

MACHINE LEARNING COURSE · CS-433

Nearest-Neighbor Classifiers and the Curse of Dimensionality

October 14, 2026

Martin Jaggi

credits to Nicolas Flammarion

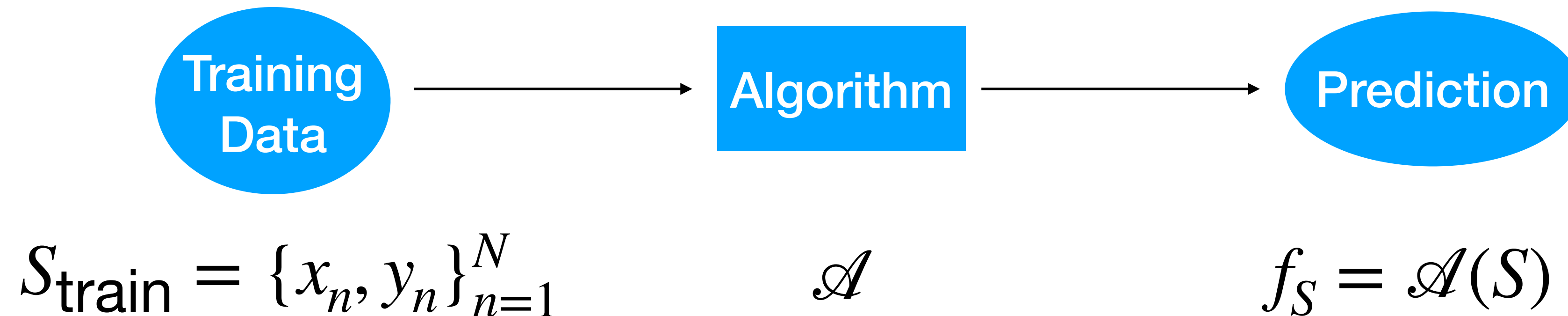
EPFL

Supervised machine learning

We observe some data $S_{\text{train}} = \{x_n, y_n\}_{n=1}^N \in \mathcal{X} \times \mathcal{Y}$

Goal: given a new observation x , we want to predict its label y

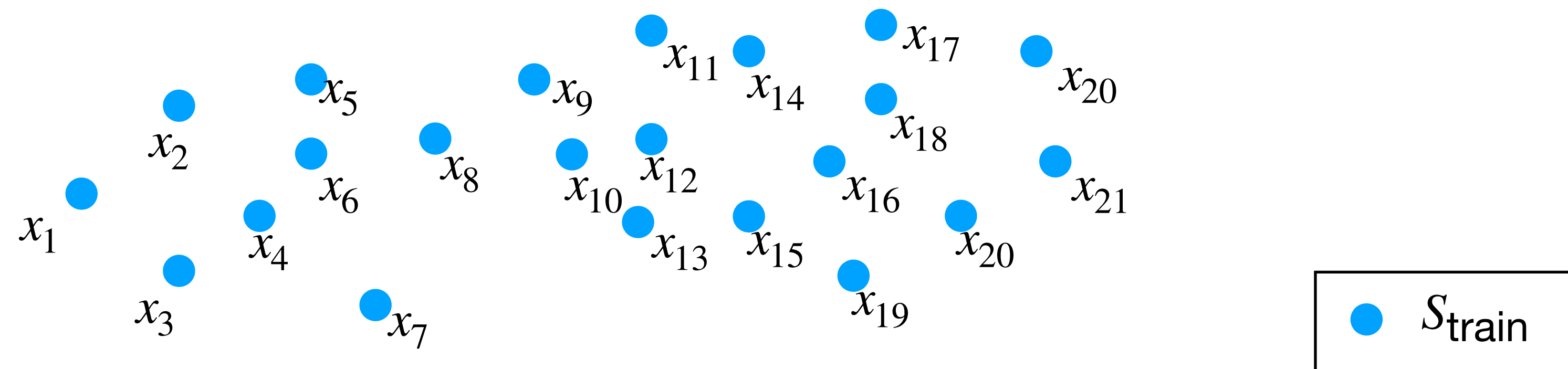
How:



Nearest neighbor function

$$\text{nbh}_{S_{\text{train}},k}: \mathcal{X} \rightarrow \mathcal{X}^k$$

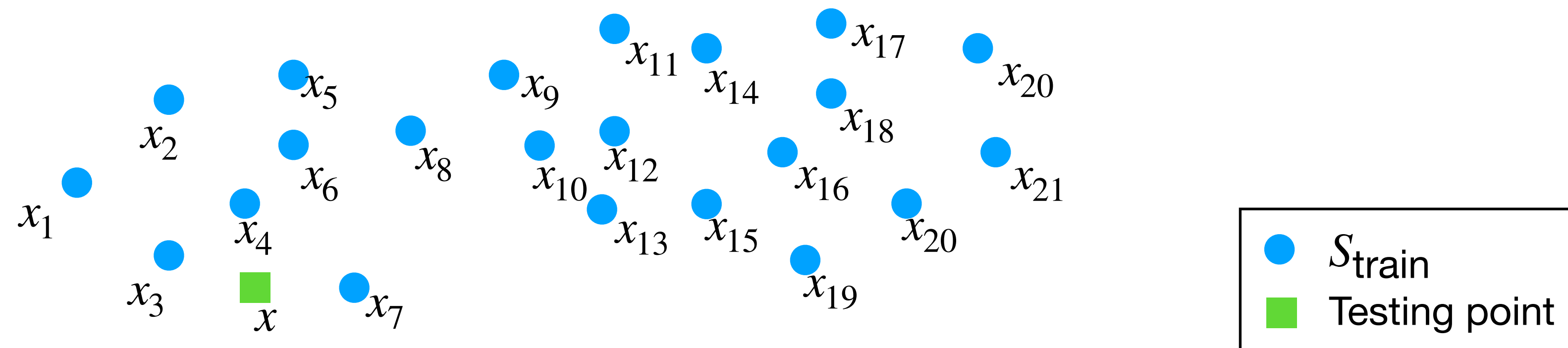
$x \mapsto \{\text{the } k \text{ elements of } S_{\text{train}} \text{ closest to } x\}$



