

# Nearest centroids, K-NN

Victor Kitov

[v.v.kitov@yandex.ru](mailto:v.v.kitov@yandex.ru)

# Table of Contents

- 1 Nearest centroids
- 2 K nearest neighbours
- 3 Special properties
- 4 Weighted account for objects

## Nearest centroids algorithm

- Consider training sample  $(x_1, y_1), \dots (x_N, y_N)$  with
  - $N_1$  representatives of 1st class
  - $N_2$  representatives of 2nd class
  - etc.

- **Training:**

Calculate centroids for each class  $c = 1, 2, \dots C$  :

$$\mu_c = \frac{1}{N_1} \sum_{n=1}^N x_n \mathbb{I}[y_n = c]$$

- **Classification:**

- 1 For object  $x$  find most close centroid:

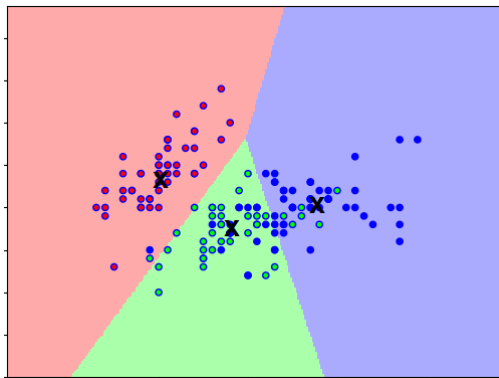
$$c = \arg \min_i \rho(x, \mu_i)$$

- 2 Associate  $x$  the class of the most close centroid:

$$\hat{y}(x) = c$$

# Illustration

Decision boundaries for 3-class nearest centroids



## Questions

- What are discriminant functions  $g_c(x)$  for nearest centroid?
- What is the complexity for:
  - training?
  - prediction?
- What would be the shape of class separating boundary?
- Can we use similar ideas for regression? Consider clustering.
- Is this method prone to the curse of dimensionality?

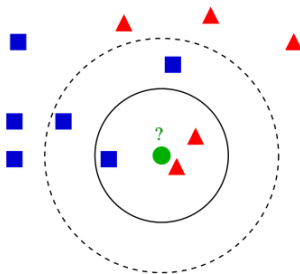
# Table of Contents

- 1 Nearest centroids
- 2 K nearest neighbours
- 3 Special properties
- 4 Weighted account for objects

# K-nearest neighbours algorithm

## Classification:

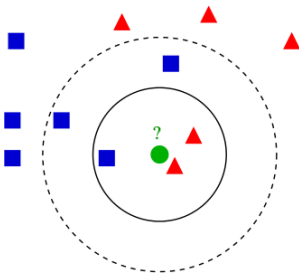
- 1 Find  $k$  closest objects to the predicted object  $x$  in the training set.
- 2 Associate  $x$  the most frequent class among its  $k$  neighbours.



# K-nearest neighbours algorithm

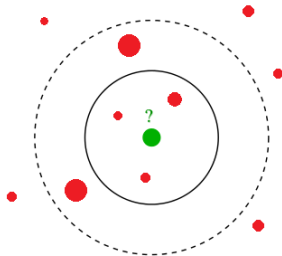
## Classification:

- 1 Find  $k$  closest objects to the predicted object  $x$  in the training set.
- 2 Associate  $x$  the most frequent class among its  $k$  neighbours.



## Regression:

- 1 Find  $k$  closest objects to the predicted object  $x$  in the training set.
- 2 Associate  $x$  average output of its  $k$  neighbours.





