

# K-nearest neighbours

Victor Kitov  
v.v.kitov@yandex.ru

Yandex School of Data Analysis



# Table of Contents

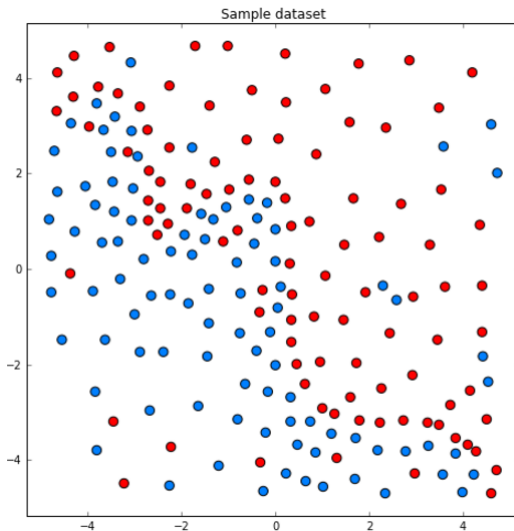
- 1 Basic variant of K-NN
- 2 Distance metric selection
- 3 Weighted voting

# K-nearest neighbours

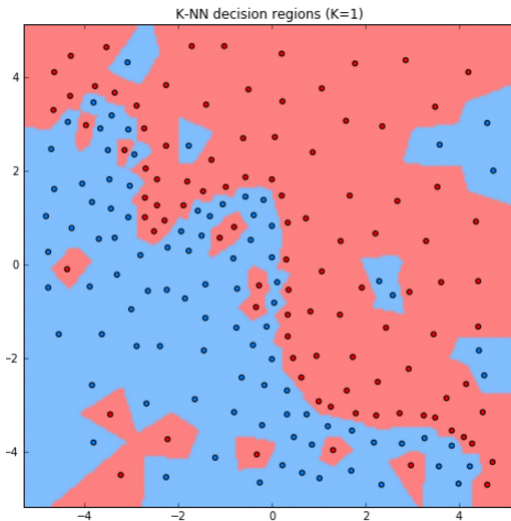
## Classification using $k$ nearest neighbours

- ❶ Find  $k$  closest objects to the predicted object  $x$  in the training set.
  - ❷ Associate  $x$  the most frequent class among its  $k$  neighbours.
- $k = 1$ : nearest neighbour algorithm
  - $k = N$ : constant prediction with the most frequent class in the training set
  - Regression case: targets of nearest neighbours are averaged
  - Base assumption of the method:
    - similar objects yield similar outputs