Nearest neighbor function

$$\text{nbh}_{S_{\text{train}},k}: \mathcal{X} \rightarrow \mathcal{X}^k$$

$x \mapsto \{\text{the } k \text{ elements of } S_{\text{train}} \text{ closest to } x\}$

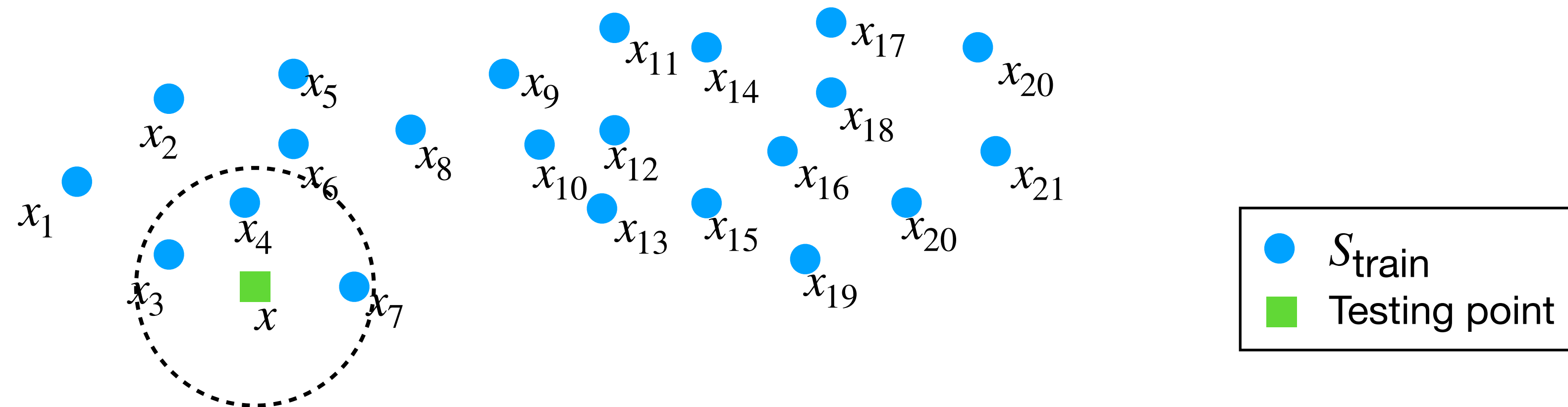


$$\text{nbh}_{S_{\text{train}},3}(x) = ?$$

Nearest neighbor function

$$\text{nbh}_{S_{\text{train}},k}: \mathcal{X} \rightarrow \mathcal{X}^k$$

$x \mapsto \{\text{the } k \text{ elements of } S_{\text{train}} \text{ closest to } x\}$

