AA 274: Principles of Robotic Autonomy | Winter 2018

Lecture 11: Localization part 2 - Parametric and Non-Parametric Filters

Scribes: Steve Burke, Claire Huang, Emma Morgan, Eley Ng, Stephanie Schneider, Gareth Weiss

11.1 Introduction

Successful robot navigation incorporates four fundamental building blocks:

- Perception: extract data from sensors
- Localization: build a map and determine position within the environment
- Cognition: decide how to move towards end goal
- Motion control: actuate motors to move along chosen path

Lecture 11 completes the description of localization strategies. In Lecture 10, Bayes' Filter Algorithm was described in its general, incalculable form. Here, we introduce some tractable solutions used for real-world localization.

These solutions can be grouped into two methods: parametric and non-parametric. The parametric class of filters forces a predefined structure onto the problem. Non-parametric filters, on the other hand, make no assumption about structure, and therefore must introduce approximations to make the algorithm computationally feasible.

11.2 Parametric Filters

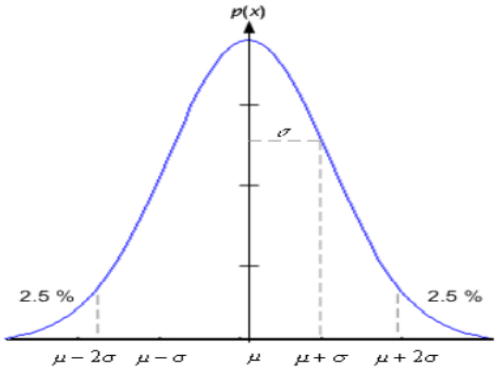
11.2.1 Gaussian Distribution

In a Gaussian belief state, the belief is represented as a mutlivariate normal distribution, given by Equation 11.1

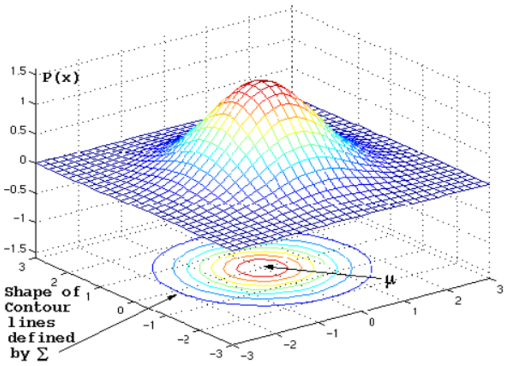
$$p(x) = det(2\pi\Sigma)^{-1/2} exp\Big(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\Big) \rightarrow \mathcal{N}(\mu,\Sigma) \tag{11.1}$$

The Gaussian distribution has a parametric density, meaning it is parameterized entirely by μ and Σ. Gausian distributions are also unimodal and symmetric about the mean. In the univariate case, the mean and variance are both scalar quantities; in the multidimensional generalization, the mean (μ) is a vector of means, and the covariance (Σ) is a matrix. The covariance matrix tells us about the uncertainty of each random variable along every dimension. A multivariance Gaussian distribution can be visualized as a contour plot (as in Figure 11.1b), where density is constant along a given contour.

Non-symmetric variance (e.g. a distribution with elliptical contours) can be explained by considering the example of a car. A self-driving car, which moves primarily in the forward/backward direction, will have more positional uncertainty in the longitudinal direction than laterally.



(a) Univariate Gaussian



(b) Multivariate Gaussian

11.2.1.1 Gaussian Properties

- A linear transformation of a Gaussian random variable is a Gaussian random variable (see [2] for derivation)

$$Y = AX + b \sim \mathcal{N}\big(A\mu + b, A\Sigma A^T\big) \tag{11.2}$$

- The sum of two independent Gaussian random variables is Gaussian

$$X_1 + X_2 \sim \mathcal{N}(\mu_1 + \mu_2, \Sigma_1 + \Sigma_1) \tag{11.3}$$

- The product of two independent Gaussian pdfs is Gaussian

$$X_1 X_2 \sim \mathcal{N}\Big(\Sigma_2(\Sigma_1 + \Sigma_2)^{-1}\mu_1 + \Sigma_1(\Sigma_1 + \Sigma_2)^{-1}\mu_2,\ \Sigma_1(\Sigma_1 + \Sigma_2)^{-1}\Sigma_2\Big) \tag{11.4}$$