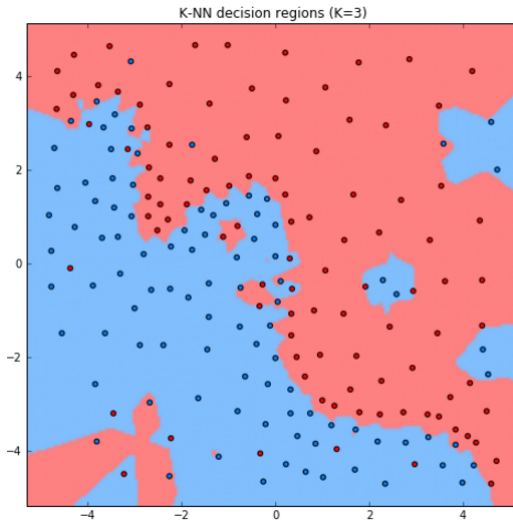
# Sample dataset



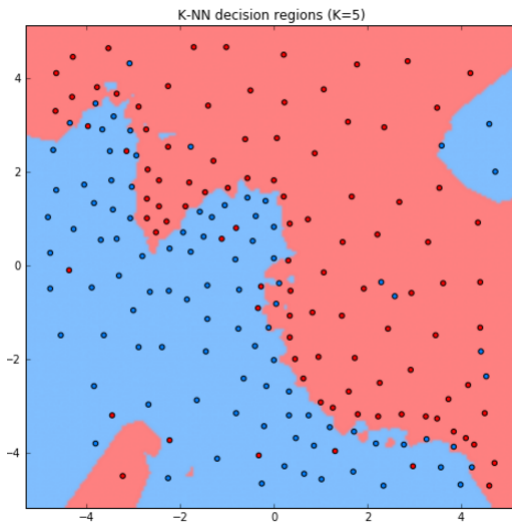
## Example: K-NN classification



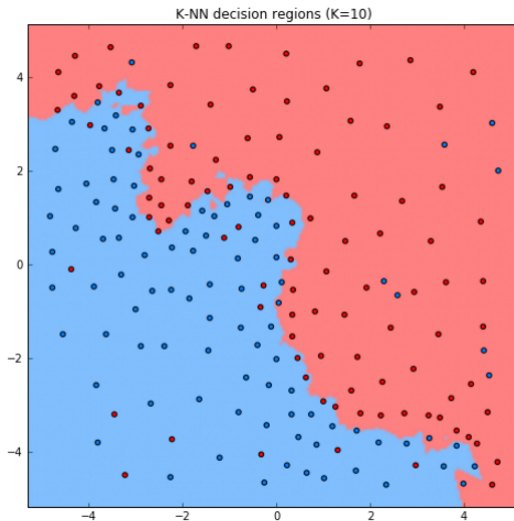
## Example: K-NN classification



## Example: K-NN classification

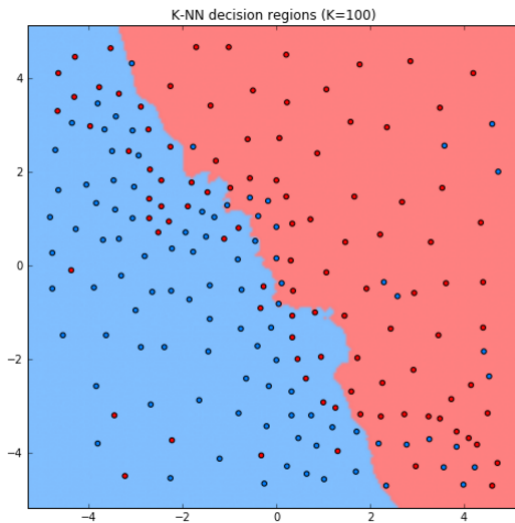


## Example: K-NN classification

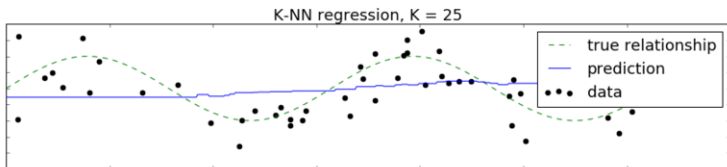
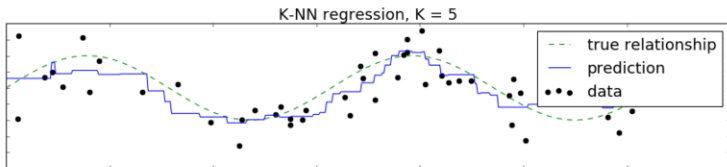
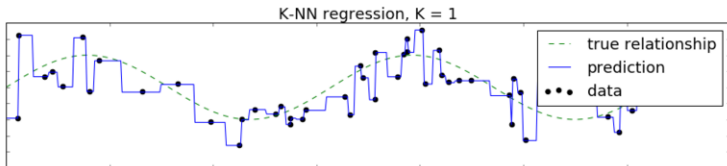




## Example: K-NN classification



## Example: K-NN regression



# Parameters of the method

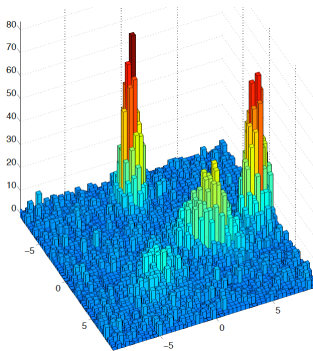
- Parameters:
  - the number of nearest neighbours  $k$
  - distance metric  $\rho(x, y)$
- Variants:
  - rejection option:
    - classification: when classes are uniformly distributed
    - regression: when variance is high
  - $k$  is locally adapted to yield maximum accuracy.
  - random subspace modification

# 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
    - K-CV may be replaced with LOO.
- Disadvantages:
  - slow classification with complexity  $O(N)$
  - accuracy deteriorates with the increase of feature space dimensionality

# The curse of dimensionality

- The curse of dimensionality in ML is a situation when the number of training samples needs to grow exponentially when  $D$  grows linearly to guarantee certain level of accuracy.
- Example: histogram estimation



# 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.

## Dealing with similar rank

When several classes get the same rank, we can:

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

# Table of Contents

- 1 Basic variant of K-NN
- 2 Distance metric selection
- 3 Weighted voting



## Distance metric selection

- Baseline case - Euclidean metric
- Necessary to normalize features.
  - Define  $\mu_j$ ,  $\sigma_j$ ,  $L_j$ ,  $U_j$  to be mean value, standard deviation, minimum and maximum value of the  $j$ -th feature.

| 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.   |

## Normalization of features

- Non-linear transformations incorporating features with rare large values:
  - $x'_i = \log(x_i)$
  - $x'_i = x^p, 0 \leq p < 1$
- For  $F_i(\alpha) = P(x^i \leq \alpha)$  transformation  $\tilde{x}^i \rightarrow F_i(x^i)$  will give feature uniformly distributed on  $[0, 1]$ .

