} = \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial F(w_{c+j}, v_c)}{\partial v_c}$
$\frac{\partial J}{\partial v_j} = 0$ if $j \neq c$
For the skip gram model $J_{CBOW}(w_{c-m \dots c+m}) = F(w_t, \hat{v})$
$\therefore \frac{\partial J_{CBOW}(w_{c-m \dots c+m})}{\partial U} = \frac{\partial F(w_t, \hat{v})}{\partial U}$
$\frac{\partial J_{CBOW}(w_{c-m \dots c+m})}{\partial v_j} = \frac{\partial F(w_t, \hat{v})}{\partial v_j}$ if $c - m \leq j \leq c + m$
$\frac{\partial J_{CBOW}(w_{c-m \dots c+m})}{\partial v_j} = 0$ otherwise

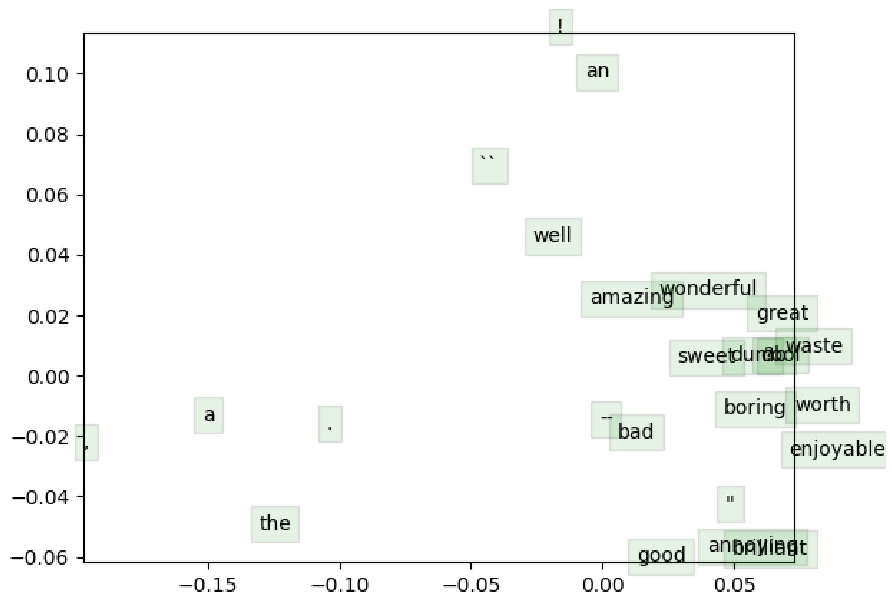

Figure 1: q3WordVectors

d Show time! Now we are going to load some real data and train word vectors with everything you just implemented! We are going to use the Stanford Sentiment Treebank (SST) dataset to train word vectors, and later apply them to a simple sentiment analysis task. You will need to fetch the datasets first. To do this, run sh get datasets.sh. There is no additional code to write for this part; just run python q3 run.py.
Note: The training process may take a long time depending on the efficiency of your implementation (an efficient implementation takes approximately an hour). Plan accordingly!
When the script finishes, a visualization for your word vectors will appear. It will also be saved as q3wordvectors.png in your project directory. Include the plot in your homework write up. Briefly explain in at most three sentences what you see in the plot.

I notice 3 things
Words like amazing, wonderful, great, sweet are together which makes sense

since they have similar meanings.
Words like boring and enjoyable are together since they would be found near similar center words .
Although a and the are grouped close together, an is much further away even though its an article word as well. So the algorithm does a good job of setting word vectors but doesn't necessarily understand nuances of the English language.

4 Sentiment Analysis

a

b Explain in at most two sentences why we want to introduce regularization when doing classication (in fact, most machine learning tasks).

Regularization penalizes the loss function and helps avoid over-fitting. It is an important tool in preventing overfitting in ML models.

c Fill in the hyperparameter selection code in q4 sentiment.py to search for the optimal regularization parameter. You need to implement both getRegularizationValues and chooseBestModel. Attach your code for chooseBestModel to your written write-up. You should be able to attain at least 36.5 percent accuracy on the dev and test sets using the pretrained vectors in part d

```
#best model parameters are off the cross validation set
bestResult=results[0]
bestScore=results[0]['dev']
     for item in results:
          if item['dev']>bestScore:
               bestResult=item
               bestScore=item['dev']
```

d Run python q4 sentiment.py –yourvectors to train a model using your word vectors from q3. Now, run python q4 sentiment.py –pretrained to train a model using pretrained GloVe vectors (on Wikipedia data). Compare and report the best train, dev, and test accuracies. Why do you think the pretrained vectors did better? Be specic and justify with 3 distinct reasons.

The pretrained vectors did better because
1. They were trained on a larger corpus so they captured the nuances more accurately.
2. The pretrained vectors have more dimensions which can give better results.
3. GloVe is a combination of word2vec and other methods that capture global statistical information. It should therefore be expected to outperform word2vec alone.

e Plot the classication accuracy on the train and dev set with respect to the regularization value for the pretrained GloVe vectors, using a logarithmic scale on the x-axis. This should have been done automatically. Include q4 reg acc.png in your homework write up. Briey explain in at most three sentences what you see in the plot.