11.2.2 Kalman Filters

11.2.2.1 Assumptions

1. Linear Dynamics

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$

where process noise is Gaussian ($\epsilon_t \sim \mathcal{N}(0, R_t)$). Considering x_{t-1} and u_t as givens (i.e. constants), the state transition probability ($p(x_t|u_t, x_{t-1})$) is a linear transformation of only the Gaussian noise. Therefore, the posterior is also Gaussian, with mean = $A_t x_{t-1} + B_t u_t$, and variance equal to variance of process noise = R_t.

2. Linear Measurement Model

$$z_t = C_t x_t + \delta_t$$

where process noise is Gaussian ($\delta_t \sim \mathcal{N}(0, Q_t)$). The posterior $p(z_t|x_t)$ in this case is also a linear transformation of the Gaussian process noise, with mean = $C_t x_t$, and variance = Q_t.

3. Initial Belief is Gaussian

$$bel(x_0) = det(2\pi\Sigma_0)^{-\frac{1}{2}} exp\Big(-\frac{1}{2}(x_0 - \mu_0)^T\Sigma_0^{-1}(x_0 - \mu_0)\Big)$$

By enforcing these three assumptions, and using the properties in Section 11.2.1.1, we can ensure that the posterior belief $bel(x_t)$ will always be Gaussian as well. We often use Gaussian assumptions even if they are invalid, in order to parameterize the problem and make it computationally feasible. This forces a unimodal belief, but is often a good approximation even for non-Gaussian data.

11.2.2.2 Algorithm

The Kalman filter algorithm follows the Bayes filter structure while dramatically simplifying computation by using Gaussians. It consists of two steps: prediction and correction. At some time t, we know the previous belief, defined by the two parameters μ_{t-1} and Σ_{t-1}, the current control action u_t, and the measurement z_t. We want to update our belief and determine the current state μ_t and Σ_t. In the prediction step, we update our prediction of belief at time t using a linear dynamics model.

$$\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t \tag{11.5}$$

$$\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \tag{11.6}$$

Once we have the predicted belief, we incorporate our current measurement to refine our belief in the correction step. We define an additional Kalman gain K_t that weights our trust in the dynamic model versus trust in the measurement.

$$K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1} \tag{11.7}$$

By this definition, K_t grows inversely with our measurement uncertainty (the terms in the denominator) – greater uncertainty means a smaller K_t. We then use the gain to calculate our final belief parameters:

$$\mu_t = \overline{\mu}_t + K_t (z_t - C_t \overline{\mu}_t) \tag{11.8}$$

$$\Sigma_t = (I - K_t C_t) \overline{\Sigma_t} \tag{11.9}$$

11.2.3 Extended Kalman Filter

11.2.3.1 Linearity Assumption

The most important property of the linearity assumption is that the linear transformation of a Gaussian random variable results in another Gaussian RV. The standard Kalman Filter algorithm relies on a linear transformation to propagate the state model forward, but unfortunately, this assumption of linear dynamics is severely restrictive for robotics applications.

11.2.3.2 EKF Set-up

The goal of the Extended KF is to relax the linearity assumptions, therefore making it more useful for robotics applications. For the EKF, the dynamics and measurement models are given by:

$$x_t = g(u_t, x_{t-1}) + \epsilon_t \tag{11.10}$$

$$z_t = h(x_t) + \delta_t \tag{11.11}$$

The key idea is to use these equations to focus on efficiently computing a Gaussian approximation of the dynamics model, rather than finding an exact posterior. We do this by linearly approximating g

and h around the most likely state. By using Jacobians (G_t and H_t), we can take advantage of the non-linear generalizations of both the dynamics and the measurement models. This results in the following equations:

$$p(x_t|u_t, x_{t-1}) =$$
$$\det(2\pi R_t)^{-1/2} \exp((-1/2)[x_t - g(u_t, \mu_{t-1}) - G_t(x_t - \mu_{t-1})]^T R_t^{-1}[x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})]) \quad (11.12)$$
$$p(z_t|x_t) = \det(2\pi Q_t)^{-1} \exp((-1/2)[z_t - h(\overline{\mu}_t) - H_t(x_t - \overline{\mu}_t)]Q_t^{-1}[z_t - h(\overline{\mu}_t) - H_t(x_t - \overline{\mu}_t)]) \quad (11.13)$$

