CSC411: Assignment #1 (bonus)

Due on Monday, February 5, 2018

Chawla Dhruv

February 5, 2018

Investigating K Nearest Neightbours

Increasing training set performance on increasing K

K Nearest Neighbours works on the assumption that data points with same labels are going to be bunched together (are going to be close together in terms of how we are measuring distance in out model)

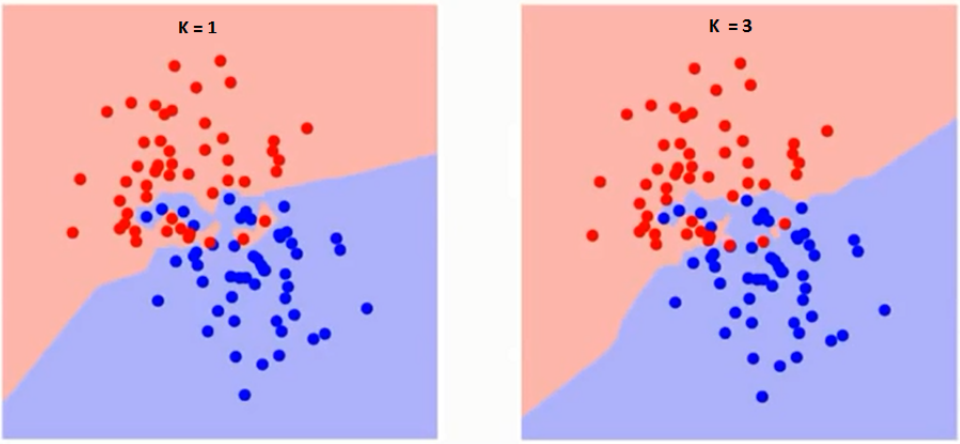

Figure 1: K Nearest Neighbours on a dataset (from analyticsvidhya.com)

As can be observed, the algorithm seems to be working pretty well as the red and blue dot seem to be clustering in specific areas.

Also, observe that in $K = 3$ case, the performance on the training set seems to be lower than on the $K = 1$ case (more blue and red dots are being classified as the other way around). This can also be seen on the error vs K chart produced (more error means worse performance on the training set).

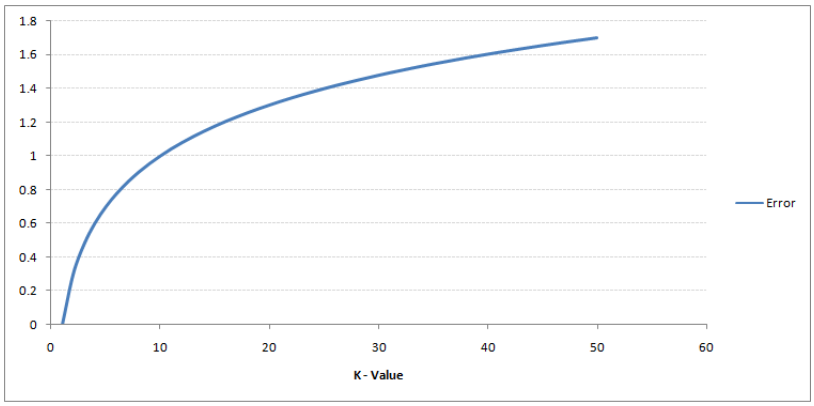

Figure 2: Error vs K (from analyticsvidhya.com)

This can be generally understood as the more you increase K, the more influence surrounding neighbours of potentially different labels have and they can swing the classifier to their favour (especially on data points that are on a boundary). As an extension, performance on $K = 1$ is always a 100% as the only neighbour we are considering is the data point itself.

In this investigation, I have tried to produce a data set which does not have data points cluster together in terms of Euclidian/Cartesian distance (although they would in terms of Polar/Radial distances). My dataset can be described as concentric circular rings with alternating labels.

This data set is produced in `generate_data.py`.

```python
def generate_data():
    """Generate random data resembling alternating concentric rings

    Returns
    X: list(list(x, y)) : coordinates of the data point
    y: list(either 0 or 1) : label of the data point
    """
    X = []
    y = []

    label = 1

    np.random.seed(5)

    for i in range(3):
        for j in range(8):
            for k in range(60):
                x_coord = (i - 0.5 + np.random.random_sample()) * np.cos(j/8 * 2*np.pi)
                y_coord = (i - 0.5 + np.random.random_sample()) * np.sin(j/8 * 2*np.pi)

                X += [[x_coord, y_coord]]
                y += [label]

        label = 1 - label

    return X, y
```

