# COMP3200: Individual Project
# Reinforcement Learning

Daniel Lee
dl2g14@ecs.soton.ac.uk

November 23, 2016

## Motivation

The purpose of this paper is to document every relevant work that I do during my background research.

## 1 Bandit Problems

In the classic bandit problem, the agent is given a limited set of actions to choose from at every time-step. Each action provides the agent with its respective stochastic reward from a particular distribution. The objective is to collect as much total reward, or *return*, from the limited amount of available time-steps. In this section, the popular greedy and $\epsilon$-greedy algorithms will be implemented to tackle this task and the resulting performances will be compared.

### 1.1 Greedy

#### 1.1.1 Theory

As mentioned above, the agent can choose an action from a limited range of options. The greedy algorithm leads the agent to choose the action that has the highest *action-value* at every single time-step. This can be numerically represented in this way:

$$Q_t(A_t^*) = max_a Q_t(a) \tag{1}$$

The action-value of an action can be computed from the following equation (2):

$$Q_t(a) = \frac{R_1 + R_2 + ... + R_{K_a}}{K_a} \tag{2}$$

The equation states that the action-value of an action at a particular time-step is determined by the total reward accumulated, divided by the number times that it was selected in the past. In this simulation, the probability distribution that was used for the immediate actions' rewards was a Gaussian distribution $\mathcal{N}(0,1)$. Due to the nature of this distribution, if an action was selected infinite times, its action-value would eventually converge to its *actual* value, $q_*$.

#### 1.1.2 Simulation

To test this algorithm, the *actual* values for each of the actions were set by selecting numbers from a $\mathcal{N}(0,1)$ distribution. Following this, the action-values were initialized by adding Gaussian noise of the form $\mathcal{N}(0,1)$. To test the performance of the greedy algorithm, the simulation was first run for 1000 steps from a bandit problem with 10 possible actions

### 1.2 $\epsilon$-Greedy

Once the convolution algorithm was tested with several arbitrarily chosen kernels of different sizes and weights, a kernel with a Gaussian distribution was created.
The kernel was created using the method `createKernelImage` with the parameters created between Lines 2 and 4. These parameters are the size of the wanted square kernel and the the standard deviation of the weights. These parameters are directly related to each other, as shown in Line 2. Moreover, it can be seen that in Line 3 there is a manipulation that avoids assigning an even number to the `size` variable.

## 2 Chapter 3

### 2.1 Low-Pass Filter

High frequencies of an image can be removed by applying convolution using a Gaussian Kernel with the appropriate standard deviation. This process acts as a low pass filter, where the standard deviation is inversely proportional to the filter's threshold. This means that a high standard deviation will drag the filter's threshold down so that only a fewer low frequency signals can pass. Therefore, by manipulating the size and standard deviation the image's blur can be regulated.

An example of this is shown in the image below using a `sigma` value of 7.
It can also be observed that the black padding is relatively big. This is because the kernel's size increases as a product of the standard deviation. The resulting image can be visualized with more easiness as the distance from our eyes to the image increases. The higher the blur, the more distance our eyes need to see the image with clarity. This effect is the basis of the concept of hybrid images.

### 2.2 High-Pass Filter

The other half of hybrid images consist of images containing high frequencies. This can be easily generated by subtracting each pixel from the original image with the corresponding pixel from the low-pass filtered image and adding 0.5, so as to move the mean from 0 to 0.5. The resulting image from applying this high pass filter to the cat image using a Gaussian kernel with `sigma` value of 5, is displayed below.
From observing the image above, we can see the small details of the image whereas in Figure 2, we see very little. This is the expected result from applying these filters. High frequencies illustrate the details and sudden changes in the pixels of the image whereas the low frequencies focus on showing the general form of the image.

### 2.3 Combining Low and High frequency images

By superimposing images filtered by a high and a low pass filter, we can generate hybrid images. Hybrid images makes use of the properties of high and low pass filtered images, where one is more visible to the human eyes in a short distance and the other is more visible from a large distance, respectively.
The first hybrid image that was generated and tested was the dog and cat image pair. The `sigma` values for these Guassian Kernels were selected with the objective of making the high pass image visible from the average distance from the eyes of a user and a laptop screen, and the low pass filter visible when the image was 4 times as small as the original image. With this objective in mind, the arbitrary `sigma` values chosen for the low pass and high pass filter kernels were 7 and 5, respectively. The resulting hybrid image can be seen below.
For this hybrid image, the dog image was passed through the low-pass filter and the cat image through the high-pass filter. Therefore, at first sight, the image seems to be showing a cat. As the distance from the image and the eyes increases, the image starts becoming to look similar to a dog. This effect it shown in Figure 5.
As stated above, Figure 5 shows that the image that is 4 times as small as the original image (second image from right) shows the low-pass filtered image, in this case, the image of the dog.
By using the same proportion constant, $k$, between the image area and `sigma` values from the dog and cat images, the `sigma` values for the low pass and high pass filters' kernels were calculated for the rest of the pair of images. The proportion constant was calculated in the following form:

## 3 Results

In this section, the resulting hybrid images and variations of its size, using values of `sigmas` calculated from the constant, $k$, for the rest of the image pairs are displayed.

## 4 Conclusion

From our result, we can see that some pairs have a more convincing effect that others. This is because there are many factors that are involved in hybrid images, such as, image size, Guassian Kernel distribution, image positioning and the general human vision uncertainty for each different person. Moreover, because the proportional constant, $k$, calculated at the beginning, was computed by choosing arbitrary values of `sigmas` for the dog and cat images' `sigma`, based on my personal judgment, this might have led to a non-optimal implementation.