6.867: Exercises (Week 7)

October 27, 2017

Contents

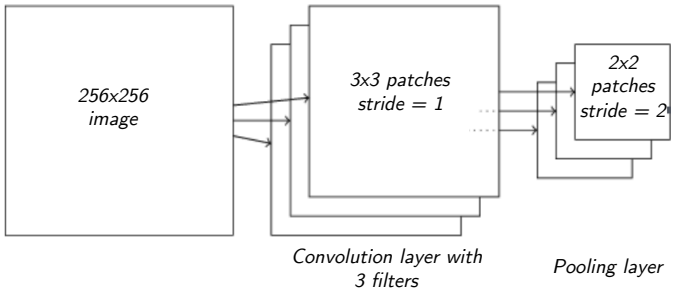
Solution: Don't look at the solutions until you have tried your absolute hardest to solve the problems.

1 ConvNets

Adapted from UofT CSC321 Final 2017.

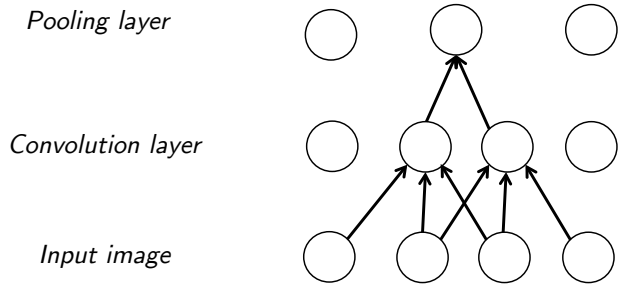
Suppose you have a convolutional network with the following architecture:

- The input is an image of size 256×256.
- The first layer is a convolution layer with 3 filters with patches of size 3×3. It uses a stride of 1 and zero padding with 1 pixel border (i.e. adding one row and column to the borders of the image with zero values so the new size is 258 * 258).
- The next layer is a max pooling layer with a stride of 2 and 2×2 filters.



(a) Determine the size of the receptive field for a single unit in the pooling layer. (i.e., determine the size of the region of the input image which influences the activation of that unit.) You may assume the receptive field lies entirely within the image. Hint: you may want to draw a one-dimensional conv net to reason about this problem.

Solution: The following figure shows a 1-D analogue:



In this figure, 4 of the input units feed into the pooling unit. This happens along both dimensions in the 2-D case, so the size of the receptive field is 4×4

(b) How many units and parameters are there in the convolution layer?

Solution: The filters are of size 3×3 and the stride is 1. We also have a zero padding with one pixel border. Hence, for each dimension of the convolution layer we have:

$$(256 + 2 * 1 - 3)/1 + 1 = 256,$$

where we add $2 * 1$ for the zero padding, subtract 3 for the filter size and divide by 1 for the stride. As a result, the size of the convolution layer is 256 * 256 * 3 and it has the same width and height as the original image.

For the number of parameters we have: 3*3+1 = 10 parameters for each filter (+1 for the bias term) and 3 * 10 = 30 for all the 3 filters.

(c) How many units are there in the max pooling layer?

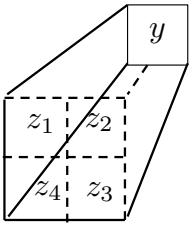
Solution: Similar to the convolution layer, we have:

$$(256 - 2)/2 + 1 = 128,$$

As a result it reduces the size of each dimension by a factor of 2 and the size of the layer will be 3 * 128 * 128.

(d) What is the derivative of the output of the max pooling layer with respect to its inputs?

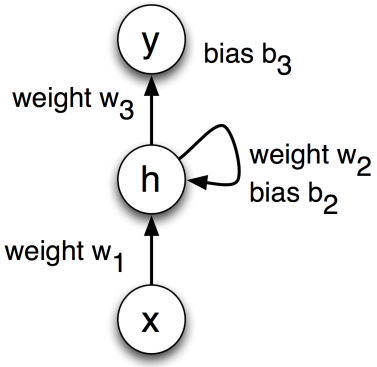
$$y = \max\{z_1, z_2, z_3, z_4\}$$



Solution: Assume $i^* = \arg\max_i z_i$, we have $\frac{\partial y}{\partial z_{i^*}} = 1$ and $\frac{\partial y}{\partial z_j} = 0$ for $j \neq i^*$. This means the gradient is passed back only to the unit with max value and the other units are not updated.

