Natural Language Processing - 097215 - HW1 - Dry

Dor Shaim - 303051502 , Jonathan Brokman - 200678415

December 26, 2019

Question 1

1)

The type of features that would solve the OOV problem are ones for which the feature gives syntactic information for that word/tag pair based on its structure. This way, if a certain class of words have the same structure (e.g. grammatical structure) and are paired with the same tag, then we don't have to learn a specific instance of that word/tag pair. When we need to tag a word that we haven't yet seen in our training set but belong to that same structure, then we can make a useful prediction of its tag based in its structure. One set of features for the representation of pairs can be all the set of possible suffixes/prefixes paired with a tag (as we saw in class). For example, words ending in the suffix 'ing' will probably be tagged by the POS 'VBG' so it gives us a lot of information about the word without caring so much about the specific instance of that word. Another set of features can be related to numbers being part of the word, for example, if the tagging task is the named-entity recognition then numbers in a word could give us information of their role in the sentence (for example a feature that checks if the word is a 4-digit number which could point to some year). Another final set of features can be related to capitalization, for example whether the first letter is capitalized (which could hint that this word can be a name) or whether all the word is capitalized (which can be an abbreviation or a company name).

2)

a)

Let's denote our vocabulary by $\mathcal{V}$ and the label set by $\mathcal{K}$. then $|\mathcal{V}| = 100,000$ and $|\mathcal{K}| = 25$. In addition, for each word/tag pair (x_i, y_i), we map it to a feature vector, given by:

$$\vec{\varphi}\,(x_i, y_i) = \begin{bmatrix} (x_{i-2}, y_{i-2}) \\ (x_{i-1}, y_{i-1}) \\ (x_i, y_i) \end{bmatrix}$$

Assuming that both $\mathcal{V}$ and $\mathcal{K}$ have the special symbol '$*$' to indicate words/tags that come before the first word in the sentences to augment features of words located on the first or second place of a sentence, then, because each feature vector have 3 dof of choosing words from $\mathcal{V}$ and 3 dof of choosing tags from $\mathcal{K}$, in total, that amounts to having $|\mathcal{V}|^3 \cdot |\mathcal{K}|^3 = 100,000^3 \cdot 25^3 = 15.625 \cdot 10^{15}$ different combinations of feature vector values for the representation of word/tag pairs under this model.

b)

Since a reasonable sized corpus will probably have around 10^6 tokens, then if we will use MLE estimates for the parameters of this model (which accroding to the above subsection has around 10^{15} parameters paired with each feature combination to estimate), we will face a critical problem of sparsity in our features and not enough statistical evidence for these estimations. Actually, the majority of parameters will have to be 0 under this model and the method of MLE estimation, because we will simply won't see most of the feature combinations in our corpus. This is not necessarily a problem for most of the features since it is true that most of the combinations are not probable. But, there will be a large set of combinations which would not get enough statistical representation in the corpus and thus we would not get a reliable estimates of the parameters. In general, it is always preferable in a supervised learning regime to have much more data than parameters to be learned for a successful generalization. Here we have the exact opposite case. We can overcome this issue by only considering features that appeared in our training corpus which makes much smaller set of combinations than the possible set of all combinations thus reducing substantially the amount of parameters to be estimated. However, we are still left with a problem of unseen words in test set and misrepresentation of the features under this model for these words. For example, it is very likely that some proper nouns like 'Mullaly' had never appeared in our training corpus and our model will have nothing to say about such words. An alternative is to smooth the estimates using a convex combination of this model with other, more simpler models such that the coefficients of this combination reflects our confidence of each estimate of the model, thus, resorting to estimation using simpler methods when we simply don't have enough data for a specific feature combination.

3)

the exponential function is used in this formulation to insure that the probability defined by that formula will be a non-negative number and also preserve the order of the original inputs (this is true since $\exp(\cdot)$ is monotinically increasing). The sum in the denominator make sure it is properly normalized so the probabilty will get values in $[0, 1]$. The exponential also amplifies big inputs so that the maximal value becomes very dominant after this operation and small values almost get nulled. Additionaly, The $\exp(\cdot)$ function is infinitely differentiable and it's derivative is itself so it is very convinient to work with this function when we need to use gradients in order to find an extremum.

