KNN Classification and Regression

#	Height (inches)	Weight (kgs)	B.P. Sys	B.P. Dia	Heart disease	Cholesterol Level
1	62	70	120	80	No	150
2	72	90	110	70	No	160
3	74	80	130	70	No	130
4	65	120	150	90	Yes	200
5	67	100	140	85	Yes	190
6	64	110	130	90	No	130
7	69	150	170	100	Yes	250
8	66	115	145	90		

KNN Classification and Regression

#	Height (inches)	Weight (kgs)	B.P. Sys	B.P. Dia	Heart disease	Cholesterol Level	Euclidean Distance
1	62	70	120	80	No	150	52.59
2	72	90	110	70	No	160	47.81
3	74	80	130	70	No	130	43.75
4	65	120	150	90	Yes	200	7.14
5	67	100	140	85	Yes	190	16.61
6	64	110	130	90	No	130	15.94
7	69	150	170	100	Yes	250	44.26
8	66	115	145	90			

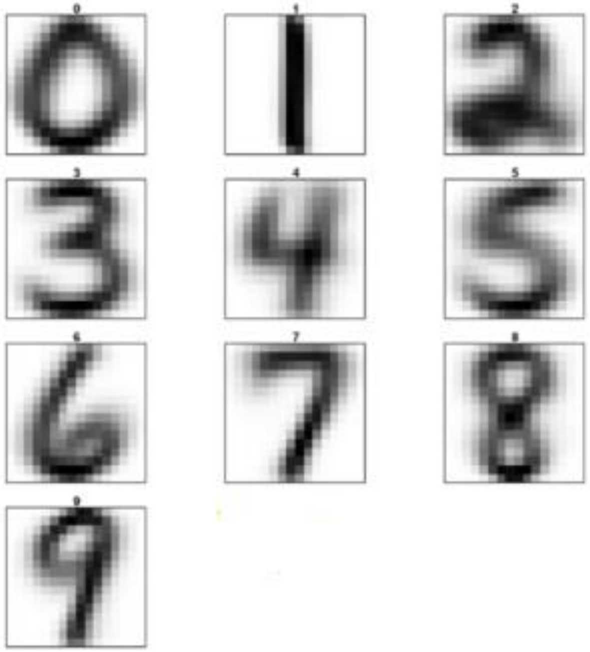
Here the data in red is p and the other data is q, so the eculedian distance will be calculated as:
(p1-q1) + (p2-q2) +-------

So here::
(66-62)^2+(115-70)^2+
(145-120)^2+(90-80)^2 ^1/2

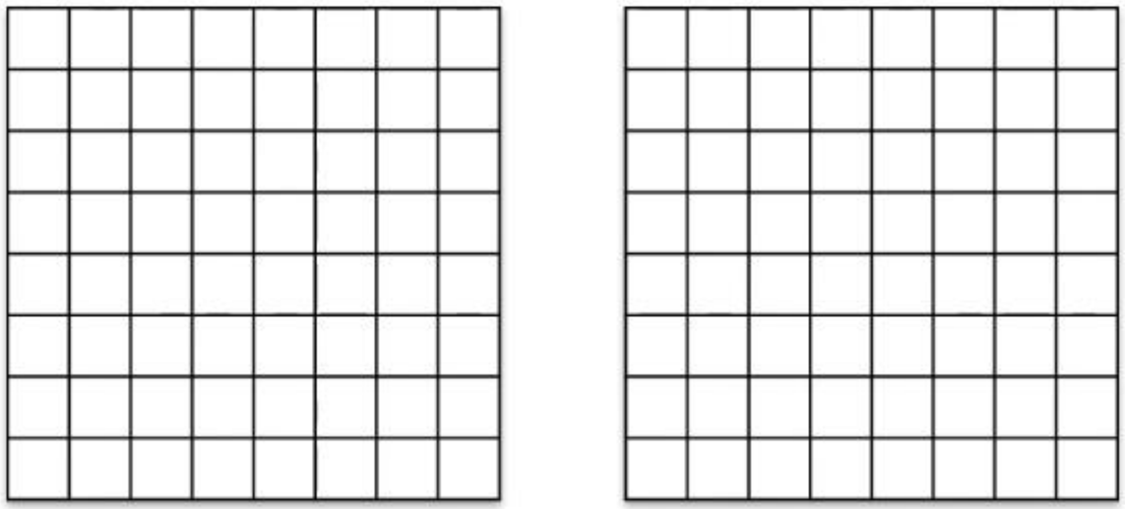
Then the distance for s#1 will be calcualted and same for others

