CSC498 - Topics in CS: Robotics Perception
Assignment 3
Due: March 22, 2022, at 4 PM

1 Pen and Paper (50 Marks)

Submission: Please upload a PDF of solutions – either TeX or hand-written and scanned. Please ensure non-TeX solutions are legible or you may lose marks.

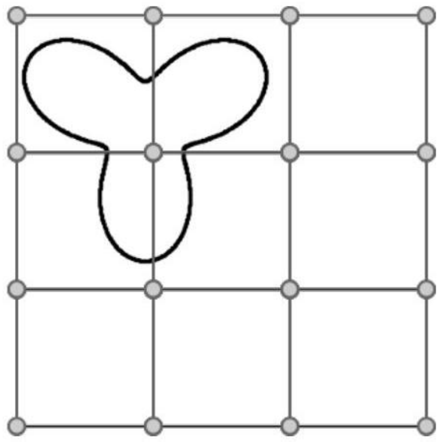
1.1 2D Occupancy Network (20 Marks)

(a) Derive the decision boundary for a linear layer $L_{\mathbf{W},b}$ with weight $\mathbf{W} = (w_1, w_2)$ and bias b, i.e. where $L_{\mathbf{W},b}(\mathbf{x}) = 0$

(b) Draw the decision boundary for: i) $\mathbf{W} = (-3, 1), b = 2$, ii) $\mathbf{W} = (2, 4), b = -2$

(c) Now consider a 2-layer fully-connected ReLU-network with weights $\mathbf{W}_1 = \begin{pmatrix} -5 & 1 \\ 2 & 3 \end{pmatrix}$, $\mathbf{b}_1 = \begin{pmatrix} 3 \\ -2 \end{pmatrix}$, and $\mathbf{W}_2 = (2, 3), b_2 = -2$. Let $\sigma(\mathbf{x}) = \begin{pmatrix} \max(0, x_1) \\ \max(0, x_2) \end{pmatrix}$ denote the ReLU activation function. Derive and draw the decision boundary for this network. You can use an online tool like geogebra [link] to plot the decision boundary.

(d) So far, we have learned how to extract the decision boundary from a trained network. Next, we want to extract a mesh from this implicitly defined boundary. The figure below shows the decision boundary for some classifier. Apply the MISE algorithm manually by coloring occupied points in red and free points in blue and refining the grid where needed. See Lecture 9 notes for an example. Refine the grid for 3 steps. Lastly, manually execute the marching squares algorithm, the 2D variant of the marching cubes algorithm, and draw the extracted mesh into the last refined grid. Assume that the classifier returns either 0 (free space) or 1 (occupied).



1.2 Differentiable Volumetric Rendering (30 Marks)

Let the occupancy network f_{θ_f} in DVR be parametrized by a single linear layer, i.e. $f_{\theta_f} = L_{\mathbf{W}}, b_f$. Further, let the texture field t_{θ_t} be defined by a linear layer with a subsequent sigmoid function σ, i.e. $t_{\theta_t} = \sigma\left(L_{\mathbf{W}_t, b_t}\right)$. Similarly to the previous exercise, we choose the weights: $\mathbf{W}_f = (-4, 1), b_f = 3$, and $\mathbf{W}_t = (2, 3), b_t = -2$. Note that in this case the texture field only predicts a single scalar that corresponds to the intensity of the pixel to simplify computations.

(a) We shoot a ray from the origin in an angle of 45°, i.e. $\mathbf{r} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \hat{d} \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$. Estimate the surface intersection point $\hat{\mathbf{p}}$ by performing a single update step with the secant method starting in the interval $\left[\hat{d}_1 = 1, \hat{d}_2 = 2\right]$.

(b) Compute the predicted texture field, i.e. intensity, at $\hat{\mathbf{p}}$.

(c) As loss $\mathcal{L}$ we choose the L2-distance between the predicted intensity $\hat{I}$ and the ground truth value I. Let us assume that $I = 0.6$. The gradient wrt. all parameters $\theta = \theta_f \cup \theta_t$ is

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta} &= \frac{\partial \mathcal{L}}{\partial \hat{I}} \frac{\partial \hat{I}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \hat{I}} \frac{d t_{\theta_t}(\hat{\mathbf{p}})}{d\theta} \\ &= \frac{\partial \mathcal{L}}{\partial \hat{I}} \left[\frac{\partial t_{\theta_t}(\hat{\mathbf{p}})}{\partial \theta} + \frac{\partial t_{\theta_t}(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \frac{\partial \hat{\mathbf{p}}}{\partial \theta} \right] \end{aligned}$$

Therefore the gradients wrt. the parameters f_{θ_f}, i.e. $\mathbf{W}_f$ and b_f are given by:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{W}_f} &= \frac{\partial \mathcal{L}}{\partial \hat{I}} \left[\frac{\partial t_{\theta_t}(\hat{\mathbf{p}})}{\partial \mathbf{W}_f} + \frac{\partial t_{\theta_t}(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \frac{\partial \hat{\mathbf{p}}}{\partial \mathbf{W}_f} \right] \\ \frac{\partial \mathcal{L}}{\partial b_f} &= \frac{\partial \mathcal{L}}{\partial \hat{I}} \left[\frac{\partial t_{\theta_t}(\hat{\mathbf{p}})}{\partial b_f} + \frac{\partial t_{\theta_t}(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}} \frac{\partial \hat{\mathbf{p}}}{\partial b_f} \right] \end{aligned}$$

Start by computing the following components of Eq. 11 , (2) $\frac{\partial \mathcal{L}}{\partial \hat{I}}, \frac{\partial t_{\theta_t}(\hat{\mathbf{p}})}{\partial \mathbf{W}_f}, \frac{\partial t_{\theta_t}(\hat{\mathbf{p}})}{\partial b_f}, \frac{\partial t_{\theta_t}(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}}$.

