Introduction to Machine Learning

PART 3: k-Nearest Neighbors

Marc Sebban

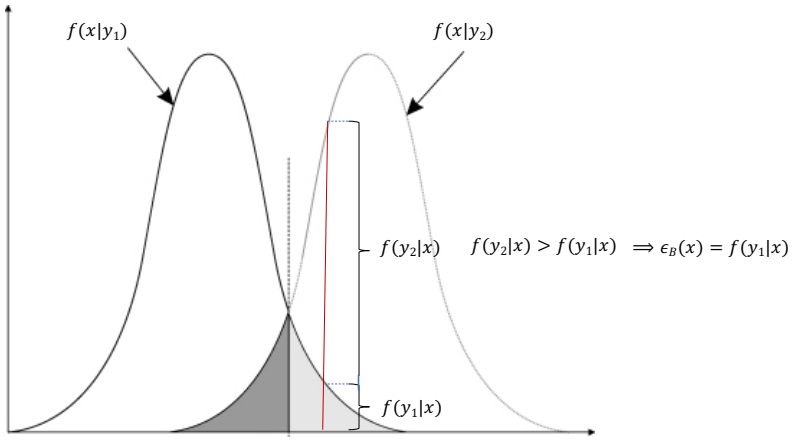
Laboratoire Hubert Curien, UMR CNRS 5516
Université Jean Monnet Saint-Étienne

Bayes Error

The Bayes error ϵ_B is the lowest possible error rate (or irreducible error) for any hypothesis h (for the sake of simplicity, f or p will be used indistinctively). In the binary setting,

$$\epsilon_B = \int_x f(y_{min}(x)|x) f(x) dx$$

where (x, y) is a labeled instance and
$y_{min}(x) = \arg\min_{y_1, y_2} \{f(y_1|x), f(y_2|x)\}$



Bayesian Classifier

Bayesian Classifier

The Bayesian classifier predicts the optimal class $y^* \in \mathcal{Y}$ given an example $\mathbf{x} \in \mathcal{X}$ by applying the Maximum a posteriori (MAP) decision rule:

$$\forall y_c \in \mathcal{Y}, p(y_c|\mathbf{x}) = \frac{p(\mathbf{x}|y_c).p(y_c)}{p(\mathbf{x})}$$

$$y^*(\mathbf{x}) = \arg\max_c p(y_c|\mathbf{x}).$$

that corresponds to $y^*(\mathbf{x}) = \arg\max_c p(\mathbf{x}|y_c).p(y_c)$

Bayesian Classifier

Bayesian Classifier

The Bayesian classifier predicts the optimal class $y^* \in \mathcal{Y}$ given an example $\mathbf{x} \in \mathcal{X}$ by applying the Maximum a posteriori (MAP) decision rule:

$$\forall y_c \in \mathcal{Y}, p(y_c|\mathbf{x}) = \frac{p(\mathbf{x}|y_c).p(y_c)}{p(\mathbf{x})}$$

$$y^*(\mathbf{x}) = \arg\max_c p(y_c|\mathbf{x}).$$

that corresponds to $y^*(\mathbf{x}) = \arg\max_c p(\mathbf{x}|y_c).p(y_c)$

If this calculation is possible, the Bayesian classifier is optimal from a probabilistic point of view, with an associated error ϵ_B.

Exercise

Reminder

$$\epsilon_B = \int_x f(y_{min}(x)|x) f(x) dx$$

Let f_{c_1} and f_{c_2} be the densities of two classes c_1 and c_2:

$$f_{c_1} = \frac{3}{2} y^2 + y \quad pour \quad 0 < y < 1$$

$$f_{c_2} = 1 \quad pour \quad \frac{3}{4} < y < \frac{7}{4}$$

Question

After having drawn these densities in a 2-dimensional space, calculate the Bayes error ϵ_B.

Underlying conditions to solve this problem

To compute ϵ_B, one needs some priors:

1. Know the *a priori* probabilities $p(y_j)$ of the different classes.
2. Know the probabilities of the observations given the classes $p(\mathbf{x}|y_j)$.

Unfortunately, $p(y_j)$ and $p(\mathbf{x}|y_j)$ are unknown. One needs to estimate these two quantities from the training sample S.

Estimation of the a priori probability of the classes $p(y_j)$

What about $p(y_j)$?

An unbiased estimate of $p(y_j)$ is given by the observed frequency $\hat{p}(y_j) = \frac{|S_j|}{|S|}$ where $|S_j|$ is the number of training examples belonging to the class y_j.

Estimation of the conditional probabilities $p(\mathbf{x}|y_j)$

What about $p(\mathbf{x}|y_j)$?

We can distinguish two types of approaches:

1. The parametric methods which assume that $p(\mathbf{x}|y_j)$ follows a given statistical distribution. In this case, the problem to solve consists in estimating the parameters of the considered distribution (*e.g.* normal distribution with μ and σ or Binomial distribution with p $\Rightarrow$ **Likelihood Maximization**).

Estimation of the conditional probabilities $p(\mathbf{x}|y_j)$

What about $p(\mathbf{x}|y_j)$?

We can distinguish two types of approaches:

1. The parametric methods which assume that $p(\mathbf{x}|y_j)$ follows a given statistical distribution. In this case, the problem to solve consists in estimating the parameters of the considered distribution (*e.g.* normal distribution with μ and σ or Binomial distribution with p $\Rightarrow$ **Likelihood Maximization**).
2. The non parametric methods which do not impose any constraint about the underlying distribution, and for which the densities $p(\mathbf{x}|y_j)$ are locally estimated around $\mathbf{x}$.

Non parametric methods

- No assumption is made on the underlying distribution...
- ... we only suppose that this target distribution is locally regular.
- The objective is to estimate $p(\mathbf{x}|y_j)$. Since this must be done $\forall y_j \in \mathcal{Y}$, for the sake of simplicity, **let us focus on $p(\mathbf{x})$ first**.

Non parametric methods

- Let p be the unknown target probability density. The probability $\mathcal{P}$ to observe x in a volume V is:

$$\mathcal{P} = \int_V p(\mathbf{x}) d\mathbf{x}$$

Non parametric methods

- Let p be the unknown target probability density. The probability $\mathcal{P}$ to observe x in a volume V is:

$$\mathcal{P} = \int_V p(\mathbf{x}) d\mathbf{x}$$

- Assuming that $p(\mathbf{x})$ does not significantly change in V (locally regular), we can approximate P such that:

$$\hat{\mathcal{P}} \simeq p(\mathbf{x}) \times V \tag{1}$$

Non parametric methods

- Let p be the unknown target probability density. The probability $\mathcal{P}$ to observe x in a volume V is:

$$\mathcal{P} = \int_V p(\mathbf{x}) d\mathbf{x}$$

- Assuming that $p(\mathbf{x})$ does not significantly change in V (locally regular), we can approximate P such that:

$$\hat{\mathcal{P}} \simeq p(\mathbf{x}) \times V \tag{1}$$

- $\mathcal{P}$ can also be estimated by the proportion of training data in V:

$$\hat{\mathcal{P}} \simeq \frac{k}{n} \tag{2}$$

Non parametric methods

- Let p be the unknown target probability density. The probability $\mathcal{P}$ to observe x in a volume V is:

$$\mathcal{P} = \int_V p(\mathbf{x}) d\mathbf{x}$$

- Assuming that $p(\mathbf{x})$ does not significantly change in V (locally regular), we can approximate P such that:

$$\hat{\mathcal{P}} \simeq p(\mathbf{x}) \times V \tag{1}$$

- $\mathcal{P}$ can also be estimated by the proportion of training data in V:

$$\hat{\mathcal{P}} \simeq \frac{k}{n} \tag{2}$$

- Therefore, we can deduce that

$$p(\mathbf{x}) \approx \frac{k}{nV} = \hat{p}(\mathbf{x}).$$

Theorem

Let us denote by $\hat{p}_n(\mathbf{x}) = \frac{k_n}{nV_n}$ *the estimate of* $p(\mathbf{x})$ *from a training sample* S *of size* n. *When* n *is increasing,* $\hat{p}_n(\mathbf{x})$ *converges to* $p(\mathbf{x})$ *if the following three conditions are fulfilled:*

$$\lim_{n\to\infty} V_n = 0$$

$$\lim_{n\to\infty} k_n = \infty$$

$$\lim_{n\to\infty} \frac{k_n}{n} = 0$$

Theorem

Let us denote by $\hat{p}_n(\mathbf{x}) = \frac{k_n}{nV_n}$ *the estimate of* $p(\mathbf{x})$ *from a training sample* S *of size* n. *When* n *is increasing,* $\hat{p}_n(\mathbf{x})$ *converges to* $p(\mathbf{x})$ *if the following three conditions are fulfilled:*

$$\lim_{n\to\infty} V_n = 0$$

$$\lim_{n\to\infty} k_n = \infty$$

$$\lim_{n\to\infty} \frac{k_n}{n} = 0$$

Interpretation

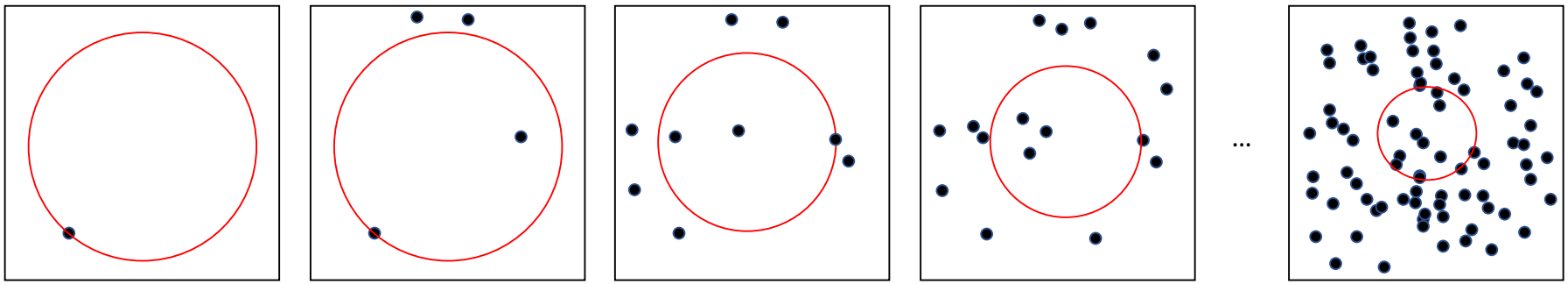
If n is high, we get:

- a good estimate $\hat{p}(x)$ (resp. $\hat{p}(x|y_j)$ if we take into account the class) of $p(x)$ (resp. $p(x|y_j)$).
- a good approximation of the Bayesian method.

Non parametric methods

k-Nearest Neighbors (Cover & Hart 1968)

It turns out that the k-nearest neighbor method satisfies the previous conditions: **fixing a number k_n of examples** (growing w.r.t. n) and **adapting a volume V_n** (*e.g.* a hypersphere centered at $\mathbf{x}$) such that k_n examples are in the volume.



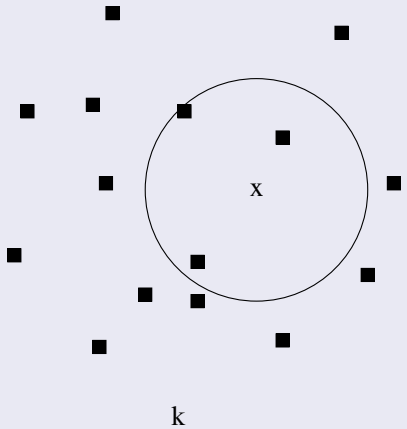
$$k_n = \sqrt{n}$$

k-nearest neighbors

kNN as a good way to estimate $p(\mathbf{x})$

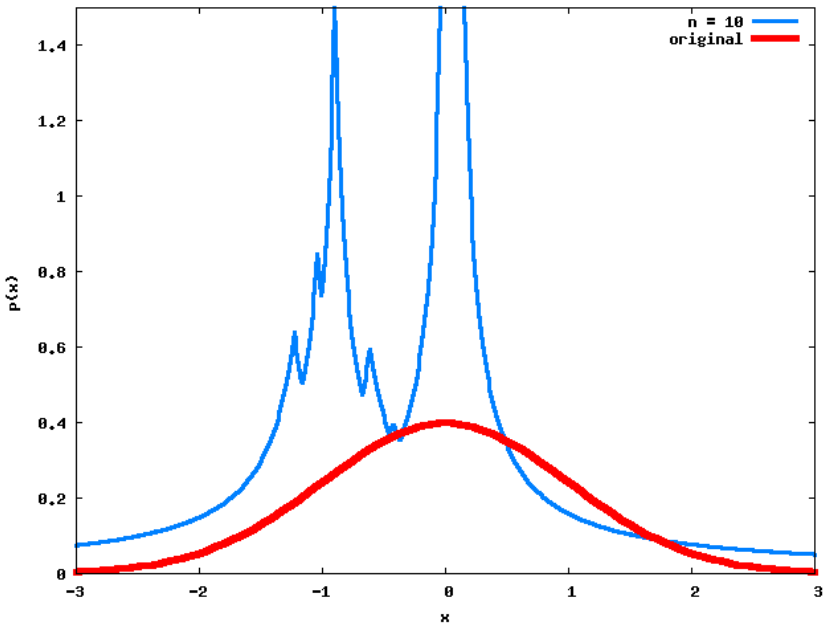
To estimate $p(\mathbf{x})$ from the n training examples of S, let us build a hypersphere centered at $\mathbf{x}$ that contains k_n examples.

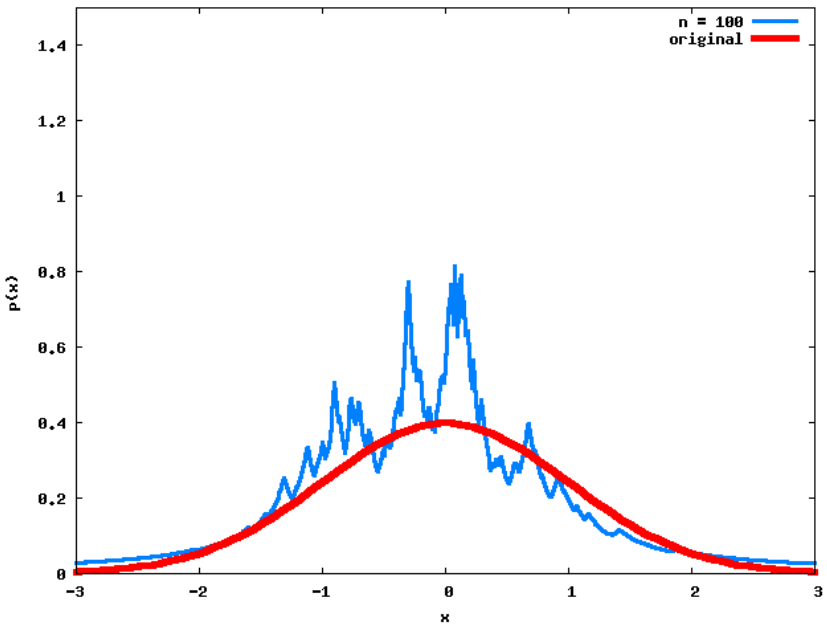
$\hat{p}_n(\mathbf{x}) = \frac{k_n/n}{V_n}$ is a good estimate of $p(\mathbf{x})$



Example 1 with $k = \sqrt{n}$

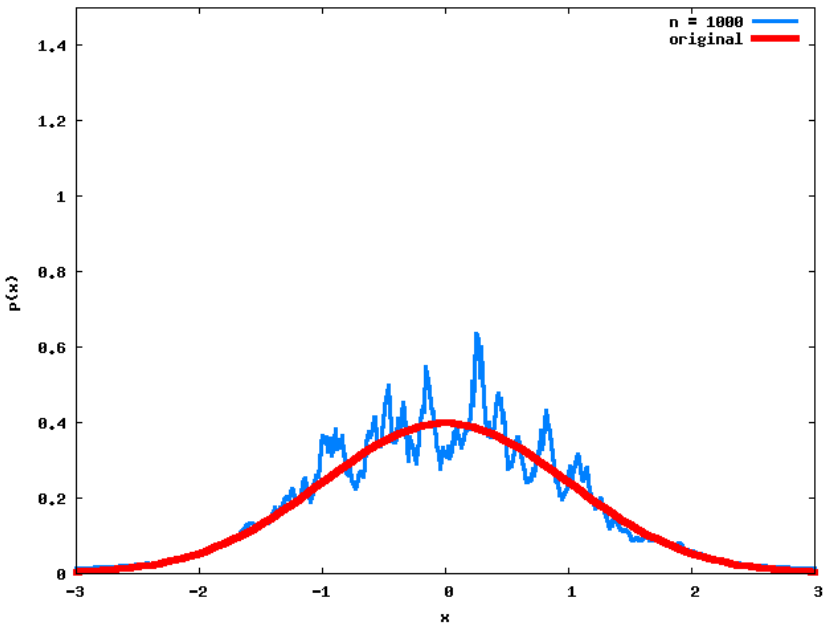
$$p(\mathbf{x}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\mathbf{x}^2}$$

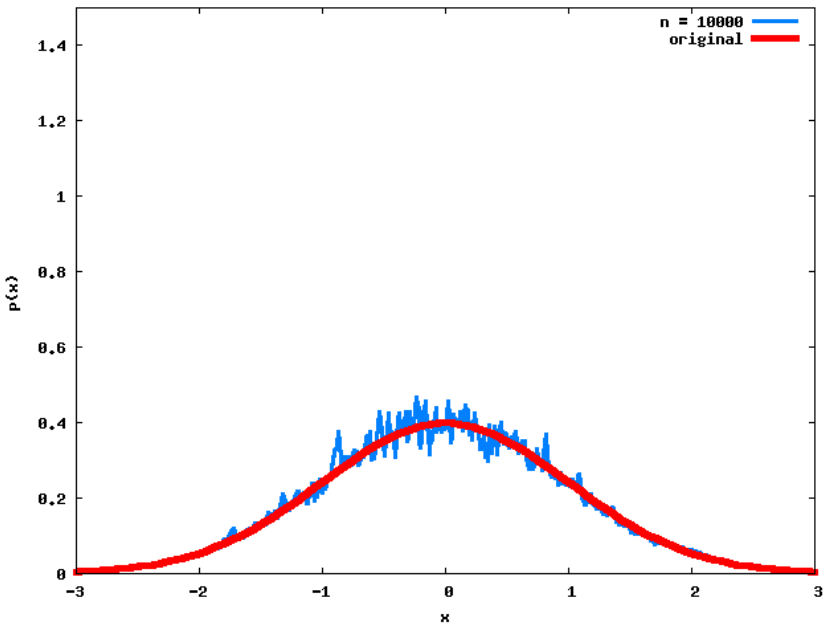




Example 1 with $k = \sqrt{n}$

$$p(\mathbf{x}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\mathbf{x}^2}$$





k-NN as a classifier

Let us suppose we have n_j examples in S of class y_j, such that $\sum_j n_j = n$. Suppose the hypersphere contains k_j examples of class y_j ($\sum_j k_j = k$). If we aim at classifying a new example $\mathbf{x}$ with its k nearest-neighbors, and applying the Bayes rule, we get:

$$h(\mathbf{x}) = arg\max_j \frac{\hat{p}(\mathbf{x}|y_j).\hat{p}(y_j)}{\hat{p}(\mathbf{x})}$$

k-NN as a classifier

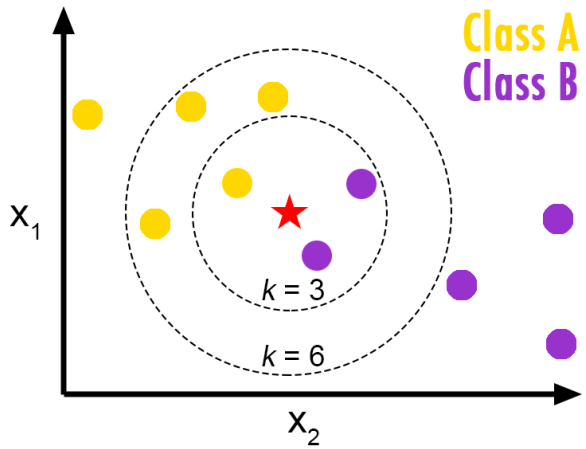
Let us suppose we have n_j examples in S of class y_j, such that $\sum_j n_j = n$. Suppose the hypersphere contains k_j examples of class y_j ($\sum_j k_j = k$). If we aim at classifying a new example $\mathbf{x}$ with its k nearest-neighbors, and applying the Bayes rule, we get:

$$h(\mathbf{x}) = arg\max_j \frac{\hat{p}(\mathbf{x}|y_j).\hat{p}(y_j)}{\hat{p}(\mathbf{x})} = arg\max_j \frac{\frac{k_j}{V_n \times n_j} \times \frac{n_j}{n}}{\frac{k}{n \times V_n}}$$

k-NN as a classifier

Let us suppose we have n_j examples in S of class y_j, such that $\sum_j n_j = n$. Suppose the hypersphere contains k_j examples of class y_j ($\sum_j k_j = k$). If we aim at classifying a new example $\mathbf{x}$ with its k nearest-neighbors, and applying the Bayes rule, we get:

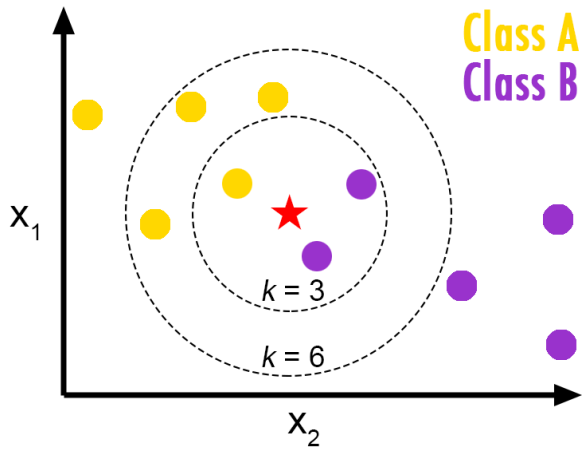
$$h(\mathbf{x}) = arg\max_j \frac{\hat{p}(\mathbf{x}|y_j).\hat{p}(y_j)}{\hat{p}(\mathbf{x})} = arg\max_j \frac{\frac{k_j}{V_n \times n_j} \times \frac{n_j}{n}}{\frac{k}{n \times V_n}} = arg\max_j \frac{k_j}{k}$$



k-nearest neighbors algorithm

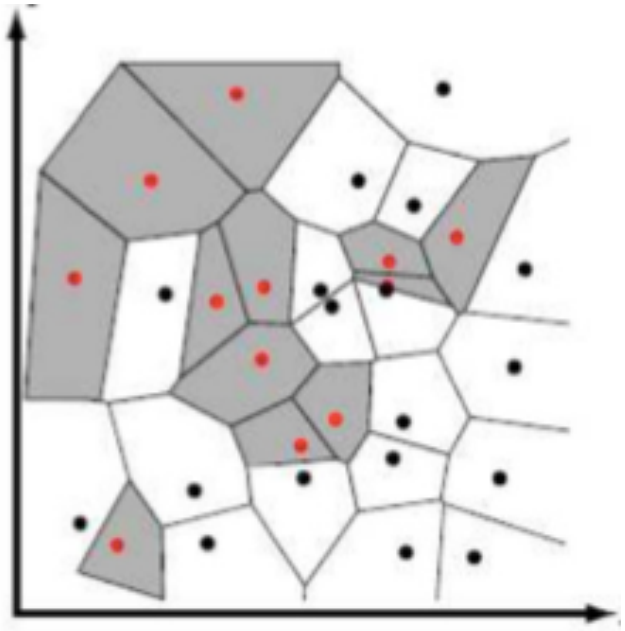
How to classify an unknown example $\mathbf{x}$ using S?