Example: Handwritten digit recognition

- 16x16 bitmaps
- 8-bit grayscale
- Euclidean distances over raw pixels



x y



- $D(x,y) = \sqrt{\sum_{i=0}^{255}(x_i - y_i)^2}$

Accuracy:

- 7-NN ~ 95.2%
- SVM ~ 95.8%
- Humans ~ 97.5%

The KNN Algorithm

Input: Training samples $D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n)\}$, Test sample $d = (\vec{x}, y)$, k. Assume $\vec{x}$ to be an m-dimensional vector.

Output: Class label of test sample d

Algorithm/steps of kNN, most important

1. Compute the distance between d and every sample in D
2. Choose the K samples in D that are nearest to d; denote the set by $S_d \in D$
3. Assign d the label y_i of the majority class in S_d

Note:

All action takes place in the test phase, the training phase is essentially to clean, normalize and store the data

Complexity of KNN

Input: Training samples $D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_n, y_n)\}$, Test sample $d = (\vec{x}, y)$, k. Assume $\vec{x}$ to be an m-dimensional vector.

Output: Class label of test sample d

1. Compute the distance between d and every sample in D

   n samples, each is m-dimensional $\Rightarrow O(mn)$

2. Choose the K samples in D that are nearest to d; denote the set by $S_d \in D$
   - Either naively do K passes of all samples costing $O(n)$ each time for $O(nk)$
   - Or use the *quickselect* algorithm (median of medians) to find the kth smallest distance in $O(n)$ and then return all distances no larger than the kth smallest distance. This will accumulate to $O(n)$

3. Assign d the label y_i of the majority class in S_d

   This is $O(k)$.

Time complexity: $O(mn + n + k) = O(mn)$, assuming k to be a constant.

Choosing the value of K – The theory

k=1:

- High variance
- Small changes in the dataset will lead to big changes in classification
- Overfitting
- Is too specific and not well-generalized
- It tends to be sensitive to noise
- The model accomplishes a high accuracy on train set but will be a poor predictor on new, previously unseen data points

k= very large (e.g. 100):

- The model is too generalized and not a good predictor on both train and test sets.
- High bias
- Underfitting

k=n:

- The majority class in the dataset wins for every prediction
- High bias

Tuning the hyperparameter K – the Method

Divide your training data into training and validation sets.

Do multiple iterations of m-fold cross-validation, each time with a different value of k, starting from k=1

Keep iterating until the k with the best classification accuracy (minimal loss) is found

- What happens if we use the training set itself, instead of a validation set? Which k wins?
  - K=1, as there is always a nearest instance with the correct label, the instance itself

KNN – The good, the bad and the ugly

KNN is a simple algorithm but is highly effective for solving various real life classification problems. Especially when the datasets are large and continuously growing.

We will show that as $n \to \infty$, the 1-NN classifier is only a factor 2 worse than the best possible classifier (remember our old friend the Bayes Optimal Classifier?).

Challenges:

1. How to find the optimum value of K?
2. How to find the right distance function?

Problems:

1. High computational time cost for each prediction.
2. High memory requirement as we need to keep all training samples.
3. The curse of dimensionality.

Bayes Error (https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html)

Assume that we know $P(y|x)$, so we can simply predict the correct label y^* as:

$$\boldsymbol{y^* = h_{opt}(x) = argmax_y P(y|x)}$$

Although the Bayes Classifier is optimal, but it is not perfect and can still make mistakes. E.g. It will predict incorrectly when a test point does not have the most likely label.

So, if $P(y^*|x)$ is the probability of correct classification then the probability of incorrect classification, the Bayes Error is given as:

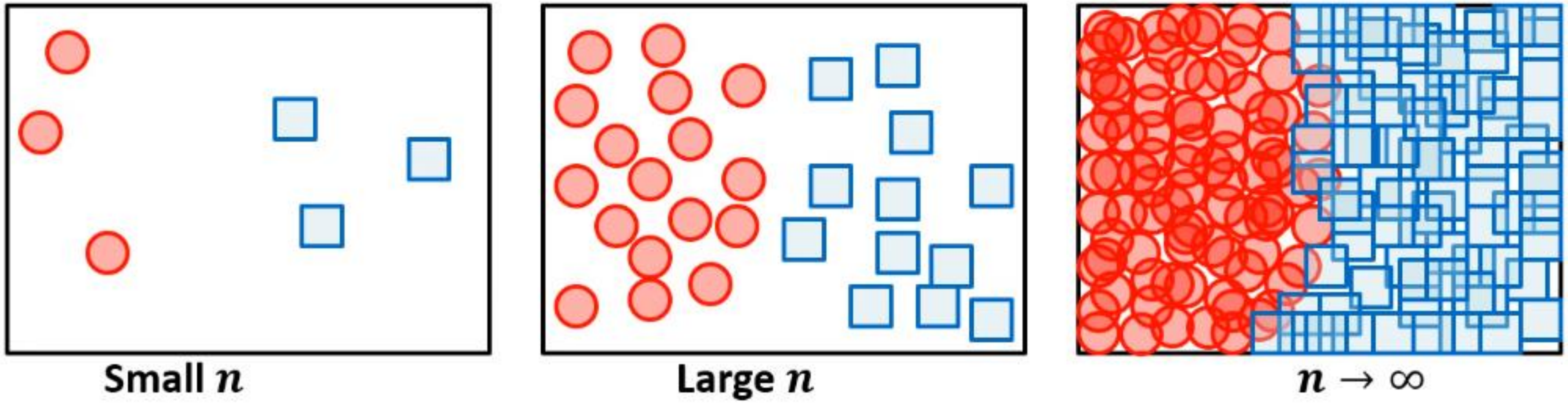
$$\boldsymbol{\epsilon_{Bayes} = 1 - P(y^*|x)}$$

1-NN Error as $n \to \infty$ (Cover and Hart 1967, Weinberger Lec 2)

Let x_{NN} be the nearest neighbor of our test point x_t

As $n \to \infty$, $dist(x_{NN}, x_t) \to 0$

- i.e. $x_{NN} \to x_t$
- 1-NN returns the label of x_{NN}



What is the probability that this is not the correct label of x_t?

- As $x_{NN} \to x_t$, the probability of misclassification is the same as the probability of x_{NN} and x_t having different labels

1-NN Error as $n \rightarrow \infty$ (Cover and Hart 1967, Weinberger Lec 2)

There are two ways this could happen.

- What's the probability that x_{NN} had the correct label y^*?
  - $P(y^*|x_{NN})$
- What's the probability that x_{NN} did not have the correct label y^*?
  - $1 - P(y^*|x_{NN})$
- Whats the probability that y^* was the correct label of x_t?
  - $P(y^*|x_t)$

1. So, what's the probability that y^* was the correct label of x_t but the nearest neighbor x_{NN} did not have that label?

$$\boldsymbol{P(y^*|x_t)(1 - P(y^*|x_{NN}))}$$

- Whats the probability that y^* was the correct label of x_t?
  - $P(y^*|x_t)$
- Whats the probability that y^* was not the correct label of x_t?
  - 1-$P(y^*|x_t)$
- What's the probability that x_{NN} had the correct label y^*?
  - $P(y^*|x_{NN})$

2. So, what's the probability that y^* was not the correct label of x_t but the nearest neighbor x_{NN} had that label?

$$\boldsymbol{P(y^*|x_{NN})(1 - P(y^*|x_t))}$$

1-NN Error as $n \to \infty$ (Cover and Hart 1967, Weinberger Lec 2)

So, the total probability of misclassification is:

$$\epsilon_{NN} = P(y^*|x_t)\big(1 - P(y^*|x_{NN})\big) + P(y^*|x_{NN})\big(1 - P(y^*|x_t)\big)$$

As $P(y^*|x_t) \leq 1$ and $P(y^*|x_{NN}) \leq 1$,

$$\epsilon_{NN} = P(y^*|x_t)\big(1 - P(y^*|x_{NN})\big) + P(y^*|x_{NN})\big(1 - P(y^*|x_t)\big)$$
$$\leq 1\big(1 - P(y^*|x_{NN})\big) + 1\big(1 - P(y^*|x_t)\big)$$

As, $x_{NN} \to x_t, P(y^*|\, x_{NN}) = P(y^*|x_t)$

$$\epsilon_{NN} \leq \big(1 - P(y^*|x_t)\big) + \big(1 - P(y^*|x_t)\big)$$

$$\epsilon_{NN} \leq 2\big(1 - P(y^*|x_t)\big)$$

$$\boldsymbol{\epsilon_{NN} \leq 2\epsilon_{Bayes}}$$