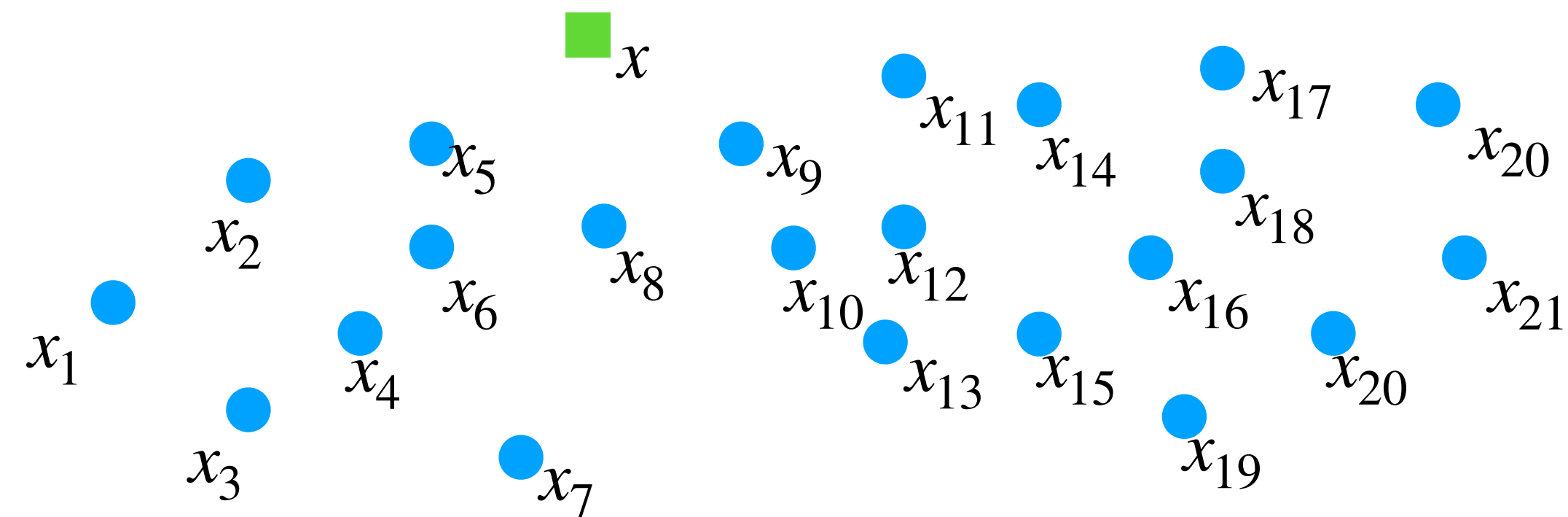


$$\text{nbh}_{S_{\text{train}},3}(x) = \{x_3, x_4, x_7\}$$

Nearest neighbor function

$$\text{nbh}_{S_{\text{train}},k}: \mathcal{X} \rightarrow \mathcal{X}^k$$

$x \mapsto \{\text{the } k \text{ elements of } S_{\text{train}} \text{ closest to } x\}$

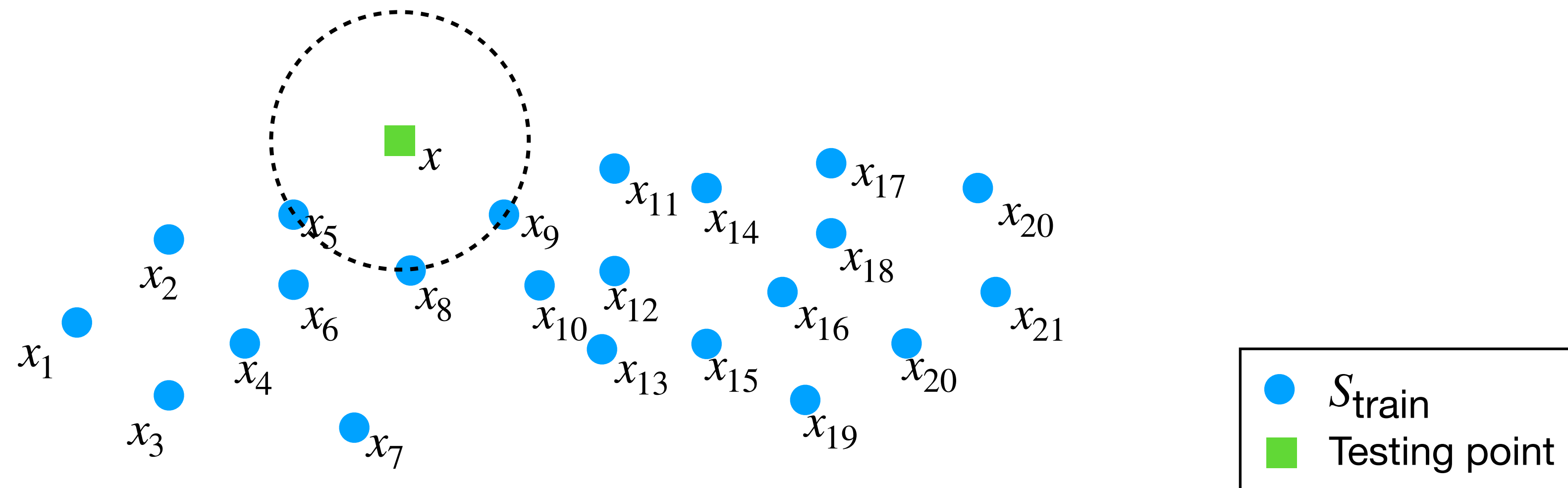


$$\text{nbh}_{S_{\text{train}},2}(x) = ?$$

Nearest neighbor function

$$\text{nbh}_{S_{\text{train}},k}: \mathcal{X} \rightarrow \mathcal{X}^k$$

$x \mapsto \{\text{the } k \text{ elements of } S_{\text{train}} \text{ closest to } x\}$