Input: $\mathbf{x}, S, d$
Output: class of $\mathbf{x}$
foreach $(\mathbf{x}', y') \in S$ **do**
  Compute the distance $d(\mathbf{x}', \mathbf{x})$;
Sort the n distances by increasing order;
Count the number of occurrences of each class y_j among the k nearest neighbors;
Assign to $\mathbf{x}$ the most frequent class;



Special case of the $(k = 1)$-nearest neighbor rule

Decision boundaries of the $(k = 1)$-nearest neighbor rule.



In a 2-dimensional space, the $(k = 1)$-nearest neighbor rule achieves a partition of the space in Voronoi cells (obtained from the perpendicular bisectors of Delaunay's triangles), each one labeled by the class of the example it contains.

Special case of the $(k = 1)$-nearest neighbor rule

Convergence properties of the $(k = 1)$-nearest neighbor rule

Theorem

Let $\mathbf{x}'$ *be the nearest neighbor of* $\mathbf{x}$,

$$\lim_{n \to \infty} p\left(d(x, x') > \epsilon\right) = 0, \forall \epsilon > 0$$

Special case of the $(k = 1)$-nearest neighbor rule

Convergence properties of the $(k = 1)$-nearest neighbor rule

Theorem

Let $\mathbf{x}'$ *be the nearest neighbor of* $\mathbf{x}$,

$$\lim_{n \to \infty} p\left(d(x, x') > \epsilon\right) = 0, \forall \epsilon > 0$$

Corrolary

If $n \to \infty$, $p(y_j|\mathbf{x}') \approx p(y_j|\mathbf{x})$.

Special case of the $(k = 1)$-nearest neighbor rule

Proof.

Let $\mathcal{P}$ be the probability that the hypersphere $s(\mathbf{x}, \epsilon)$ centered at $\mathbf{x}$ of radius ϵ does not contain any point of S:

Special case of the $(k = 1)$-nearest neighbor rule

Proof.

Let $\mathcal{P}$ be the probability that the hypersphere $s(\mathbf{x}, \epsilon)$ centered at $\mathbf{x}$ of radius ϵ does not contain any point of S:

$$\mathcal{P} = p(\mathbf{x_1} \notin s(\mathbf{x}, \epsilon), ..., \mathbf{x_n} \notin s(\mathbf{x}, \epsilon)) = \prod_{i=1}^{n} p(\mathbf{x_i} \notin s(\mathbf{x}, \epsilon))$$

Special case of the $(k = 1)$-nearest neighbor rule

Proof.

Let $\mathcal{P}$ be the probability that the hypersphere $s(\mathbf{x}, \epsilon)$ centered at $\mathbf{x}$ of radius ϵ does not contain any point of S:

$$\mathcal{P} = p(\mathbf{x_1} \notin s(\mathbf{x}, \epsilon), ..., \mathbf{x_n} \notin s(\mathbf{x}, \epsilon)) = \prod_{i=1}^{n} p(\mathbf{x_i} \notin s(\mathbf{x}, \epsilon))$$

$$= \prod_{i=1}^{n} (1 - p(\mathbf{x_i} \in s(\mathbf{x}, \epsilon))) = (1 - p_\epsilon)^n.$$

Special case of the $(k = 1)$-nearest neighbor rule

Proof.

Let $\mathcal{P}$ be the probability that the hypersphere $s(\mathbf{x}, \epsilon)$ centered at $\mathbf{x}$ of radius ϵ does not contain any point of S:

$$\mathcal{P} = p(\mathbf{x_1} \notin s(\mathbf{x}, \epsilon), ..., \mathbf{x_n} \notin s(\mathbf{x}, \epsilon)) = \prod_{i=1}^{n} p(\mathbf{x_i} \notin s(\mathbf{x}, \epsilon))$$

$$= \prod_{i=1}^{n} (1 - p(\mathbf{x_i} \in s(\mathbf{x}, \epsilon))) = (1 - p_\epsilon)^n.$$

Since the nearest neighbor $\mathbf{x}'$ of $\mathbf{x}$ is also outside of the sphere, we get

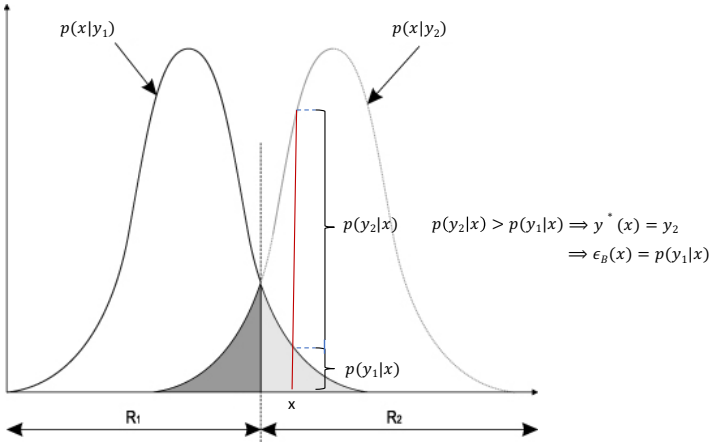
$$p(d(\mathbf{x}, \mathbf{x}') > \epsilon) = (1 - p_\epsilon)^n$$

$$\text{therefore, } \lim_{n \to \infty} p(d(\mathbf{x}, \mathbf{x}') > \epsilon) = \lim_{n \to \infty} (1 - p_\epsilon)^n = 0$$

Theorem

The generalization error ϵ_{1NN} of the 1-nearest neighbor rule is bounded by twice the (optimal) bayes error ϵ_B.

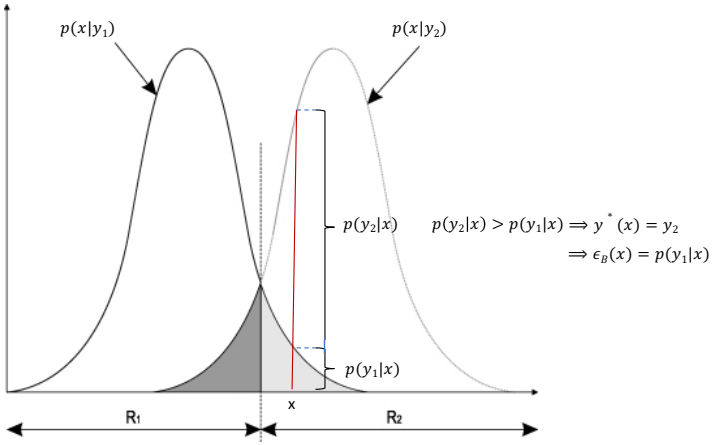
$$\epsilon_{1NN} \leq 2\epsilon_B$$



Proof.

Bayes error:

$$\forall \mathbf{x} \in \mathcal{X},\ \epsilon_B(\mathbf{x}) \quad = \quad \min\{p(y_1|\mathbf{x}), p(y_2|\mathbf{x})\} = p(y_{min}|\mathbf{x})$$



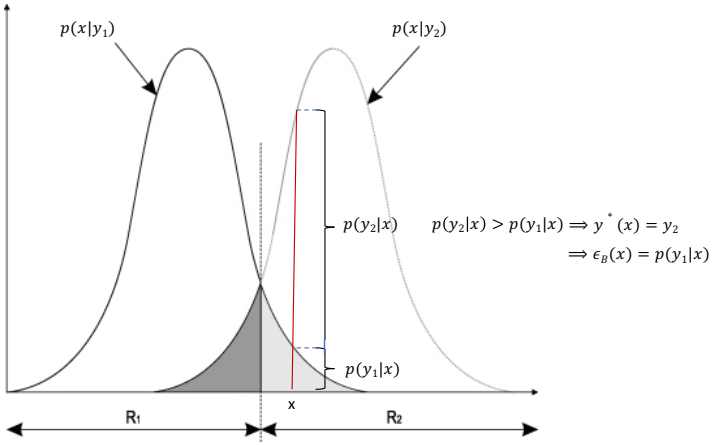
Proof.

Bayes error:

$$\forall \mathbf{x} \in \mathcal{X},\ \epsilon_B(\mathbf{x}) \quad = \quad \min\{p(y_1|\mathbf{x}), p(y_2|\mathbf{x})\} = p(y_{min}|\mathbf{x})$$

1NN error:

$$\epsilon_{1NN}(\mathbf{x}) \quad = \quad p(y_1|\mathbf{x})p(y_2|\mathbf{x}') + p(y_2|\mathbf{x})p(y_1|\mathbf{x}')$$



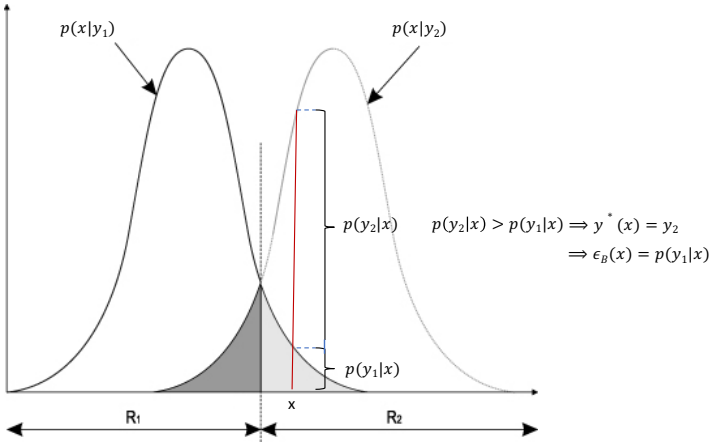
Proof.

Bayes error:

$$\forall \mathbf{x} \in \mathcal{X},\ \epsilon_B(\mathbf{x}) = \min\{p(y_1|\mathbf{x}), p(y_2|\mathbf{x})\} = p(y_{min}|\mathbf{x})$$

1NN error:

$$\begin{aligned} \epsilon_{1NN}(\mathbf{x}) &= p(y_1|\mathbf{x})p(y_2|\mathbf{x}') + p(y_2|\mathbf{x})p(y_1|\mathbf{x}') \\ \text{(if } n \text{ is large)} &\approx p(y_1|\mathbf{x})p(y_2|\mathbf{x}) + p(y_2|\mathbf{x})p(y_1|\mathbf{x}) \end{aligned}$$



Proof.

Bayes error:

$$\forall \mathbf{x} \in \mathcal{X},\ \epsilon_B(\mathbf{x}) \quad = \quad \min\{p(y_1|\mathbf{x}), p(y_2|\mathbf{x})\} = p(y_{min}|\mathbf{x})$$

1NN error:

$$\begin{aligned} \epsilon_{1NN}(\mathbf{x}) &= p(y_1|\mathbf{x})p(y_2|\mathbf{x}') + p(y_2|\mathbf{x})p(y_1|\mathbf{x}') \\ \text{(if } n \text{ is large)} \quad &\approx p(y_1|\mathbf{x})p(y_2|\mathbf{x}) + p(y_2|\mathbf{x})p(y_1|\mathbf{x}) \\ &= 2p(y_{min}|\mathbf{x})(1 - p(y_{min}|\mathbf{x})) \leq 2p(y_{min}|\mathbf{x}) = 2\epsilon^*(\mathbf{x}) \end{aligned}$$

We can deduce that:

In the discrete case

$$\epsilon_B = \sum_{x \in \mathcal{X}} p(y_{min}|\mathbf{x}).p(\mathbf{x})$$

$$\epsilon_{1NN} = \sum_{x \in \mathcal{X}} \left(p(y_1|\mathbf{x})p(y_2|\mathbf{x}') + p(y_2|\mathbf{x})p(y_1|\mathbf{x}')\right).p(\mathbf{x}) \leq 2\epsilon^*$$

In the continuous case

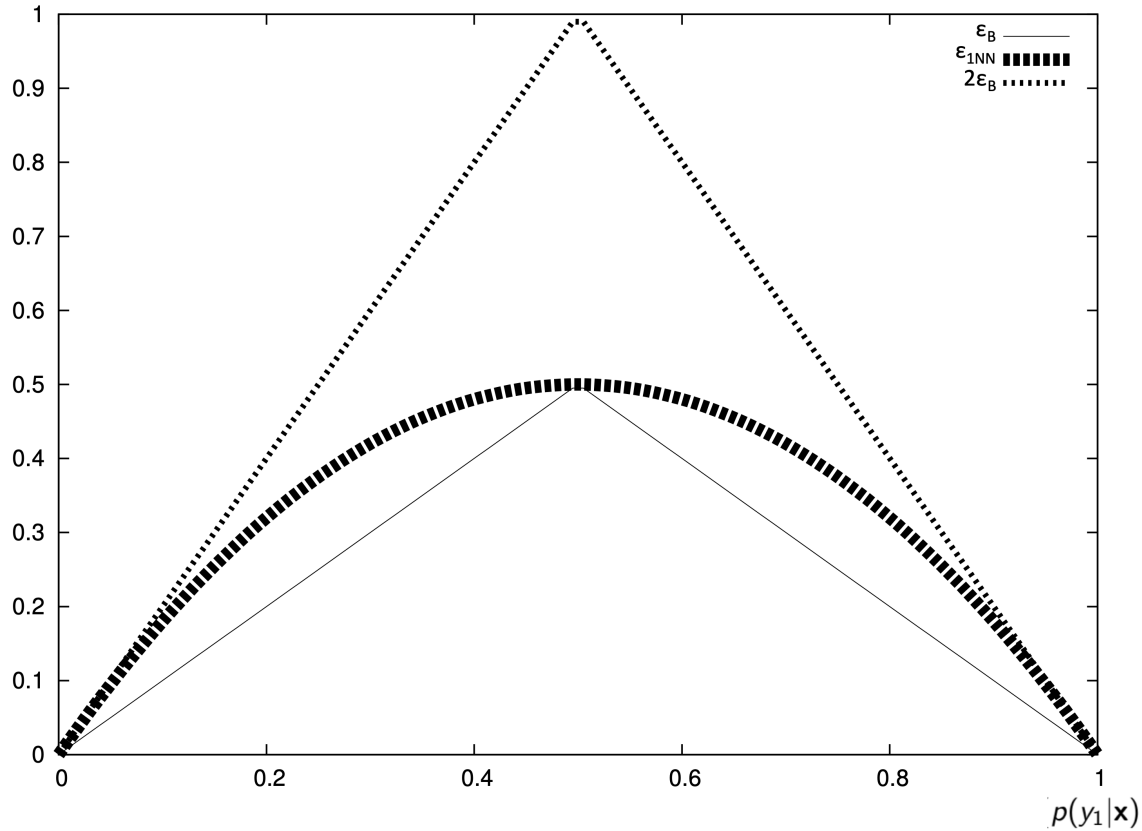
$$\epsilon_B = \int_{x \in \mathcal{X}} f(y_{min}|\mathbf{x}).f(\mathbf{x})d\mathbf{x}$$

$$\epsilon_{1NN} = \int_{x \in \mathcal{X}} \left(f(y_1|\mathbf{x})f(y_2|\mathbf{x}') + f(y_2|\mathbf{x})f(y_1|\mathbf{x}')\right).f(\mathbf{x})\, d\mathbf{x} \leq 2\epsilon^*$$

Graphically

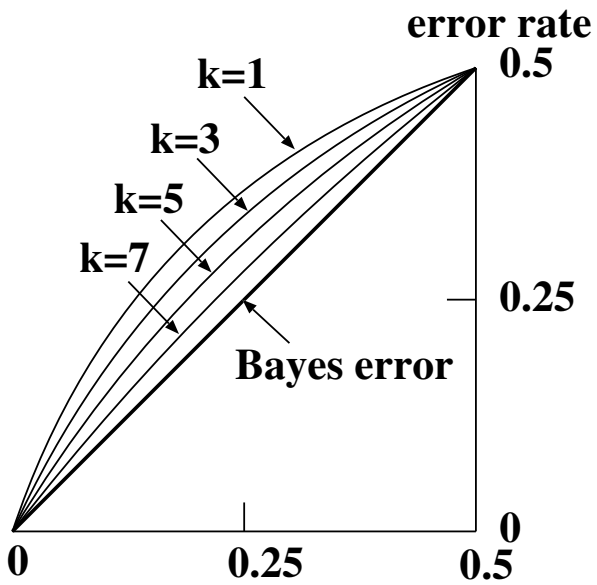
$$\epsilon_B(\mathbf{x}) = \min\{p(y_1|\mathbf{x}), p(y_2|\mathbf{x})\} = p(y_{min}|\mathbf{x})$$

$$\epsilon_{1NN}(\mathbf{x}) = 2p(y_{min}|\mathbf{x})(1 - p(y_{min}|\mathbf{x})) \leq 2\epsilon^*(\mathbf{x})$$



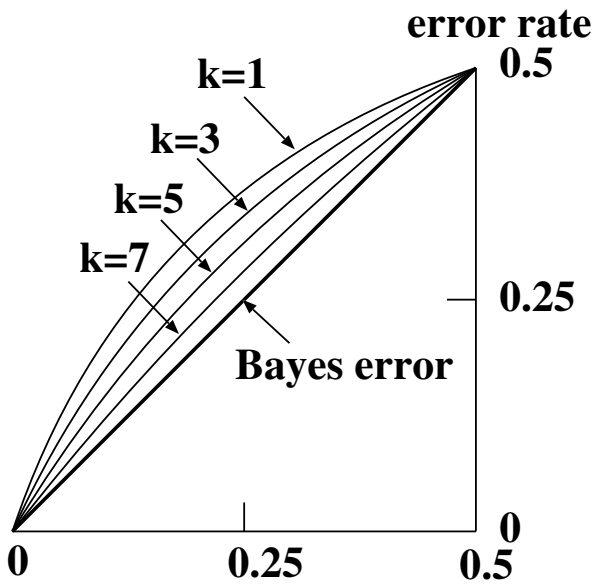
Effect of k on the estimation quality

$$\epsilon_B \leq \epsilon_{kNN} \leq \epsilon_{(k-1)NN} \leq \ldots \leq \epsilon_{1NN} \leq 2\epsilon_B$$



Effect of k on the estimation quality

$$\epsilon_B \leq \epsilon_{kNN} \leq \epsilon_{(k-1)NN} \leq \ldots \leq \epsilon_{1NN} \leq 2\epsilon_B$$



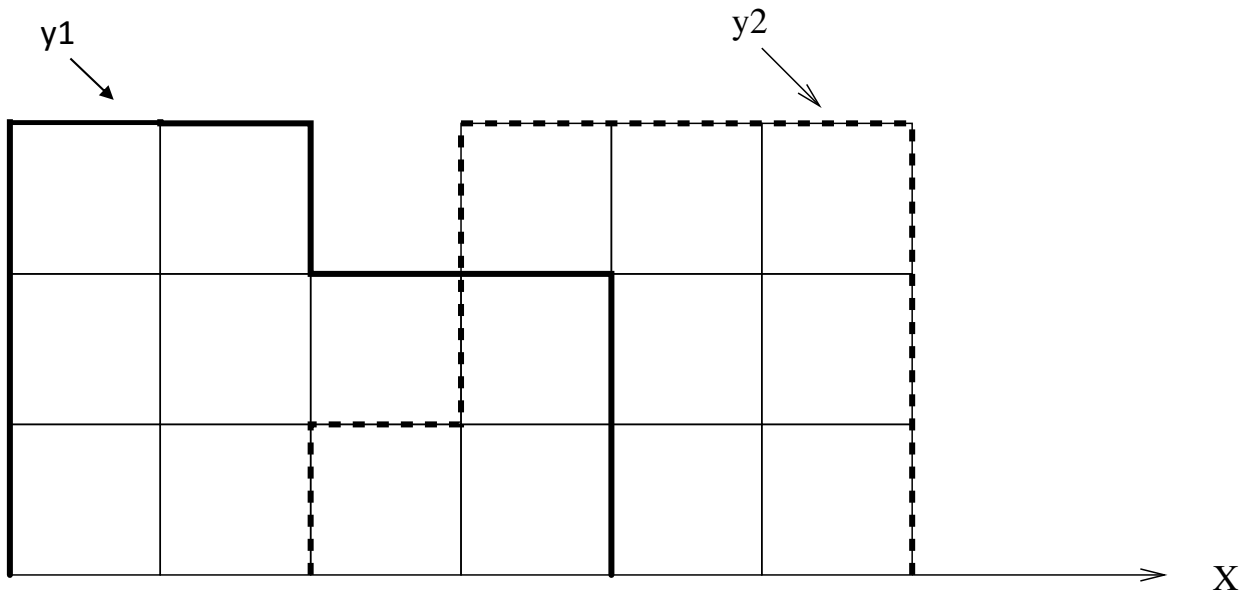
Reminder

The previous property holds only asymptotically ($n \to \infty$)

$\to$ possible compromise between k and n: $k = \sqrt{\frac{n}{|\mathcal{Y}|}}$.