2 Mystery RNN

Adapted from UofT CSC321 Winter 2015.



You are given an RNN with the architecture shown in the figure above where all inputs and outputs are binary valued (ie. $x, y \in \{0, 1\}$). You observe that it initially outputs 1, but as soon as it gets an input of 0, it switches to outputting 0 for all subsequent time steps. For example, the sequence of inputs 1110101 produced the outputs 1110000. You know that the hidden unit has an initial value of 0 and that it uses the binary threshold function for its non-linearity, which is defined as follows:

$$f(z) = \begin{cases} 1, & z > 0 \\ 0, & otherwise. \end{cases}$$

The output unit is linear. The network can be described by the following set of equations where x_t is the input and y_t is the output at time t and $f()$ is the binary threshold function:

$$h_t = f(w_1 x_t + w_2 h_{t-1} + b_2)$$
$$y_t = w_3 h_t + b_3$$

Provide a set of values for the weights and biases that this network could be using (there is more than one correct answer).

Solution: This network can be modelled as two functions parameterized by the unknown weights and biases:

$$h_t = g(x_t, h_{t-1})$$
$$y_t = k(h_t)$$

Since y_t is a function of h_t alone, we know that the change in y from ones to zeros caused by an $x = 0$ must be due to a change in h. Since h is the result of a binary threshold function which can only take the values $\{0, 1\}$ and $h_0 = 0$, the value of h must flip from 0 to 1 after the

first $x = 0$ and then continue to be 1 regardless of subsequent values of x. This means:

$$0 = g(1, 0)$$
$$1 = g(0, 0)$$
$$1 = g(1, 1)$$
$$1 = g(0, 1)$$

A possible set of parameters that gives rise to this function is $w_1 = -1, w_2 = 1, b_2 = 0.5$. Finally, our analysis gives the following constraints for the second function:

$$1 = k(0)$$
$$0 = k(1)$$

For which a possible set of parameters is $w_3 = -1, b_3 = 1$. Again, the exact parameter values for the solution are not unique.

3 RNNs For Language Models

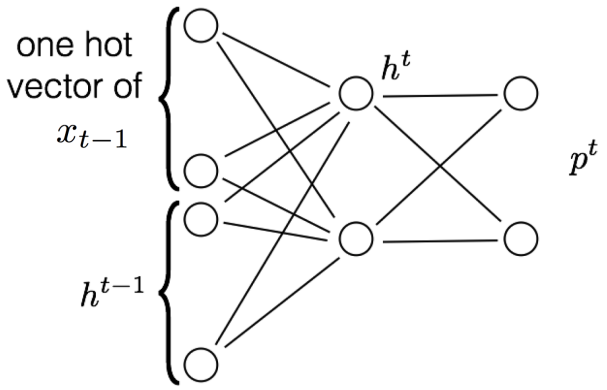
Adapted from MIT 6.864 Fall 2015

Language models often make use of n-gram models, which mean the probability of a given word in a sequence is only dependent on the previous n words. For example, a bi-gram model would model the probability of a given word at position t of a sequence as $p(x_t|x_{t-1})$ whereas a tri-gram model would be $p(x_t|x_{t-1}, x_{t-2})$. In this question we examine the use of RNNs for language models.

<start>	This	is	a	very	very	unimaginative	example	sentence
t=0	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8

(a) Consider the example sentence shown above. Is it possible for a bi-gram model to assign a likelihood of 1 to this sentence? Explain.

Solution: No because $p(x_4|x_3)$ and $p(x_5|x_4)$ cannot both be 1 since $x_4 = x_3$.