# Comments

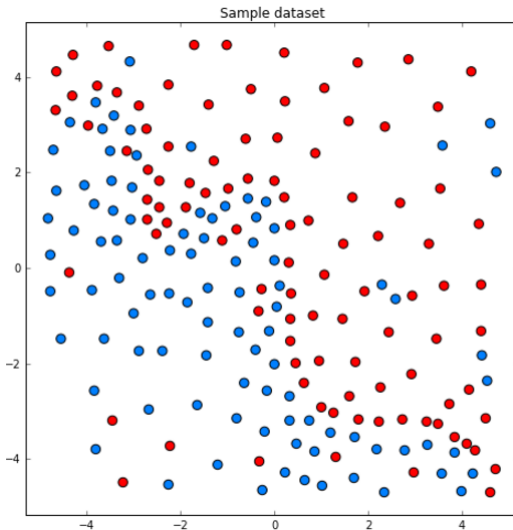
- K nearest neighbours algorithm is abbreviated as K-NN.
- $k = 1$ : nearest neighbour algorithm<sup>1</sup>
- Base assumption of the method<sup>2</sup>:
  - similar objects yield similar outputs

---

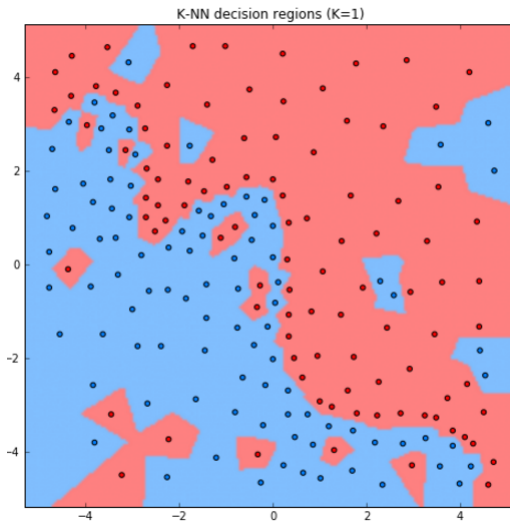
<sup>1</sup>what will happen for  $K = N$ ?

<sup>2</sup>what is simpler - to train K-NN model or to apply it?