11.2.3.3 Algorithm

The main difference between the Kalman Filter and EKF is that the EKF uses non-linear generalizations, while the regular Kalman Filter models the dynamics using linear predictions. Mathematically, the EKF uses Jacobians instead of linear system matrices. Both algorithms follow the same high-level derivation, and both use the Bayesian structure of having a prediction step followed by a correction step.

The inputs for the EKF are:

- μ_{t-1}: expected value of the state at time $t-1$
- Σ_{t-1}: variance of the estimate at $t-1$
- z_t: measurement taken at time t

The EKF outputs a Gaussian approximation for the posterior belief, parameterized by μ_t and Σ_t.

The prediction step involves propagating forward the expected value, assuming no noise and using the non-linear motion model. Then, the covariance is propagated with the linearized motion model. Using the Jacobians, the prediction is then corrected to a new value: our output. This is repeated at each new state. Below is the algorithm, with the differences from the standard Kalman filter highlighted in red.

Prediction:

$$\begin{aligned} \overline{\mu}_t &= g(u_t, \mu_t - 1) \\ \overline{\Sigma}_t &= G_t \Sigma_t - 1 {G_t}^T + R_t \end{aligned}$$

Correction:

$$\begin{aligned} K_t &= \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1} \\ \mu_t &= \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t)) \\ \Sigma_t &= (I - K_t H_t)\overline{\Sigma}_t \end{aligned}$$

11.2.4 Unscented Kalman Filter

11.2.4.1 UKF Overview

Another way to utilize Gaussian approximations, but without linearization, is the Unscented Kalman Filter. While the EKF propagates an entire Gaussian using a approximated motion and measurement models, the UKF approximates the current belief by selecting sigma-points, then uses the full non-linear model to propagate those points forward. This tends to mean that the UKF is a bit slower due to the need to process individual particles. However, it is more accurate and more robust in high variance situations.

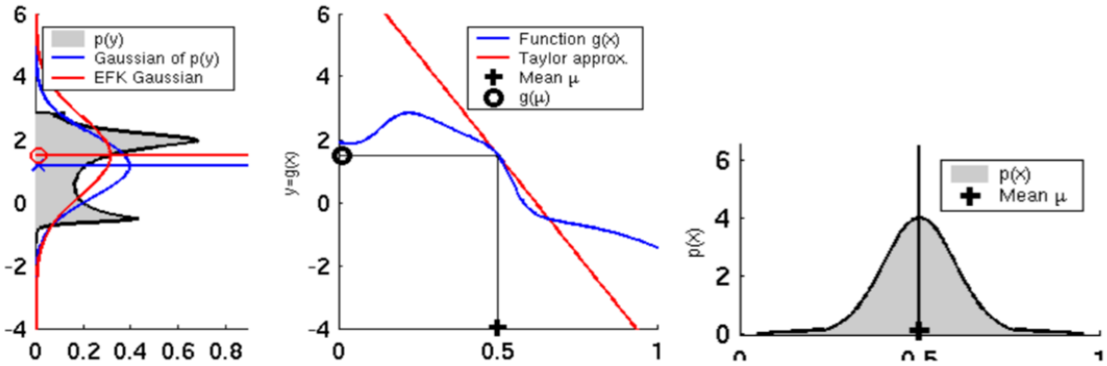


Figure 11.2: EKF Examples

11.2.4.2 UKF Set-up

To implement the UKF, we first have to develop equations to compute relevant sigma points, find corresponding weights for those sigma points, and compute Gaussian parameters from a set of transformed points. Sigma points should be chosen symmetric relative to the eigenaxis of the uncertainty ellipse; the set of sigma points should contain at least the mean, as well as points some displacement along the axis. The equations for determining sigma points ($X^{[i]}$) and weights ($w_m^{[i]}, w_c^{[i]}$) are described below. Assume n-dimensional Gaussian $N(\mu, \Sigma)$