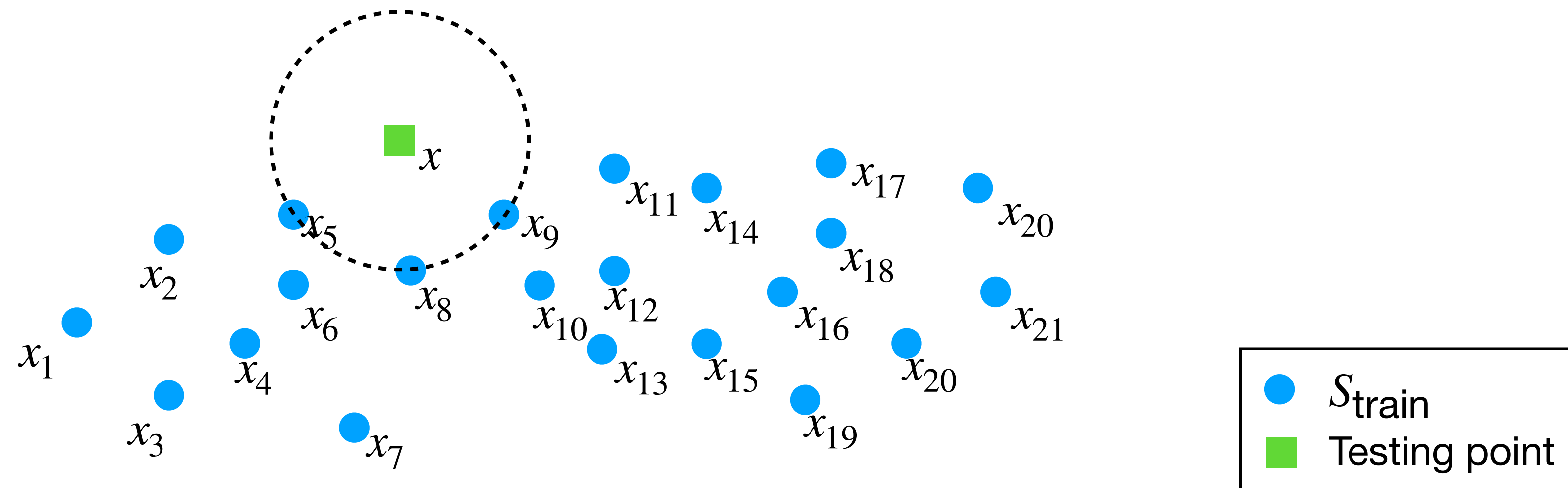


$$\text{nbh}_{S_{\text{train}},2}(x) = ?$$

Nearest neighbor function

$$\text{nbh}_{S_{\text{train}},k}: \mathcal{X} \rightarrow \mathcal{X}^k$$

$x \mapsto \{\text{the } k \text{ elements of } S_{\text{train}} \text{ closest to } x\}$



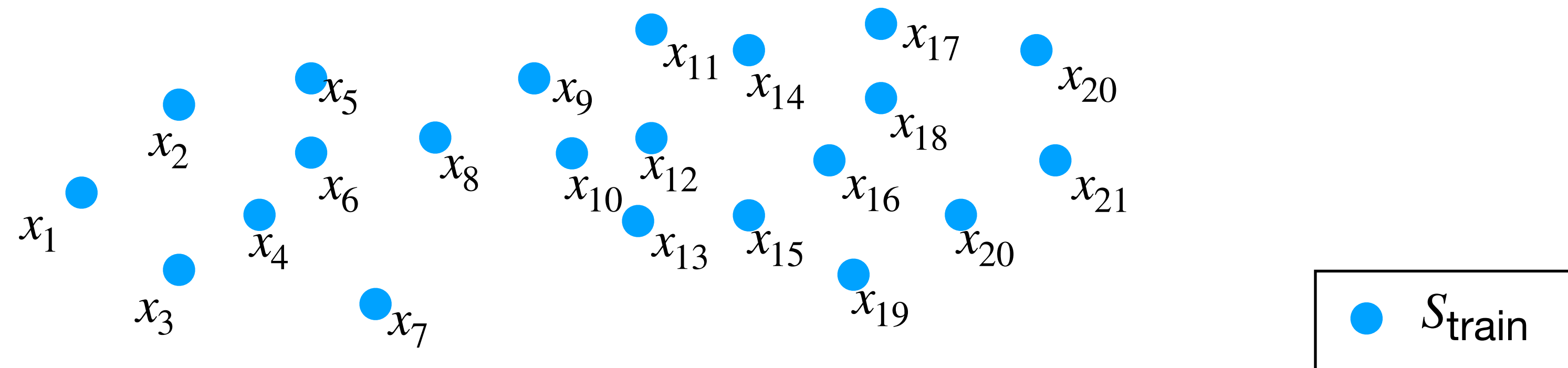
$$\text{nbh}_{S_{\text{train}},2}(x) = \{x_5, x_8\}$$

Not uniquely defined!
The result depends on the implementation
Ties are often broken randomly

Nearest neighbor function

$$\text{nbh}_{S_{\text{train}},k}: \mathcal{X} \rightarrow \mathcal{X}^k$$

$x \mapsto \{\text{the } k \text{ elements of } S_{\text{train}} \text{ closest to } x\}$