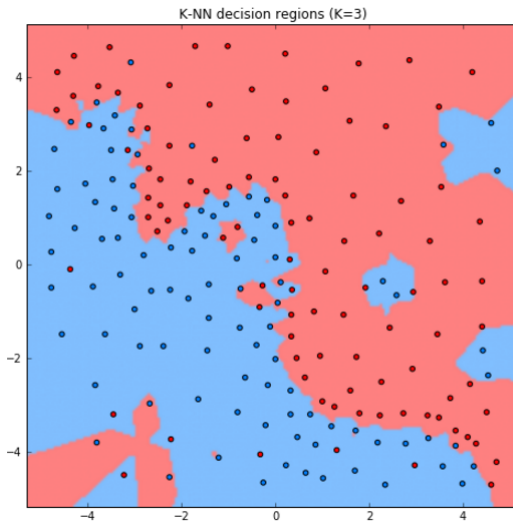
# Sample dataset



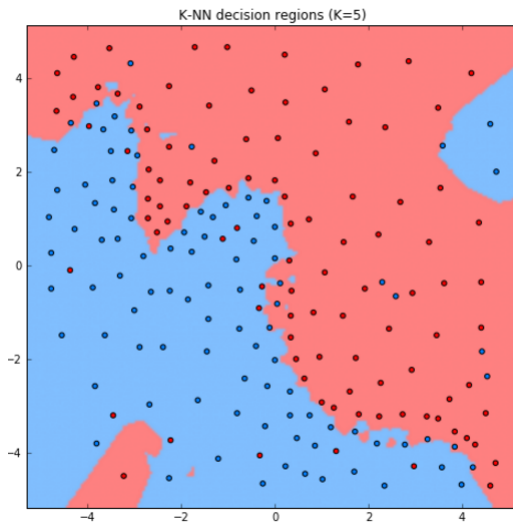
## Example: K-NN classification



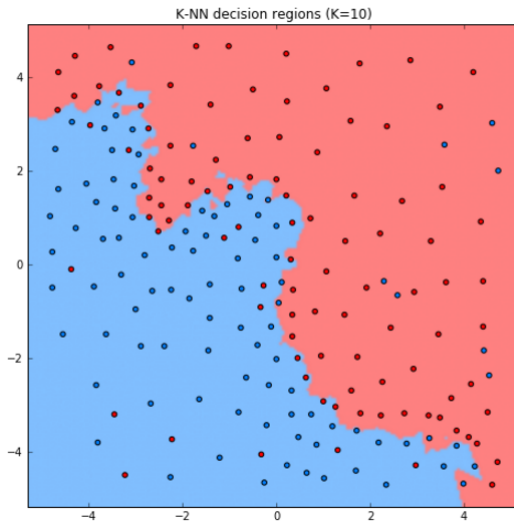
## Example: K-NN classification



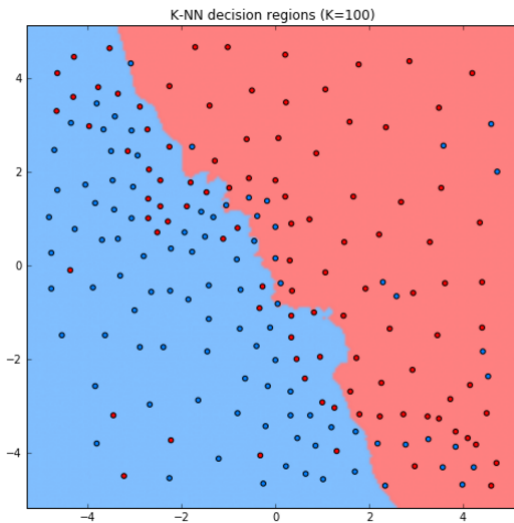
## Example: K-NN classification



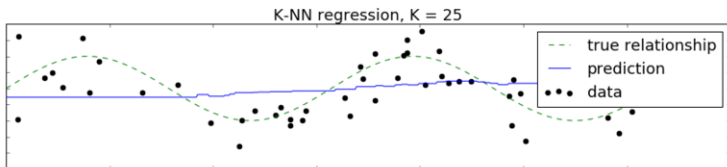
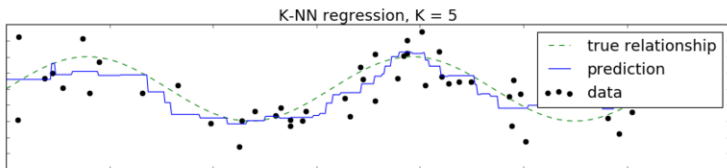
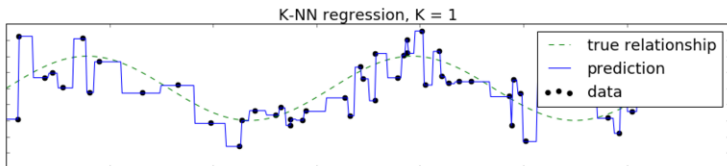
## Example: K-NN classification



## Example: K-NN classification



# Example: K-NN regression





## Dealing with similar rank

When several classes get the same rank, we can assign to class:

## Dealing with similar rank

When several classes get the same rank, we can assign to class:

- with higher prior probability
- having closest representative
- having closest mean of representatives (among nearest neighbours)
- which is more compact, having nearest most distant representative

# Parameters of the method

- Parameters:
  - the number of nearest neighbours  $K$
  - distance metric  $\rho(x, x')$
- Modifications:
  - forecast rejection option<sup>3</sup>
  - variable  $K$ <sup>4</sup>

---

<sup>3</sup>Propose a rule, under what conditions to apply rejection in a) classification  
b) regression

<sup>4</sup>Propose a method of K-NN with adaptive variable  $K$  in different parts of the feature space

# Properties

- **Advantages:**

- only similarity between objects is needed, not exact feature values.
  - so it may be applied to objects with arbitrary complex feature description
- simple to implement
- interpretable (case based reasoning)
- does not need training
  - may be applied in online scenarios
  - Cross-validation may be replaced with LOO.

- **Disadvantages:**

- slow classification with complexity  $O(N)$
- accuracy deteriorates with the increase of feature space dimensionality

# Table of Contents

- 1 Nearest centroids
- 2 K nearest neighbours
- 3 Special properties**
- 4 Weighted account for objects

## Normalization of features

- Feature scaling affects predictions of K-NN?