(d) To compute the gradients for $\mathbf{W}_f$ and b_f the only components missing for Eq. 1 , 2 are $\frac{\partial \hat{\mathbf{p}}}{\partial \mathbf{W}_f}$ and $\frac{\partial \hat{\mathbf{p}}}{\partial b_f}$, respectively. However, we cannot directly compute them because $\hat{\mathbf{p}}$ is defined only implicitly. Use implicit differentiation on $f_{\theta_f} = \tau$, where $\tau = 0$, to obtain $\frac{\partial \hat{\mathbf{p}}}{\partial \mathbf{W}_f}, \frac{\partial \hat{\mathbf{p}}}{\partial b_f}$. With these results, compute the gradients for $\mathbf{W}_f$ and b_f.

Hint: Write $\hat{\mathbf{p}}$ as the ray from the origin $\hat{\mathbf{p}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \hat{d} \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$ to link it to the depth prediction $\hat{d}$. Since the depth is predicted by f_{θ_f}, this will allow you to formulate $\frac{\partial \hat{\mathbf{p}}}{\partial \mathbf{W}_f}, \frac{\partial \hat{\mathbf{p}}}{\partial b_f}$ in terms of $\frac{\partial f_{\theta_f}(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}}$ and $\frac{\partial f_{\theta_f}(\hat{\mathbf{p}})}{\partial \mathbf{W}_f}$ and $\frac{\partial f_{\theta_f}(\hat{\mathbf{p}})}{\partial b_f}$, respectively. For computing the result this means you also need to compute $\frac{\partial f_{\theta_f}(\hat{\mathbf{p}})}{\partial \hat{\mathbf{p}}}, \frac{\partial f_{\theta_f}(\hat{\mathbf{p}})}{\partial \mathbf{W}_f}$ and $\frac{\partial f_{\theta_f}(\hat{\mathbf{p}})}{\partial b_f}$.

2 Coding Exercises (50 Marks)

The following exercises will be coding exercises. They are all contained in the colab notebook [link]. The notebook itself is self-contained but you can also use this document as guidance.

Submission: Submit your code and results in an IPython notebook (.ipynb) with all the **images and video outputs embedded** in it. You will lose marks if your notebook works but doesn't contain outputs.

2.1 Occupancy Network

In this exercise, you will train an occupancy network to represent a single 3D object. More specifically, you will start from the point cloud of a chair together with its occupancy values, see Fig. 1. Then, you will train an occupancy network to encode the shape and reconstruct the mesh from the trained network.

(a) Prepare your data by implementing the function get_train_val_split.

(b) Complete the class OccNet such that you obtain a network with 4 hidden layers of hidden dimension 64. The network takes a batch of 3D locations as input and predicts a single occupancy value for each point.

(c) Complete the training loop, i.e. the function train_model. Do the curves for training and validation loss look good? Or is there under / overfitting?

(d) Load the trained model and predict the occupancy values for all points on a 3D grid. You can use the test_loader to load the grid points in batches.

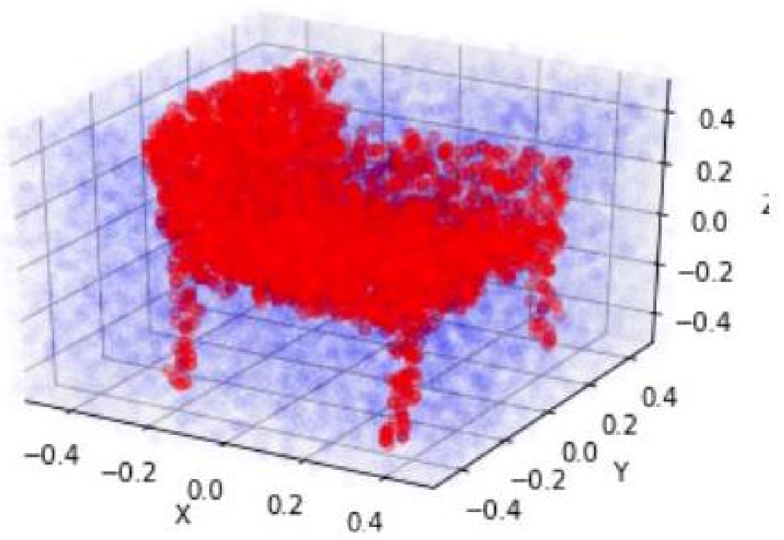


Figure 1: 3D Point Cloud of a Chair. Red points belong to the object while blue points represent free space.