Exercise

Let us consider the following discrete probability functions of two classes y_1 (solid line) and y_2 (dashed line).

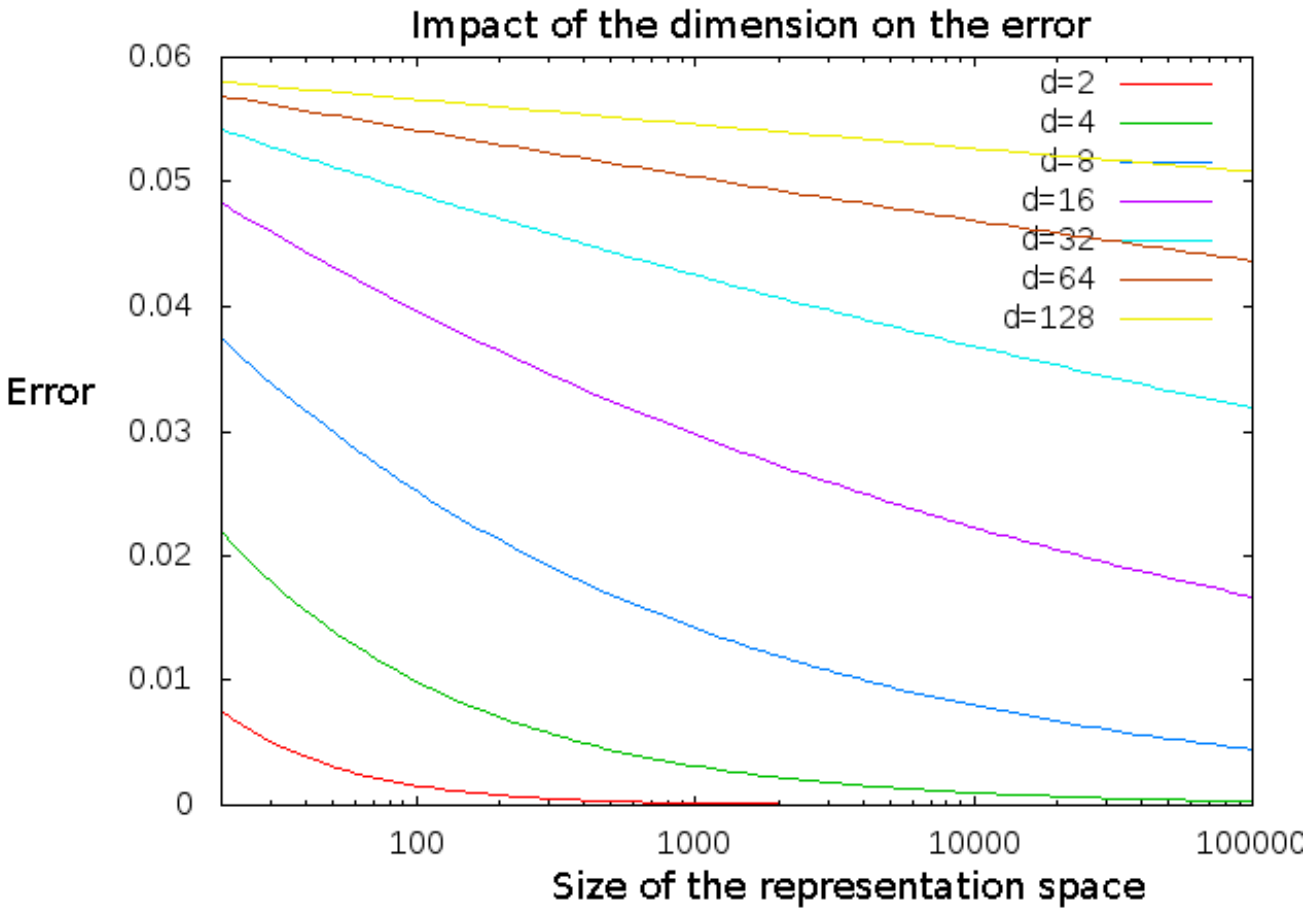


Question

Verify that the previous theorem holds

Curse of dimensionality

Fukunaga [1987] studied the effect of the representation dimension on the NN behavior (curse of dimensionality).



Problems of k-NN

- To prevent the kNN algorithm from facing the curse of dimensionality, dimensionality reduction is required (PCA, feature selection, metric learning, etc.).

Problems of k-NN

- To prevent the kNN algorithm from facing the curse of dimensionality, dimensionality reduction is required (PCA, feature selection, metric learning, etc.).
- To converge, the k-nearest neighbor algorithm requires a large number of training examples.
- However, a large number of training examples implies a large space/time complexity.

Problems of k-NN

- To prevent the kNN algorithm from facing the curse of dimensionality, dimensionality reduction is required (PCA, feature selection, metric learning, etc.).
- To converge, the k-nearest neighbor algorithm requires a large number of training examples.
- However, a large number of training examples implies a large space/time complexity.

Two strategies to overcome these problems:

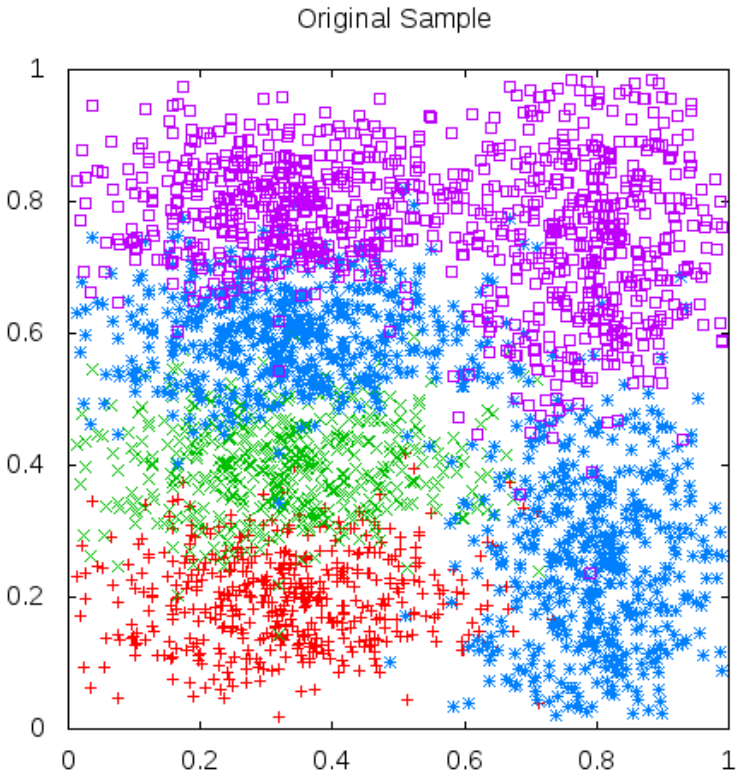
- Reduce the size of S while keeping the most relevant examples (*e.g.* the condensed nearest neighbor rule (Hart 1968)).
- Simplify the calculation of the nearest-neighbor.

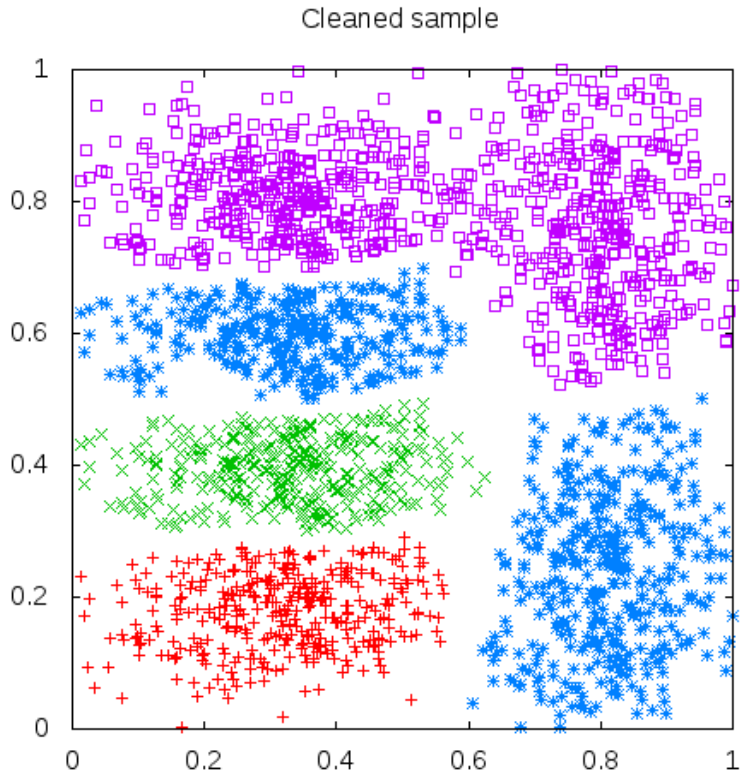
Data reduction techniques

Preliminary step: remove from S the outliers and the examples of the bayes error region.

Input: S
Output: $S_{cleaned}$
Split randomly S into two subsets S_1 and S_2;
while *no stabilization of* S_1 *and* S_2 **do**
    Classify S_1 with S_2 using the 1-NN rule;
    Remove from S_1 the misclassified instances;
    Classify S_2 with the new set S_1 using the 1-NN rule;
    Remove from S_2 the misclassified instances;
$S_{cleaned} = S_1 \cup S_2$;

Illustration



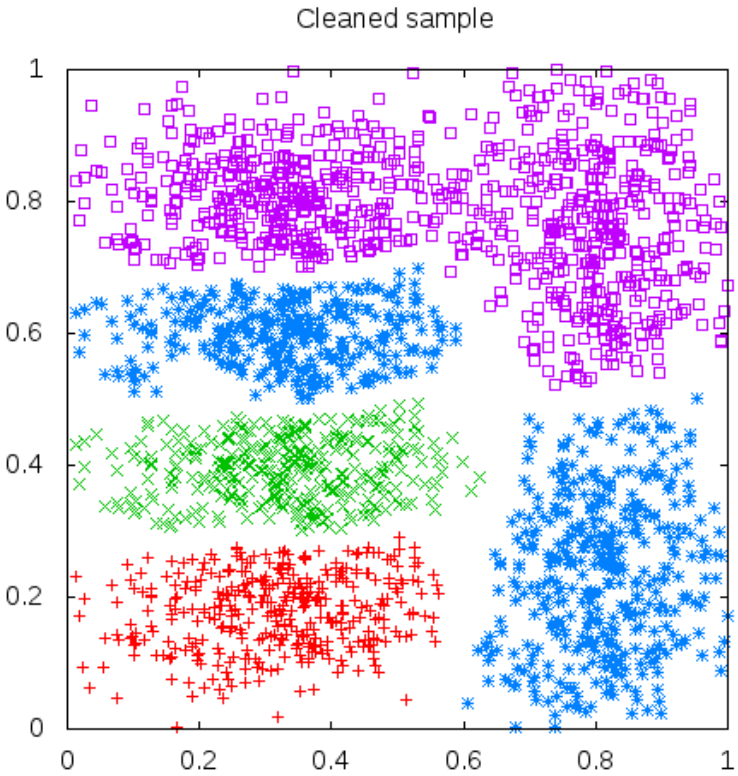


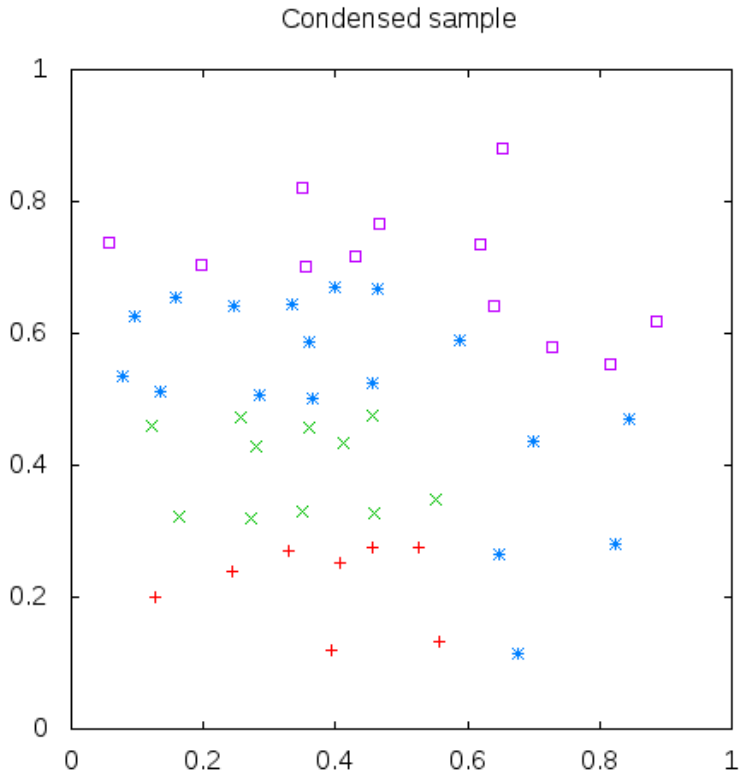
The condensed nearest neighbor rule (CNN)

Second step: remove the irrelevant examples.

Input: S
Output: STORAGE
STORAGE $\leftarrow \emptyset$; DUSTBIN $\leftarrow \emptyset$;
Draw randomly a training example from S and put it in STORAGE;
while *no stabilization of STORAGE* **do**
  foreach $\mathbf{x_i} \in S$ **do**
    if $\mathbf{x_i}$ *is correctly classified with STORAGE using the* 1*-NN rule*
    then
      DUSTBIN $\leftarrow \mathbf{x_i}$
    else
      STORAGE $\leftarrow \mathbf{x_i}$
return *STORAGE*;

Illustration





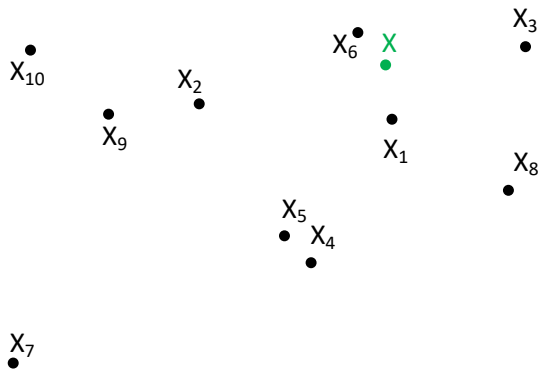
How to speed-up the nearest-neighbor calculation?

X: new query to classify

X': current NN of X at a distance d_{min}

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}

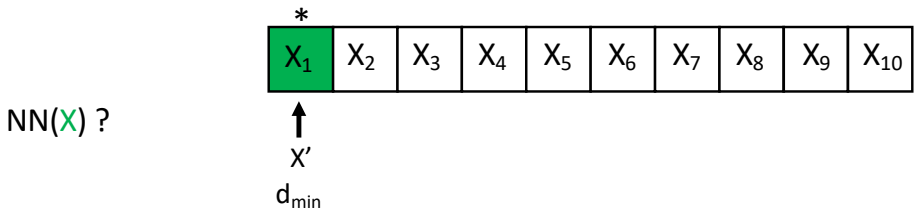
NN(X) ?

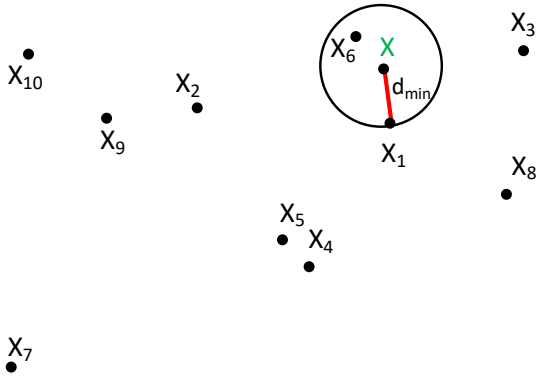


How to speed-up the nearest-neighbor calculation?

X: new query to classify

X': current NN of X at a distance d_{min}

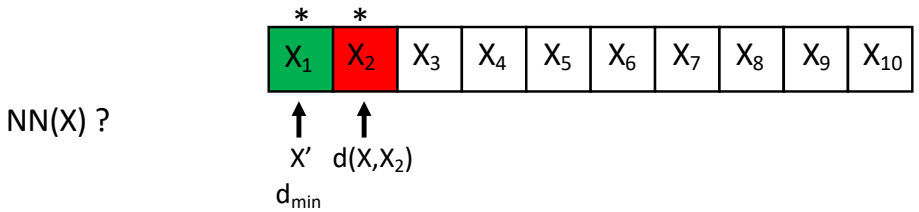


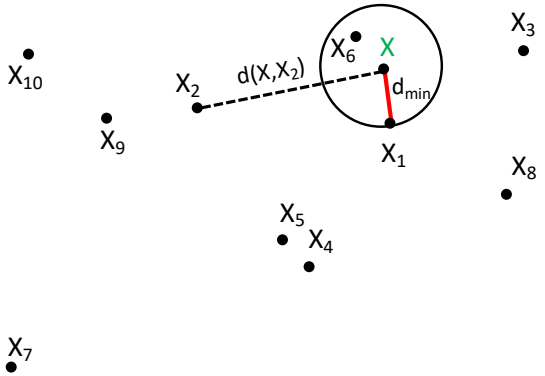


How to speed-up the nearest-neighbor calculation?

X: new query to classify

X': current NN of X at a distance d_{min}

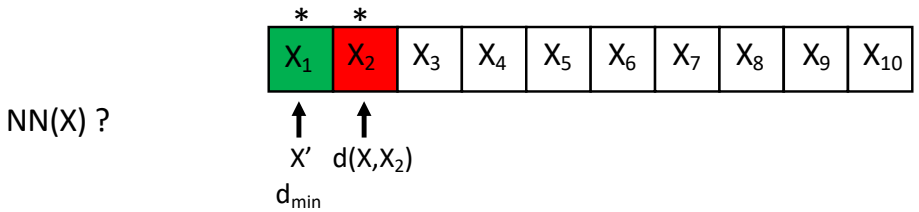




How to speed-up the nearest-neighbor calculation?

X: new query to classify

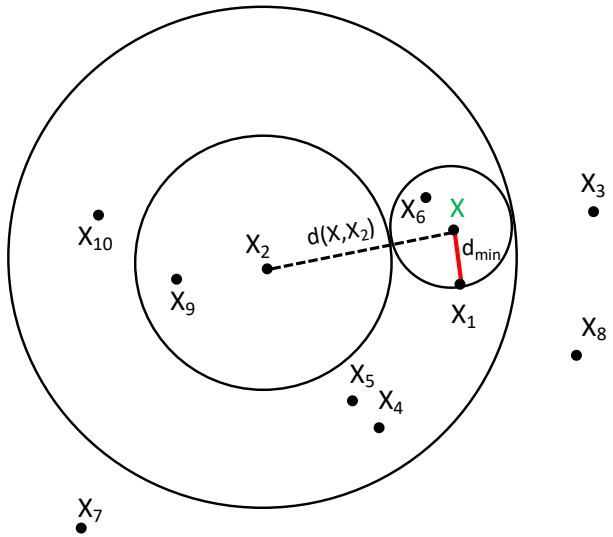
X': current NN of X at a distance d_{min}



Since $d(X,X_2) > d_{min}$ the training points that are

-in the sphere centered at X_2 of radius $d(X,X_2)-d_{min}$
-out of the sphere centered at X_2 of radius $d(X,X_2)+d_{min}$

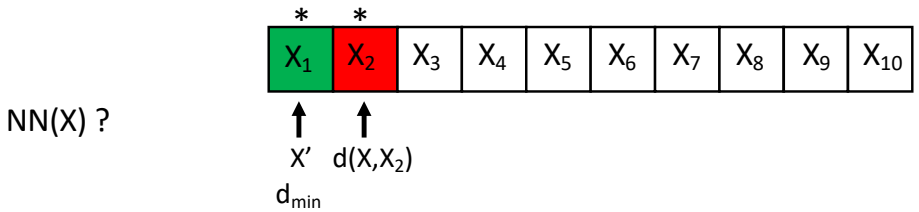
cannot be NN(X)



How to speed-up the nearest-neighbor calculation?