# Distance metric selection

| Metric         | $d(x, z)$   |
|----------------|---|
| Euclidean      | $\sqrt{\sum_{i=1}^D (x^i - z^i)^2}$                       |
| $L_p$          | $\sqrt[p]{\sum_{i=1}^D (x^i - z^i)^p}$                    |
| $L_\infty$     | $\max_{i=1,2,\dots,D}  x^i - z^i $                        |
| $L_1$          | $\sum_{i=1}^D  x^i - z^i $                                |
| Canberra       | $\frac{1}{D} \sum_{i=1}^D \frac{ x^i - z^i }{x^i + z^i}$  |
| Lance-Williams | $\frac{\sum_{i=1}^D  x^i - z^i }{\sum_{i=1}^D x^i + z^i}$ |

## Cosine similarity

### Scalar product property

$$\langle x, z \rangle = \|x\| \|z\| \cos(\alpha)$$

where  $\alpha$  - is the angle between  $x$  and  $y$ .

## Cosine similarity

### Scalar product property

$$\langle x, z \rangle = \|x\| \|z\| \cos(\alpha)$$

where  $\alpha$  - is the angle between  $x$  and  $y$ .

### Cosine similarity

$$s(x, z) = \frac{\langle x, z \rangle}{\|x\| \|z\|} = \frac{\sum_{i=1}^D x^i z^i}{\sqrt{\sum_{i=1}^D (x^i)^2} \sqrt{\sum_{i=1}^D (z^i)^2}}$$

# Cosine similarity

## Scalar product property

$$\langle x, z \rangle = \|x\| \|z\| \cos(\alpha)$$

where  $\alpha$  - is the angle between  $x$  and  $y$ .

## Cosine similarity

$$s(x, z) = \frac{\langle x, z \rangle}{\|x\| \|z\|} = \frac{\sum_{i=1}^D x^i z^i}{\sqrt{\sum_{i=1}^D (x^i)^2} \sqrt{\sum_{i=1}^D (z^i)^2}}$$

- $s(x, z) \in [-1, 1]$ ,  $d(x, z) \in [0, 1]$ ,  $d(x, z) = 0 \Leftrightarrow x \uparrow\uparrow y$ .
- Cosine distance:  $d(x, z) = 1 - \frac{1}{\pi} \arccos(s(x, z))$
- frequently used for comparing documents in bag-of-words model
- $d(x, z)$  satisfies:  $d(x, x') \leq d(x, z) + d(z, x') \quad \forall x, x', z$ .

## Whitening transformation

- $x \sim F(\mu, \Sigma)$ ,  $\mu = \mathbb{E}[\mu]$ ,  $\Sigma = \text{cov}(x, x)$ ,  $\mu \in \mathbb{R}^D$ ,  $\Sigma \in \mathbb{R}^{D \times D}$
- Task: find linear standardization of  $x \rightarrow z$ , so that  $\mathbb{E}z = \mathbf{0}$  and  $\text{cov}[z, z] = I$ .

## Whitening transformation

- $x \sim F(\mu, \Sigma)$ ,  $\mu = \mathbb{E}[\mu]$ ,  $\Sigma = \text{cov}(x, x)$ ,  $\mu \in \mathbb{R}^D$ ,  $\Sigma \in \mathbb{R}^{D \times D}$
- Task: find linear standardization of  $x \rightarrow z$ , so that  $\mathbb{E}z = \mathbf{0}$  and  $\text{cov}[z, z] = I$ .

### Whitening transformation

$$z = \Sigma^{-1/2}(x - \mu), \quad \mathbb{E}z = \mathbf{0}, \quad \text{cov}[z, z] = I.$$



# Whitening transformation

- $x \sim F(\mu, \Sigma)$ ,  $\mu = \mathbb{E}[\mu]$ ,  $\Sigma = \text{cov}(x, x)$ ,  $\mu \in \mathbb{R}^D$ ,  $\Sigma \in \mathbb{R}^{D \times D}$
- Task: find linear standardization of  $x \rightarrow z$ , so that  $\mathbb{E}z = \mathbf{0}$  and  $\text{cov}[z, z] = I$ .