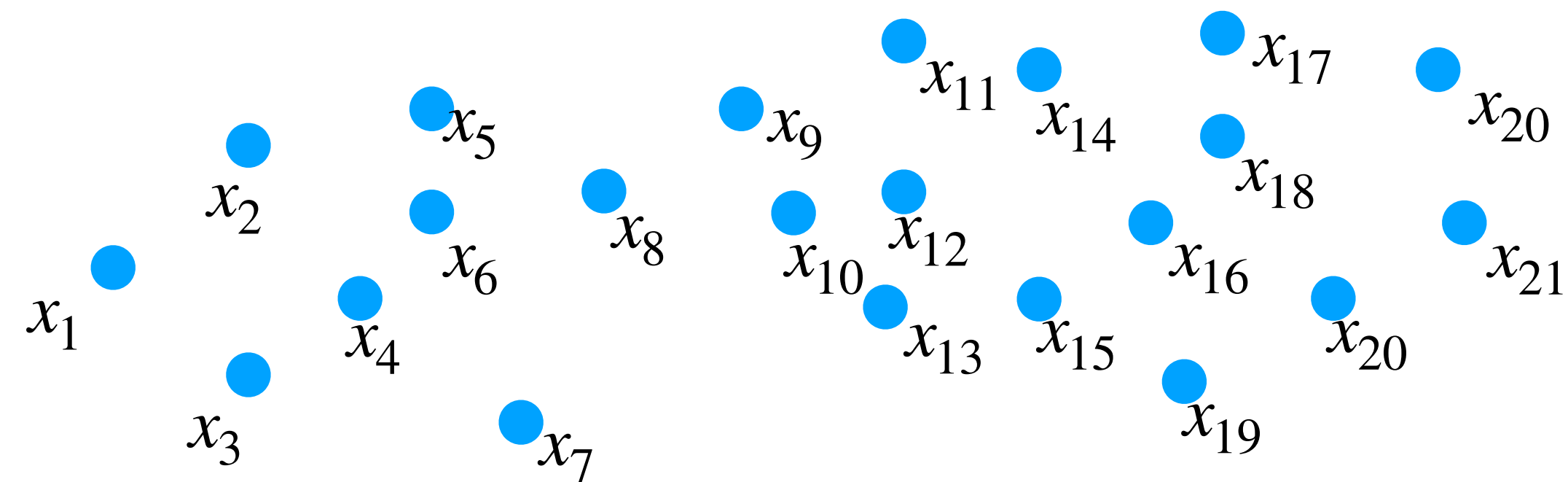


Remarks:

- Different metrics can be employed
- High computational complexity for large N (but efficient data structure may exist)

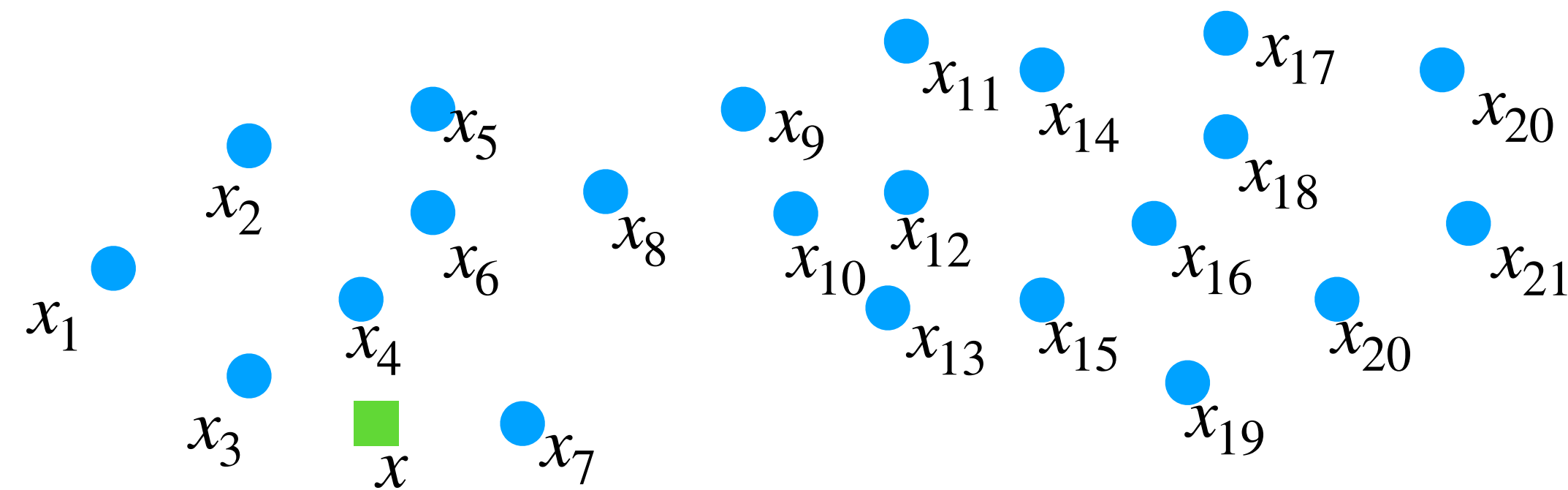
k-NN can be used for regression ($y \in \mathbb{R}$)

$$f_{S_{train},k}(x) = \frac{1}{k} \sum_{n: x_n \in nbh_{S_{train},k}(x)} y_n$$



k-NN can be used for regression ($y \in \mathbb{R}$)

