COGNIZANCE

# Technical Interview

Gerhard Viljoen

## Question 1:

**What is the difference between Supervised and Unsupervised learning?**

These are two of the main branches of Machine Learning, which itself is the most successful branch of Artificial Intelligence (the third main branch of Machine Learning is Reinforcement Learning).

**Supervised Learning** is Machine Learning with the goal of making predictions on unseen data, after training either a regression or classification model. In Supervised Learning we either want to predict a real-valued quantity (regression) or a class label (classification). In this case it is important to have a response value $(y)$ in addition to the input data $(x)$. In order to quantify how well a Supervised Learning model is performing, a loss function, e.g. $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ can be used to quantify the average deviation of the predicted values $(\hat{y}_i)$ from the desired values $(y)$. The gradient of the loss function informs the updates to the parameters $(\theta)$ of the ML model: $f_\theta(x) = \hat{y}$, with the goal of obtaining a theoretically optimal mapping function $f_\theta{}^*(x) = \hat{y} = y$.

**Unsupervised Learning**, in contrast, is instead concerned with finding interesting and useful features regarding the structure of the input data, the main branches of Unsupervised Learning are: Clustering, Association and Dimensionality Reduction.

Clustering is concerned with finding a defined number of natural groups based on a set of provided features $(x)$, there are no labels provided in such a setup and the loss function in this case would rather be concerned with maximizing within-group similarity, while minimizing between-group dissimilarity.

Association is important for algorithms such as recommendation systems, which focus on finding association rules, mapping item sets (e.g. items purchased, movies watched) to additional entries in the item set (people who purchase beer also purchase diapers, people who watch Game of Thrones also watch Vikings).

Lastly, dimensionality reduction is concerned with discovering the major factors of variation in a dataset and is useful for compression of a large dataset with highly correlated features, to a reduced number of features where the most important factors of variation are kept and those that might only contribute noise are excluded, this can be used for feature selection (before supervised learning) or to find interesting ways of visualizing a dataset during the exploratory data analysis phase of an ML project.

**What are exploding gradients with regards to machine learning?**

Exploding gradients occur when very large ("exploding") gradient-based updates cause a model to become unstable. Sometimes gradients can become so large that they result in `NaN` values, rendering the model untrainable. Exploding gradients in neural networks for example occur via exponential growth when gradients are repeatedly multiplied through network layers with values greater than 1. One can pick up that exploding gradients are occurring when the loss

function “jumps around” erratically from epoch to epoch: going up and down cyclically, when the loss function doesn’t change much at all or when the model loss goes to `NaN`. One can remedy this situation by having fewer layers (it is good practise to start with fewer layers anyway), using a smaller batch size, using gradient clipping, which limits gradients to be within a defined range, using weight regularization, reducing the learning rate, etc.

### What is a confusion matrix?

A confusion matrix is a useful table of values used to quantify how well a classification model performs on unseen data. The important values to look out for are the True Positives and True Negatives on the main diagonal (see green cells in Table 1: Confusion matrix), these cells will contain the number of observations that were predicted to be positive (1 in a binary classification problem) and were in fact positive (True Positives: TP) and those that were predicted to be negative (0 in a binary classification problem) and were in fact negative (True Negatives). The off-diagonal entries indicate Type I- (False Positive: FP) and Type II (False Negative: FN) errors. The calculations in the margins of this table indicate various measures that might be important to optimise for, depending on the particular problem and are all based on various ratios of TP, FP, TN and TN. For example, if we are detecting cancer we might want to optimize for sensitivity, whereas if we are predicting potential high-value customers that require special effort we might want to optimise for specificity.

*Table 1: Confusion matrix*

| | | **Predicted Class** | | |
|---|---|---|---|---|
| | | **Positive** | **Negative** | |
| **Actual Class** | **Positive** | True Positive | False Negative (Type II Error) | Sensitivity $\frac{TP}{TP+FN}$ |
| | **Negative** | False Positive (Type I Error) | True Negative | Specificity $\frac{TN}{TN+FP}$ |
| | | Precision $\frac{TP}{TP+FP}$ | Negative Predictive Value $\frac{TN}{TN+FN}$ | Accuracy $\frac{TP+TN}{TP+TN+FP+FN}$ |

### What is regularisation and why is it important?

When building a machine learning algorithm, we want to keep the bias-variance trade-off in mind, see Figure 1: Bias-variance trade-off, i.e. we want a model which performs well on our training dataset (we want it to have low bias), but we also want to be able to generalize well to unseen data (we also want it to have low variance).

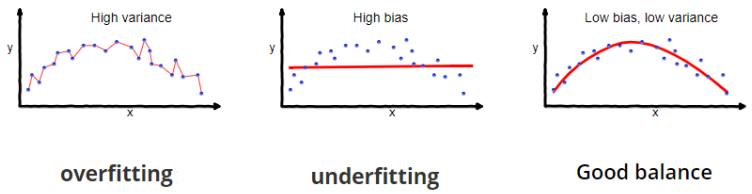


*Figure 1: Bias-variance trade-off*

In order for our model to perform well on unseen data, we need to prevent overfitting, which can occur when a model has sufficient capacity to memorise features that are specific to the

training dataset and will not generalise well to unseen data, since it has fitted itself to “noise” in the training data.

Regularisation is a set of techniques which prevents overfitting from occurring, by for example limiting the capacity of a model (fewer layers or nodes in a neural network, fewer trees or shallower depth in tree-based methods), by early stopping (when training and validation loss start to diverge), or more explicitly by constraining the parameters of a model.

In linear regression we can use techniques such as lasso or ridge regression (adding a weighted penalty to the cost function, based on the square of the magnitude of the regression coefficients), or lasso regression (similar to ridge, but instead basing the penalty on the magnitudes of the coefficients into account) to shrink the importance, or in the case of lasso, completely eliminate features; this is done in order to reduce model complexity as well as multi-collinearity.

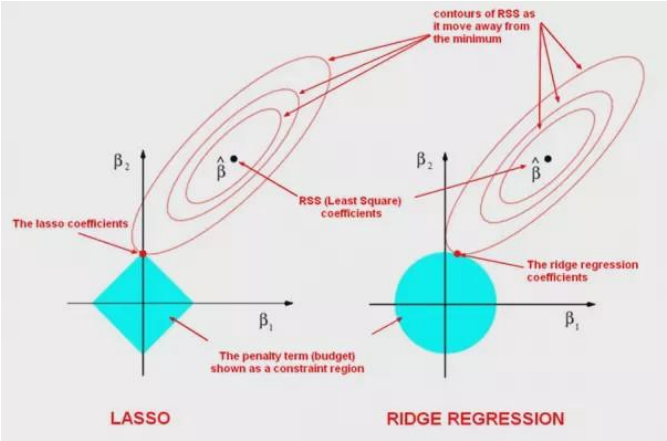


*Figure 2: Lasso and Ridge Regression*

In neural networks, we can use dropout (randomly setting a subset of weights to zero, by using a binary mask during training), L1- and L2-regularization (similar to ridge and lasso), apply Gaussian noise to the input data during training, or perform dataset augmentation by stretching, skewing and otherwise transforming the input data during training in order to help it generalise better.

### Why is a softmax function used in some networks as the last layer?

The softmax function is generally used in multi-class classification in order to normalise the values of the final (output) layer of a neural network to a probability distribution. In other words, softmax normalises the outputs to all be in the interval (0,1), ensuring that the sum of all outputs adds up to 1. The predicted class will simply be the $argmax$ function, which returns the index of the value which is largest in the set of normalised outputs.