Sigma Points:

$$X^{[0]} = \mu \tag{11.14}$$

$$X^{[i]} = \mu + (\sqrt{(n+\lambda)\Sigma})_i \quad for \quad i = 1, ..., n \tag{11.15}$$

$$X^{[i]} = \mu - (\sqrt{(n+\lambda)\Sigma})_{i-n} \quad for \quad i = n+1, ..., 2n \tag{11.16}$$

Weights:

$$\omega_m^{[0]} = \frac{\lambda}{n+\lambda} \tag{11.17}$$

$$\omega_c^{[0]} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 +) \tag{11.18}$$

$$\omega_m^{[i]} = \omega_c^{[i]} = \frac{1}{2(n+\lambda)} \quad for \quad i = 1, ..., 2n \tag{11.19}$$

Once the new points have been computed (see section 11.2.5), it is necessary to fit the new points to a Gaussian by recovering the mean and covariance:

$$\mu' = \sum_{i=0}^{2n} \omega_m^{[i]} Y^{[i]} \tag{11.20}$$

$$\Sigma' = \sum_{i=0}^{2n} \omega_c^{[i]} (Y^{[i]} - \mu')(Y^{[i]} - \mu')^T \tag{11.21}$$

11.2.5 Algorithm

The UKF instantiates the Bayes filter with a prediction step and a correction step. The prediction step takes original sigma points X_{t-1} and propagates them forward with the full dynamics model $Y^{[i]} = g(X^{[i]})$ to get resulting points X_t^*. These points are then fit to a Gaussian by recovering the mean and covariance.

$$X_{t-1} = (\mu_{t-1}, \mu_{t-1} + \gamma\sqrt{\Sigma_{t-1}}, \mu_{t-1} - \gamma\sqrt{\Sigma_{t-1}}$$

$$X_t^* = g(\mu_t, X_{t-1})$$

$$\overline{\mu_t} = \sum_{i=0}^{2n} \omega_m^{[i]} \overline{X_t^{*[i]}}$$

$$\overline{\Sigma_t} = \sum_{i=0}^{2n} \omega_c^{[i]} (\overline{X_t^{*[i]}} - \overline{\mu_t})(\overline{X_t^{*[i]}} - \overline{\mu_t})^T + R_t$$

The correction step accounts for measurements by approximating the predicted belief X_t with sigma points. Then predict the measurements Z_t for the sigma points using the nonlinear measurement model. Finally, reconstruct the Gaussian.

$$\overline{X_t} = \overline{\mu_t}, \overline{\mu_t} + \gamma\sqrt{\overline{\Sigma_t}}, \overline{\mu_t} - \gamma\sqrt{\overline{\Sigma_t}}$$

$$\overline{Z_t} = h(\overline{X_t})$$

$$\hat{z}_t = \sum_{i=0}^{2n} \omega_m^{[i]} \overline{Z_t^{[i]}}$$

$$S_t = \sum_{i=0}^{2n} \omega_c^{[i]} (\overline{Z_t^{[i]}} - \hat{z}_t)(\overline{Z_t^{[i]}} - \hat{z}_t)^T + Q_t$$

$$\overline{\Sigma_t^{x,z}} = \sum_{i=0}^{2n} \omega_c^{[i]} (\overline{X_t^{[i]}} - \mu_t)(\overline{Z_t^{[i]}} - \hat{z}_t)^T$$

$$K_t = \overline{\Sigma_t^{x,z}} S_t^{-1}$$

$$\mu_t = \overline{\mu_t} + K_t(z_t - \hat{z}_t)$$

$$\Sigma_t = \overline{\Sigma_t} - K_t S_t K_t^T$$

11.3 Non-Parametric Filters

Non-Parametric Filters do not force assumptions upon what the belief should look like (e.g. Gaussian). This allows for more flexible belief spaces, such as multi-modal. However, it requires a certain amount of approximation to provide a computational simplification of the Bayes filter algorithm.

11.3.1 Histogram Filters (Discrete Bayes Filter)

Histogram filters use a discrete approximations of a continuous distributions, i.e. considering pmfs or histograms as opposed to pdfs.