4)

We would use binary features and not integer features. The problem with integer features is that they come with dimensionality reduction of our representation (which is good) but at the expense of not differentiating between characteristics that we originally wanted to represent. The dot product $w \cdot f(x, y)$ sums products of correponding weights and features and the integer value of f can be merged inside w. Moreover, if we use integer features, then only one scalar in w help us in giving scores to all the combinations of that feature. But if we use binary features than we actually give score to each combination of that feature separately. So if we use binary features it is possible to tune w such that inside that template we can give low scores to combinations of that feature template which don't occur much and high scores for combinations of that feature template which occur a lot. But if we use integer features, then by making w higher we raise the score for all the combinations of that feature template, so effectively, we have less control in our tuning of scores inside each feature template. Another possible reason for not choosing integer features is the fact that we don't have a natural ordering of our features since they are of a combinatorial nature without an inherent partial order defined on them.

5)

The problem with this formulation is that each time we have to compute $p(x_i = x \mid y_i = y)$ we have to compute the denominator term:

$$\sum_{\hat{x} \in \hat{X}} \exp(w \cdot f(\hat{x}, y))$$

but the set $\hat{X}$ is hugh so this sum can be too computationally prohibitive. Effectively, this problem makes our model computationally intractable.

6)

By changing the above formulation we now have a model with probabilities $p(y_{i+1} = y \mid x_i = x\,,\, y_i = y')$ which are computationally tractable due to the sum in the denominator now changed to be over the set Y which is much smaller (for example in POS tagging, Y would be the set of tags which is approximately of size 45), so the above problem has been solved.

7)

There can be 2 advantages of HMM over MEMM. The first is that HMM is a much more simple model to understand and also to train. The optimization of HMM just uses counts of words and tags over the training corpus whereas the optimization of MEMM applies more sophisticated gradient descent techniques which is more computationally expensive so applying HMM model can be faster then applying MEMM. The second advantage is that the MEMM is a discriminative model whereas the HMM is a generative model. This means that given an HMM model we can generate examples from the learned probability distributions. In some domains this have a hugh impact. A good generative model can produce synthetic examples and effectively create new datasets (or augment existing datasets). Generative models can also discriminate between tags (using Bayes' rule like we saw in class) so in some sense they are generalizations of discriminative models.

Question 2

1)

We want to first note that the probability $p\left(w|c\right)$ should be changed to:

$$p\left(w|c\right)=\frac{p\left(w,c\right)}{p\left(c\right)}=\frac{p\left(w,c\right)}{\sum\limits_{w'\in V}p\left(w',c\right)}=\frac{e^{v_w\cdot v_c}}{\sum\limits_{w'\in V}e^{v_{w'}\cdot v_c}}$$

instead of :

$$p\left(w|c\right)=\frac{e^{v_w\cdot v_c}}{\sum\limits_{c'\in V}e^{v_w\cdot v_{c'}}}$$

so the summation in the denominator should be on all the $w'\in V$ instead of all the $c'\in V$ because we're conditioning on c. However, we will continue with the probability formula given in the HW as is. Let's start by writing the log-likelihood:

$$\sum_{(w,c)\in D}\log\left(p\left(w\mid c\right)\right)=\sum_{(w,c)\in D}\log\left(\frac{e^{v_w\cdot v_c}}{\sum\limits_{c'\in V}e^{v_w\cdot v_{c'}}}\right)=\sum_{(w,c)\in D}\left[\log\left(e^{v_w\cdot v_c}\right)-\log\left(\sum_{c'\in V}e^{v_w\cdot v_{c'}}\right)\right]=$$

$$=\sum_{(w,c)\in D}v_w\cdot v_c-\sum_{(w,c)\in D}\log\left(\sum_{c'\in V}e^{v_w\cdot v_{c'}}\right)$$