X: new query to classify

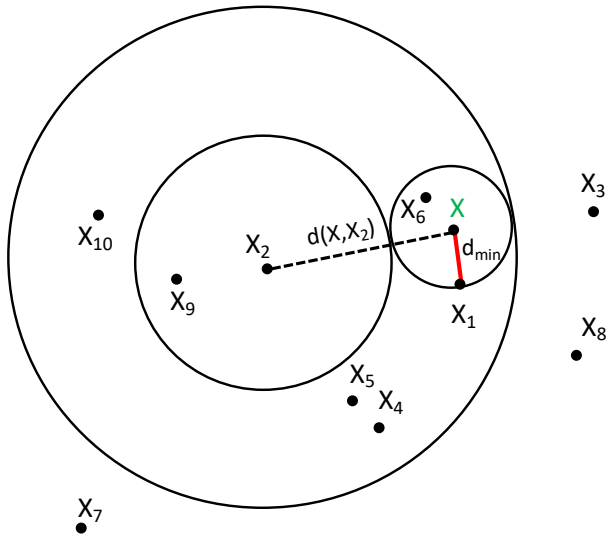
X': current NN of X at a distance d_{min}



Since $d(X,X_2) > d_{min}$ the training points that are

-in the sphere centered at X_2 of radius $d(X,X_2)-d_{min}$
-out of the sphere centered at X_2 of radius $d(X,X_2)+d_{min}$

cannot be NN(X)



Explanation

$$d(X_9, X_2) \leq d(X, X_2) - d_{min}$$
$$\Longleftrightarrow \quad d(X_9, X_2) + d_{min} \leq d(X, X_2) \qquad (1)$$

By the triangle inequality:

$$d(X, X_2) \leq d(X, X_9) + d(X_9, X_2) \qquad (2)$$

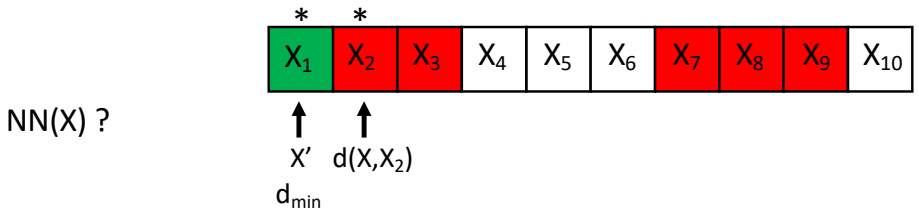
From (1) and (2), we get:

$$d(X_9, X_2) + d_{min} \leq d(X, X_9) + d(X_9, X_2)$$
$$\Longleftrightarrow \quad d_{min} \leq d(X, X_9)$$

How to speed-up the nearest-neighbor calculation?

X: new query to classify

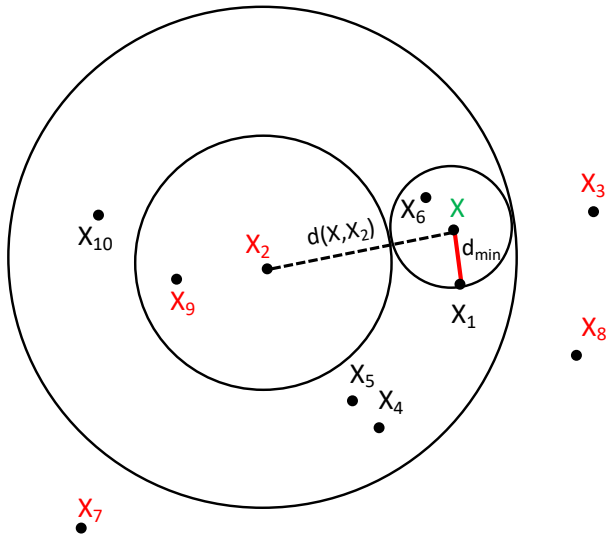
X': current NN of X at a distance d_{min}



Since $d(X,X_2) > d_{min}$ the training points that are

-in the sphere centered at X_2 of radius $d(X,X_2)-d_{min}$
-out of the sphere centered at X_2 of radius $d(X,X_2)+d_{min}$

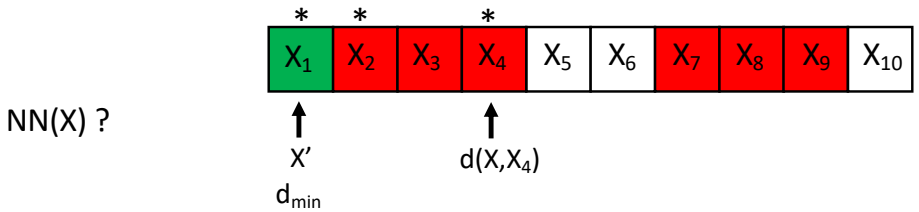
cannot be NN(X)

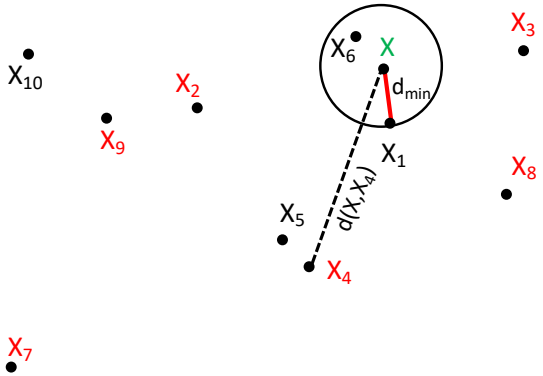


How to speed-up the nearest-neighbor calculation?

X: new query to classify

X': current NN of X at a distance d_{min}

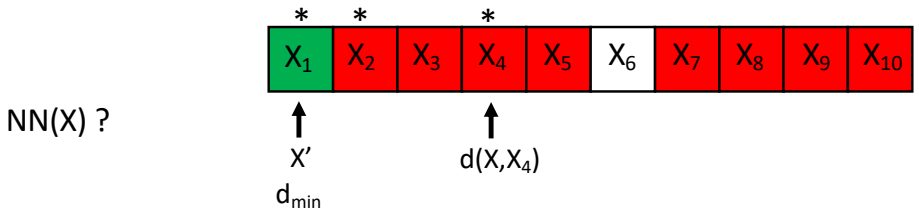




How to speed-up the nearest-neighbor calculation?

X: new query to classify

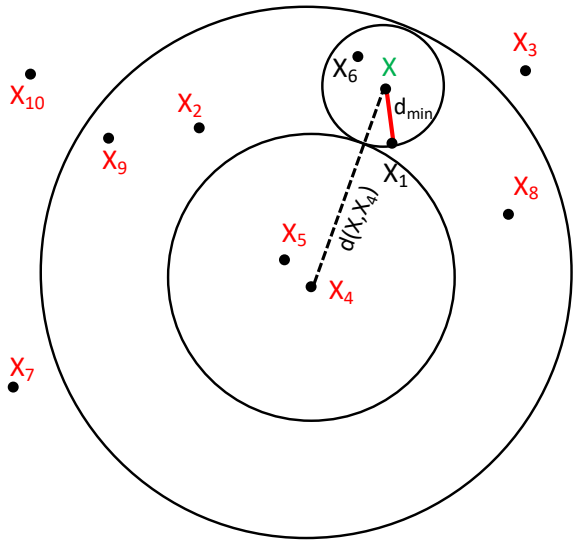
X': current NN of X at a distance d_{min}



Since $d(X,X_4) > d_{min}$ the training points that are

-in the sphere centered at X_4 of radius $d(X,X_4)-d_{min}$
-out of the sphere centered at X_2 of radius $d(X,X_2)+d_{min}$

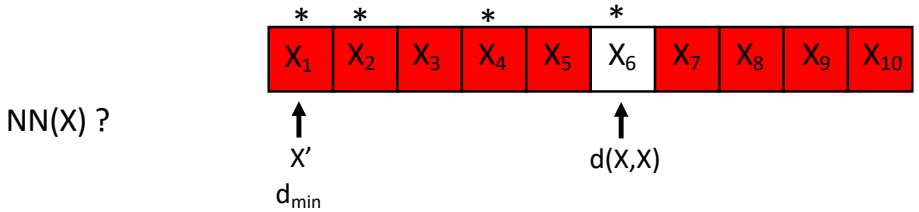
cannot be NN(X)

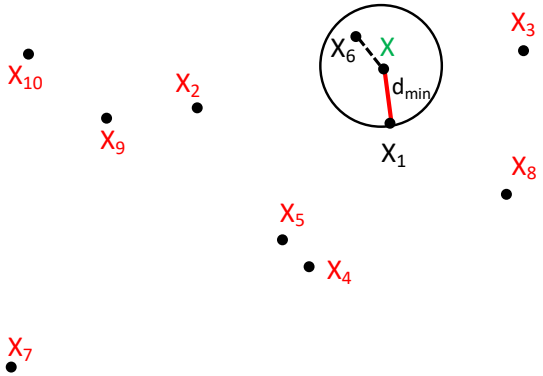


How to speed-up the nearest-neighbor calculation?

X: new query to classify

X': current NN of X at a distance d_{min}

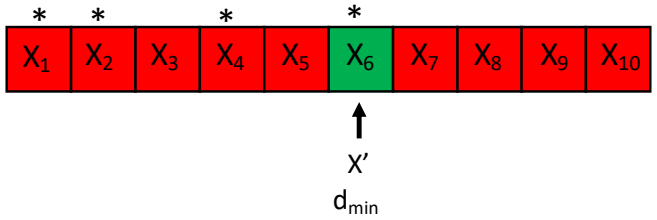


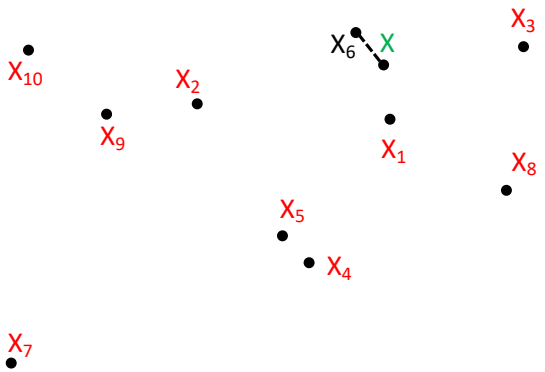


How to speed-up the nearest-neighbor calculation?

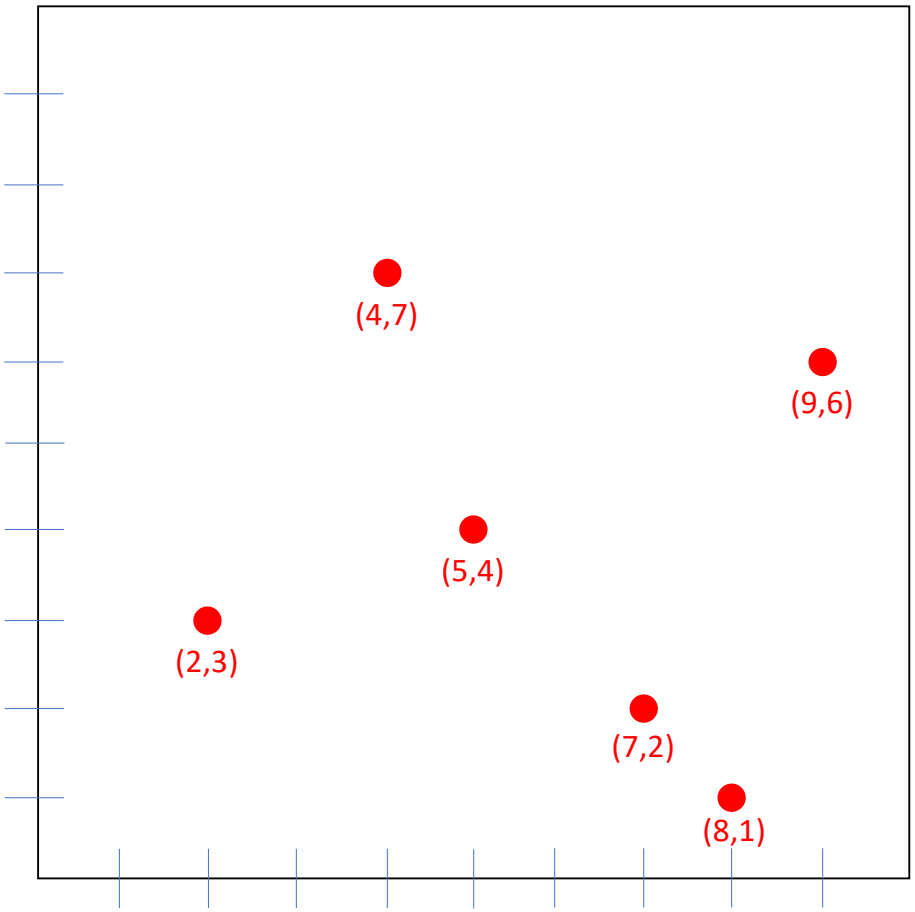
X: new query to classify

X': current NN of X at a distance d_{min}

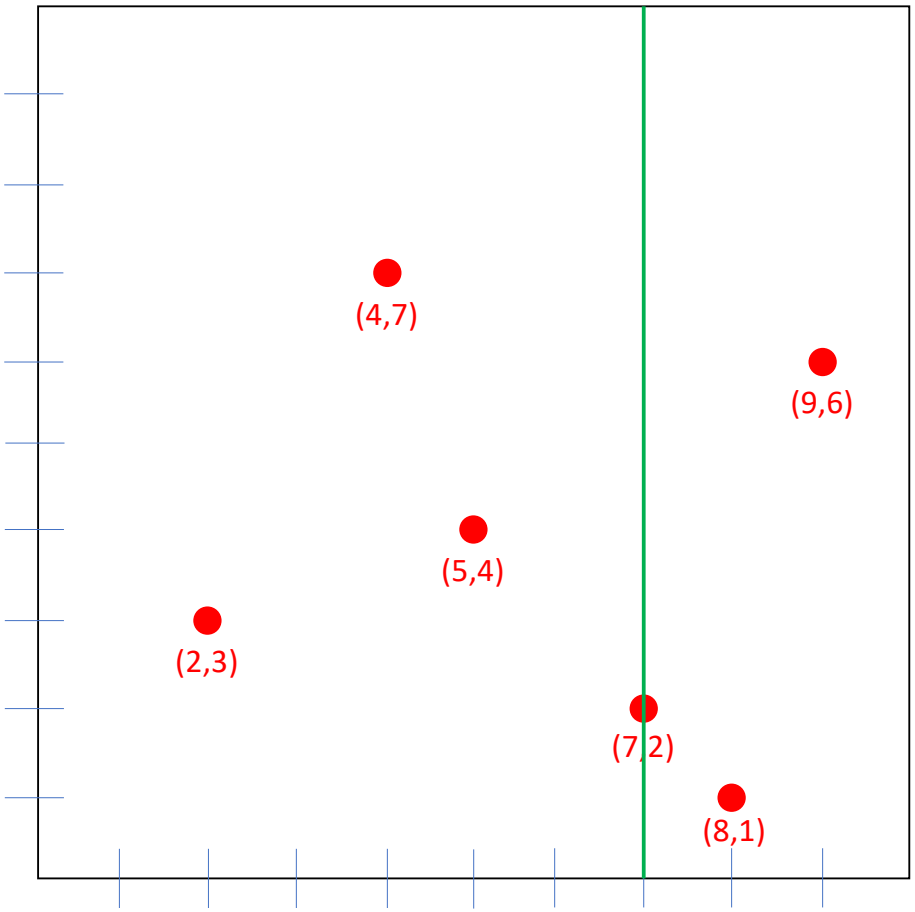




kNN and KD-trees

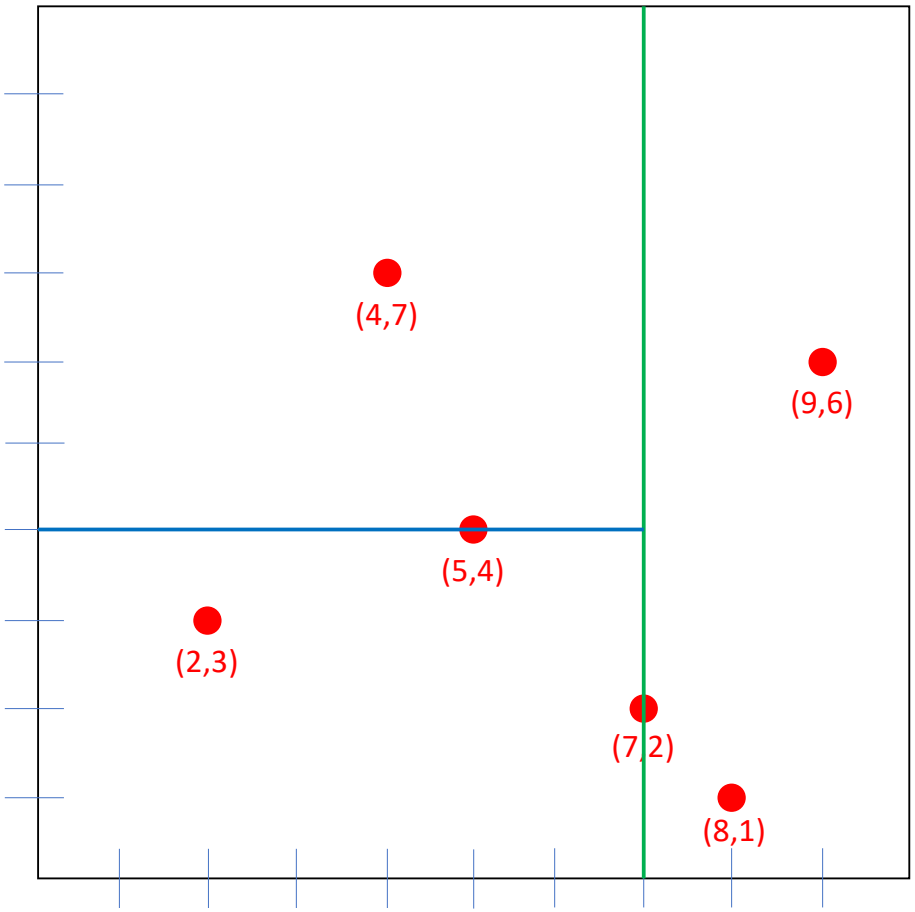


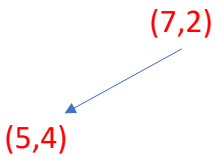
kNN and KD-trees



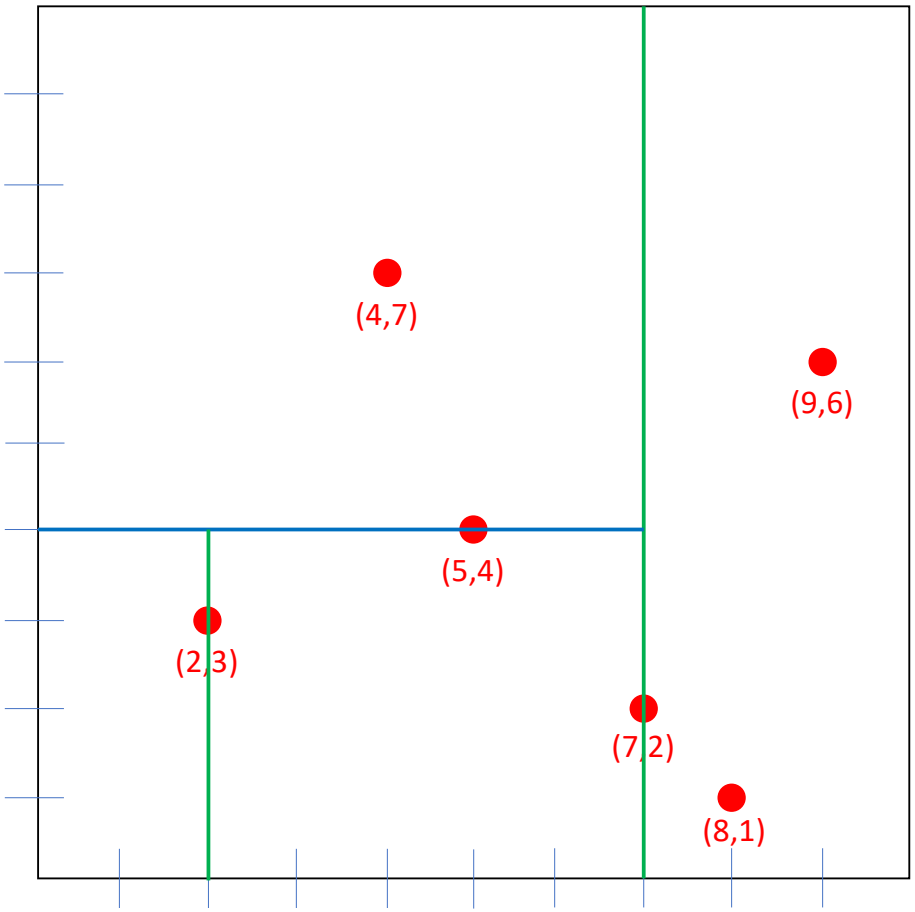
(7,2)

