I used feature based approach to do my project. I considered the frog’s immediate surrounding features as the state of the frog. These features are chosen the below features:

[up\_danger ,right\_danger ,left\_danger ,down\_danger ]

In each direction, (up, down , left, right and stay(no-op)), I set the possible dangers as the possible danger in that direction. For instance, if the frog goes left, is there any car that is coming toward the frog? Or is there any empty spot (river) that the frog might fall in, is there any turtle? If any of this are true about that location, I consider that to the danger of going to that direction. And the same for other actions.

I tried these abstract or more generic features for some reasons which I am going to explain here. At first, I tried several different approaches which each encountered several problems and when I analyzed them I came up with the above final features.

At first, I considered numbers of cars around the frog in its area in a window of 3, number of dangerous cars, those that are coming toward the frog (I figured out their direction based on their location in the previous step and the current step), number of river slots, number of turtles that are coming toward the frog, number of bridges that are on the frog’s way immediately. In that model, the frog would learn the road side, but when entering the river, it would die immediately. It even didn’t get the chance to learn the river’s environment.

As it didn’t work I realized that my features are specific and don’t remain practical as the environment changes from the road to the river part. For instance, at the road side, the frog learns after several failings, that when it is crushed by a car, it will die. So it learns that whenever there is no danger in terms of car, it is safe to go up! (as we considered number of cars in each direction when we trained the frog. But it is not like that when entering the river side, of course because there is no cars at all there. The frog sees no car and thinks it is great and It should go in that direction, which is obviously wrong and the frog dies immediately. So now that we have two kinds of environment, where deciding features are not shared (like there is no cars on the river side and no river danger in the road sides) we need to have more general features. I consider the overall danger that the frog might encounter in each direction as feature. For instance, if it goes left, how the surrounding is (and not will be as we cannot have look ahead! The frog can only see its current state and not the next stat. e of the environment)

For the danger, I take into account if there is any possible danger, such as if there is a car coming toward the frog, if there is any river (which the frog can fall in and die), and if there is any crocodile in the home place. Over time, the frog learns how each of this element affects its life and the final reward it gets. After noticeable amount of training, it learns when there is a car, or there is a river spot, or maybe a crocodile, it shouldn’t go toward them and it is better to stay away from them. And the agent learns them only based on q-learning.

The trained weights for my feature set are saved in a separate files.

When running the program, it is obvious that how the frog is doing bad at frist but when being killed by the car several times it learn that it is better to be away from the car, so when a car getting closer, the frog knows it should go go somewhere safe. It sometimes goes down and sometimes up or left and write in this case, based on what the value of doing each action has. For instance, if there is a car coming from right side and there are some cars getting close to the above, the frog knows that it should go down to a safer place. There are all learnt by the frog after being in each step and experiencing the consequences.

As for the concept of exploration, to let the agent act randomly sometimes to prevent its conservative behavior and to deal with its fear of death, I forced it to act randomly in 5 percent of the time, with equal random distribution of each direction.

Over this project, I learned that when the environemt is dynamic for an agent, we should come up with more abstract and generic features, features that can be applied in different part of the area. Otherwise the model can not be representative of the space at all.

I need to mention that there is no look ahead in my approach as the frog only uses the current state (at the time t) to get the feature of the surrounding and over times learn how each feature might affect the final reward. Also, I didn’t add any additional reward and trick so my agent, the frog, only learns based on the concept of the Q learning. For instance, I didn’t encourage the frog to go up and I let the frog itself to explore the area based on these 5 actions and learn how to act in next run only and merely based on what the frog receives as its reward and the value of taking each action and ending being in the next state. It only acts based on learning how to act when going to a similar pattern of state.

As for the efficacy, I consider time and also the number of time it takes for my agent to learn how to act in each situation. Of course, based on the value of rate of learning these factors wary. I count the iteration times it takes for the frog to win. The learning rate is slow for this agent as I don’t consider any memory for that and it merely learns based on learning the features and their weights. Besides, since there are so many chances of dyeing in the river side, and I only consider danger as feature, the frog cannot make good decision at first. And it takes a lot of time for the frog to learn. In fact, it has a tendancy to stay alie than to risk going to new area and dies. But hopefully, after good amount of training, it learns how to get itself to the safe house, as I could get 9 frogs home at the end. Of course, if we wait linger we can see the learning improves further and frog can find its way home faster over time.

HOW to run the code:

If you just type python randomAgentFinall.py the program statrs with random weight

but if you typed " randomAgentFinall.py weights1.pkl" or any other weight file name (among these 5 that I uploaded), the program will starts by predefined weights and not randomized ones.

The first weight file belongs to when the first frog wins the second one has been saved when the second one won the third one is saved when the fourth frog won and the last one has been saved when the fifth frog got home.