θ is the vectors of parameters. In our case this just the vector of all vector representations $v_w\in\mathbb{R}^d$. So θ has $d\cdot|V|$ parameters. So finally we have:

$$\theta^*=\underset{\theta\in\mathbb{R}^{d\cdot|V|}}{\operatorname{argmax}}\sum_{(w,c)\in D}\log\left(p\left(w\mid c\right)\right)=\underset{\theta\in\mathbb{R}^{d\cdot|V|}}{\operatorname{argmax}}\left\{\sum_{(w,c)\in D}v_w\cdot v_c-\sum_{(w,c)\in D}\log\left(\sum_{c'\in V}e^{v_w\cdot v_{c'}}\right)\right\}$$

2)

We assume that in the following partial derivative $c\neq\hat{w}$ for all $(\hat{w},c)\in D$ (we need this assumption in order to avoid mixed dot products and ugly formulas for the gradient).

$$\nabla_{v_w}\left[\sum_{(\hat{w},c)\in D}\log\left(p\left(\hat{w}\mid c\right)\right)\right]=\nabla_{v_w}\left[\sum_{(\hat{w},c)\in D}v_{\hat{w}}\cdot v_c-\sum_{(\hat{w},c)\in D}\log\left(\sum_{c'\in V}e^{v_{\hat{w}}\cdot v_{c'}}\right)\right]=$$

$$=\sum_{(\hat{w},c)\in D}\nabla_{v_w}\left(v_{\hat{w}}\cdot v_c\right)-\sum_{(\hat{w},c)\in D}\nabla_{v_w}\log\left(\sum_{c'\in V}e^{v_{\hat{w}}\cdot v_{c'}}\right)=$$

$$=\sum_{(\hat{w},c)\in D}v_c\delta_{\hat{w},w}-\sum_{(\hat{w},c)\in D}\frac{1}{\sum\limits_{\tilde{c}\in V}e^{v_{\hat{w}}\cdot v_{\tilde{c}}}}\cdot\nabla_{v_w}\left(\sum_{c'\in V}e^{v_{\hat{w}}\cdot v_{c'}}\right)=$$

$$=\sum_{c\in V\,:\,(w,c)\in D}v_c-\sum_{(\hat{w},c)\in D}\frac{\sum\limits_{c'\in V}\nabla_{v_w}\left(e^{v_{\hat{w}}\cdot v_{c'}}\right)}{\sum\limits_{\tilde{c}\in V}e^{v_{\hat{w}}\cdot v_{\tilde{c}}}}=$$

$$=\sum_{c\in V\,:\,(w,c)\in D}v_c-\sum_{(\hat{w},c)\in D}\frac{\sum\limits_{c'\in V}e^{v_{\hat{w}}\cdot v_{c'}}\cdot\nabla_{v_w}\left(v_{\hat{w}}\cdot v_{c'}\right)}{\sum\limits_{\tilde{c}\in V}e^{v_{\hat{w}}\cdot v_{\tilde{c}}}}=$$

$$=\sum_{c\in V\,:\,(w,c)\in D}v_c-\sum_{(\hat{w},c)\in D}\frac{\sum\limits_{c'\in V}e^{v_{\hat{w}}\cdot v_{c'}}\cdot v_{c'}\cdot\delta_{\hat{w},w}}{\sum\limits_{\tilde{c}\in V}e^{v_{\hat{w}}\cdot v_{\tilde{c}}}}=$$

$$= \sum_{c\in V\,:\,(w,c)\in D} v_c - \sum_{c\in V\,:\,(w,c)\in D} \sum_{c'\in V} \frac{e^{v_w\cdot v_{c'}}}{\sum\limits_{\tilde{c}\in V} e^{v_w\cdot v_{\tilde{c}}}} \cdot v_{c'} =$$

$$= \sum_{c\in V\,:\,(w,c)\in D} \left[v_c - \sum_{c'\in V} p\left(w \,|\, c'\right) \cdot v_{c'} \right]$$

3)

a)