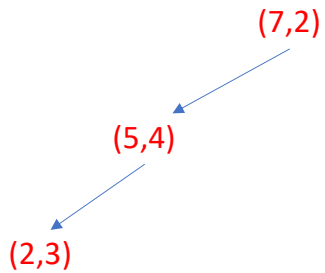
kNN and KD-trees



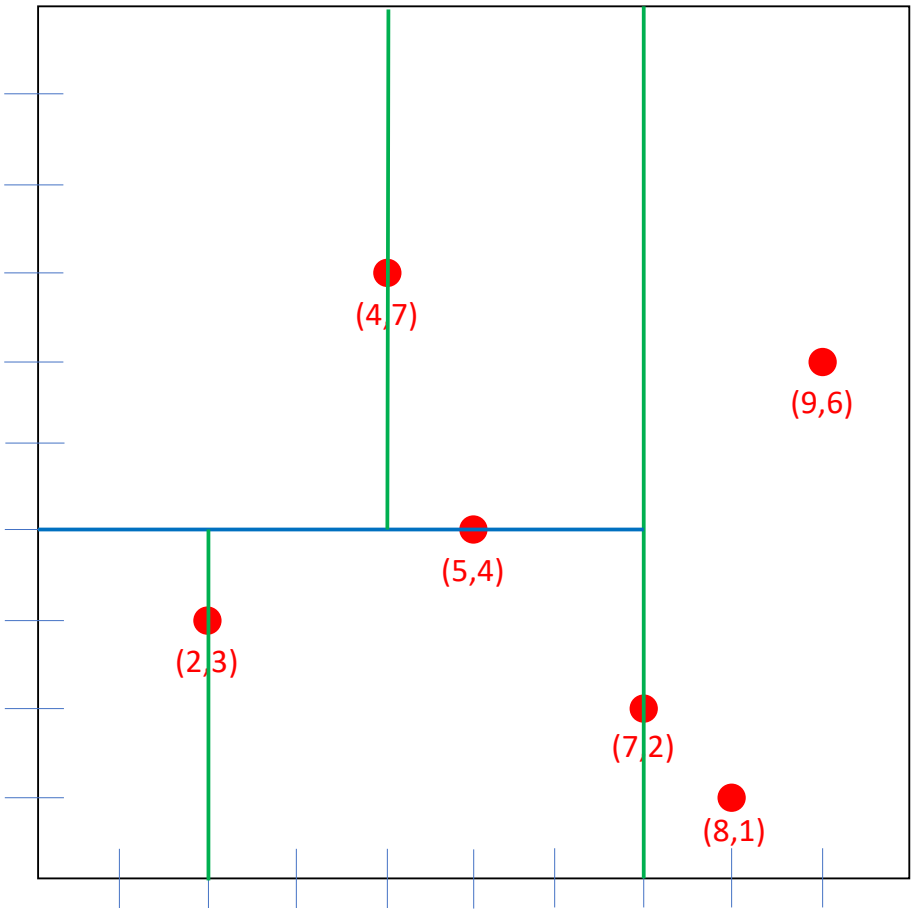


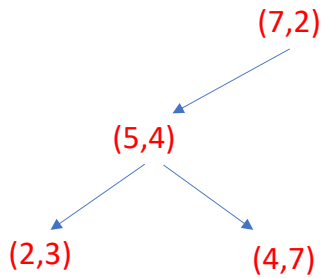
kNN and KD-trees



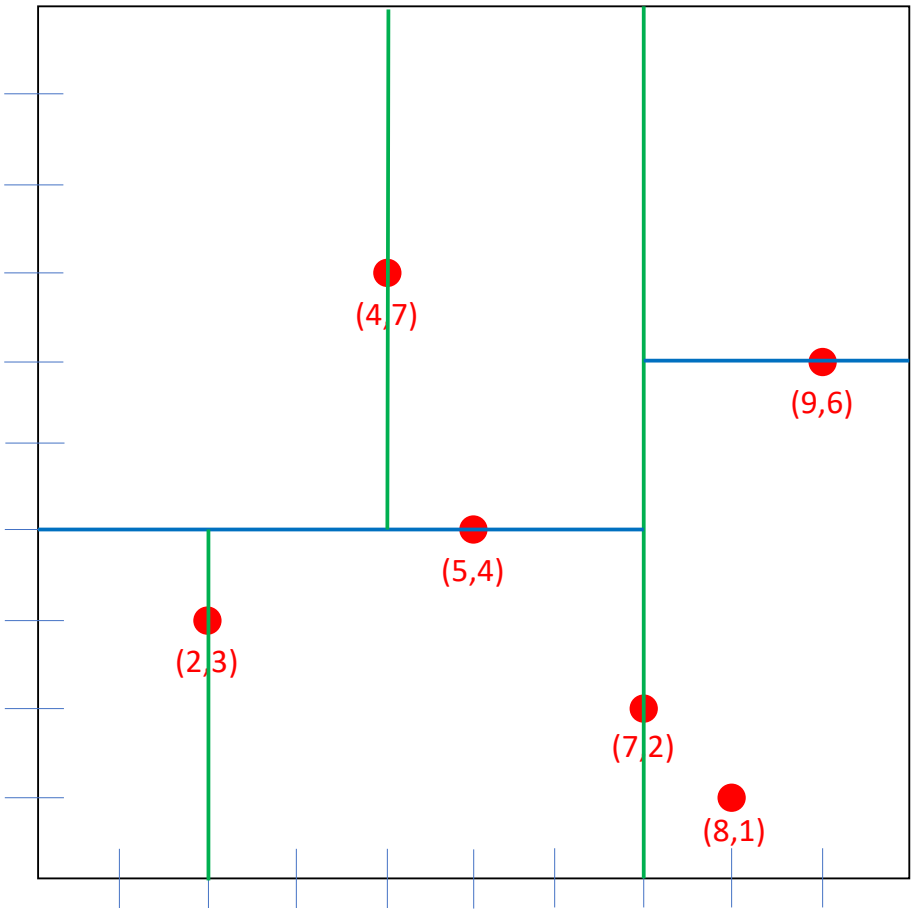


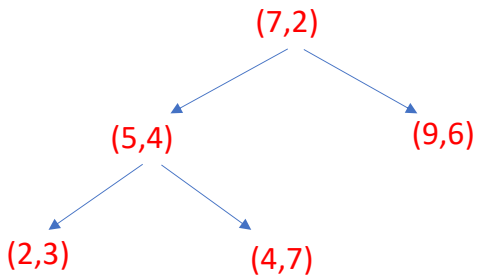
kNN and KD-trees



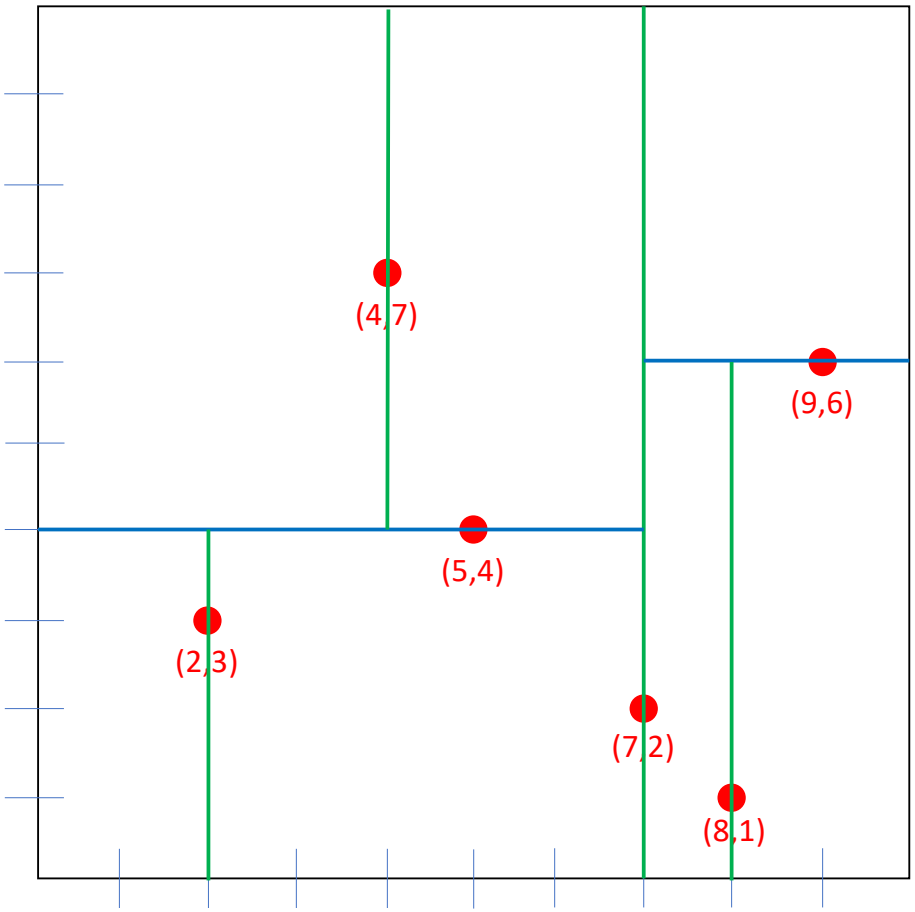


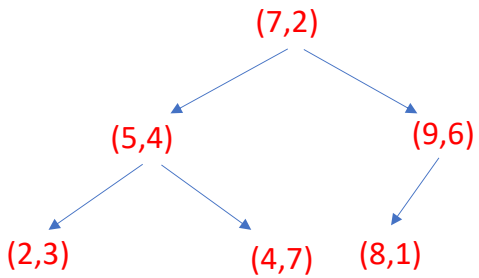
kNN and KD-trees



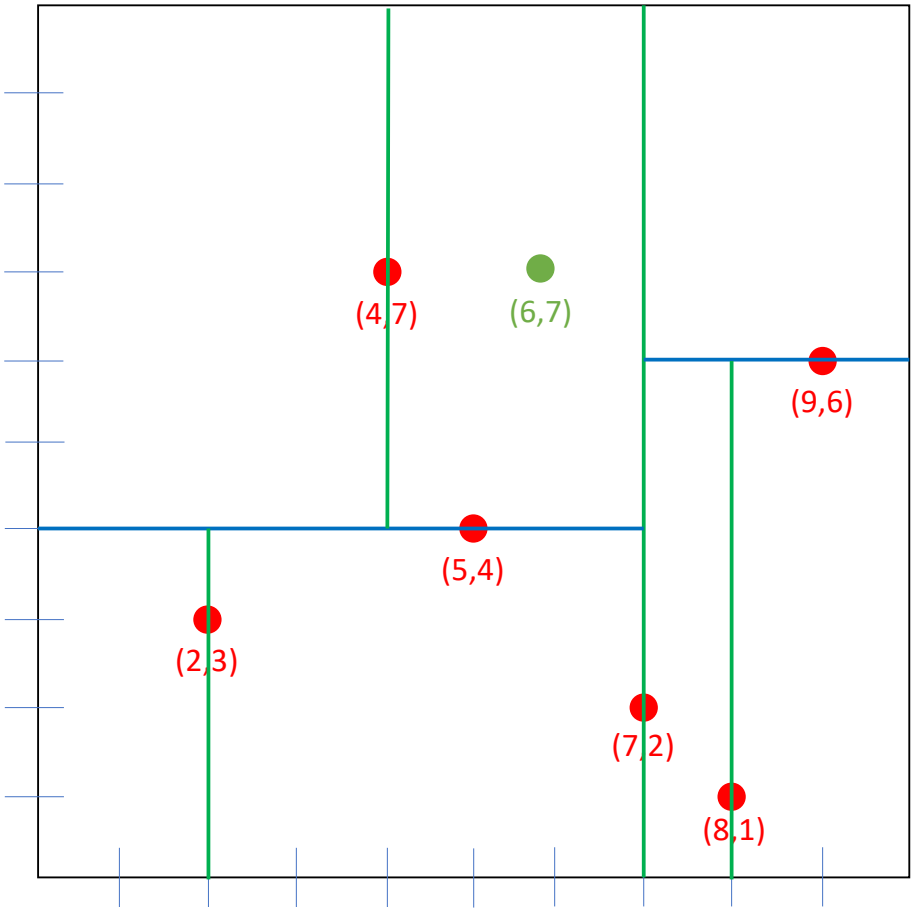


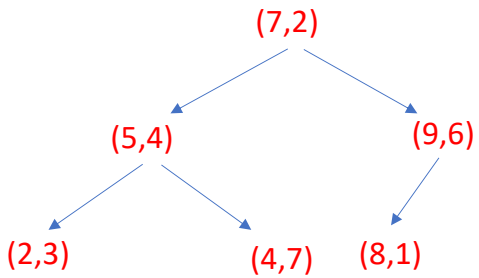
kNN and KD-trees





kNN and KD-trees

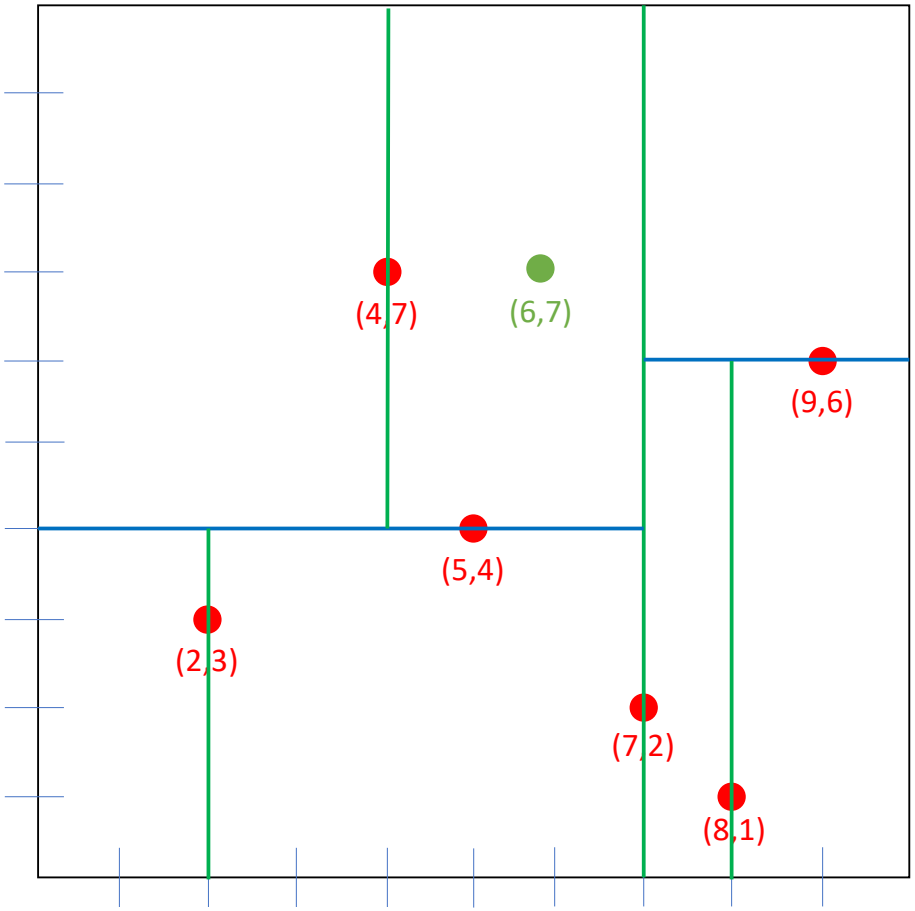




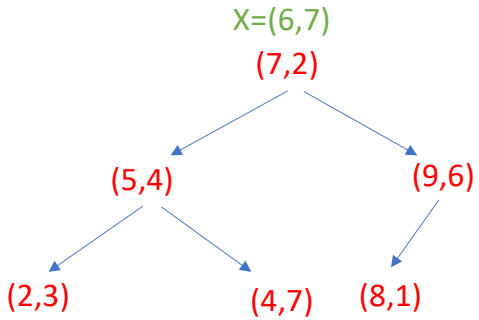
How to search the NN for a new query point?

X=(6,7)

kNN and KD-trees

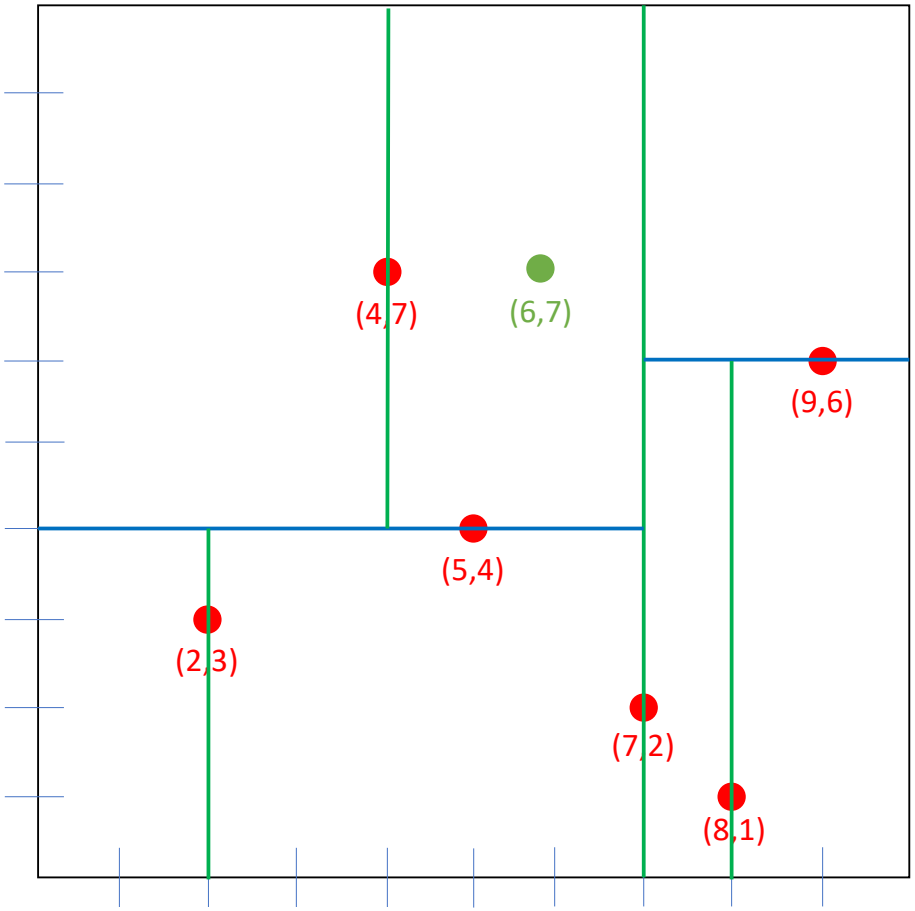


How to search the NN for a new query point?

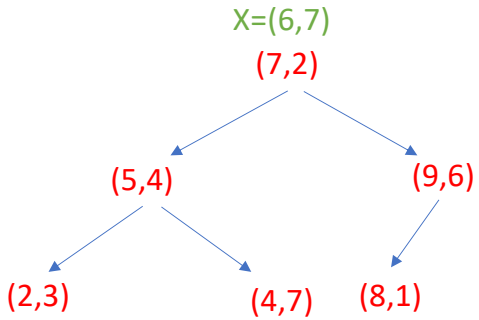


Best distance = infinity
Current best = none

kNN and KD-trees



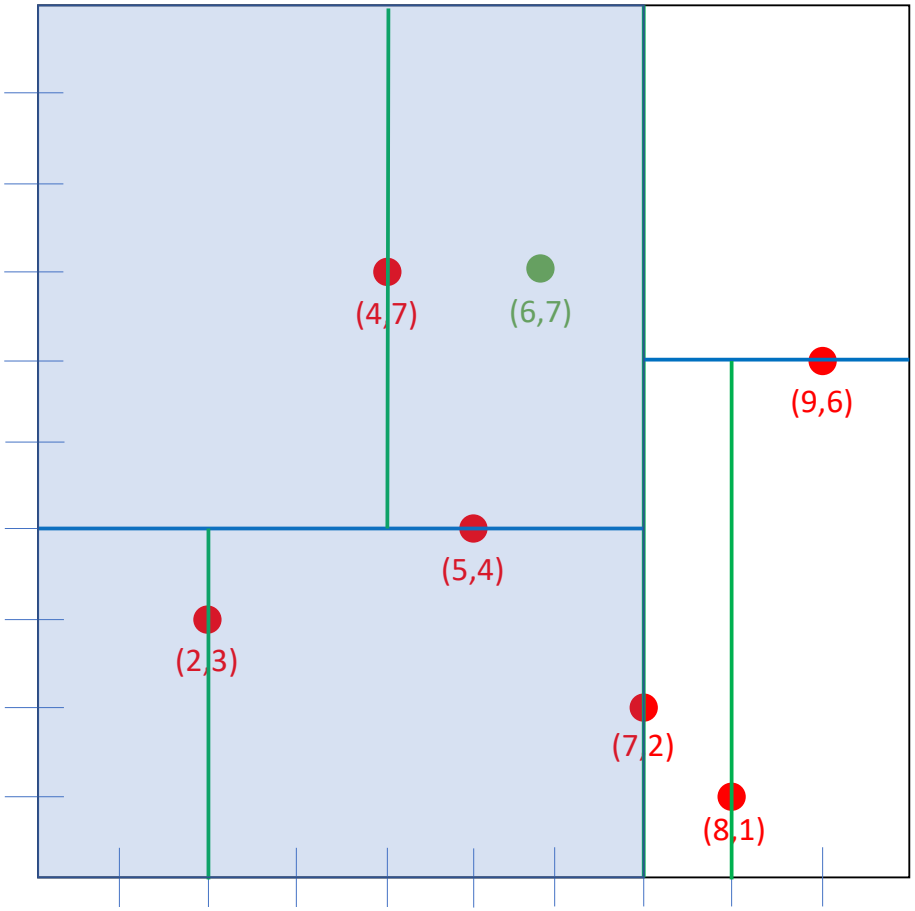
How to search the NN for a new query point?