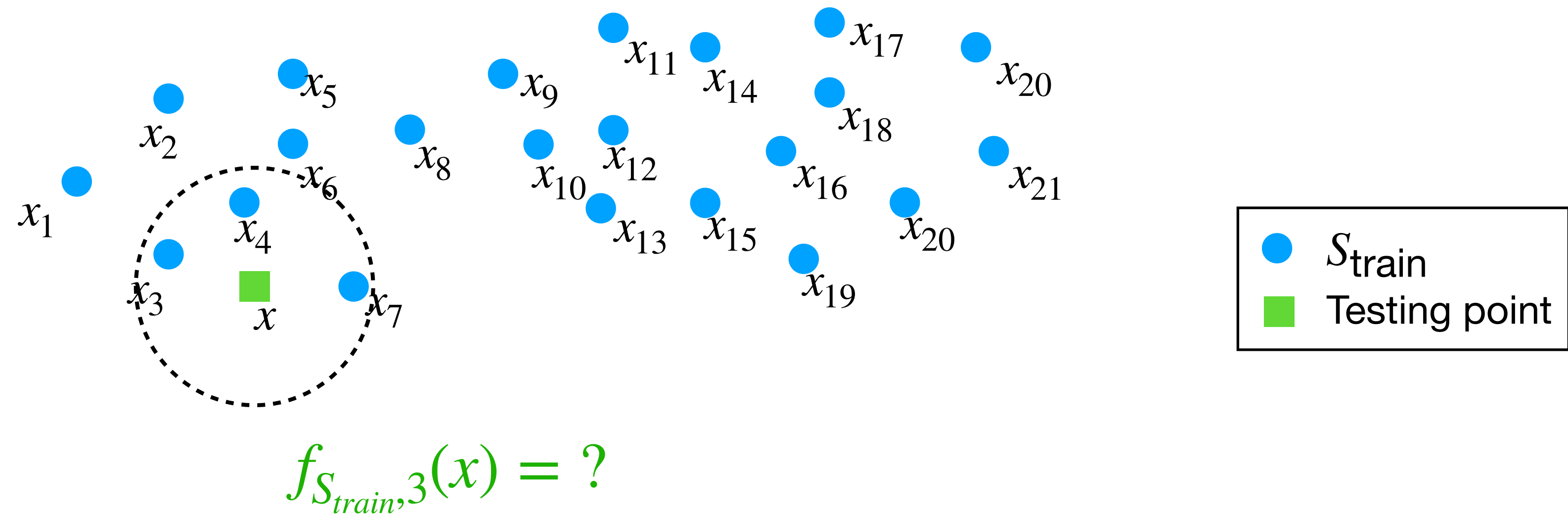
$$f_{S_{train},k}(x) = \frac{1}{k} \sum_{n: x_n \in nbh_{S_{train},k}(x)} y_n$$



$$f_{S_{train},3}(x) = ?$$

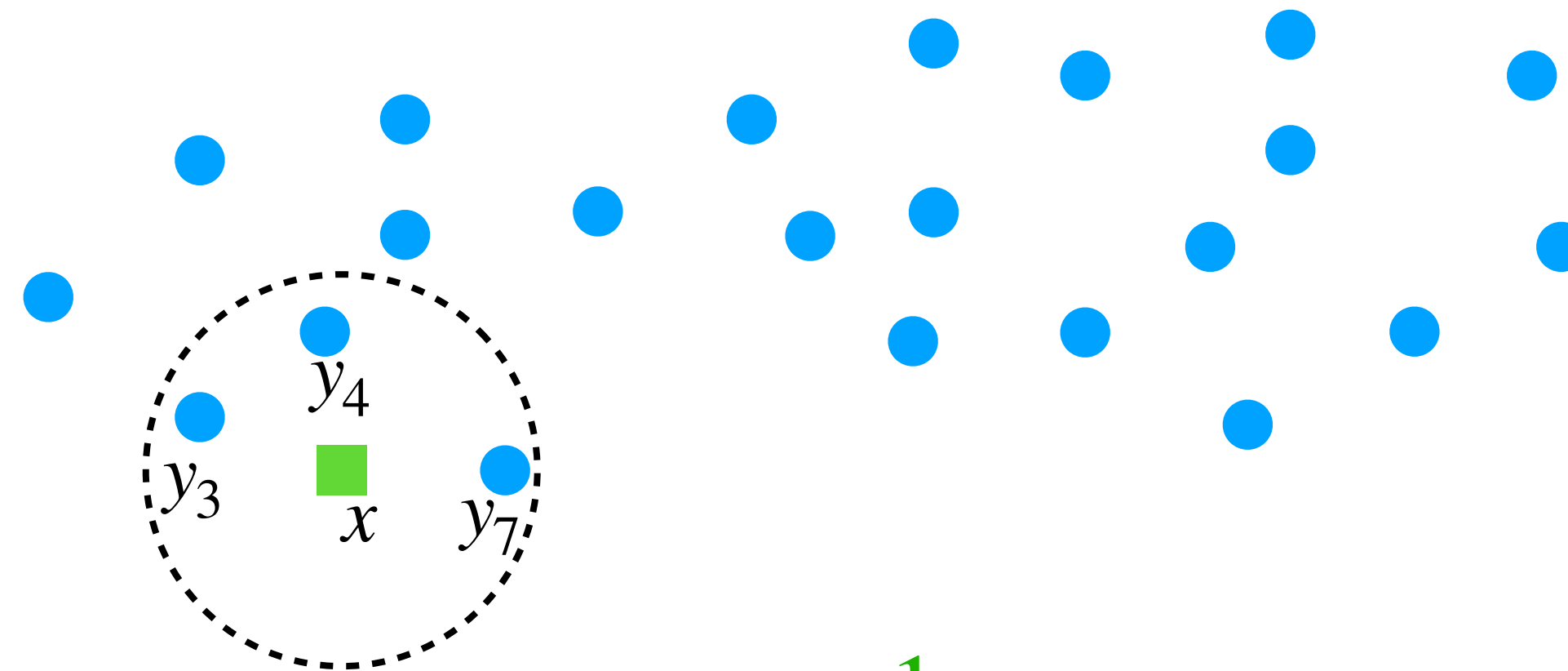
k-NN can be used for regression ($y \in \mathbb{R}$)

$$f_{S_{train},k}(x) = \frac{1}{k} \sum_{n: x_n \in nbh_{S_{train},k}(x)} y_n$$



k-NN can be used for regression ($y \in \mathbb{R}$)