As we already mentioned, the parameter vector θ contains $d \cdot |V| = 500 \cdot 500{,}000 = 2.5 \cdot 10^8$ entries.

b)

As already mentioned in the first question, in a supervised learning regime we need to have much more data then learned parameters in order for the model to generalize well. So we should have a corpus containing at least 10^9 tokens which is hugh. Usually, a rule of thumb for logistic regression is that the ratio of available train data to learned parameters should be greater or equal to 10 in order to get a successful model.

c)

The optimization over this model will be too slow and efficitively will take infinite time to complete due to the hugh number of parameters to optimize and the terms we are calculating (e.g calculating the denominator term require us to sum exponents of dot products where we sum over $500{,}000$ terms!) thus making it useless to work with. So this model is not computationally feasible.

4)

The sum is on all of V^2 so some of the (w, c) pairs will be in D (and hence we will use the probability $p\left(D = 1|w, c; \theta\right)$ for them) and the other (w, c) pairs will not be in D (and hence we will use the probability $p\left(D = 0|w, c; \theta\right)$ for them). So the sum over V^2 should be splitted into 2 sums. one over D and the other on $D^c = V^2 \backslash D$ which is the complement set of D in V^2:

$$\sum_{(w,c)\in V^2} \log p\left(D|w,c;\theta\right) = \sum_{(w,c)\in D} \log p\left(D=1|w,c;\theta\right) + \sum_{(w,c)\in D^c} \log p\left(D=0|w,c;\theta\right) =$$

$$= \sum_{(w,c)\in D} \log \sigma\left(v_w \cdot v_c\right) + \sum_{(w,c)\in D^c} \log \sigma\left(-v_w \cdot v_c\right) =$$

$$= \sum_{(w,c)\in D} \log\left(\frac{1}{1+e^{-v_w\cdot v_c}}\right) + \sum_{(w,c)\in D^c} \log\left(\frac{1}{1+e^{v_w\cdot v_c}}\right) =$$

$$= - \sum_{(w,c)\in D} \log\left(1+e^{-v_w\cdot v_c}\right) - \sum_{(w,c)\in D^c} \log\left(1+e^{v_w\cdot v_c}\right)$$

where we have used the fact that:

$$1 - p\left(D=1|w,c;\theta\right) = 1 - \sigma\left(v_w \cdot v_c\right) = 1 - \frac{1}{1+e^{-v_w\cdot v_c}} = \frac{\cancel{1} + e^{-v_w\cdot v_c} - \cancel{1}}{1+e^{-v_w\cdot v_c}} =$$

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

Thus:

$$\theta^* = \underset{\theta\in\mathbb{R}^{d\cdot|V|}}{\operatorname{argmax}} \left\{ - \sum_{(w,c)\in D} \log\left(1+e^{-v_w\cdot v_c}\right) - \sum_{(w,c)\in D^c} \log\left(1+e^{v_w\cdot v_c}\right) \right\}$$

5)

As in section 2 of this question, we assume here that $c \neq \hat{w}$ for all $(\hat{w}, c) \in V^2$ to make the expressions for the gradients easier to manage.