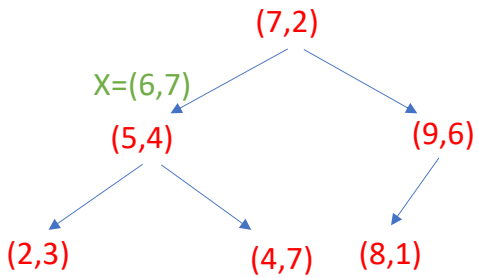
Best distance = 5.09
Current best = (7,2)

$$d([6,7];[7,2]) = 5.09$$

kNN and KD-trees



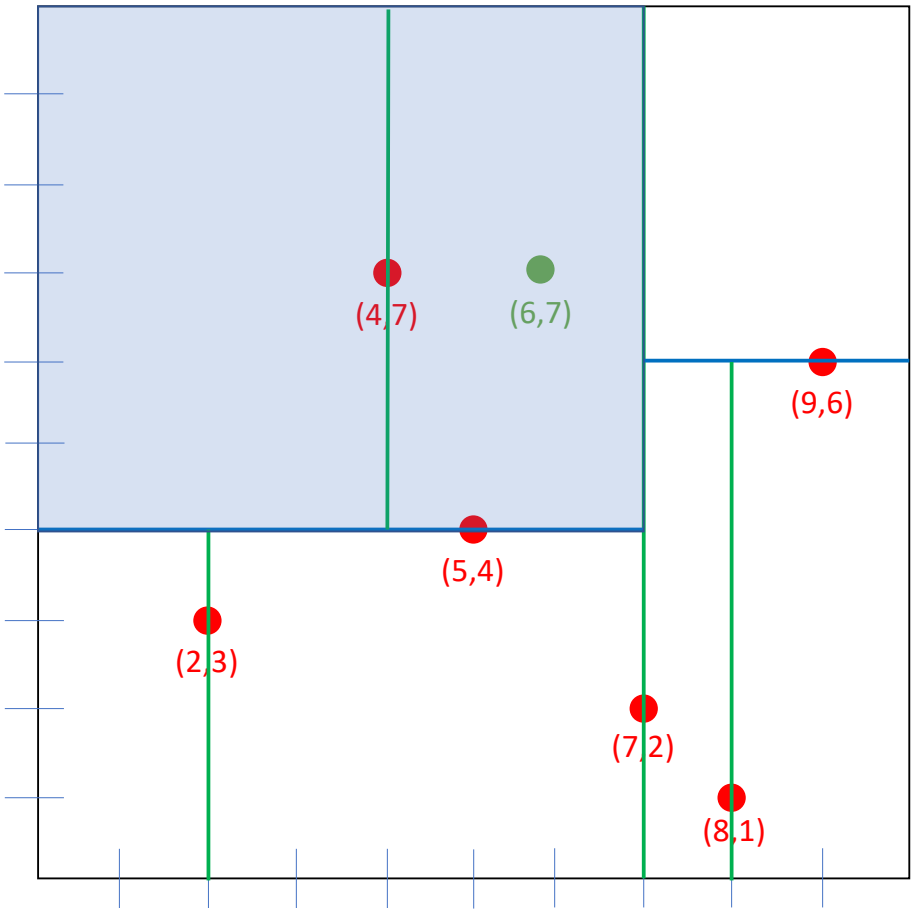
How to search the NN for a new query point?



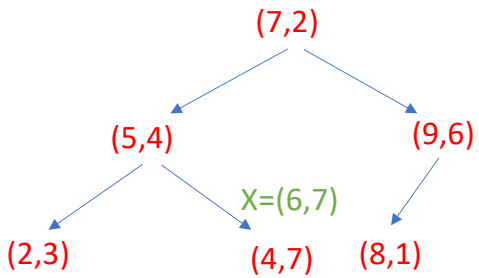
Best distance = 3.16
Current best = (5,4)

$$d([6,7];[5,4]) = 3.16$$

kNN and KD-trees



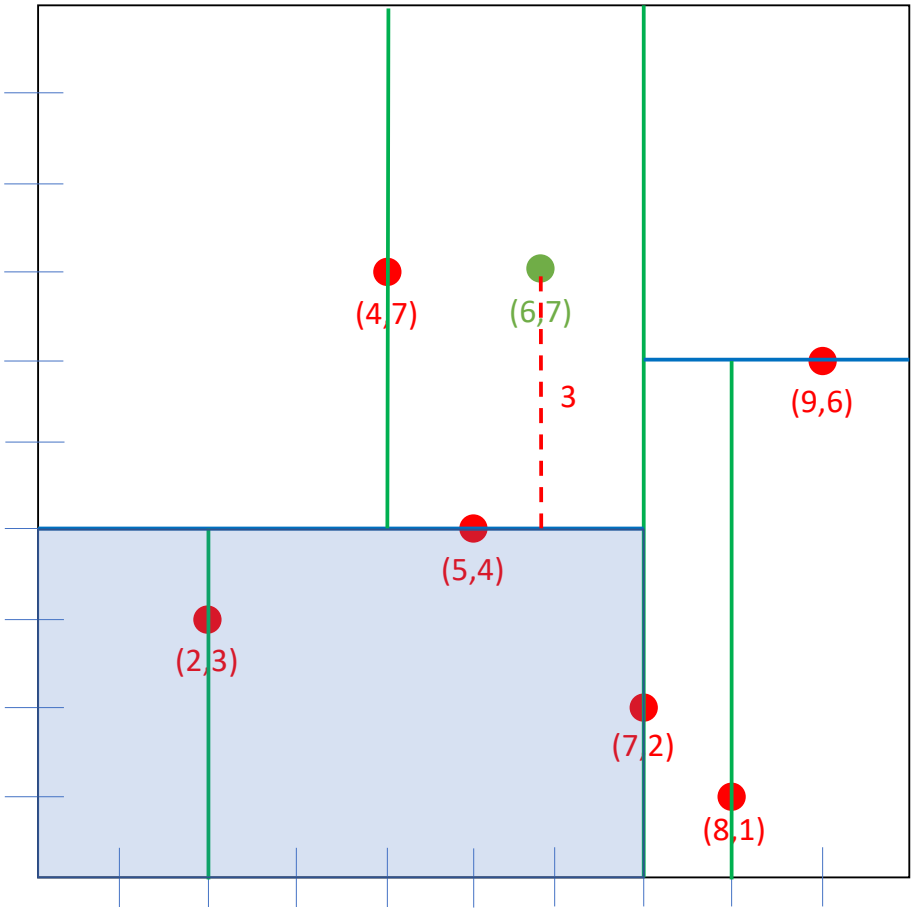
How to search the NN for a new query point?



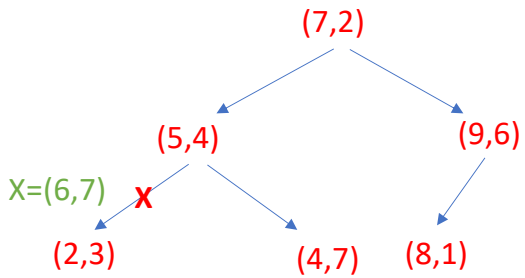
Best distance = 2
Current best = (4,7)

$$d([6,7];[4,7]) = 2$$

kNN and KD-trees



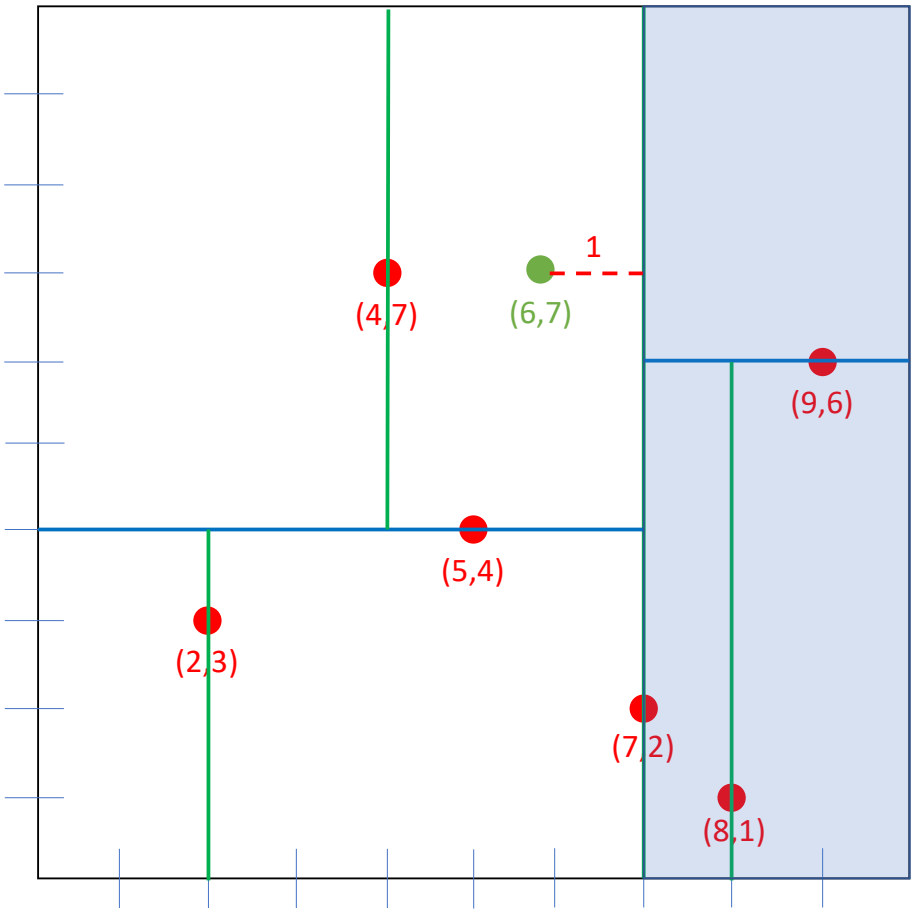
How to search the NN for a new query point?



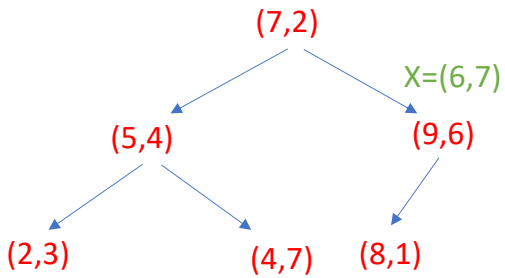
Best distance = 2
Current best = (4,7)

$$d([6,7];[4,7]) = 2$$

kNN and KD-trees



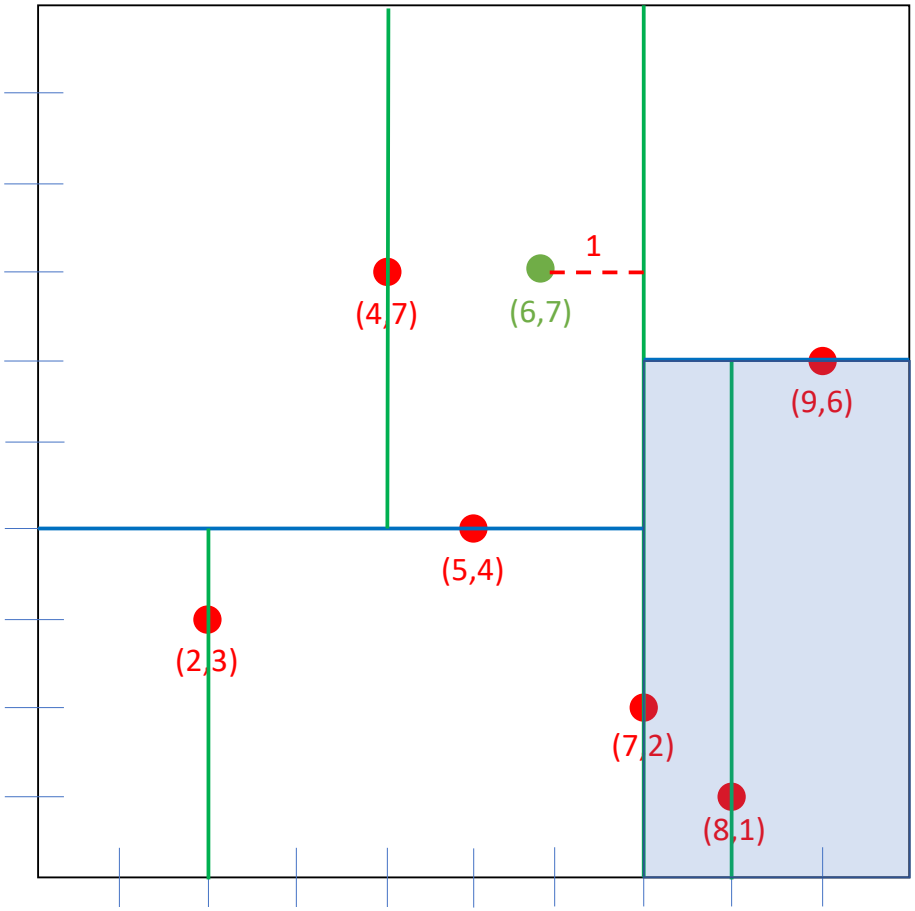
How to search the NN for a new query point?



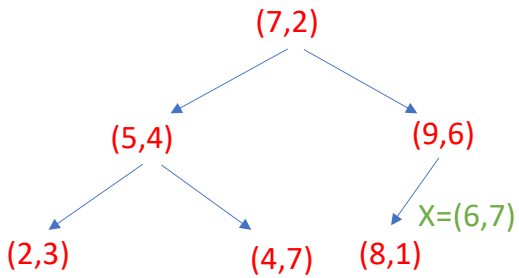
Best distance = 2
Current best = (4,7)

$$d([6,7];[9,6]) = 3.16$$

kNN and KD-trees



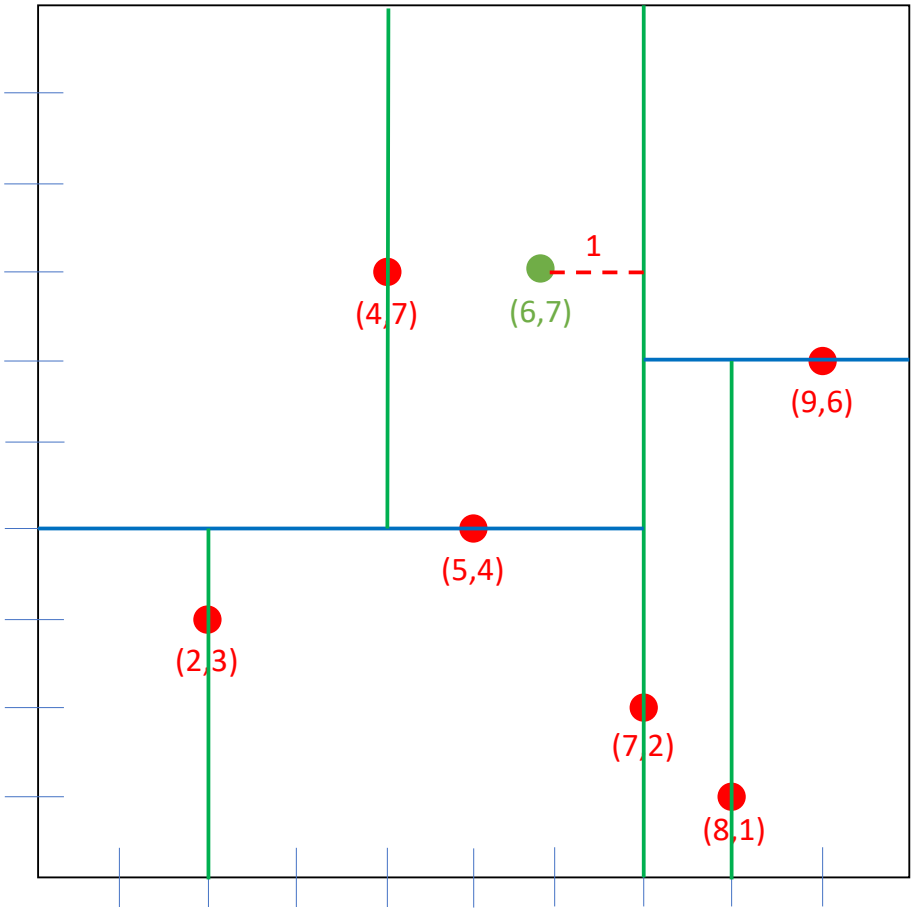
How to search the NN for a new query point?



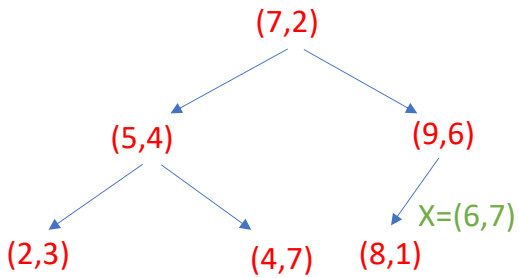
Best distance = 2
Current best = (4,7)

$$d([6,7];[8,1]) = 6.32$$

kNN and KD-trees



How to search the NN for a new query point?



Best distance = 2
Current best = (4,7)

$d([6,7];[8,1]) = 6.32$

Conclusion

- With a sufficiently large number of training examples, a NN classifier is able to converge towards very complex target functions.
- It is simple and theoretically well founded.
- There exist several solutions to overcome its algorithmic complexity issues (time and space).

Back to the Bayes Classifier

Bayes Classifier

Let $\mathbf{x}$ be a d-dimensional feature vector $\mathbf{x} = (x_1, ..., x_d)$.

$$y^*(\mathbf{x}) = \arg\max_c p(y_c|\mathbf{x}) = \arg\max_c p(y_c|x_1, ..., x_d).$$

that corresponds to

$$y^*(\mathbf{x}) = \arg\max_c p(\mathbf{x}|y_c).p(y_c) = \arg\max_c p(x_1, ..., x_d|y_c).p(y_c).$$

Back to the Bayes Classifier

Bayes Classifier

Let $\mathbf{x}$ be a d-dimensional feature vector $\mathbf{x} = (x_1, ..., x_d)$.

$$y^*(\mathbf{x}) = \arg\max_c p(y_c|\mathbf{x}) = \arg\max_c p(y_c|x_1, ..., x_d).$$

that corresponds to

$$y^*(\mathbf{x}) = \arg\max_c p(\mathbf{x}|y_c).p(y_c) = \arg\max_c p(x_1, ..., x_d|y_c).p(y_c).$$

Naive Bayes Classifier

The **naive** Bayes classifier assumes that the value of a particular feature is independent of the value of any other feature, given the class variable.

$$\hat{y}(\mathbf{x}) = \hat{y}(x_1, ..., x_d) = \arg\max_c \prod_{i=1}^{d} p(x_i|y_c).p(y_c)$$

Naive Bayes Classifier

PROS

- Highly scalable. Each probability can be independently estimated as a one-dimensional distribution.
- This helps alleviate problems stemming from the curse of dimensionality (the need for examples that scale exponentially with the number of features).
- Allows us to smooth the probabilities.

Naive Bayes Classifier

PROS

- Highly scalable. Each probability can be independently estimated as a one-dimensional distribution.
- This helps alleviate problems stemming from the curse of dimensionality (the need for examples that scale exponentially with the number of features).
- Allows us to smooth the probabilities.

Example

Given all the previous patients I've seen (below are their symptoms and diagnosis)...

chills	runny nose	headache	fever	flu?
Y	N	Mild	Y	N
Y	Y	No	N	Y
Y	N	Strong	Y	Y
N	Y	Mild	Y	Y
N	N	No	N	N
N	Y	Strong	Y	Y
N	Y	Strong	N	N
Y	Y	Mild	Y	Y

Do I believe that a patient with the following symptoms has the flu?

chills	runny nose	headache	fever	flu?
Y	N	Mild	Y	?

