**Learning Vector Quantization**

* **Supervised learning + Dimensionality Reduction**

**Description**

This section provides a brief introduction to the Learning Vector Quantization algorithm and the Ionosphere classification problem that we will use in this tutorial

**Learning Vector Quantization**

The Learning Vector Quantization (LVQ) algorithm is a lot like k-Nearest Neighbors.

Predictions are made by finding the best match among a library of patterns. The difference is that the library of patterns is learned from training data, rather than using the training patterns themselves.

The library of patterns are called codebook vectors and each pattern is called a codebook. The codebook vectors are initialized to randomly selected values from the training dataset. Then, over a number of epochs, they are adapted to best summarize the training data using a learning algorithm.

The learning algorithm shows one training record at a time, finds the best matching unit among the codebook vectors and moves it closer to the training record if they have the same class, or further away if they have different classes.

Once prepared, the codebook vectors are used to make predictions using the k-Nearest Neighbors algorithm where k=1.

The algorithm was developed for classification predictive modeling problems, but can be adapted for use with regression problems.

**Ionosphere Dataset**

The Ionosphere dataset predicts the structure of the ionosphere given radar return data.

Each instance describes the properties of radar returns from the atmosphere and the task is to predict whether or not there is structure in the ionosphere.

There are 351 instances and 34 numerical input variables, 17 pairs of 2 for each radar pulse that generally have the same scale of 0-1. The class value is a string with a value of either a “g” for good return or “b” for a bad return.

Using the Zero Rule Algorithm that predicts the class with the most observations, a baseline accuracy of 64.286% can be achieved.

You can learn more and download the dataset from the [UCI Machine Learning Repository](https://archive.ics.uci.edu/ml/datasets/Ionosphere).

Download the dataset and place it in your current working directory with the name **ionosphere.csv**.

**Tutorial**

This tutorial is broken down into 4 parts:

1. Euclidean Distance.
2. Best Matching Unit.
3. Training Codebook Vectors.
4. Ionosphere Case Study.

These steps will lay the foundation for implementing and applying the LVQ algorithm to your own predictive modeling problems.

**1. Euclidean Distance**

The first step needed is to calculate the distance between two rows in a dataset.

Rows of data are mostly made up of numbers and an easy way to calculate the distance between two rows or vectors of numbers is to draw a straight line. This makes sense in 2D or 3D and scales nicely to higher dimensions.

We can calculate the straight line distance between two vectors using the Euclidean distance measure. It is calculated as the square root of the sum of the squared differences between the two vectors.

### 2. Best Matching Unit

The Best Matching Unit or BMU is the codebook vector that is most similar to a new piece of data.

To locate the BMU for a new piece of data within a dataset we must first calculate the distance between each codebook to the new piece of data. We can do this using our distance function above.

Once distances are calculated, we must sort all of the codebooks by their distance to the new data. We can then return the first or most similar codebook vector.

We can do this by keeping track of the distance for each record in the dataset as a tuple, sort the list of tuples by the distance (in descending order) and then retrieve the BMU.

### 3. Training Codebook Vectors

The first step in training a set of codebook vectors is to initialize the set.

We can initialize it with patterns constructed from random features in the training dataset.

This is done iteratively.

1. **Epochs**: At the top level, the process is repeated for a fixed number of epochs or exposures of the training data.
2. **Training Dataset**: Within an epoch, each training pattern is used one at a time to update the set of codebook vectors.
3. **Pattern Features**: For a given training pattern, each feature of a best matching codebook vector is updated to move it closer or further away.

The best matching unit is found for each training pattern and only this best matching unit is updated. The difference between the training pattern and the BMU is calculated as the error. The class values (assumed to be the last value in the list) are compared. If they match, the error is added to the BMU to bring it closer to the training pattern, otherwise, it is subtracted to push it further away.

The amount that the BMU is adjusted is controlled by a learning rate. This is a weighting on the amount of change made to all BMUs. For example, a learning rate of 0.3 means that BMUs are only moved by 30% of the error or difference between training patterns and BMUs.

Further, the learning rate is adjusted so that it has maximum effect in the first epoch and less effect as training continues until it has a minimal effect in the final epoch. This is called a linear decay learning rate schedule and can also be used in artificial neural networks.

### 4. Ionosphere Case Study

In this section, we will apply the Learning Vector Quantization algorithm to the Ionosphere dataset.

The first step is to load the dataset and convert the loaded data to numbers that we can use with the Euclidean distance calculation. For this we will use the helper function **load\_csv()** to load the file, **str\_column\_to\_float()** to convert string numbers to floats and **str\_column\_to\_int()** to convert the class column to integer values.

We will evaluate the algorithm using k-fold cross-validation with 5 folds. This means that 351/5=70.2 or just over 70 records will be in each fold. We will use the helper functions **evaluate\_algorithm()** to evaluate the algorithm with cross-validation and **accuracy\_metric()** to calculate the accuracy of predictions.