Instance-Based Learning

Prof. Dan A. Simovici

UMB
1. *k*-nearest neighbor learning

2. kNN in R
Instance-based learning is a process that consists in using accumulated examples followed by a classification decision that occurs when a certain new instance is presented.

When a new instance is presented comparisons with existing stored examples are performed and a classification decision based on this comparisons with existing instances is rendered.
Basic assumption in $k$-nearest neighbor learning (kNNL):

- all instances correspond to points in $\mathbb{R}^n$;
- $T$ is the set of training examples, and $V$ is a finite set of labels;
- a discrete-valued target function $f : T \rightarrow V$ is defined.
The training algorithm:

- for each training example \((x, f(x))\) add the example to the list \(T\);
- given a query instance \(x\) to be classified let \(x_1, \ldots, x_k\) be the list of training examples that are nearest to \(x\);
- return \(\hat{f}(x) = \arg\max_{v \in V} \delta(v, f(x_i))\), where

\[
\delta(a, b) = \begin{cases} 
1 & \text{if } a = b, \\
0 & \text{otherwise}.
\end{cases}
\]
Example
Consider the set that consists of 10 examples.

\[ \text{x is classified as positive by an 1-NN classifier and as negative by a 5-NN classifier.} \]
Suppose that we apply the 1NNL to a finite set $S$ of training examples in $\mathbb{R}^2$, where

$$S = \{x_1, x_2, \ldots, x_n\}.$$  

**Definition**

Let $S \subseteq \mathbb{R}^2$, and let $x, y \in S$. The *dominance* of a $x$ over a site $y$ is the set

$$\text{dmn}(x, y) = \{u \in \mathbb{R}^2 \mid d(x, u) \leq d(y, u)\}.$$ 

Geometrically, $\text{dmn}(x, y)$ is a closed half-plane determined by the perpendicular bisector of the segment $[x, y]$. 
The region of a site \( x \) is the intersection \( \text{reg}(x) \) of all dominance regions of \( x \) relative to the other sites in \( S \) that is,
\[
\text{reg}(x) = \bigcap \{ \text{dmn}(x, y) \mid y \in S - \{x\} \}.
\]

Since regions are intersection of finite collections of half-planes, each region is a **convex polygon** and the boundary of a region consists of at most \( |S| - 1 \) edges.

Since \( x \in \text{reg}(x) \) for every \( x \in S \), no region of a site is empty. Thus, the collection of regions constitute a polygonal partition \( \pi_S \) of \( \mathbb{R}^2 \). This partition is referred to as a **Voronoi diagram**.
Example

Let $S = \{x, y, z\}$ be a set in $\mathbb{R}^2$ such that $x, y, z$ are non-collinear. Note that the perpendicular bisectors of the segments $[x, y]$, $[y, z]$, and $[z, x]$ all pass through the center of the circumcircle of the triangle $(x, y, z)$. Thus, they divide the plane into three regions.
Example

Consider a set $S = \{x_1, \ldots, x_7\}$ of points in $\mathbb{R}^2$ whose coordinates are specified as

$$
\begin{align*}
x_1 &= \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \\
x_2 &= \begin{pmatrix} 3 \\ 6 \end{pmatrix}, \\
x_3 &= \begin{pmatrix} 5 \\ 3 \end{pmatrix}, \\
x_4 &= \begin{pmatrix} 5 \\ 8 \end{pmatrix}, \\
x_5 &= \begin{pmatrix} 7 \\ 1 \end{pmatrix}, \\
x_6 &= \begin{pmatrix} 7 \\ 5 \end{pmatrix}, \\
x_7 &= \begin{pmatrix} 9 \\ 4 \end{pmatrix}.
\end{align*}
$$
Example cont’d
If $S = \{x_1, \ldots, x_n\}$ is a set of $n$ sites in general position (that is, no three sites are collinear), then we have $\binom{n}{2}$ perpendicular bisectors of segments of the form $[x_i, x_j]$ with $i < j$.