## Normalization of features

- Feature scaling affects predictions of K-NN?
  - yes, so normalize them
- Equal scaling - equal impact of features
- Non-equal scaling - non-equal impact of features
- Typical normalizations:

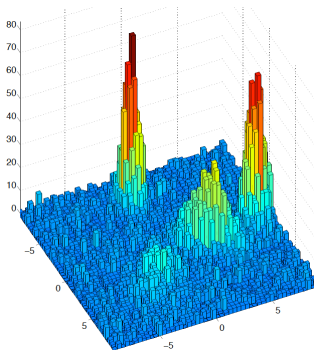
Name	Transformation	Properties of resulting feature
Autoscaling	$x'_j = \frac{x_j - \mu_j}{\sigma_j}$	zero mean and unit variance.
Range scaling	$x'_j = \frac{x_j - L_j}{U_j - L_j}$	belongs to $[0, 1]$ interval.

where  $\mu_j$ ,  $\sigma_j$ ,  $L_j$ ,  $U_j$  are mean value, standard deviation, minimum and maximum value of the  $j$ -th feature.

- for non-negative features range scaling does not affect feature sparsity:  $0 \rightarrow 0$ .

# The curse of dimensionality

- The curse of dimensionality: with growing  $D$  data distribution becomes sparse and insufficient.
- Example: histogram estimation<sup>5</sup>



---

<sup>5</sup>At what rate should training size grow with increase of  $D$  to compensate curse of dimensionality?



# Curse of dimensionality

- Case of K-nearest neighbours:
  - assumption: objects are distributed uniformly in feature space
  - ball of radius  $R$  has volume  $V(R) = CR^D$ , where
$$C = \frac{\pi^{D/2}}{\Gamma(D/2+1)}.$$
  - ratio of volumes of balls with radius  $R - \varepsilon$  and  $R$ :

$$\frac{V(R - \varepsilon)}{V(R)} = \left( \frac{R - \varepsilon}{R} \right)^D \xrightarrow{D \rightarrow \infty} 0$$

- most of volume concentrates on the border of the ball, so there lie the nearest neighbours.
  - nearest neighbours stop being close by distance
- Good news: in real tasks the true dimensionality of the data is often less than  $D$  and objects belong to the manifold with smaller dimensionality.

# Table of Contents

- 1 Nearest centroids
- 2 K nearest neighbours
- 3 Special properties
- 4 Weighted account for objects

## Equal voting

- Define  $K$  nearest neighbors:  $(z_1, y_1), (z_2, y_2), \dots, (z_K, y_K)$ .

$$\rho(x, z_1) \leq \rho(x, z_2) \leq \dots \leq \rho(x, z_K)$$

- Regression:

$$\hat{y}(x) = \frac{1}{K} \sum_{k=1}^K y_k$$

- Classification:

$$g_c(x) = \sum_{k=1}^K \mathbb{I}[y_k = c], \quad c = 1, 2, \dots, C.$$

$$\hat{y}(x) = \arg \max_c g_c(x)$$

## Weighted voting

- Weighted regression:

$$\hat{y}(x) = \frac{\sum_{k=1}^K w(k, \rho(x, z_k)) y_k}{\sum_{k=1}^K w(k, \rho(x, z_k))}$$

## Weighted voting

- Weighted regression:

$$\hat{y}(x) = \frac{\sum_{k=1}^K w(k, \rho(x, z_k)) y_k}{\sum_{k=1}^K w(k, \rho(x, z_k))}$$