Learning from Imbalanced Datasets

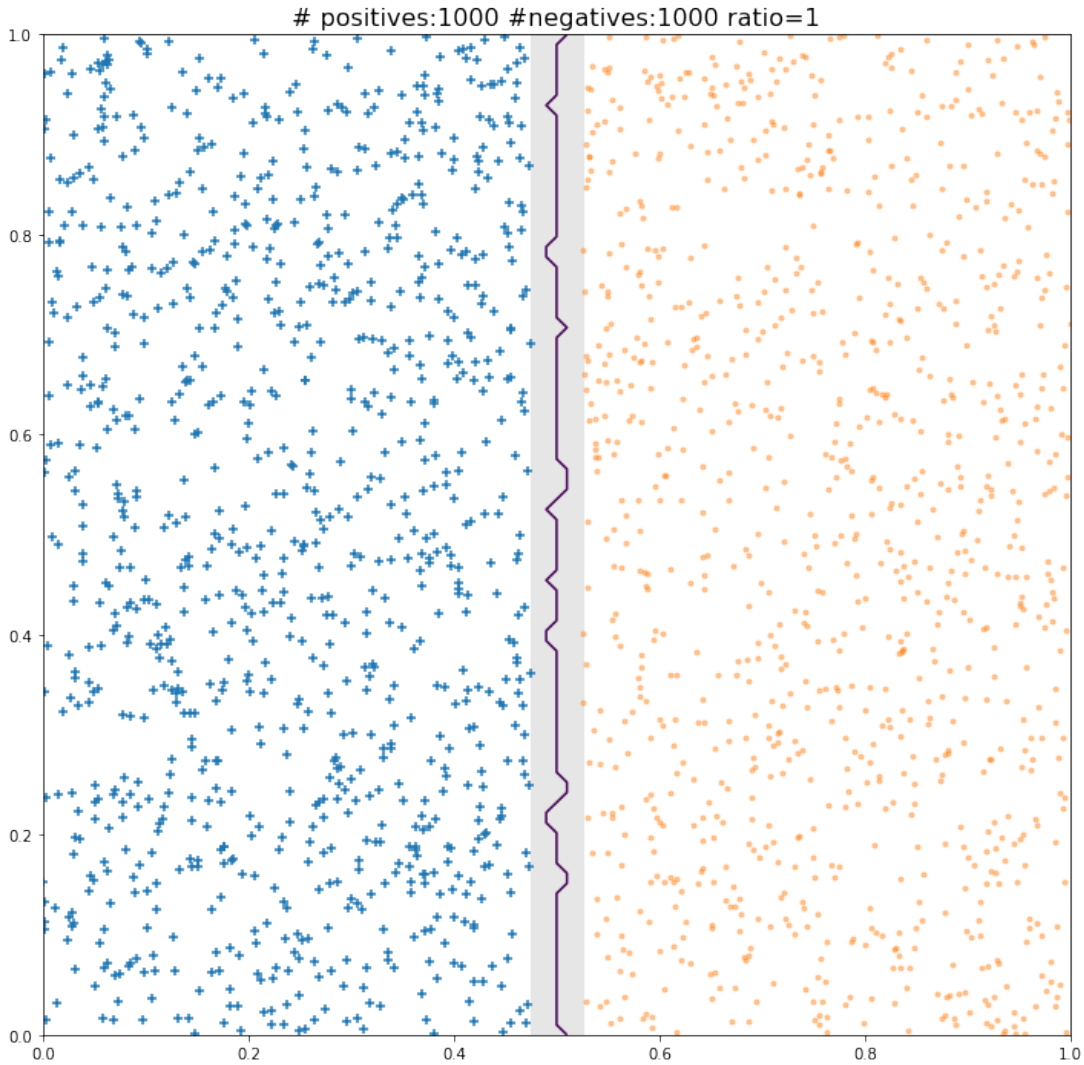
Learning from Imbalanced Datasets

In real-world applications, datasets can be (higly) imbalanced:

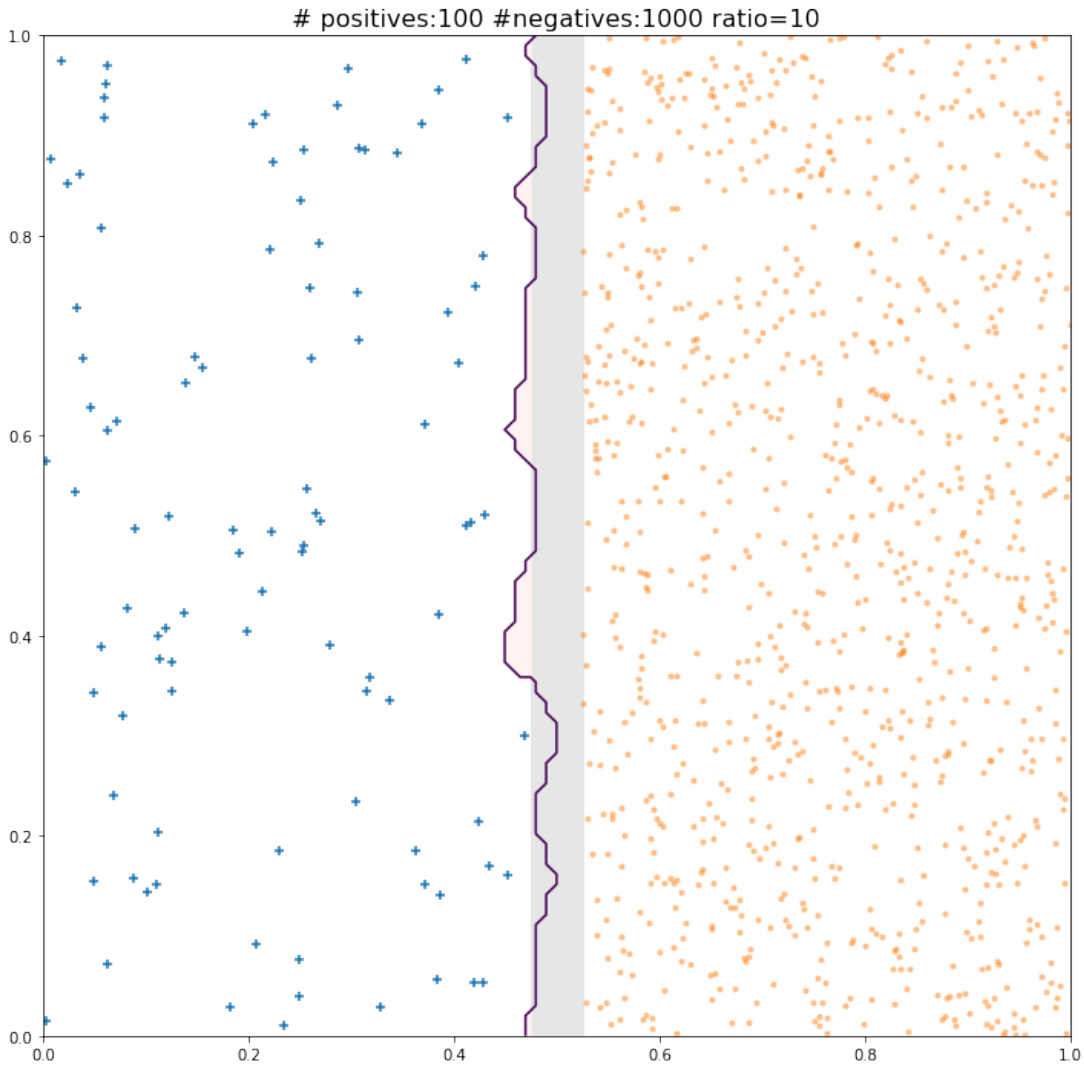
- Credit card transactions: $< 0.2\%$ frauds and 99.8% genuine transactions
- Medical screening: e.g. HIV prevalence in the USA is 0.4%
- Disk drive failure: $< 1\%$
- etc.

What is the impact of an imbalanced dataset on the performance of the classifiers?

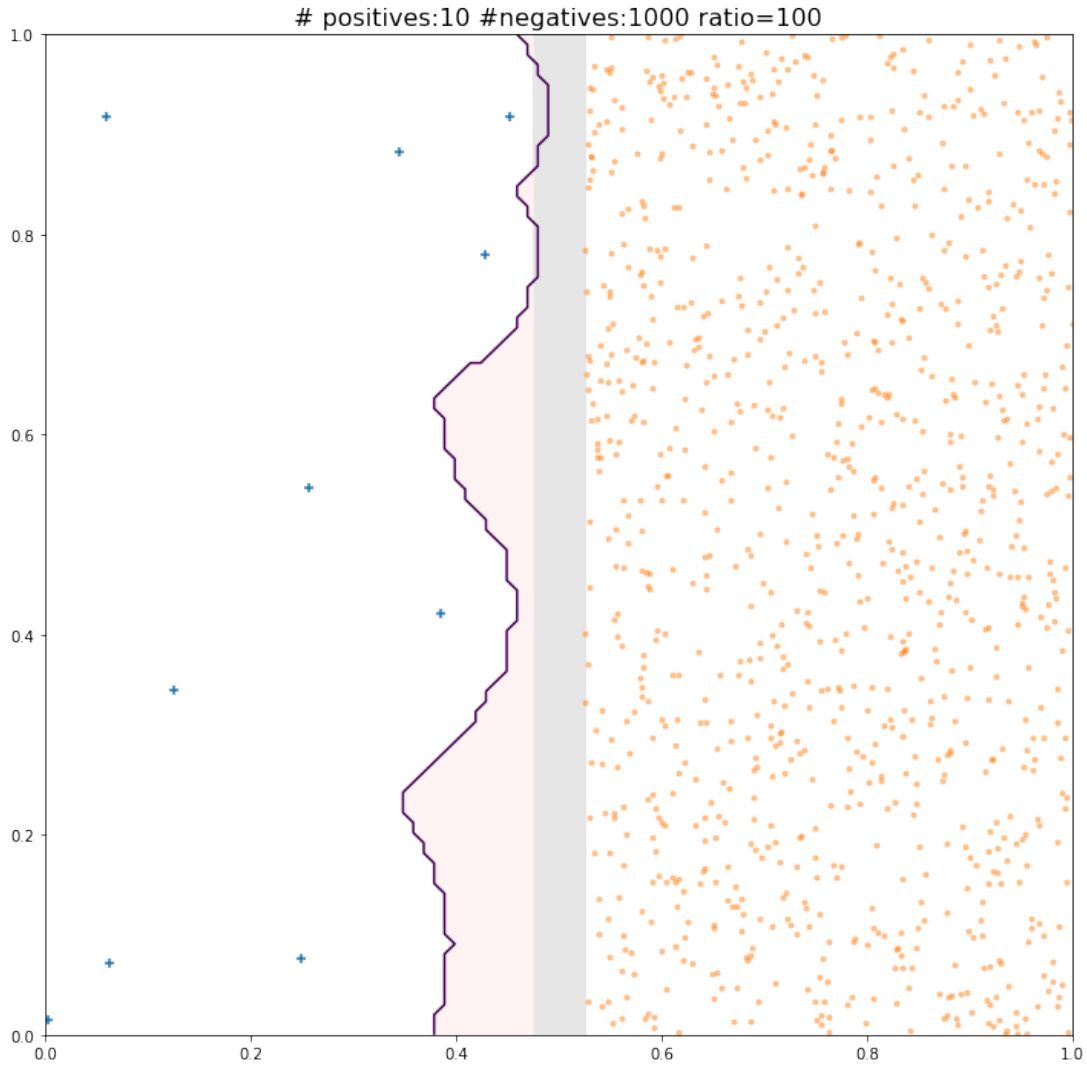
k-NN and imbalanced datasets



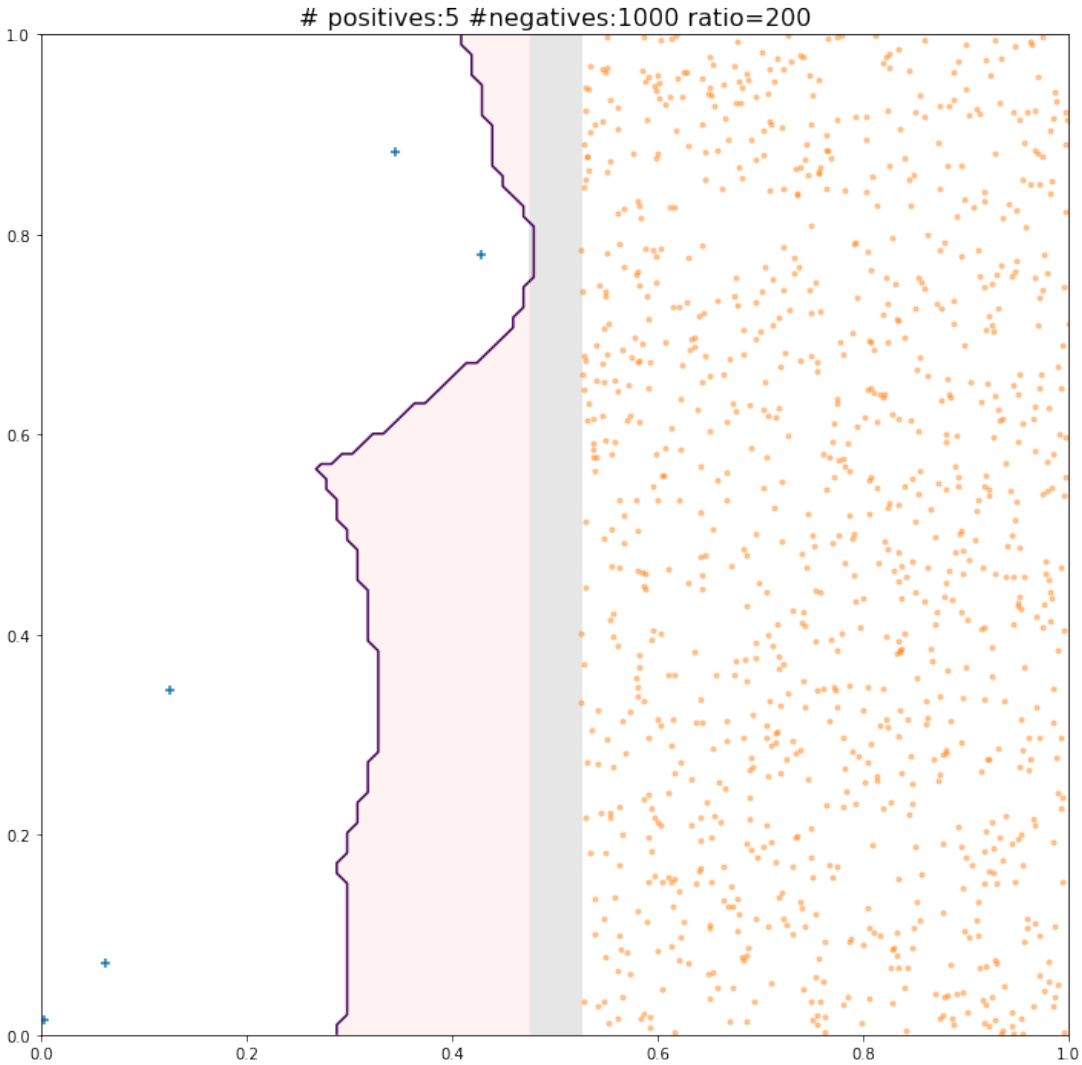
k-NN and imbalanced datasets



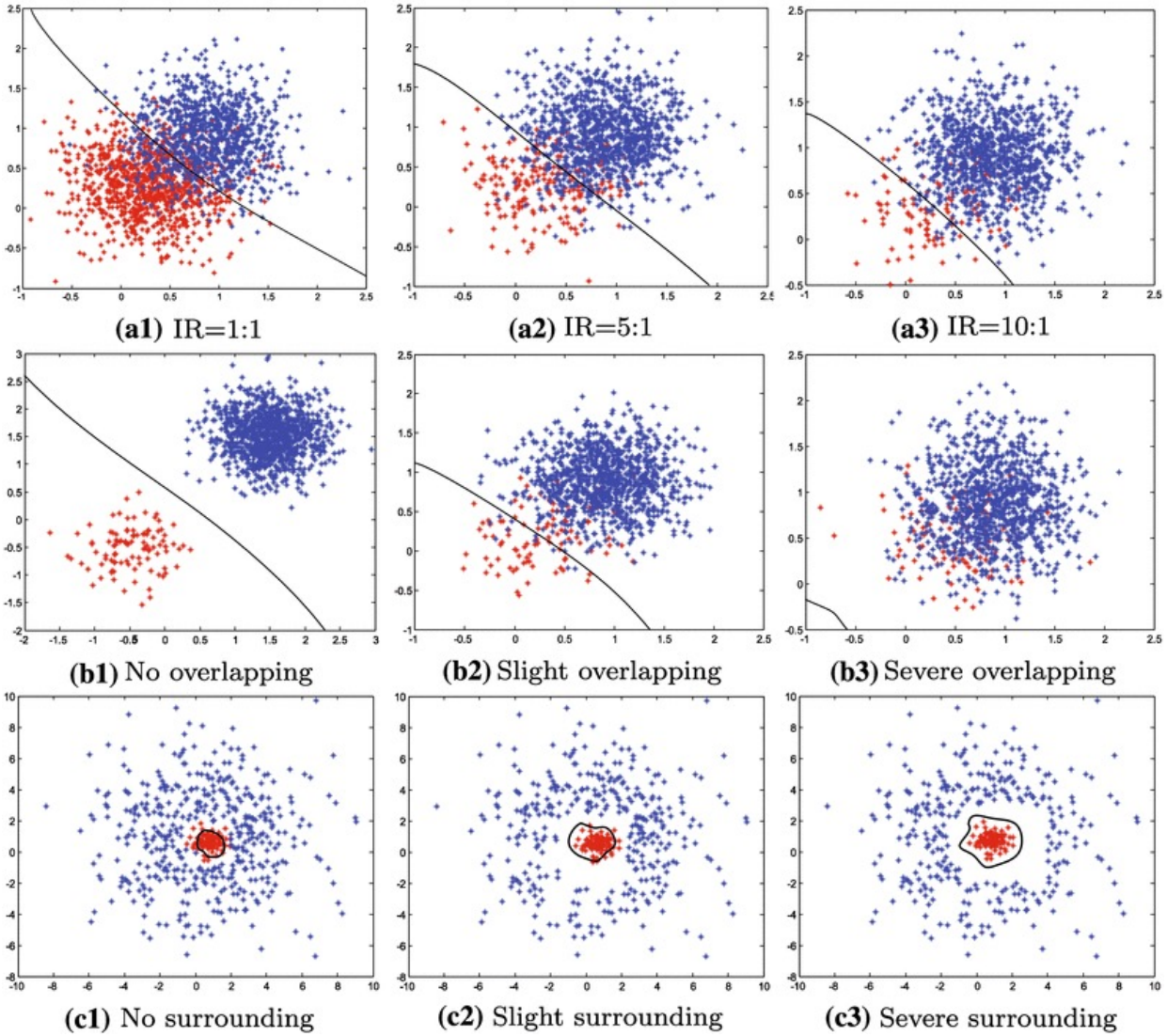
k-NN and imbalanced datasets



k-NN and imbalanced datasets



Neural Network and imbalanced datasets



« Parallel one-class extreme learning machine for imbalance learning based on Bayesian approach »
Journal of Ambient Intelligence and Humanized Computing
Li et al. , 2018

Linear model with Gaussian noise

Learning from imbalanced datasets

More generally, how to learn from (highly) imbalanced datasets?

In credit card transactions:

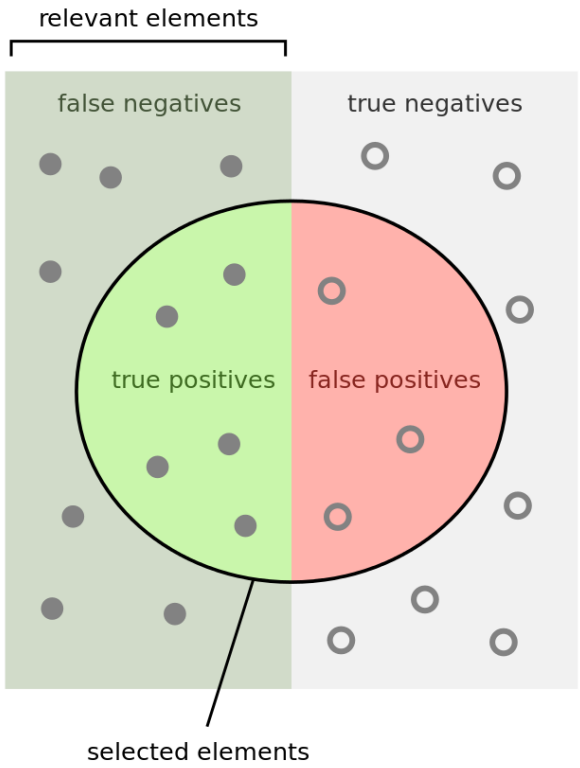
- $< 0.2\%$ correspond to frauds and 99.8% genuine transactions.
- Binary classification problem where
  - $y = +1$ for a fraud (called **positive example**)
  - $y = -1$ for a genuine transaction (called **negative example**)
  - we are looking for an hypothesis $h(x)$ as accurate as possible.

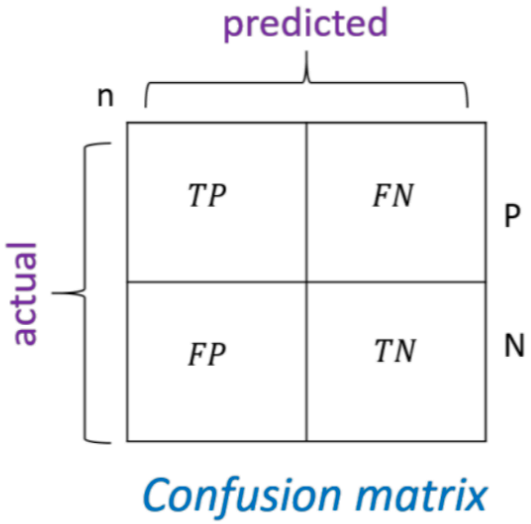
Then...

A classifier $h(x)$ which would predict $h(x) = -1 \ \forall x$ is 99.8% accurate while missing all the frauds!!!