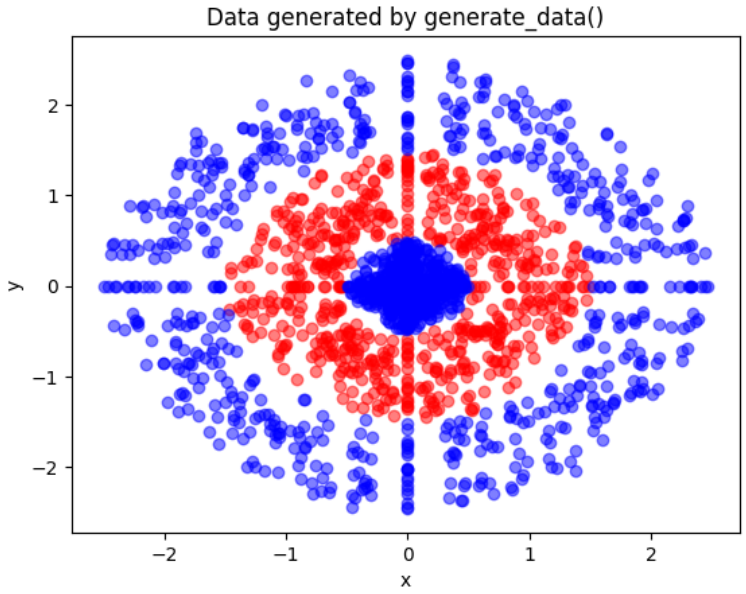

Figure 3: Data set visualized

This data set challenges the underlying assumption of K Nearest Neighbours and produces a dip in training set performance at around $K = 400$. Performance picks up soon after as the outermost ring starts to influence the inner circle classification. The performance finally settles at roughly 67%, which is due to the fact that there are twice as many data points labelled as 1 than labelled as 0.

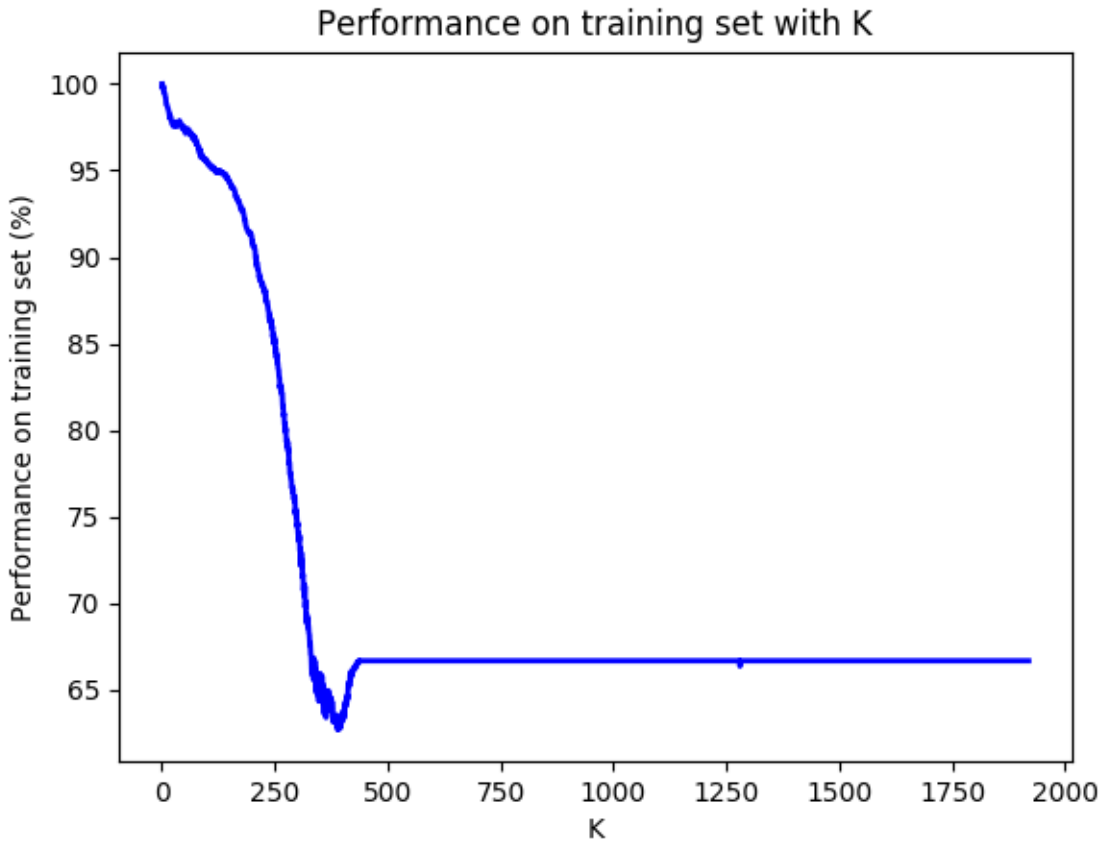

Figure 4: Performance vs K on training set

Insight
The circumstances under which performance increases on increasing K in K Nearest Neighbours is when the data set in arranged in a way that when K is increased slightly at first, data points find themselves surrounded by data points of a different label.

In this case, this was when $K > 1$ and $K < 400$ and the data points in inner circle and data points in outermost ring were being influenced by the middle ring data points.

On increasing K further, the performance would start rising towards the final performance when K is infinitely large as we are considering nearly all points in the training set and the performance just becomes a majority vote.

In this case, this can be seen when K is between 400 and 500. The inner circle and outermost ring come in influence of each other and make the performance rise towards the final value of 67%.

References

1. https://www.analyticsvidhya.com/blog/2014/10/introduction-k-neighbours-algorithm-clustering/