Before the discrete Bayes' filter can be executed, several discretizations must be done. First, a (continuous) state space must be discretized into a finite number of bins/cells. Each state, $x_{k,t}$, is then assigned a probability, $p_{k,t}$, which represents the probability of being in a particular cell in the state space. For all $x_t \in x_{k,t}$,

$$p(x_t) \equiv \frac{p_{k,t}}{|x_{k,t}|} \tag{11.22}$$

so these probabilities are assumed to be uniform.
Next, the motion and measurement models must also be discretized without taking into account the exact state we are in. To do this, we choose a *mean state* $\hat{x}_{k,t}$ (which can be thought of as the central mass of the cell) as a representative state:

$$\hat{x}_{k,t} = |x_{k,t}|^{-1} \int_{x_{k,t}} x_t dx_t \tag{11.23}$$

And approximate the measurement and transition models using the mean state:

$$p(z_t|x_{k,t}) \approx p(z_t|\hat{x}_{k,t}) \tag{11.24}$$

$$p(x_{k,t}|u_t, x_{k,t-1}) \approx \eta|\hat{x}_{k,t}|p(\hat{x}_{k,t}|\hat{u}_t, \hat{x}_{k,t-1}) \tag{11.25}$$

The algorithm for the discrete Bayes' filter is essentially the same as the continuous Bayes' filter. The only difference is replacing the integrals with summations for each cell.

11.3.2 Particle Filters

The particle filter is another non-parametric implementation of Bayes' filter. Similar to the histogram filter, it approximates the posterior belief with a finite number of discrete parameters. In this case, a number of particles represents the posterior belief. This filter allows for both the use of nonlinear transformations and also the representation of non-Gaussian distributions, including multi-modal beliefs.

For this filter, samples of particles (represented by vertical bars in Figure 11.3) represent the belief of state, where a higher density of particles corresponds to a higher belief that the state lies in a certain region.

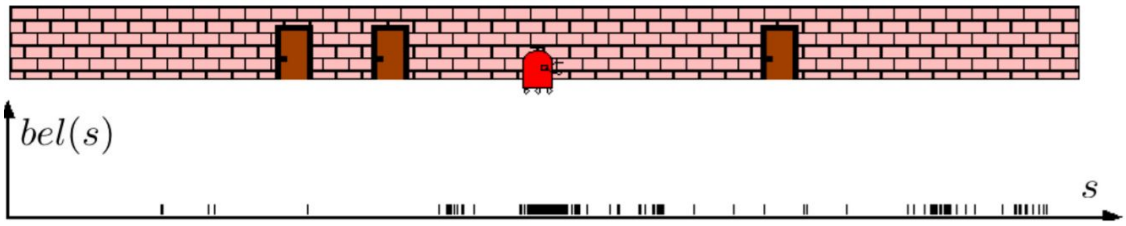


Figure 11.3: Particle Representation of Belief [2]

The belief is represented by a set of M particles:

$$\chi_t := x_t^{[1]},\ x_t^{[2]},\ \ldots\ ,\ x_t^{[M]} \tag{11.26}$$

Each particle represents a hypothesis about what the true state could be. Ideally the particles are distributed according to the probability of the current state, given all measurements and control inputs up to that time:

$$x_t^{[m]} \sim p(x_t \mid z_{1:t},\ u_{1:t}) = bel(x_t) \tag{11.27}$$

Maintaining such a series of particles for each time step allows us to fit a probability distribution across the particle samples to understand the belief of state. The particle filter algorithm recursively constructs a particle set at the current time from a particle set at the previous time step, with the goal that the new set matches the target posterior distribution. In order to do so, the algorithm exploits principles from Bayesian filtering.