## Whitening transformation

$$z = \Sigma^{-1/2}(x - \mu), \quad \mathbb{E}z = \mathbf{0}, \text{cov}[z, z] = I.$$

- Proof:

$$\mathbb{E}z = \mathbb{E} \left\{ \Sigma^{-1/2}(x - \mu) \right\} = \Sigma^{-1/2} \mathbb{E} \{x - \mu\} = \mathbf{0} \in \mathbb{R}^D$$

$$\begin{aligned} \text{cov}[z, z] &= \mathbb{E}(z - \mathbb{E}z)(z - \mathbb{E}z)^T = \mathbb{E}zz^T \\ &= \mathbb{E} \left\{ \Sigma^{-1/2}(x - \mu)(x - \mu)^T (\Sigma^{-1/2})^T \right\} = \\ &= \Sigma^{-1/2} \mathbb{E} \left\{ (x - \mu)(x - \mu)^T \right\} (\Sigma^{-1/2})^T = \\ &= \Sigma^{-1/2} \Sigma \Sigma^{-1/2} = I \end{aligned}$$

## Distance between normalized feature vectors

- Distance between normalized  $x$  and  $x'$  is equal to Euclidean distance between  $z = \Sigma^{-1/2}(x - \mu)$  and  $z' = \Sigma^{-1/2}(x' - \mu)$ :

$$\begin{aligned}\rho_M(x, x') &= \rho_E(z, z') = \sqrt{(z - z')^T(z - z')} = \\ &= \sqrt{(x - x')^T \Sigma^{-1/2} \Sigma^{-1/2} (x - x')} \\ &= \sqrt{(x - x')^T \Sigma^{-1} (x - x')}\end{aligned}$$

- This is known as *Mahalanobis distance*.
- Special case when features are uncorrelated and  $\text{Var}[x^i] = \sigma_i^2$ :

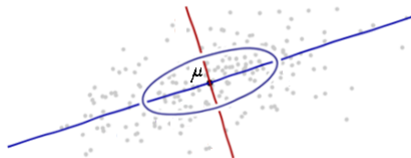
$$\rho_M(x, \tilde{x}) = \sqrt{\sum_i \frac{(x^i - \tilde{x}^i)^2}{\sigma_i^2}}$$

## Mahalanobis distance - illustration

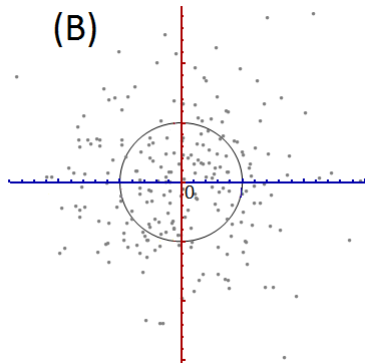
(A): object in initial feature space with Mahalanobis sphere

$G_\alpha = \{x : \rho_M(x, \mu) = \alpha\}$ . (B): the image of objects and sphere in normalized space ( $Im[G_\alpha] = \{z : \rho_E(z, 0) = \alpha\}$ ).

(A)



(B)



# Table of Contents

- 1 Basic variant of K-NN
- 2 Distance metric selection
- 3 Weighted voting

## Weighted voting

Let training set  $x_1, x_2, \dots, x_N$  be rearranged to  $x_{i_1}, x_{i_2}, \dots, x_{i_N}$  by increasing distance to the test pattern  $x$ :

$$d(x, x_{i_1}) \leq d(x, x_{i_2}) \leq \dots \leq d(x, x_{i_N}).$$

Define  $z_1 = x_{i_1}, z_2 = x_{i_2}, \dots, z_K = x_{i_K}$ .