The figure above shows the architecture for an RNN that we will try to use for language modelling. The hidden state h^t consists of m units with a binary threshold activation function defined as follows:

$$f(z) = \begin{cases} 1, & z > 0 \\ 0, & otherwise. \end{cases}$$

The initial hidden state h^0 is always $h^0_1 = 1$ and $h^0_j = 0$ for $j = 2, \ldots, m$. The output layer is linear followed by a softmax to represent the probabilities over the next word x_t. You can assume all bias terms are -1.

(b) Suppose we eliminate h^{t-1} as input to the network. Is it still possible to set the remaining parameters of the network to assign a likelihood of 1 to the example sentence above? Explain.

Solution: No. This would just be a bi-gram model which we have already shown is insufficient.

(c) Suppose we eliminate x_{t-1} as input to the network. Is it still possible to set the remaining parameters of the network to assign a likelihood of 1 to the example sentence above? Explain.

Solution: Yes. For example you can set h^t to keep track of the index t and have the final layer be a mapping from index to the corresponding word in the sentence.

The last few parts will consider the whole network (with both h^{t-1} and x_{t-1} as inputs) and show how it can be used to exactly replicate a tri-gram model.

(d) Why is having h^t be the vector corresponding to the concatenation of one hot vectors for x_{t-1} and x_{t-2} insufficient for producing a tri-gram model?

Solution: Since the output layer is linear, a vector of size $2|V|$, where $|V|$ is the vocabulary size, is insufficient for representing the $|V|^2$ mappings of a tri-gram model.

(e) What is a possible vector representation of h^t that would allow the network to replicate a tri-gram model?

Solution:

$$concat(\phi(x_{t-1}), vec(\phi(x_{t-1})\phi(x_{t-2})^T))$$

Where $vec()$ takes a matrix as input and returns its rows concatenated as a vector and $\phi()$ is a one hot encoding.

(f) What is a possible parameterization of the network that results in this hidden state representation and the network replicating a tri-gram model?

Solution: Let W_1 be the weight matrix connecting the $\phi(x_{t-1})$ input nodes to the nodes in h^t representing $\phi(x_{t-1})$. $W_1 = 2I$ where I is the identity matrix. This results in the exact input values mapped to h^t as desired.

Let W_2 be the weight matrix connecting the $\phi(x_{t-1})$ input nodes to the nodes in h^t representing $vec(\phi(x_{t-1})\phi(x_{t-2})^T)$. If $|V|$ is the vocabulary size, and therefore the size of a one hot vector, W_2 is a $|V|^2 \times |V|$ matrix composed by stacking $|V|$ copies of the identity matrix I.

Let W_3 be the $|V|^2 \times |V|$ weight matrix connecting the h^{t-1} input nodes to the nodes in h^t representing $vec(\phi(x_{t-1})\phi(x_{t-2})^T)$. W_3 is constructed by stacking $|V|$ $|V| \times |V|$ matrices where matrix i has all ones in column i and all other columns are zeros.

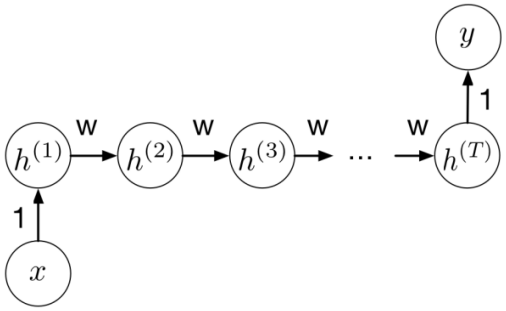
W_2 and W_3 together compute the cross product terms of h^t where only the unit corresponding to the exact word pair x_{t-1}, x_{t-2} is one and all others are zero.

Also, the h^t nodes are connected to the h^{t-1} nodes via a weight matrix equal to 2I allowing the exact h^t vector to become h^{t-1} for the next time step.

Finally the output layer maps each of the $vec(\phi(x_{t-1})\phi(x_{t-2})^T)$ nodes of h^t to the output with weights corresponding to the desired distribution over x_t.

All weights that were not specified are zero.

4 Exploding RNN



Consider the following RNN, which has a scalar input at the first time step, makes a scalar prediction at the last time step, and uses a shifted logistic activation function:

$$\phi(z) = \sigma(z) - 0.5$$

(a) Let z^t denote the input to the activation function at time t. Write the formula for the derivative of the loss $\frac{\delta l}{\delta h^{(t)}}$ in terms of $\frac{\delta l}{\delta h^{(t+1)}}$ for $t < T$.