There is a tradeoff between training accuracy and cross validation accuracy. This actually demonstrates the use of regularization to prevent overfitting. We get a lower training accuracy but higher Cross Validation accuracy.

f We will now analyze errors that the model makes (with pretrained GloVe vectors). When you ran python q4 sentiment.py –pretrained, two les should have been generated. Take a look at q4devconf.png and include it in your homework writeup. Interpret the confusion matrix in at most three sentences

Overall the results are not great. Positive sentiments being classified as negative and vice versa. Also its doing a poor job of very negative and positive sentiments.

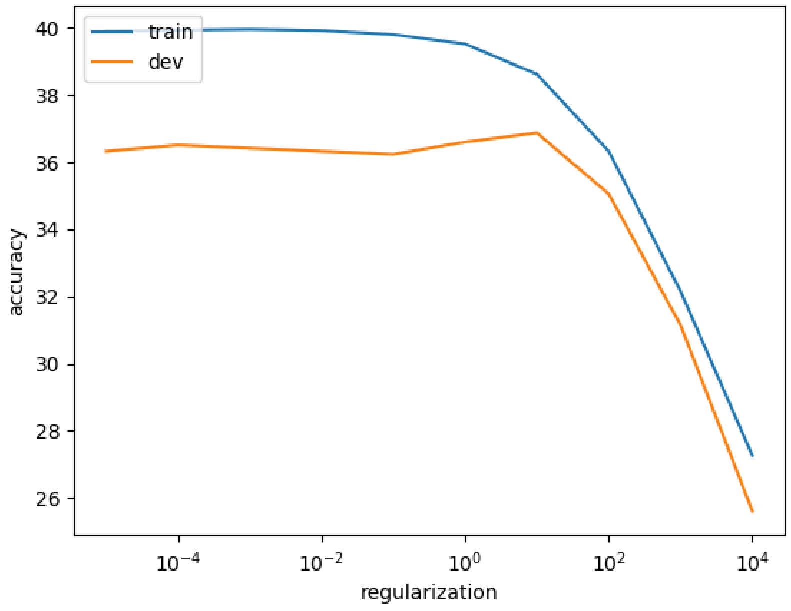

Figure 2: q4regvacc

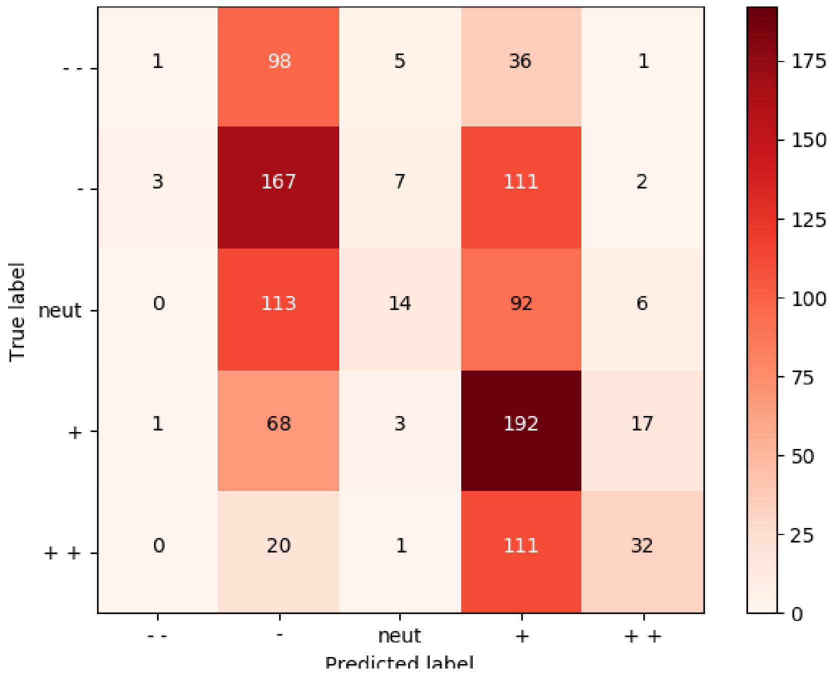

Figure 3: q4devconf

g Next, take a look at q4 dev pred.txt. Choose 3 examples where your classifier made errors and briefly explain the error and what features the classifier would need to classify the example correctly (1 sentence per example). Try to pick examples with different reasons.

`3 1 whether you like rap music or loathe it , you can't deny either the tragic loss of two young men in the prime of their talent or the power of this movie .`
Words like loathe and tragic probably causing a negative prediction.

`1 3 a subject like this should inspire reaction in its audience ; the pianist does not .`
Non standard sentence structure is throwing off the classification. Should be a negative comment but being classified as positive.

`3 1 the jabs it employs are short , carefully placed and dead-center .` The words used dont convey very strong sentiment which is probably why its misclassifying this.