- Weighted classification:

$$g_c(x) = \sum_{k=1}^K w(k, \rho(x, z_k)) \mathbb{I}[y_k = c], \quad c = 1, 2, \dots, C.$$

$$\hat{y}(x) = \arg \max_c g_c(x)$$

## Commonly chosen weights

Index dependent weights:

$$w_k = \alpha^k, \quad \alpha \in (0, 1)$$

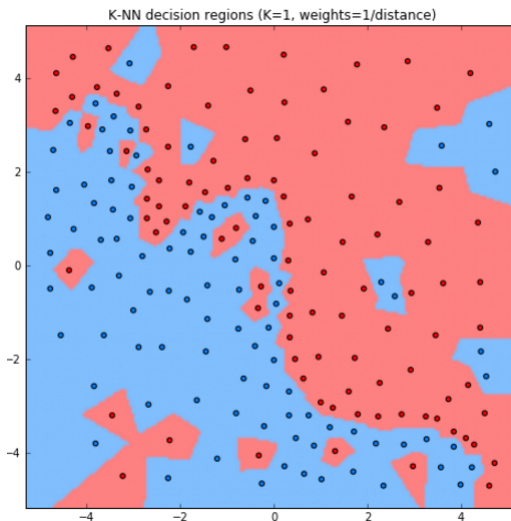
$$w_k = \frac{K + 1 - k}{K}$$

Distance dependent weights:

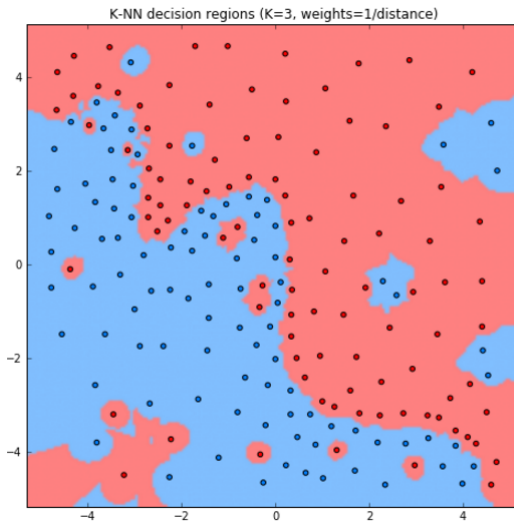
$$w_k = \begin{cases} \frac{\rho(z_K, x) - \rho(z_k, x)}{\rho(z_K, x) - \rho(z_1, x)}, & \rho(z_K, x) \neq \rho(z_1, x) \\ 1 & \rho(z_K, x) = \rho(z_1, x) \end{cases}$$