Solution:

$$\frac{\delta l}{\delta h^{(t)}} = \frac{\delta l}{\delta h^{(t+1)}} \phi'(z^t) w$$

(b) Suppose the input to the network is $x = 0$. Notice that $h^{(t)} = 0$ for all t. Based on your answer to the previous part, determine the value α such that if $w < \alpha$, the gradient vanishes, while if $w > \alpha$, the gradient explodes. You may use the fact that $\sigma'(0) = 0.25$.

Solution: If we have $\frac{\delta l}{\delta h^{(t)}} = \frac{\delta l}{\delta h^{(t+1)}}$, then the gradient will be stable. This means we want $\phi'(z^t)w = 1$. For an input of 0, we will have $z^t = 0$ for all t regardless of w. So with $\phi'(0)w = 1$, $w = \alpha = 4$ gives the desired properties.

(c) Suppose now that we make this RNN a residual network. Instead of $h^{(t+1)} = \phi(z^t)$, we will connect the input of a recurrent unit to its output, so that we now have

$$h^{(t+1)} = \phi(z^t) + h^t$$

Give a setting of w such that the new gradient does not vanish or explode.

Solution:

$$\frac{\delta l}{\delta h^{(t)}} = \frac{\delta l}{\delta h^{(t+1)}}(\phi'(z^t)w + 1)$$

As before, we desire $\phi'(z^t)w + 1 = 1$. Setting $w = 0$ will achieve this.

5 More Exploding RNN

Assume we have an RNN with T hidden layers and activation function ϕ. Let h_t and h_T be the hidden unit vectors for hidden layers T and t of the RNN. Assume $t \ll T$. We have the following:

$$z_{t+1} = \mathbf{W}_t h_t + \mathbf{V}_t x_{t+1}$$

$$h_{t+1} = \phi(z_{t+1})$$

(a) Write the derivative of the loss l with respect to h_t in terms of $\delta l/\delta h_T$, $\phi'(z)$ and $\mathbf{W}$ for each layer.

Solution: Let $\mathbf{D}_{k+1} = Diag(\phi'(z_{k+1}))$

$$\frac{\delta l}{\delta h_t} = \frac{\delta l}{\delta h_T} \prod_{k=t}^{T-1} \mathbf{D}_{k+1} \mathbf{W}_k^T$$

(b) Let's look at the norm of $\delta l/\delta h_t$. We define the norm of a vector to be the usual L2-norm, and the norm of a matrix to be the operator 2-norm. Using the derivative from above, come up with an upper bound for $\|\delta l/\delta h_t\|$. Hint: use the facts that for matrix $\mathbf{A}$ and vector v, $\|\mathbf{A}v\| \leqslant \|\mathbf{A}\| \, \|v\|$.

$$\left\| \frac{\delta l}{\delta h_t} \right\| = \left\| \frac{\delta l}{\delta h_T} \prod_{k=t}^{T-1} \mathbf{D}_{k+1} \mathbf{W}_k^T \right\|$$

$$\leqslant \left\| \frac{\delta l}{\delta h_T} \right\| \prod_{k=t}^{T-1} \left\| \mathbf{D}_{k+1} \mathbf{W}_k^T \right\|$$

(c) If we are looking to prevent exploding or vanishing gradients, what might be a useful property of the weight matrices $\mathbf{W}$? Assuming the weights indeed satisfy this property, simplify the bound given for $\|\delta l/\delta h_t\|$ in the previous part. Hint: if $\mathbf{A}$ is orthogonal, we have $\|\mathbf{AB}\| = \|B\|$. (Recall that an orthogonal matrix is a square matrix whose columns and rows are perpendicular to each other and have norm 1. For orthogonal matrix $\mathbf{A}$, $\mathbf{A}^T\mathbf{A} = \mathbf{I}$.)

