Practical understanding

An HMM problem is defined in terms of three matrices. The initial probability matrix which holds initial estimates of what are the probabilities of the system being at time=0, the transition matrix which holds the conditional probabilities of the system transitioning from the one hidden state to the next and the emission matrix which holds the conditional probabilities that an observation is recorded given a specific state. It is important to say that the transimission and emission matrices don't change in the case of the forward algorithm.

$$\mathbf{P_{initial}} = \begin{pmatrix} P_{state_1} \\ P_{state_2} \\ \vdots \\ P_{state_n} \end{pmatrix} (size = N \times 1)$$

$$\mathbf{Transition} = \begin{pmatrix} P_{state_{1_t}|state_{1_{t-1}}} & P_{state_{2_t}|state_{1_{t-1}}} & \cdots & P_{state_{n_t}|state_{1_{t-1}}} \\ P_{state_{1_t}|state_{2_{t-1}}} & P_{state_{2_t}|state_{2_{t-1}}} & \cdots & P_{state_{n_t}|state_{2_{t-1}}} \\ \vdots & \vdots & \vdots & \vdots \\ P_{state_{1_t}|state_{n_{t-1}}} & P_{state_{2_t}|state_{n_{t-1}}} & \cdots & P_{state_{n_t}|state_{n_{t-1}}} \end{pmatrix} (size = N \times N)$$

$$\mathbf{Emission} = \begin{pmatrix} P_{observation_1|state_1} & P_{observation_2|state_1} & \cdots & P_{observation_k|state_1} \\ P_{observation_1|state_2} & P_{observation_2|state_2} & \cdots & P_{observation_k|state_2} \\ \vdots & \vdots & \vdots & \vdots \\ P_{observation_1|state_n} & P_{observation_2|state_n} & \cdots & P_{observation_k|state_n} \end{pmatrix} (size = N \times K)$$

For ease we are going to use the matrices in the example code

$$\mathbf{P_{initial}} = \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix}$$

$$\mathbf{Transition} = \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix}$$

$$\mathbf{Emission} = \begin{pmatrix} 0.5 & 0.4 & 0.1 \\ 0.1 & 0.3 & 0.6 \end{pmatrix}$$

The states are the weather conditions rain R (1^{st} state) and sunny S (2^{nd} state). The observations are some specific person cleaning C (1^{st} column of $Emission$), shopping Sh (2^{nd} column of $Emission$) and going for a walk W (3^{rd} column of $Emission$), conditional on the weather.

Example

Let's say we want to calculate the probability of observing that person first going shopping the first day and then going for a walk the following day.

Initialise the first step

We are at t=0 and we want to calculate $P(Sh)$ conditional an all possible states, thus
$P(Sh) = P(R)P(Sh|R) + P(S)P(Sh|S)$

we can get $P(R), P(S)$ from the $P_{initial}$ matrix

$P(Sh|R), P(Sh|S)$ from the $Emission$ matrix

So $P(Sh) = 0.6x0.4 + 0.4x0.3 = 0.24 + 0.12 = 0.36$. Note that in the calculations in the code below the variable `fwd` will hold these two values after the execution of `forward_init_op` based on the following tensor operations

$$\mathbf{fwd_{shop}} = \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \odot \begin{pmatrix} 0.4 \\ 0.3 \end{pmatrix} = \begin{pmatrix} 0.24 \\ 0.12 \end{pmatrix}$$

Forward propagation

No we are at step t=1. From the previous step we calculated $P(Sh_{t=0}) = 0.36$ which can be broken down to $P(Sh_{t=0}) = P(R_{t=0}) + P(S_{t=0})$, where $P(R_{t=0}) = 0.24, P(S_{t=0}) = 0.12$

The general formula for the forward propagation is given by the following:

$$P_t(i) = \sum_j P_{t-1}(j) \times P(Q_t = S_i|Q_{t-1} = S_j) \times P(O_t|Q_t = S_i)$$

where

$P_t(i)$ is the probability of being in the state i at time t

$P_{t-1}(j)$ is the probability of being in state j at time $t-1$. This would be given from the initialisation step as $P_{t-1}(R) = 0.24, P_{t-1}(S) = 0.12$. In general, for each step this would be given from the result of the previous step.

$P(O_t|Q_t = S_i)$ is given by the $Emission$ matrix, for example $P(C|R) = 0.5, P(C|S) = 0.1, \ldots$

$P(Q_t = S_i|Q_{t-1} = S_j)$ is given by the $Transition$ matrix, for example $P(R_t|R_{t-1}) = 0.7, P(R_t|S_{t-1}) = 0.4, \ldots$

So now we have all we need to calculate the $P_t(R)$ and $P_t(S)$

$$P_t(R) = P_{t-1}(R) \times P(W|R) \times P(R_t|R_{t-1}) + P_{t-1}(S) \times P(W|R) \times P(R_t|S_{t-1}) = 0.24 \times 0.1$$
$$\times P(R_t|S_{t-1}) = 0.024 \times P(R_t|R_{t-1}) + 0.012 \times P(R_t|S_{t-1})$$

note that if we run the code for the same sequence of the observations the values [0.024, 0.012] would correspond to the first column of the tensor `transitions` in the `forward_op` function.

$$\mathbf{transitions_{walk}} = \begin{pmatrix} 0.24 \\ 0.12 \end{pmatrix} \times \begin{pmatrix} 0.1 \\ 0.6 \end{pmatrix}^T = \begin{pmatrix} 0.024 & 0.144 \\ 0.012 & 0.072 \end{pmatrix}$$

$$P_t(R) = P_{t-1}(R) \times P(W|R) \times P(R_t|R_{t-1}) + P_{t-1}(S) \times P(W|R) \times P(R_t|S_{t-1}) = 0.24 \times 0.1$$
$$\times P(R_t|S_{t-1}) = 0.024 \times 0.7 + 0.012 \times 0.4 = 0.0168 + 0.0048$$

note that the values correspond to the first column of the `weighted_transitions` in the `forward_op` function.

$$\mathbf{weighted_transitions_{walk_i}} = \begin{pmatrix} 0.024 & 0.144 \\ 0.012 & 0.072 \end{pmatrix} \odot \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix} = \begin{pmatrix} 0.0168 & 0.0432 \\ 0.0048 & 0.0432 \end{pmatrix}$$

Finally we can write $P_t(R) = 0.0216$

we repeat the same to calculate $P_t(S)$

$$P_t(S) = P_{t-1}(R) \times P(W|S) \times P(S_t|R_{t-1}) + P_{t-1}(S) \times P(W|S) \times P(S_t|S_{t-1}) = 0.24 \times 0.6 \times$$
$$+ 0.072 \times 0.6 = 0.0432 + 0.0432 = 0.0864$$

Note how the results of this operation are contained in the 2^{nd} column of tensors `transitions` and `weighted_transitions` in the `forward_op` function.

Final probability

We calculated $P_t(R) = 0.0216$ and $P_t(S) = 0.0864$ and we add them together to calculate the final probability of observing the sequence {shop,walk} $0.0216 + 0.0864 = 0.108$

Run the following code with input `observations = [1,2]` to make sure you get the correct answer

Code

Oof this code's a bit complicated if you don't already know how HMMs work. Please see the book chapter for step-by-step explanations. I'll try to improve the documentation, or feel free to send a pull request with your own documentation!

First, let's import TensorFlow and NumPy:

In [1]:

```
import numpy as np
import tensorflow as tf
```

Define the HMM model:

In [2]:

```
class HMM(object):
    def __init__(self, initial_prob, trans_prob, obs_prob):
        self.N = np.size(initial_prob)
        self.initial_prob = initial_prob
        self.trans_prob = trans_prob
        self.emission = tf.constant(obs_prob)

        assert self.initial_prob.shape == (self.N, 1)
        assert self.trans_prob.shape == (self.N, self.N)
        assert obs_prob.shape[0] == self.N

        self.obs_idx = tf.placeholder(tf.int32)
        self.fwd = tf.placeholder(tf.float64)

    def get_emission(self, obs_idx):
        slice_location = [0, obs_idx]
        num_rows = tf.shape(self.emission)[0]
        slice_shape = [num_rows, 1]
        return tf.slice(self.emission, slice_location, slice_shape)

    def forward_init_op(self):
        obs_prob = self.get_emission(self.obs_idx)
        fwd = tf.multiply(self.initial_prob, obs_prob)
        return fwd

    def forward_op(self):
        transitions = tf.matmul(self.fwd, tf.transpose(self.get_emission(self.ob
s_idx)))
        weighted_transitions = transitions * self.trans_prob
        fwd = tf.reduce_sum(weighted_transitions, 0)
        return tf.reshape(fwd, tf.shape(self.fwd))
```

Explanation of extracting the emission probabilities for a specific observation

i.e. explain function:

```
def get_emission(self, obs_idx):
        slice_location = [0, obs_idx]
        num_rows = tf.shape(self.emission)[0]
        slice_shape = [num_rows, 1]
        return tf.slice(self.emission, slice_location, slice_shape)
```

We want to extract a column from the observation probabilities tensor corresponding to the emission probabilities of a specific observation

$$\mathbf{P_{emission}} = \begin{pmatrix} 0.5 & 0.4 & 0.1 \\ 0.1 & 0.3 & 0.6 \end{pmatrix}$$

where each row is the hidden state (rainy[0], sunny[1]) and each column corresponds to an observed activity: clean[0], shop[1], walk[2]

`obs_idx` is a tensor holding the index of the observation activity 0->clean, 1-> shop, 2-> walk. `slice location` is a list with the location in the 2D $P_{emission}$ of where the slice should begin. For each of the observations this `slice_location` would be clean->[0,0], shop->[0,1], walk->[0,2].

`num_rows` is a tensor holding the number of rows of $P_{emission}$ and it corresponds to the end of the slice. It evaluates to an integer (i.e. 2).

`slice_shape` is a list holding the shape of slice to extract from $P_{emission}$. It holds the tensor `num_rows` and the number 1. It tells `tf.slice` to return an tensor with `num_rows`(2) rows and 1 column.

examples for `tf.slice(self.emission, slice_location, slice_shape)`:

```
tf.slice(self.emission, [0,0], [2,1])
Out[0]:
[[0.5]
 [0.1]]

tf.slice(self.emission, [0,1], [2,1])
Out[0]:
[[0.4]
 [0.3]]

tf.slice(self.emission, [0,2], [2,1])
Out[0]:
[[0.1]
 [0.6]]
```

Explanation of initialising the forward propagation

i.e. explain function:

```
def forward_init_op(self):
        obs_prob = self.get_emission(self.obs_idx)
        fwd = tf.multiply(self.initial_prob, obs_prob)
        return fwd
```

Element-wise multiplication of the initial probabilities of each state with the emission probabilities for a specific observation.

For example

$$\mathbf{FW_{clean}} = \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \odot \begin{pmatrix} 0.5 \\ 0.1 \end{pmatrix} = \begin{pmatrix} 0.3 \\ 0.04 \end{pmatrix}$$

$$\mathbf{FW_{shop}} = \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \odot \begin{pmatrix} 0.4 \\ 0.3 \end{pmatrix} = \begin{pmatrix} 0.24 \\ 0.12 \end{pmatrix}$$

$$\mathbf{FW_{walk}} = \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \odot \begin{pmatrix} 0.1 \\ 0.6 \end{pmatrix} = \begin{pmatrix} 0.06 \\ 0.24 \end{pmatrix}$$

Explanation of forward propagation

i.e. explain function:

```
def forward_op(self):
        transitions = tf.matmul(self.fwd, tf.transpose(self.get_emission(
self.obs_idx)))
        weighted_transitions = transitions * self.trans_prob
        fwd = tf.reduce_sum(weighted_transitions, 0)
        return tf.reshape(fwd, tf.shape(self.fwd))
```

See practical understanding section for explanation

Step 1: Calculation tranistion matrix

For a specific observation, multiply the previous result of forward propagation with the emission probabilities of that observation

$$\mathbf{Transition_i} = \mathbf{FW_{i-1}} \times \mathbf{P_{emission_{observation}}}^T$$

example for observation 3 'walk':

$$\mathbf{Transition_{walk_i}} = \begin{pmatrix} 0.06 \\ 0.24 \end{pmatrix} \times \begin{pmatrix} 0.1 \\ 0.6 \end{pmatrix}^T = \begin{pmatrix} 0.006 & 0.036 \\ 0.024 & 0.144 \end{pmatrix}$$

Step 2: Calculate the weighted transitions

Perform element-wise multiplication between original transition matrix with the transition matrix calculated from the previous step.

$$\mathbf{WeightedTransition_i} = \mathbf{Transition_i} \odot \mathbf{OriginalTransition}$$

example for observation 3 'walk':

$$\mathbf{WeightedTransition_{walk_i}} = \begin{pmatrix} 0.006 & 0.036 \\ 0.024 & 0.144 \end{pmatrix} \odot \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix} = \begin{pmatrix} 0.0042 & 0.0108 \\ 0.0096 & 0.0864 \end{pmatrix}$$

Step 3: Update forward step

Sum weighted transitions along rows to calculate next forward vector and transpose matrix to get final forward step

example for observation 3 'walk':

$$\begin{pmatrix} 0.0042 & 0.0108 \\ 0.0096 & 0.0864 \end{pmatrix}$$

summation:

$$\begin{pmatrix} 0.0138 & 0.0972 \end{pmatrix}$$

transpose:

$$\begin{pmatrix} 0.0138 \\ 0.0972 \end{pmatrix}$$

Define the forward algorithm:

In [3]:

```
def forward_algorithm(sess, hmm, observations):
    fwd = sess.run(hmm.forward_init_op(), feed_dict={hmm.obs_idx: observations[0
]})
    for t in range(1, len(observations)):
        fwd = sess.run(hmm.forward_op(), feed_dict={hmm.obs_idx: observations[t
], hmm.fwd: fwd})
    prob = sess.run(tf.reduce_sum(fwd))
    return prob
```

Let's try it out:

In [10]:

```
if __name__ == '__main__':
    initial_prob = np.array([[0.6], [0.4]])
    trans_prob = np.array([[0.7, 0.3], [0.4, 0.6]])
    obs_prob = np.array([[0.5,0.4,0.1], [0.1, 0.3, 0.6]])
    print('Initial matrix = \n{}'.format(initial_prob))
    print('Trasition matrix = \n{}'.format(trans_prob))
    print('Observation matrix = \n{}'.format(obs_prob))

    hmm = HMM(initial_prob=initial_prob, trans_prob=trans_prob, obs_prob=obs_pro
b)

#     observations = [0, 1, 1, 2, 1]
    observations = [1, 2]
    with tf.Session() as sess:
        prob = forward_algorithm(sess, hmm, observations)
        print('Probability of observing {} is {}'.format(observations, prob))
```

```
Initial matrix =
[[0.6]
 [0.4]]
Trasition matrix =
[[0.7 0.3]
 [0.4 0.6]]
Observation matrix =
[[0.5 0.4 0.1]
 [0.1 0.3 0.6]]
Probability of observing [1, 2] is 0.10799999999999998
```