Generative ML Notes

1 Neural Networks

Neural Networks (NNs) consist of *layers* which contain units which are connected to the units in the previous layer. The connections are assoicated with multiplicative weights. At each unit an *actiation function* is applied to the weighted sum of the inputs.

A *Dense* layer is a layer where all its units are connected to all of the units in the previous layer.

A *Multilayer Percepton* (MLP) is a NN where all the layers are Dense.

There are three key activation functions. *ReLU*, *LeakyReLU*, and *sigmoid*. ReLU stands for rectified linear unit. These activation functions are defined below. Note $\alpha > 0$.

$$\text{ReLU}(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}$$

$$\text{LeakyReLU}(x) = \begin{cases} \alpha x & x < 0 \\ x & x \geq 0 \end{cases}$$

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

ReLU is the standard activation function for neural networks. They are good because they:

- are easy to evaluate and to find the gradient of for a given input,
- introduce non-linearity which allows for more complex patterns to be learned, and
- don't suffer from the vanishing gradient problem (sigmoid/tanh functions can result in gradients that dimish rapidly as the number of layers increase).

One issue with ReLUs is they can suffer from the vanishing gradient problem if the weights are such that the unit only outputs zero. This can be solved by adjusting to use the LeakyReLU which has a small positive slope α for when $x < 0$.

If it is desirable for the output of the unit to be between zero and one then a sigmoid activation function is sensible.

Sometimes referred to as an activation function, there is also the softmax function which can be applied to a layer to make the sum of the outputs of all of the units in a layer equal 1 as well as constraining the outputs of the individual units to $(0, 1)$. The softmax is defined as

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$$

for a layer with J units where x_j is the output of the jth unit. In Keras, the softmax can be used as the activation fuction for a Dense layer.

Once a NN archtecture has been chosen, a loss function and optimization algorithm still needs to be chosen so that the optimal weights can be found.

For regression problems, often the *mean-squared error* (MSE) can be used. It is defined as

$$\text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where $\mathbf{y}$ is the vector of true outputs, $\hat{\mathbf{y}}$ is the vector of predicted outputs and $n = \dim \mathbf{y}$.

For classification problems where each observation belongs to only one class, then the categorical cross-entropy can be used.