11.3.2.1 Algorithm

The algorithm inputs are the particle representation of belief at the previous time step as well as the control and measurement of the current time step. The algorithm outputs another set of particles representing the posterior belief. This algorithm can be broken down into two key components:

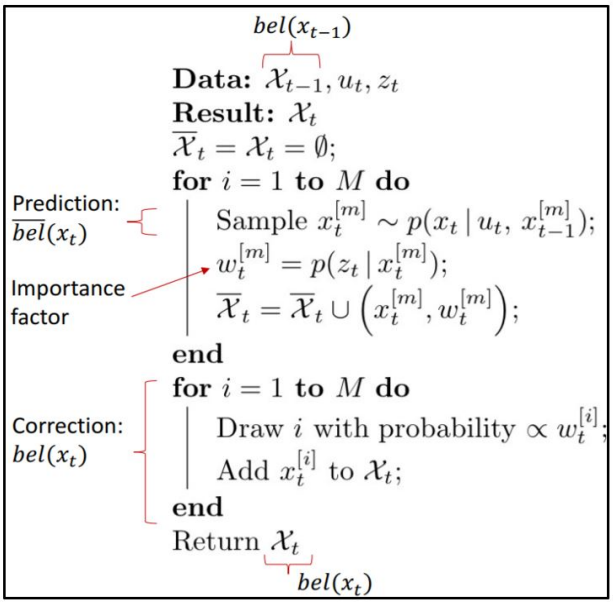


Figure 11.4: Particle Filter Algorithm [2]

1. Prediction Step
    - The algorithm first uses the set of particles representing the prior belief along with the control input and state transition model to sample a new set of particles that represent the new belief for time t.

- For each new particle that is sampled, a weight is computed (referred to as an importance factor), which represents the probability of that sampled particle given the observed measurement.
- The set of sampled particles with corresponding importance factors represents the prediction belief and is passed on to the next part of the algorithm.

2. Correction Step

   - The correction step draws particles from the prediction set, with probability matching the importance factor of each particle.
   - This effectively ensures that particles in the prediction set that are deemed implausible by the current measurement are unlikely to make it into the final belief set, as these particles do not represent legitimate hypotheses for the state. This enforces that the final belief distribution complies with the observed measurement.

After the prediction and correction resampling, if the number of total particles M approaches a very large number, the belief can be represented by the below equation. Typically, a value of roughly 1000 samples can adequately represent the posterior belief.

$$bel(x_t) = \eta p(z_t \mid x_t^{[m]}) \overline{bel}(x_t) \tag{11.28}$$

The particle filter has certain computational benefits because of its process of using a subset of samples to represent a probability distribution. But there is a potential risk that the incomplete sampling of the belief distribution could lead to a robot losing an accurate representation of its true state, especially in the case where the robot receives accurate but unlikely measurement data [1].

11.4 Localization Overview

In order to select an appropriate filtering algorithm, we must consider the specifics of the problem. For example, position tracking presents as a problem in which initial pose is known (i.e. there is little variance around the mean), so can be well-represented with one of the parametric filters discussed in Section 11.2. Conversely, in a global localization problem, the initial state is unknown; the initial belief may be multimodal, and therefore these problems are better represented by one of the non-parametric filters in 11.3. One example of global localization is called the "kidnapped robot" problem, where the robot is effectively teleported to another (unknown) location. This can be a helpful model for when our estimated state diverges from true state, and we need to reset the filter. There are several other characteristics to consider when choosing a filtering algorithm:

- Static environment (robot position is the only changing variable) vs.
  Dynamic environment (other moving objects exist in the environment)
- Passive localization (robot motion aims towards a goal unrelated to localization) vs.
  Active localization (actions explicitly made to reduce uncertainty in localization)
- Single-robot localization (individual robot only) vs.
  Multi-robot localization (team of robots engaged in cooperative localization)

Robot localization involves integrating data over time in order to determine the robot's position relative to a given map. In Lecture 10, we learned how to represent localization problems using the Markovian

state assumption; we also learned about the Bayes' Filter Algorithm, which consists of a prediction step, followed by a taking a measurement and performing a correction update. The Bayes' Filter provides a general, but unsolvable, algorithm for localization. In Lecture 11, we introduced several different assumptions to make the filter tractable. The suitability of these assumptions relies on the specifications of the problem.

References

[1] R. Siegwart, I. R. Nourbakhsh, D. Scaramuzza. *Introduction to Autonomous Mobile Robots.* MIT Press, 2nd Edition, 2011.

[2] Pavone, M. (2018). AA274: Principles of Robotic Autonomy, lecture 11 notes [PowerPoint Slides]. Retrieved from https://canvas.stanford.edu/courses/75589/files/folder/Lectures

[3] Chapter 4 Bayesian Decision Theory. www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm