

k-NN Diagnosing Breast Cancer

Prof. Eric A. Suess

February 5, 2020

Example

Breast cancer screening allows the disease to be diagnosed and treated prior to it causing noticeable symptoms. The process of early detection involves examining the breast tissue for abnormal lumps and masses.

If a lump is found, cells are extracted and examined to determine if the mass is malignant or benign.

Machine Learning

If machine learning could automate the identification of cancerous cells, that would be beneficial. It would speed up the process.

There might be less human subjectivity.

UCI Machine Learning Repository

The UCI Machine Learning Repository is a where many datasets are posted that can be used for machine learning practice.

<http://archive.ics.uci.edu/ml/>

Step 1 - collecting data

Breast Cancer Wisconsin Diagnostic dataset

Measurements from digitized images of fine-needle aspirate of a breast mass. The values represent characteristics of the cell nuclei present in the digital image.

- ▶ 569 observations/examples/instances
- ▶ 32 variables/features/attributes
- ▶ 10 different characteristics measured.

Diagnosis M = malignant (cancer), B = benign (no cancer)

Step 2 - exploring and preparing the data

ID variable should be excluded. A perfect predictor. **Over fitting.**

Target usually needs to be a **factor** in R.

Since k-NN is dependent upon the measurement scale of the input features, *normalization* is used to transform the data to a common scale.

Step 2 - normalizing

In R

- ▶ **function()**
- ▶ **lapply()**
- ▶ **as.data.frame()**

Step 2 - creating training and test datasets

How well does the *learner* perform on a dataset of unlabeled data?

Next 100 measurements in the lab.

Or we can simulate this scenario.

- ▶ Training dataset, first 469 observations/records is used to build the kNN model
- ▶ Test dataset, remaining 100 to simulate new patients

Note the data is already in *random order*.

Step 3 - training a model on the data

For the k-NN algorithm, the training phase involves no model fitting – the process of training a lazy learner like k-NN involves storing the input data in a structured format.

knn() function in the **class** package is used.

Step 3 - classifying

A test instance is classified by taking a “vote” among the k -Nearest Neighbors – specifically, this involves assigning the class of the majority of the k neighbors. A tie is broken at random.

The function `knn()` returns a factor vector of predicted classes for each row in the test data frame.

Step 4 - evaluating model/algorithm performance

Evaluate how well the predicted classes match with the known values.

CrossTable() in the **gmodels** packages is used.

- ▶ TN true negatives
- ▶ TP true positives
- ▶ FN false negatives - very bad
- ▶ FP false positive - bad also

Prediction involves balancing between FP and FN

Step 5 - improving model/algorithm performance

- ▶ different rescaling
- ▶ different k

Step 5 - z-score

The z-score does not restrict between 0 and 1.

The z-score allows for more weight to be given to larger distances.

- ▶ **scale()**

In this example, the results are not as good.

Step 5 - alternative k

k of 1 is too small, increases FP

Summary

k-NN **does not do any learning.**

It simply stores the training data verbatim.

Unlabeled test observations/examples/instances/records are then matched to the most similar observations in the training set using a distance function, and then unlabeled examples are assigned to the label of its neighbors.