Lecture 4 Quiz

7 questions

1 point

1.
The squared error cost function with n linear units is equivalent to:

Clarification:

Let's say that a network with n linear output units has some weights w. w is a matrix with n columns, and w_i indexes a particular column in this matrix and represents the weights from the inputs to the i^{th} output unit.

Suppose the target for a particular example is j (so that it belongs to class j in other words).

The squared error cost function for n linear units is given by:

$\frac{1}{2}\sum_{i=1}^{n}(t_i - w_i^T x)^2$

where t is a vector of zeros except for 1 in index j.

The cross-entropy cost function for an n-way softmax unit is given by:

$$\text{-}\log\left(\frac{\exp\left(w_j^T x\right)}{\sum_{i=1}^{n}\exp\left(w_i^T x\right)}\right) = -w_j^T x + \log\left(\sum_{i=1}^{n}\exp\left(w_i^T x\right)\right)$$

Finally, n logistic units would compute an output of $\sigma(w_i^T x) = \frac{1}{1+\exp(-w_i^T x)}$ *independently* for each class i. Combined with the squared error the cost would be:

$\frac{1}{2}\sum_{i=1}^{n}(t_i - \sigma(w_i^T x))^2$

Where again, t is a vector of zeros with a 1 at index j (assuming the true class of the example is j).

Using this same definition for t, the cross-entropy error for n logistic units would be the sum of the individual cross-entropy errors:

$$-\sum_{i=1}^{n} t_i \log(\sigma(w_i^T x)) + (1 - t_i) \log(1 - \sigma(w_i^T x))$$

For any set of weights w, the network with n linear output units will have some cost due to the squared error (cost function). The question is now asking whether we can define a new network with a set of weights w^* using some (possibly different) cost function such that:

a) $w^* = f(w)$ for some function f

b) For every input, the cost we get using w in the linear network with squared error is the same cost that we would get using w^* in the new network with the possibly different cost function.

- ☐ The cross-entropy cost function with an n-way softmax unit.
- ☐ The cross-entropy cost function with n logistic units.
- ☐ The squared error cost function with n logistic units.
- ☐ None of the above.

---

1 point

2.

A logistic unit with the cross-entropy cost function is equivalent to:

Clarification:

In a network with a logistic output, we will have a single vector of weights w. For a particular example with target t (which is 0 or 1), the cross-entropy error is given by:

$-t\log\big(\sigma(w^T x)\big) - (1-t)\log\big(1-\sigma(w^T x)\big)$ where $\sigma(w^T x) = \frac{1}{1+\exp(-w^T x)}$.

The squared error if we use a single linear unit would be:

$\frac{1}{2}\left(t - w^T x\right)^2$

Now notice that another way we might define t is by using a vector with 2 elements, [1,0] to indicate the first class, and [0,1] to indicate the second class. Using this definition, we can develop a new type of classification network using a softmax unit over these two classes instead. In this case, we would use a weight *matrix* w with two columns, where w_i is the column of the i^{th} class and connects the inputs to the i^{th} output unit.

Suppose an example belonged to class j (where j is 1 or 2 to indicate [1,0] or [0,1]). Then the cross-entropy cost for this network would be:

$$-\log\left(\frac{\exp(w_j^T x)}{\exp(w_1^T x)+\exp(w_2^T x)}\right) = -w_j^T x + \log\big(\exp(w_1^T x) + \exp(w_2^T x)\big)$$

For any set of weights w, the network with a logistic output unit will have some error due to the cross-entropy cost function. The question is now asking whether we can define a new network with a set of weights w^* using some (possibly different) cost function such that:

a) $w^* = f(w)$ for some function f

b) For every input, the cost we get using w in the network with a logistic output and cross-entropy error is the same cost that we would get using w^* in the new network with the possibly different cost function.

- ☐ A 2-way softmax unit (a softmax unit with 2 elements) with the cross entropy cost function.
- ☐ A linear unit with the squared error cost function.
- ☐ A 2-way softmax unit (a softmax with 2 elements) with the squared error cost function.

- [ ] None of the above.

---

1 point

3.

The output of a neuro-probabilistic language model is a large softmax unit and this creates problems if the vocabulary size is large. Andy claims that the following method solves this problem:

At every iteration of training, train the network to predict the current learned feature vector of the target word (instead of using a softmax). Since the embedding dimensionality is typically much smaller than the vocabulary size, we don't have the problem of having many output weights any more. Which of the following are correct? Check all that apply.