$$\begin{aligned}
\nabla_{v_w}\left[\sum_{(\hat{w},c)\in V^2} \log p\left(D|\hat{w},c;\theta\right)\right] &= \nabla_{v_w}\left[\sum_{(\hat{w},c)\in D} \log\sigma\left(v_{\hat{w}}\cdot v_c\right) + \sum_{(\hat{w},c)\in D^c} \log\sigma\left(-v_{\hat{w}}\cdot v_c\right)\right] = \\
&= \sum_{(\hat{w},c)\in D} \nabla_{v_w}\left[\log\sigma\left(v_{\hat{w}}\cdot v_c\right)\right] + \sum_{(\hat{w},c)\in D^c} \nabla_{v_w}\left[\log\sigma\left(-v_{\hat{w}}\cdot v_c\right)\right] = \\
&= \sum_{(\hat{w},c)\in D} \frac{1}{\sigma\left(v_{\hat{w}}\cdot v_c\right)}\cdot\nabla_{v_w}\left[\sigma\left(v_{\hat{w}}\cdot v_c\right)\right] + \sum_{(\hat{w},c)\in D^c} \frac{1}{\sigma\left(-v_{\hat{w}}\cdot v_c\right)}\cdot\nabla_{v_w}\left[\sigma\left(-v_{\hat{w}}\cdot v_c\right)\right] = \\
&= \sum_{(\hat{w},c)\in D} \frac{1}{\cancel{\sigma\left(v_{\hat{w}}\cdot v_c\right)}}\cdot\cancel{\sigma\left(v_{\hat{w}}\cdot v_c\right)}\overbrace{\left(1-\sigma\left(v_{\hat{w}}\cdot v_c\right)\right)}^{\sigma(-v_{\hat{w}}\cdot v_c)}\nabla_{v_w}\left[v_{\hat{w}}\cdot v_c\right] + \\
&+ \sum_{(\hat{w},c)\in D^c} \frac{1}{\cancel{\sigma\left(-v_{\hat{w}}\cdot v_c\right)}}\cdot\cancel{\sigma\left(-v_{\hat{w}}\cdot v_c\right)}\overbrace{\left(1-\sigma\left(-v_{\hat{w}}\cdot v_c\right)\right)}^{\sigma(v_{\hat{w}}\cdot v_c)}\nabla_{v_w}\left[-v_{\hat{w}}\cdot v_c\right] = \\
&= \sum_{(\hat{w},c)\in D} \sigma\left(-v_{\hat{w}}\cdot v_c\right)v_c\cdot\delta_{\hat{w},w} - \sum_{(\hat{w},c)\in D^c} \sigma\left(v_{\hat{w}}\cdot v_c\right)v_c\cdot\delta_{\hat{w},w} = \\
&= \sum_{c\in V\,:\,(w,c)\in D} \sigma\left(-v_w\cdot v_c\right)v_c - \sum_{c\in V\,:\,(w,c)\in D^c} \sigma\left(v_w\cdot v_c\right)v_c
\end{aligned}$$

6)

The time complexity of the computation of the objective function is $O\left(|V|^2\right)$ and for the full gradient the situation is even worse because we need to compute sum of vector terms where the sum occurs also over $|V|^2$ terms and this needs to be repeated for $|V|$ times (for each $w \in V$) in order to get the full gradient and not just one partial derivate w.r.t to v_w. if $|V| \sim 10^5 - 10^7$ then $|V|^2 \sim 10^{10} - 10^{14}$ which is too prohibitive.

7)

The set D^c is much much bigger then the set D because given a word $c \in V$, most of the words $w \in V$ in the vocabulary will not show up right after it. That means that the when we optimize over v_w under the above objective we will pay most of our attention on word representations for v_w such that the similarity between it and the words v_c which are not in the same context will be small. In general this is a good property for the final representation of words but if not enough emphesize put on high similarity between words in the same context, then it is like asking our model to find representation for all words as far as possible from one another which makes it useless when we want to ask how close is one word to another word. This explanation can be rephrased as putting too much weight on the penalty term in the objective that deal with noise words rather then on the context words and thus the model will be unable to learn good representation for context words (which is the goal of this model).

8)

In addition to the computational complexity aspect, Negative sampling helps to the problem discussed above (in section 7). The objective now is almost the same as before only the second sum is carried now over D' instead of D^c. The set D' is much smaller then D^c because in D' we add few noise words for each word $c \in V$ so it is on the order of $|V|$ while D^c is on the order of $|V|^2$. The reason it helps ot the problem discussed in section 7 is that now much less emphesize is put on the penalty term of the noise words and the overall objective is more balanced. This way the model can pay more attention to learning good representation for the context words.

9)

We would expect words with similar Word2Vec vectors to be highly associated because the model learns vector representation of words that occur in their context and usually words that occur in context of other words are not similar but associated. For example, it is much more likely that we will find in a text the word 'Maradona' near the word 'football' and not 'soccer' (which is similar to the word football) near 'football'.