Usual K-NN algorithm can be defined, using  $C$  discriminant functions:

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

Weighted K-NN algorithm uses weighted voting scheme:

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

## Commonly chosen weights

Index dependent weights:

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

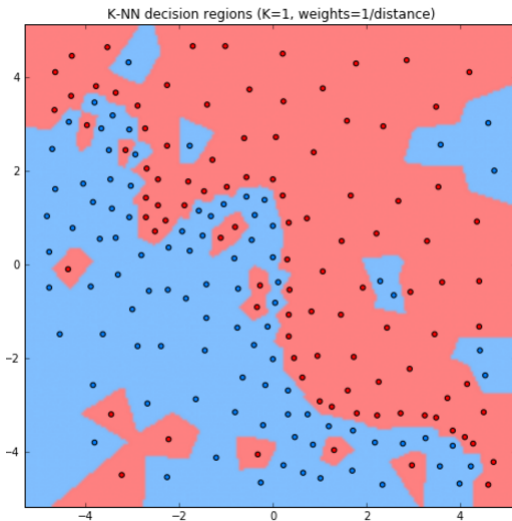
$$w_k = \frac{K + 1 - k}{K}$$

Distance dependent weights:

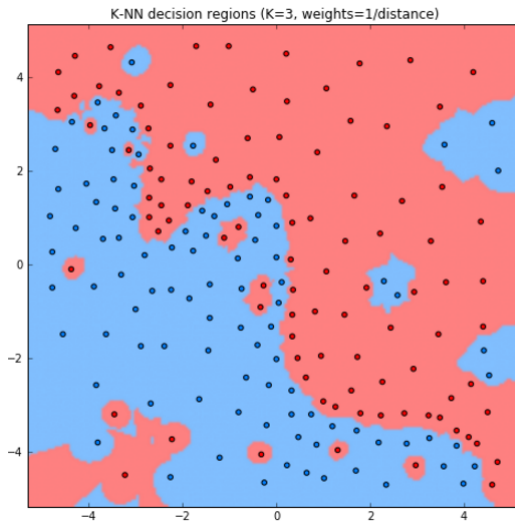
$$w_k = \begin{cases} \frac{d(z_K, x) - d(z_k, x)}{d(z_K, x) - d(z_1, x)}, & d(z_K, x) \neq d(z_1, x) \\ 1 & d(z_K, x) = d(z_1, x) \end{cases}$$

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

## Example: K-NN classification with weights

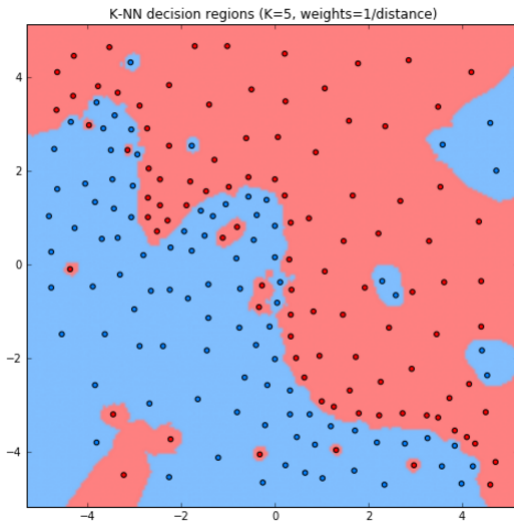


## Example: K-NN classification with weights

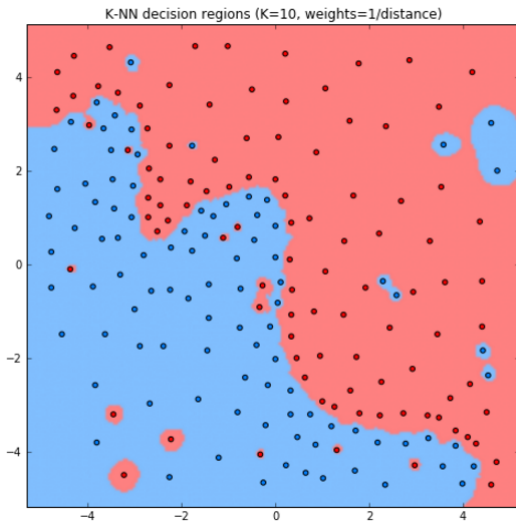




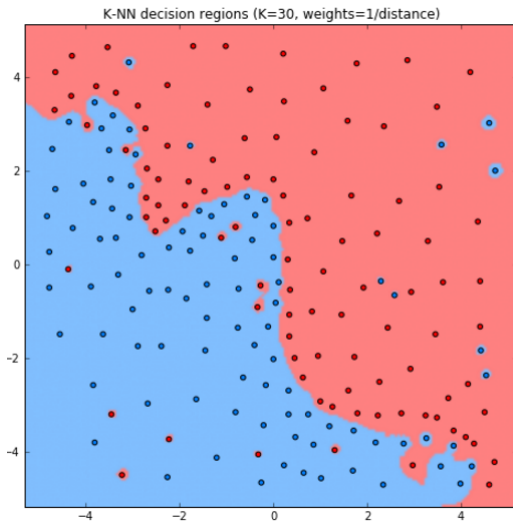
## Example: K-NN classification with weights