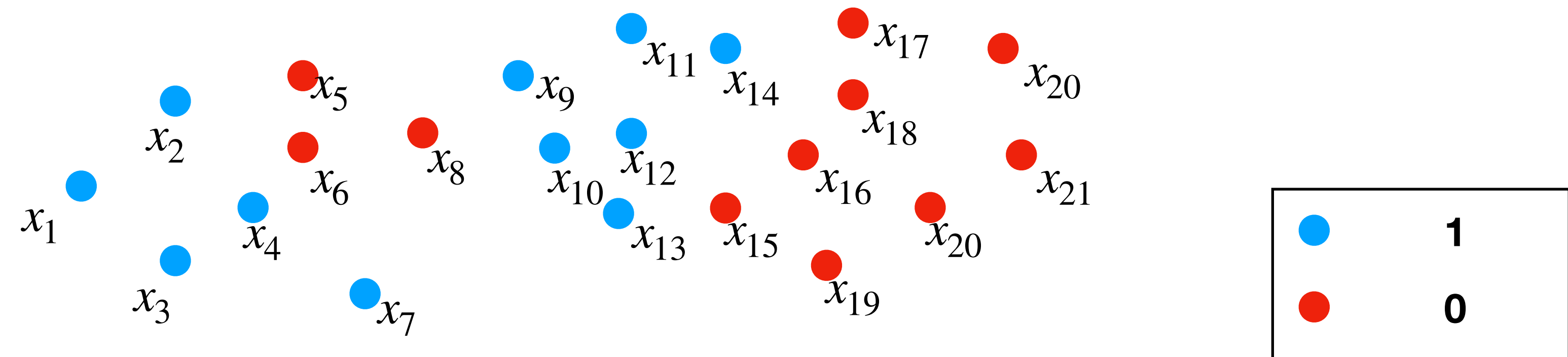
$$f_{S_{train},k}(x) = \frac{1}{k} \sum_{n: x_n \in nbh_{S_{train},k}(x)} y_n$$



$$f_{S_{train},3}(x) = \frac{1}{3} (y_3 + y_4 + y_7)$$

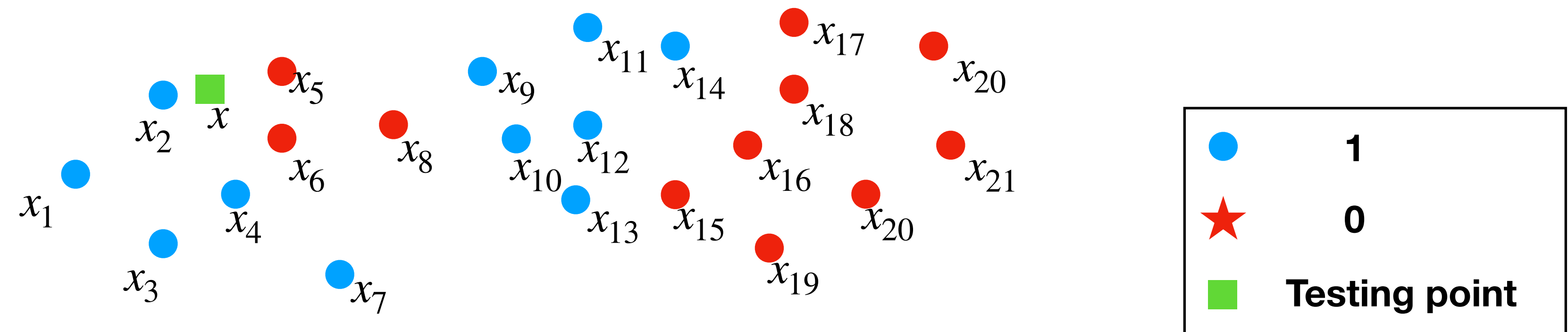
k-NN can be used for classification ($y \in \{0,1\}$)

$$f_{S_{train},k}(x) = \text{majority} \{ y_n : x_n \in \text{nbh}_{S_{train},k}(x) \}$$



k-NN can be used for classification ($y \in \{0,1\}$)