The Voronoi diagram of $S = \{x_1, \ldots, x_n\}$ can be regarded as a planar graph $G = (\{\text{reg}(x_i) \mid 1 \leq i \leq n\}, E)$ having the vertices $\text{reg}(x_1), \ldots, \text{reg}(x_n)$. An edge exists between $\text{reg}(x_i)$ and $\text{reg}(x_j)$ if these regions are adjacent.
The decision surface is a combination of convex polyhedra surrounding each training example. The polygon in $\mathbb{R}^2$ that contains $x_i$ defines examples that will have the same class as $x_i$. 
The choice of $k$ is very important:

- If $k$ is very large, say equal to the number of observations, then the predicted class will always be the class of the majority of observations.
- If $k = 1$ noisy data or outliers will unduly influence the classification of examples. Any unlabeled example that is located nearest to an incorrectly labeled example, will be contaminated by this example.
The \( kNN \) technique for continuous-valued target function calculates the mean values of the \( k \) nearest training examples and assigns to \( \mathbf{x} \) the value

\[
f(\mathbf{x}) \rightarrow \frac{1}{k} \sum_{i=1}^{k} f(\mathbf{x}_i).
\]
kNN can be refined by assigning weights $w_i$ to each of the $k$ neighbors according to their distance to the query points with greater weight for the closest neighbors. If

$$w_i = \frac{1}{d(x, x_i)^2}$$

we can define $f(x)$ as

$$f(x) = \begin{cases} \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i)) & \text{if } x \not\in \{x_1, \ldots, x_k\} \\ f(x_i) & \text{if } x = x_i \text{ for some } x_i. \end{cases}$$
The function knn is a part of the class R package, which includes various functions for classification, including k-nearest neighbour, vector quantization and self-organizing maps.

Make sure that you installed the packages ISRL and class!
We use the Default data set of the R package ISLR (Introduction to Statistical Learning) which contains information on ten thousand customers.

The variables are:

- default: a factor with levels No and Yes indicating whether the customer defaulted on their debt;
- student: a factor with levels No and Yes indicating whether the customer is a student;
- balance: the average balance on the account;
- income: income of customer.
knn requires that all predictors be numeric, so we coerce student to be a 0 and 1 dummy variable instead of a factor:

```r
Default$student = as.numeric(Default$student) - 1
```

The response (default) must remain a factor.

The training set and the test set are defined as:

```r
set.seed(42)
default_idx = sample(nrow(Default), 5000)
default_trn = Default[default_idx, ]
default_tst = Default[-default_idx, ]
```
Here, `knn` takes four arguments:

- `train`, the predictors for the train set;
- `test`, the predictors for the test set; `knn` will output results (classifications) for these cases;
- `cl`, the true class labels for the train set;
- `k`, the number of neighbors to consider.
Consider the following definitions:

```r
# training data
X_default_trn = default_trn[, -1]
y_default_trn = default_trn$default

# testing data
X_default_tst = default_tst[, -1]
y_default_tst = default_tst$default

to be used next.
```
The function `calc_class_err()` function is used to assess how well knn works with this data. Its definition is:

```r
calc_class_err = function(actual, predicted) {
    mean(actual != predicted)
}
```

This function is used as

```r
calc_class_err(actual = y_default_tst, 
               predicted = knn(train = X_default_trn, 
                               test = X_default_tst, 
                               cl = y_default_trn, 
                               k = 5))
```

## [1] 0.0312
Leveling the scales of the predictors variables may help the performance. If one variable is much larger numbers it will dominate other variables in the distance measurements. It is desirable to scale the predictors to have a mean of zero and unit variance, which can be done by writing:

```r
calc_class_err(actual = y_default_tst, 
                predicted = knn(train = scale(X_default_trn),
                                test = scale(X_default_tst),
                                cl = y_default_trn, 
                                k = 5))
```

```
## [1] 0.0284
```
To determine which k works best use the code:

```r
set.seed(42)
k_to_try = 1:100
err_k = rep(x = 0, times = length(k_to_try))

for (i in seq_along(k_to_try)) {
  pred = knn(train = scale(X_default_trn),
              test = scale(X_default_tst),
              cl = y_default_trn,
              k = k_to_try[i])
  err_k[i] = calc_class_err(y_default_tst, pred)
}
```
(Test) Error Rate vs Neighbors

- k, number of neighbors
- classification error
- Error rate decreases as k increases, reaching a minimum at around k = 20, after which it increases again.
# plot error vs choice of k
plot(err_k, type = "b", col = "dodgerblue", cex = 1, pch = 20,
     xlab = "k, number of neighbors", ylab = "classification error",
     main = "(Test) Error Rate vs Neighbors")
# add line for min error seen
abline(h = min(err_k), col = "darkorange", lty = 3)
# add line for minority prevalence in test set
abline(h = mean(y_default_tst == "Yes"), col = "grey", lty = 2)