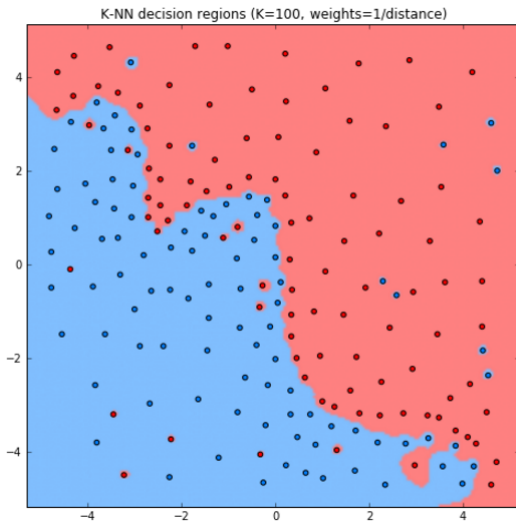
## Example: K-NN classification with weights



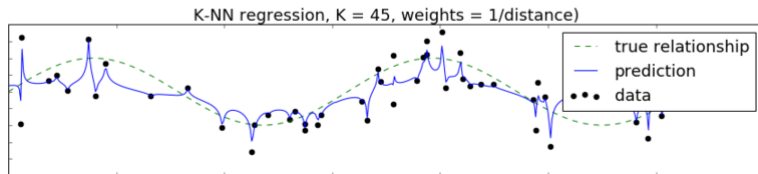
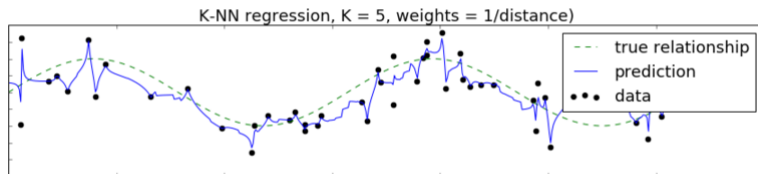
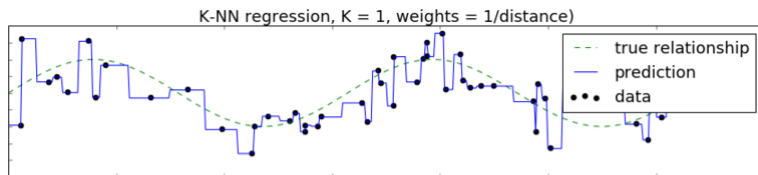
## Example: K-NN classification with weights



## Example: K-NN classification with weights



# Example: K-NN regression with weights



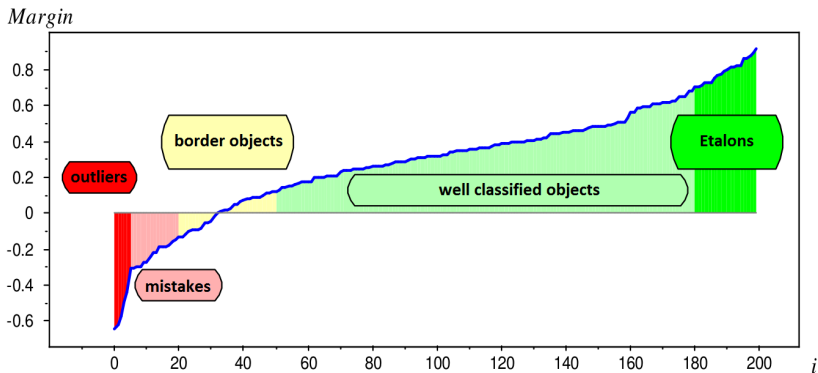
## Margin definition

- Consider the training set:  $(x_1, c_1), (x_2, c_2), \dots (x_N, c_N)$ , where  $c_i$  is the correct class for object  $x_i$ , and  $\mathbf{C} = \{1, 2, \dots, C\}$  - is the set of all classes.
- Define the margin:

$$M(x_i, c_i) = g_{c_i}(x_i) - \max_{c \in \mathbf{C} \setminus \{c_i\}} g_c(x_i)$$

- margin is negative  $\iff$  object  $x_i$  was incorrectly classified
- the value of margin shows how much the classifier is inclined to vote for class  $c_i$

# Categorization of objects based on margin



Good classifier should:

- minimize the number of negative margin region
- classify correctly with high margin