$$w_k = \frac{1}{\rho(z_k, x)}$$

## Example: K-NN classification with weights

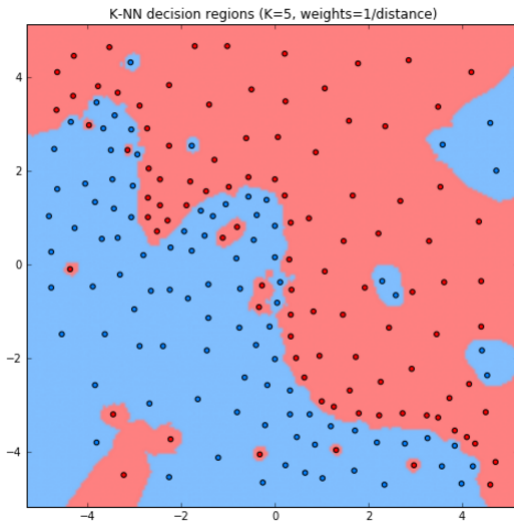


## Example: K-NN classification with weights

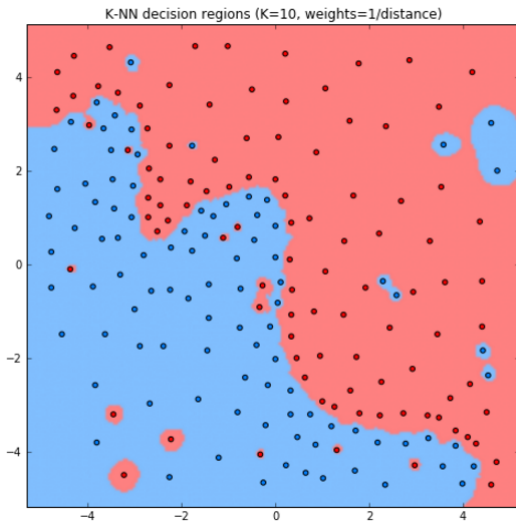




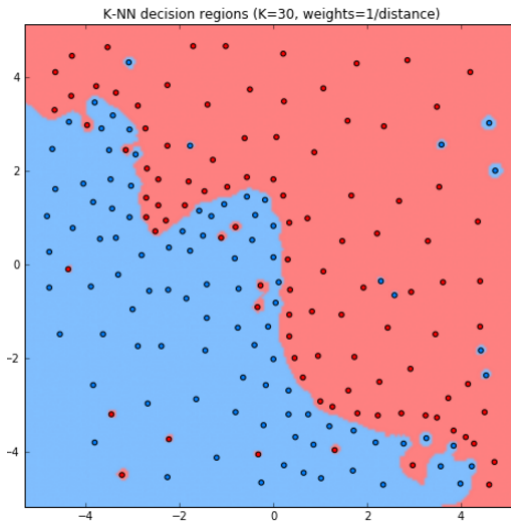
## Example: K-NN classification with weights



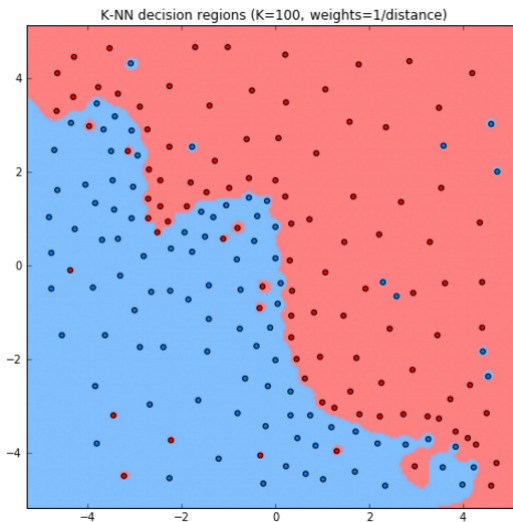
## Example: K-NN classification with weights



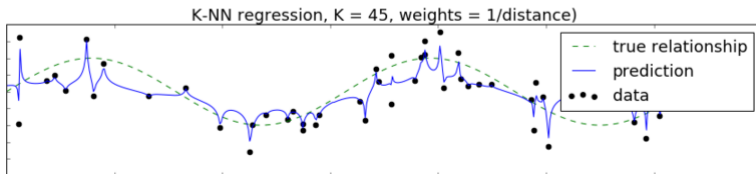
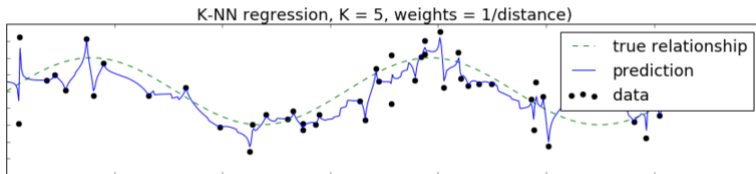
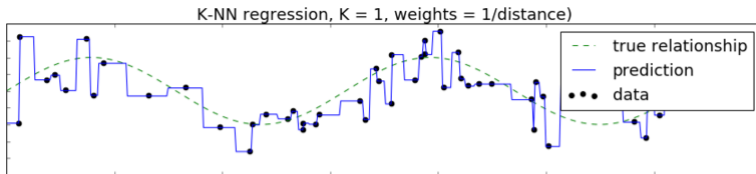
## Example: K-NN classification with weights



## Example: K-NN classification with weights



# Example: K-NN regression with weights



## Summary

- Important parameters of K-NN:
  - $K$ : controls model complexity
  - $\rho(x, x')$
- Output depends on feature scaling.
  - scaling to equal / non-equal scatter possible.
- Prone to curse of dimensionality.
- Fast training but long prediction.
  - some efficiency improvements are possible though
- Weighted account for objects possible.
- Nearest centroid has different properties.

dfssdf