Lecture 4 Quiz

Quiz, 7 questions

5/7 points (71%)

Try again once you are ready.

Required to pass: 80% or higher

You can retake this quiz up to 3 times every 8 hours.

Back to Week 4

Retake

0 / 1
points

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.

Un-selected is correct

The cross-entropy cost function with n logistic units.

Un-selected is correct

The squared error cost function with n logistic units.

This should not be selected

None of the above.

This should be selected

---

1 / 1
points

2.

A 2-way softmax unit (a softmax unit with 2 elements) 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:

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

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

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

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:

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

For any set of weights w, the network with a softmax output unit over 2 classes 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 softmax output unit over 2 classes and cross-entropy error is the same cost that we would get using w^* in the new network with the possibly different cost function.

- [x] A logistic unit with the cross-entropy cost function.

Correct

With weights w_1 and w_2 we can write the softmax probability for an input x as:

$$P(\text{x belongs to class 1}) = \frac{\exp(w_1^Tx)}{\exp(w_1^Tx)+\exp(w_2^Tx)}.$$

Dividing the top and bottom by $\exp(w_1^T x)$:

$$P(\text{x belongs to class 1}) = \frac{1}{1+\exp(w_2^T x - w_1^T x)} = \frac{1}{1+\exp(-(w_1-w_2)^T x)}.$$

So this is equivalent to a logistic unit with weights $w_1 - w_2$. Since the cross-entropy is the negative log-probability, it will be the same for both representations.

- [ ] A 2-way softmax unit (a softmax unit with 2 elements) with the squared error cost function.
- [ ] Two linear units with the squared error cost function.
- [ ] None of the above.

---

1 / 1 points

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.

- [x] 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.

Correct

- [ ] Andy is correct: this is equivalent to the serialized version of the model discussed in the lecture.

Un-selected is correct

- [ ]

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.

Un-selected is correct

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.

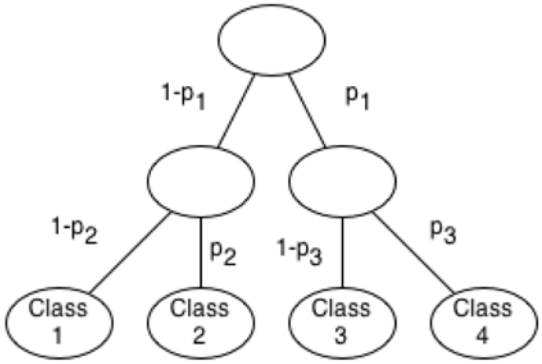
Correct

---

1 / 1
points

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 with 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.

If $p_1 = 0.45$, $p_2 = 0.6$, and $p_3 = 0.95$, then which class will the following methods report for x?

a) Evaluate the probabilities of each of the four classes (the leaf nodes) and report the class of the leaf node with the highest probability. This is the standard approach but may take quite some time.

b) The proposed alternative approach: greedily traverse the tree by choosing the branch with the highest probability and report the class of the leaf node that this finds.

- Method a) will report class 1

  Method b) will report class 4

- Method a) will report class 4

  Method b) will report class 2

Correct

- [ ] Method a) will report class 3

  Method b) will report class 2

- [ ] Method a) will report class 4

  Method b) will report class 1

- [ ] Method a) will report class 2

  Method b) will report class 2

- [ ] Method a) will report class 2

  Method b) will report class 3

---

0 / 1
points

5.
True or false: the neural network in the lectures that was used to predict relationships in family trees had "bottleneck" layers (layers with fewer dimensions than the input). The reason these were used was to prevent the network from memorizing the training data without learning any meaningful features for generalization.

- [x] False

This should not be selected

- [ ] True

---

1 / 1
points

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:

- [x] Learning a binary classifier.

Correct

The model is made to answer this binary question: "is the word in the middle correct, or is it just some random word?"

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

---

1 / 1
points

7.