Solution: If the weight matrices were orthogonal, then the product over weights in all layers will not explode:

$$\left\| \frac{\delta l}{\delta h_t} \right\| \leqslant \left\| \frac{\delta l}{\delta h_T} \right\| \prod_{k=t}^{T-1} \left\| \mathbf{D}_{k+1} \mathbf{W}_k^T \right\|$$

$$= \left\| \frac{\delta l}{\delta h_T} \right\| \prod_{k=t}^{T-1} \left\| \mathbf{D}_{k+1} \right\|$$

(d) Notice that the values of the matrices $\mathbf{D}$ now determine whether or not the gradients explode or vanish. Based on this, why might the ReLU be a good choice for the activation function?

Solution: Unless all the activations are killed at one layer (in which case $\mathbf{D}$ is 0, and the gradient is 0), the maximum entry of $\mathbf{D_k}$ is 1, resulting in $\|\mathbf{D_k}\| = 1$ for all layers k. This leaves us with the nice, not-exploding gradient:

$$\left\| \frac{\delta l}{\delta h_t} \right\| \leqslant \left\| \frac{\delta l}{\delta h_T} \right\| \prod_{k=t}^{T-1} \left\| \mathbf{D}_{k+1} \right\| = \left\| \frac{\delta l}{\delta h_T} \right\|$$

6 Multi-task learning

You come across an exciting data set consisting of a million images of cats. Each image is annotated with (i) the cat's *breed*, (ii) the cat's *gender*, and (iii) the *time* of the day when the picture was taken. You set out to test how predictive these target variables are given the image using deep CNNs. This can be viewed as a *multi-task learning* problem where predicting each target is considered a separate task. You consider the following two approaches:

- Approach 1: Train a separate network for each target variable.
- Approach 2: Train a single network that jointly predicts different target variables based on the last hidden layer. Everything up to the last hidden layer is shared.

(a) You are informed by a cat expert that both (i) and (ii) can be accurately determined based on a common set of morphological features of a cat. Which of the two approaches is preferable for predicting (i) and (ii) together? Why?

Solution: Approach 2. Given the similarity of the two tasks, learned features that can predict *both* (i) and (ii) well are less prone to overfitting and are expected to generalize better compared to learned features that are specialized for either task.

(b) Assume the features that were useful for (i) and (ii) contain no predictive signal for (iii) and vice versa. With proper regularization, which approach is preferable for predicting (i) and (iii) together and why? What could happen if we are in the regime of overfitting?

Solution: Approach 1. Since the two tasks are mostly orthogonal, there is no synergistic effect in jointly tackling them. The performance may actually suffer if the model capacity is too limited to extract all relevant signals for both tasks from the image. However, if Approach 1 is not properly regularized and overfits the data, then Approach 2 may achieve better generalization, because limited capacity can counteract overfitting.

(c) You hypothesize that the first two hidden layers of a network trained for any of the three tasks capture rudimentary image features (e.g., edges, brightness, and contrast) that are useful for any task. Assuming this is indeed the case, find a compromise between Approaches 1 and 2 to design a *single* network for predicting all three target variables together, using the insights from (a) and (b).

Solution: Share the first two layers across all tasks. Create a fork with two branches, both with desired number of hidden layers: one for (i) and (ii) and the other for (iii). The former branch is shared between (i) and (ii) until the last hidden layer, at which point it outputs predictions for both (i) and (ii). This way, we allow the pooling of signals for more effective feature learning at different levels of abstraction, while still allowing task-specific behaviors where appropriate.

(d) You found another data set containing a million images of *dogs*. Each image is annotated with the dog's *breed*, which we would like to predict. How would you modify your model in (c) to simultaneously tackle this fourth task, which is defined over a whole new data set? Briefly describe how the new model would be trained.

Solution: The filters that extract morphological features from a cat that are informative for predicting the cat's breed is likely to be similar to that for dogs. We can simply insert another output unit that corresponds to the predicted breed of a dog at the end of the branch for (i) and (ii) and share all hidden layers with (i) and (ii). When we compute the gradient given a cat image during training, the output node for the dog is neglected. On the other hand, given a dog image, it is the only output that contributes to the gradient for backpropagation.