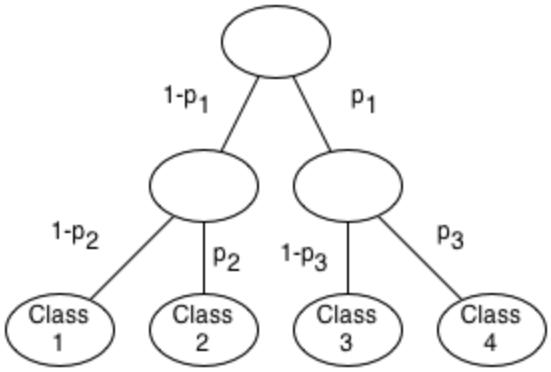
- [ ] In theory there's nothing wrong with Andy's idea. However, the number of learnable parameters will be so far reduced that the network no longer has sufficient learning capacity to do the task well.
- [ ] If we add in extra derivatives that change the feature vector for the target word to be more like what is predicted, it may find a trivial solution in which all words have the same feature vector.
- [ ] Andy is correct: this is equivalent to the serialized version of the model discussed in the lecture.
- [ ] The serialized version of the model discussed in the slides is using the current word embedding for the output word, but it's optimizing something different than what Andy is suggesting.

---

1 point

4.

We are given the following tree that we will use to classify a particular example x:



In this tree, each p value indicates the probability that x will be classified as belonging to a class in the right subtree of the node at which that p was computed. For example, the probability that x belongs to Class 2 is $(1 - p_1) \times p_2$. Recall that at training time this is a very efficient representation because we only have to consider a single branch of the tree. However, at test-time we need to look over all branches in order to determine the probabilities of each outcome.

Suppose we are not interested in obtaining the exact probability of every outcome, but instead we just want to find the class with the maximum probability. A simple heuristic is to search the tree greedily by starting at the root and choosing the branch with maximum probability at each node on our way from the root to the leaves. That is, at the root of this tree we would choose to go right if $p_1 \geq 0.5$ and left otherwise.

For this particular tree, what would make it more likely that these two methods (exact search and greedy search) will report the same class?

- ☐ It helps if the value of each p is close to 0 or 1.
- ☐ It helps if p_1 is further from 0.5. It hurts if p_2 is further from 0.5.
- ☐ It helps if the value of each p is close to 0.5.
- ☐ It helps if p_1 is close to 0.5 while p_2 and p_3 are close to 0 or 1.

---

1 point

5.

Brian is trying to use a neural network to predict the next word given several previous words. He has the following idea to reduce the amount of computation needed to make this prediction.

Rather than having the output layer of the network be a 100,000-way softmax, Brian says that we can just encode each word as an integer: 1 will correspond to the first word, 2 to the second word and so on up to 100,000. For the output layer, we can then simply use a single linear neuron with a squared error cost function. Is this a good idea?

- No. Brian is implicitly imposing the belief that words with similar integers are more related to each other than words with very different integers, and this is usually not true.
- Yes. With this method, there are fewer parameters, while the network can still learn equally well.
- Partly. Brian's method should only be used at the input layer. The output layer must always report probabilities, and squared error loss is not appropriate for that.

---

1 point

6.

In the Collobert and Weston model, the problem of learning a feature vector from a sequence of words is turned into a problem of:

- Learning to predict the middle word in the sequence given the words that came before and the words that came after.
- Learning to reconstruct the input vector.
- Learning a binary classifier.
- Learning to predict the next word in an arbitrary length sequence.

---

1 point

7.

Suppose that we have a vocabulary of 3 words, "a", "b", and "c", and we want to predict the next word in a sentence given the previous two words. Also suppose that we don't want to use feature vectors for words: we simply use the local encoding, i.e. a 3-component vector with one entry being 1 and all other two entries being 0.

In the language models that we have seen so far, each of the context words has its own dedicated section of the network, so we would encode this problem with two 3-dimensional inputs. That makes for a total of 6 dimensions; clearly, the more context words we want to include, the more input units our network must have. Here's a method that uses fewer input units:

We could instead encode the **counts** of each word in the context. So a context of "a a" would be encoded as input vector [2 0 0] instead of [1 0 0 1 0 0], and "b c" would be encoded as input vector [0 1 1] instead of [0 1 0 0 0 1]. Now we only need an input vector of the size of our vocabulary (3 in our case), as opposed to the size of our vocabulary times the length of the context (which makes for a total of 6 in our case). Are there any significant problems with this idea?

- Yes: even though the input has a smaller dimensionality, each entry of the input now requires more bits to encode, because it's no longer just 1 or 0. Therefore, there would be no significant advantage.
- Yes: the network loses the knowledge of the location at which a context word occurs, and that is valuable knowledge.
- Yes: although we could encode the context in this way, we would then need a smaller bottleneck layer than we did before, thereby lowering the learning capacity of the model.
- Yes: the neural networks shown in the course so far cannot deal with integer inputs (as opposed to binary inputs).

I, **Jian Wang**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account. Learn more about Coursera's Honor Code

3 questions unanswered

Submit Quiz