$\rightarrow$ **Conclusion: minimizing the error rate (or a surrogate function) is not adapted to an imbalance setting**

Metrics for learning from imbalanced datasets





How many selected items are relevant?

$$\text{Precision} = \frac{\text{green}}{\text{green} + \text{red}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{green}}{\text{green} + \text{dark green}}$$

$$\begin{aligned} Accuracy &= \frac{TP + TN}{TP + FP + TN + FN} \\ &= 1 - Error \end{aligned}$$

F-Measure

When # positives << # negatives we rather optimize other metrics such as the $F_\beta - Measure$:

$$F_\beta - Measure = \frac{(1+\beta^2) \times Precision \times Recall}{(\beta^2 \times Precision) + Recall}$$

where:

$$Precision = \frac{TP}{TP + FP}$$

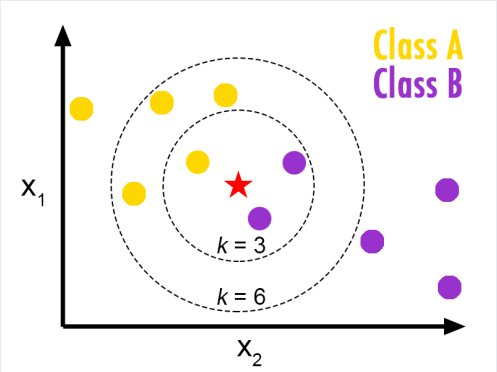
$$Recall = \frac{TP}{TP + FN}$$

$F_\beta - Measure$ is a non convex (even with surrrogates), and non separable function ($F_\beta \neq \sum_{(x_i,y_i) \in S} ...$):

- The loss for one point depends on the others.
- Impossible to optimize directly

Learning from imbalanced datasets

$F_\beta - Measure$ depends on the **decision threshold** defined for the classifier.



Example (k-NN): Let $p^+(x)$ (resp. $p^-(x)$) the proportion of positives (negatives) in the neighborhood of a query x.

In a standard k-NN, we usually use the following decision rule: if $p^+(x) > p^-(x)$ that is equivalent to $p^+(x) > 0.5$ then $h(x) = +1$ otherwise $h(x) = -1$.

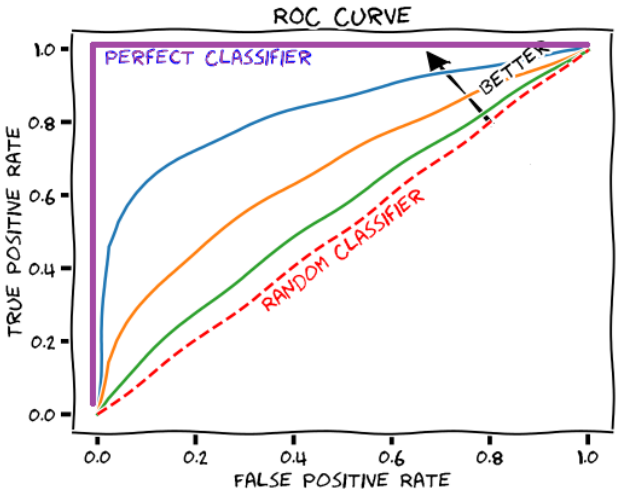
The **threshold** 0.5 is well suited to the balanced scenario. For imbalanced datasets, we can play with this threshold to increase the F-Measure. One can also **tune k so as to maximize the F-Measure**.

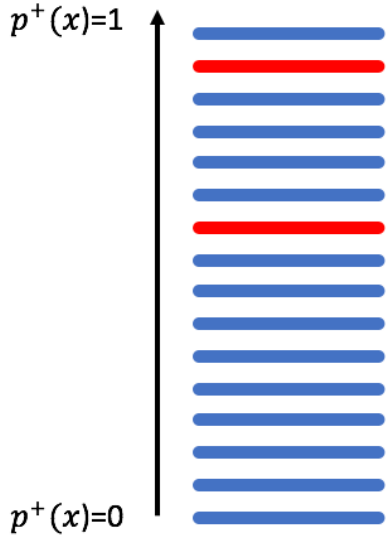
Ranking Metrics independent of a threshold

$$AUCROC = \frac{1}{PN} \sum_{i=1}^{P} \sum_{j=1}^{N} [p^+(x_i) > p^+(x_j)]$$

$$AP = \frac{1}{P} \sum_{i=1}^{P} precision(k_i)$$

where $precision(k_i)$ is the precision with respect to the rank k_i of the positive example x_i.





AUCROC versus *AP*

AP is much better when the budget in terms of number of allowed checks by human controllers is limited, like in bank or tax fraud detection.

Fig. 1. Evaluation of the *AUC-ROC* and *AP* on two rankings. Blue (resp. grey) lines represent positive (resp. negative) samples. While *AUC-ROC* behaves similarly on both cases (*AUC-ROC* $= \frac{1}{2}$), the average precision is equal to 0.43 on the left and 0.68 on the right, illustrating that *AP* favors the ranking that put (at least some) positives at the very top of the list.

Summary

F-Measure, AUCROC and AP are not differentiable and not convex measures. Therefore, they are difficult to directly optimize.

Summary

F-Measure, AUCROC and AP are not differentiable and not convex measures. Therefore, they are difficult to directly optimize.

Other strategies:

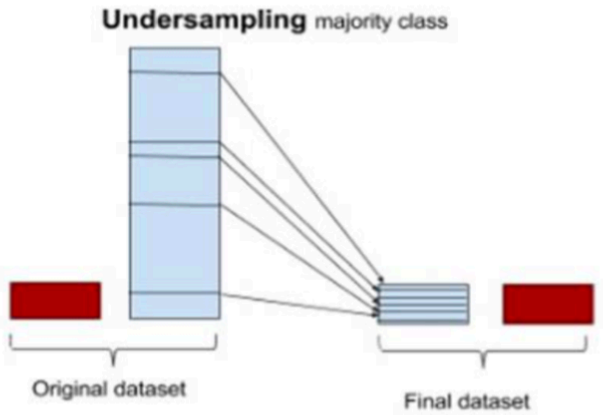
- Sampling methods
- Modification of the decision boundaries (e.g. γ-NN)

Sampling methods

Re-sampling the training set

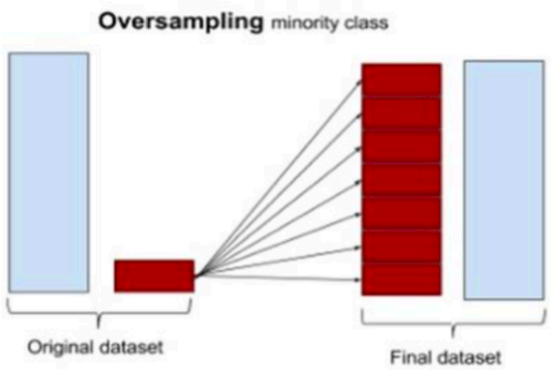
- Undersampling:

Randomly downsample the majority class



- Oversampling:

Randomly replicates minority instances



Sampling methods

Impact on the classifier

- Undersampling:

Pros:

  - Faster training (reduce training set size)
  - Add more weight for the minority class

Cons:

  - Potential loss of information

- Oversampling:

Pros:

  - Add more weight for the minority class

Cons:

  - Slows training (increases training set size)
  - Reduce the variance of the minority class (model is prone to overfitting)

SMOTE: Synthetic Minority Oversampling Technique

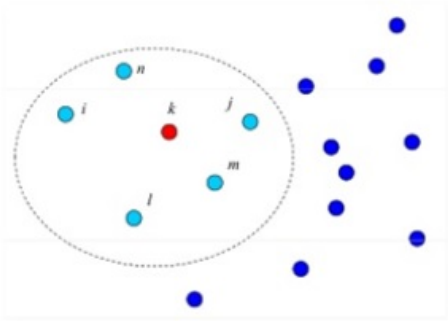
SMOTE

- Idea : Include more variance in the new examples

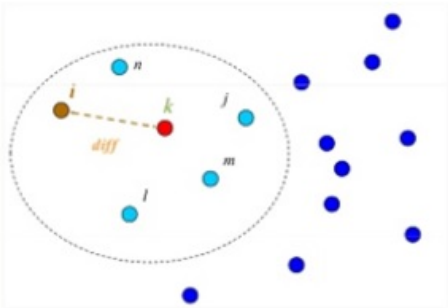
- For each minority Sample
  - Find its k-nearest minority neighbors
  - Randomly select j of these neighbors
  - Randomly generate synthetic samples along the lines joining the minority sample and its j selected neighbours

  (j depends on the amount of oversampling desired)

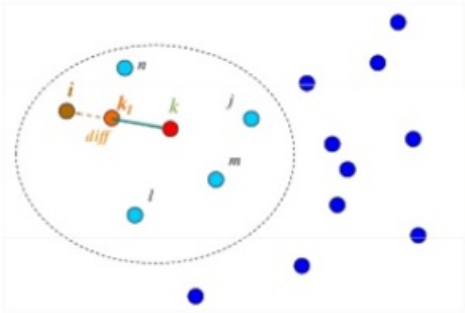
Sampling methods



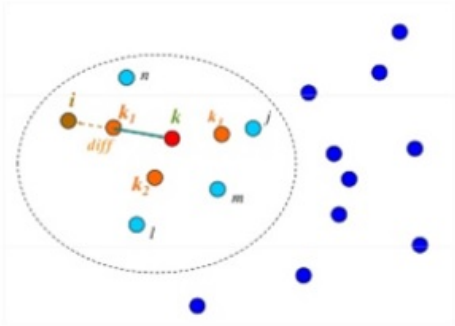
1. For each minority example k compute nearest minority class examples (i, j, l, n, m)



2. Randomly choose an example out of 5 closest points

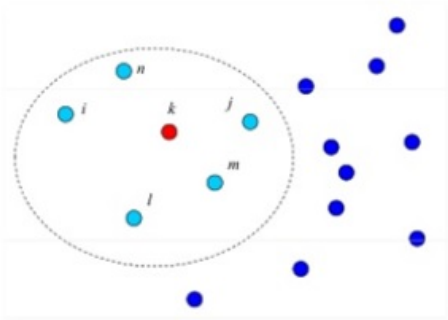


3. Synthetically generate event k_1, such that k_1 lies between k and i

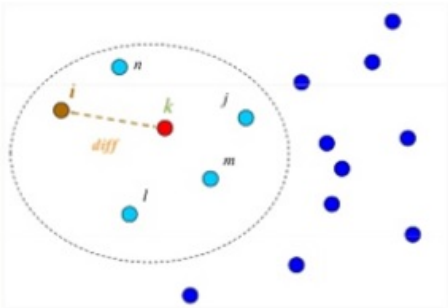


4. Dataset after applying SMOTE 3 times

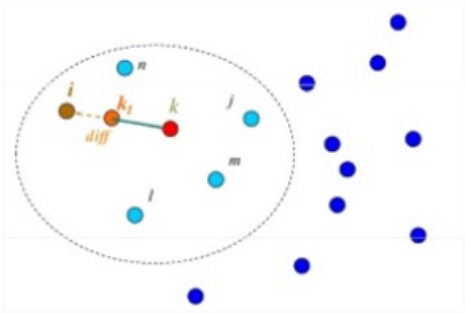
Sampling methods



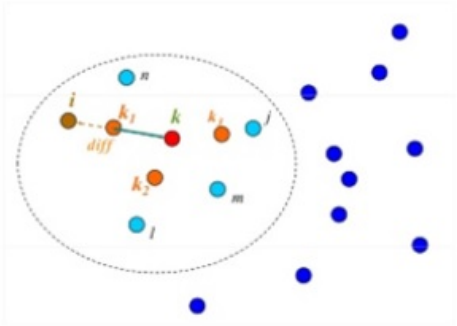
1. For each minority example k compute nearest minority class examples (i, j, l, n, m)



2. Randomly choose an example out of 5 closest points



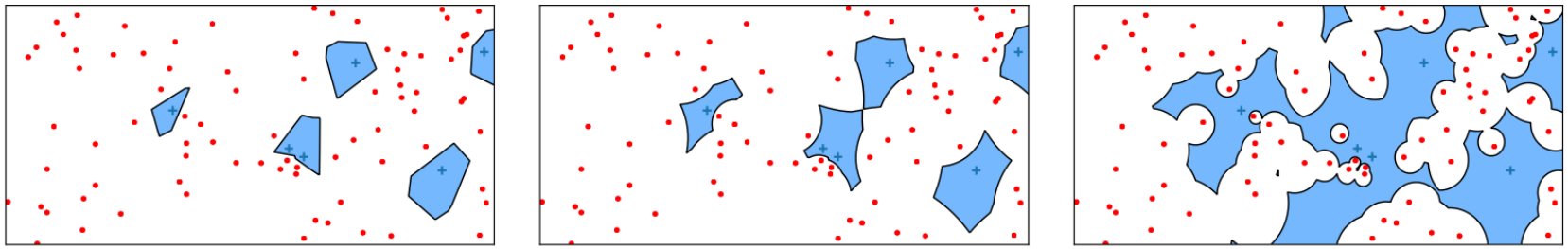
3. Synthetically generate event k_1, such that k_1 lies between k and i



4. Dataset after applying SMOTE 3 times

Can be unstable by generating positives in "negative" regions.

Modification of the boundaries: γ-NN [Viola et al. 2021]



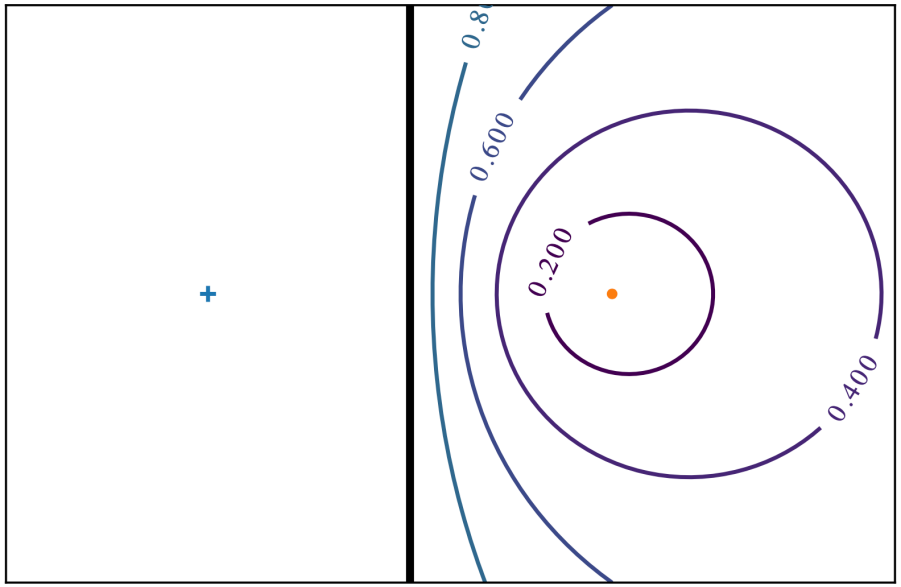
Approach

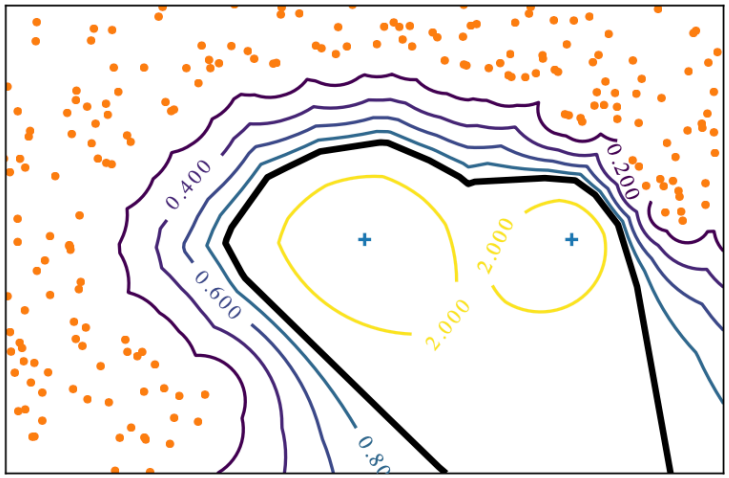
- artificially make $+$ closer to new points
- how? by using a different distance for $+$ and $-$
- the base distance to $+$ gets multiplied by a parameter γ

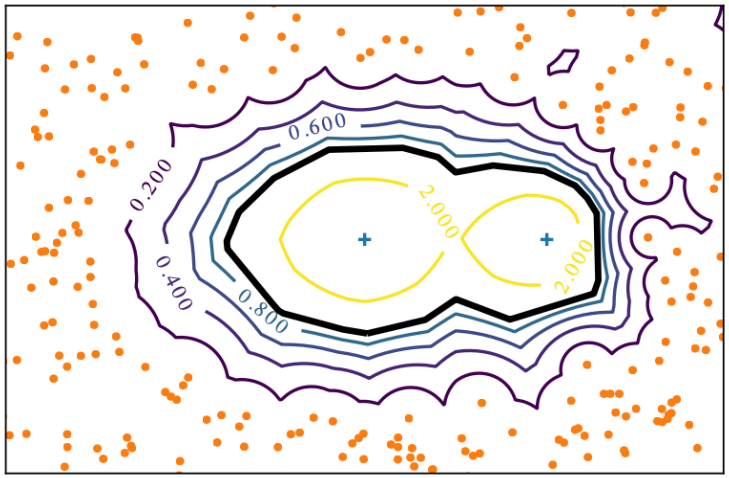
  (intuitively $\gamma \leq 1$ if $+$ is rare)

$$d_\gamma(x, x_i) = \begin{cases} d(x, x_i) & \text{if } x_i \in S_-, \\ \gamma \cdot d(x, x_i) & \text{if } x_i \in S_+. \end{cases}$$

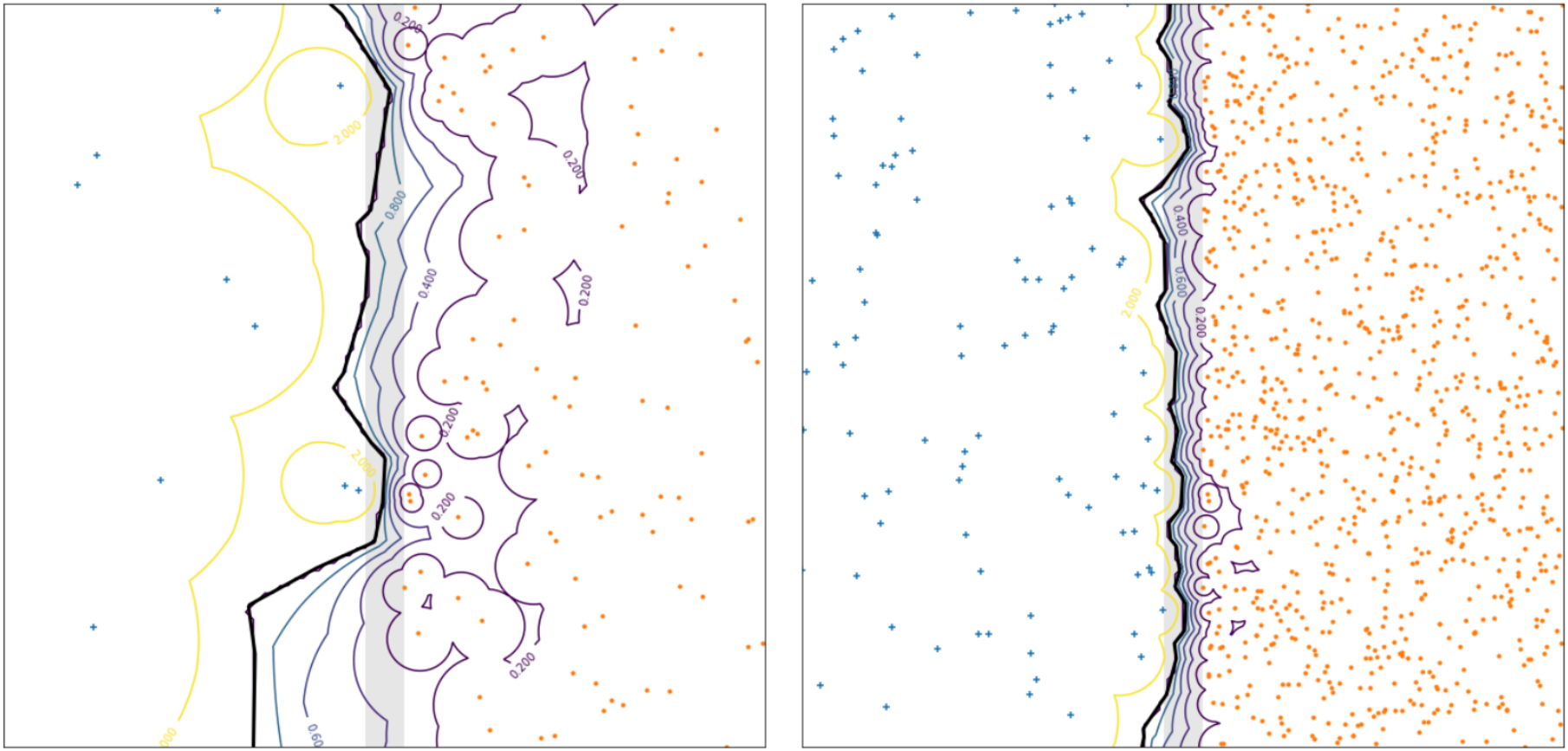
Modification of the boundaries: γ-NN [Viola et al. 2021]







Modification of the boundaries: γ-NN [Viola et al. 2021]



Presentation of the Project

+

Multiple Choice Questions **MCQ3**