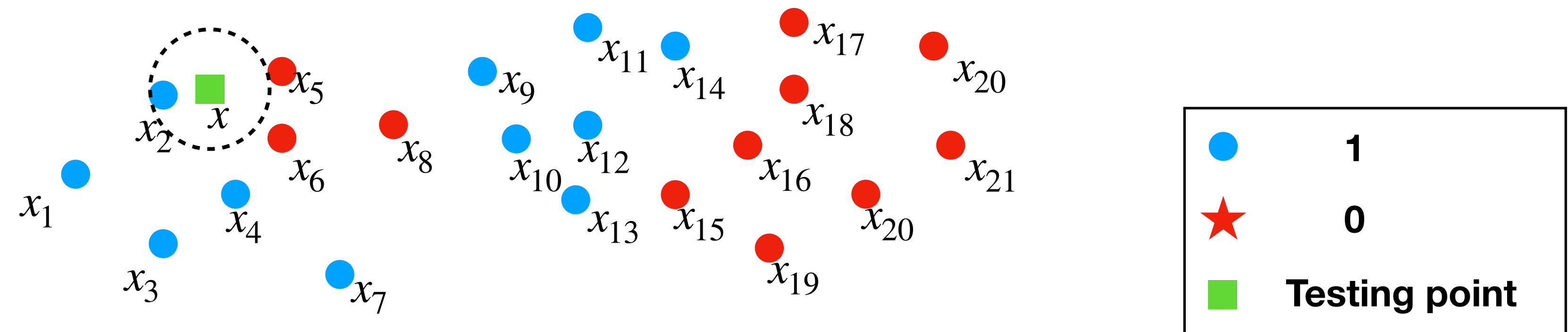
$$f_{S_{train},k}(x) = \text{majority} \{ y_n : x_n \in \text{nbh}_{S_{train},k}(x) \}$$



$$f_{S_{train},1}(x) = ?$$

k-NN can be used for classification ($y \in \{0,1\}$)

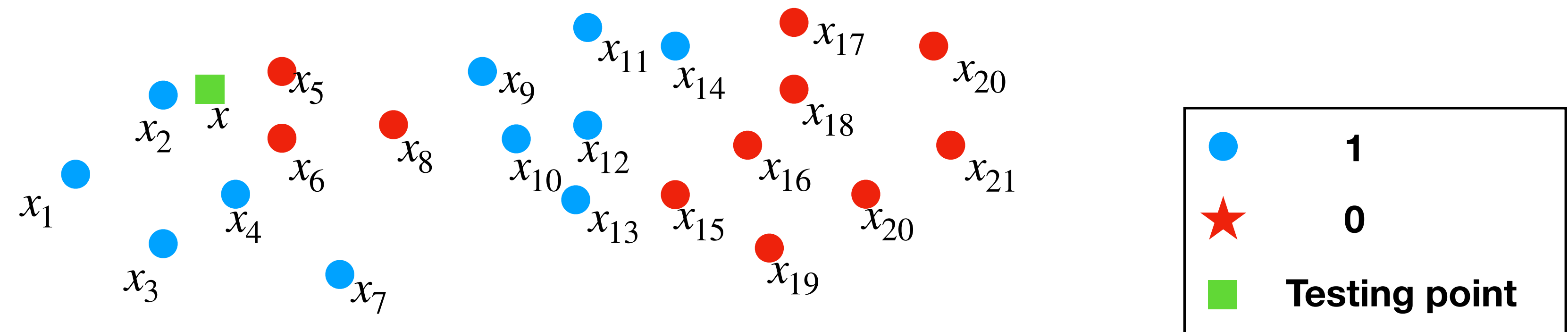
$$f_{S_{train},k}(x) = \text{majority} \{ y_n : x_n \in \text{nbh}_{S_{train},k}(x) \}$$



$$f_{S_{train},1}(x) = 1$$

k-NN can be used for classification ($y \in \{0,1\}$)

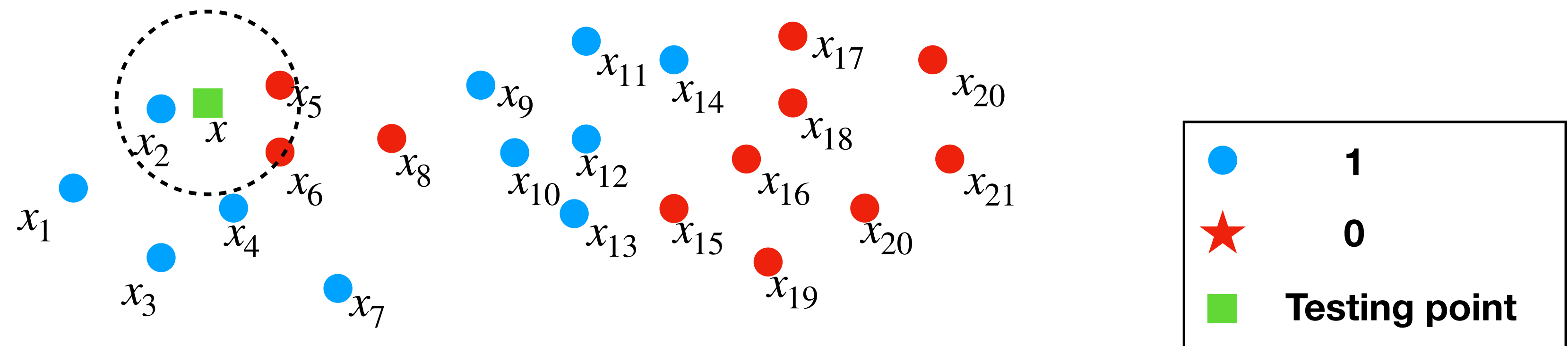
$$f_{S_{train},k}(x) = \text{majority} \{ y_n : x_n \in \text{nbh}_{S_{train},k}(x) \}$$



$$f_{S_{train},3}(x) = ?$$

k-NN can be used for classification ($y \in \{0,1\}$)

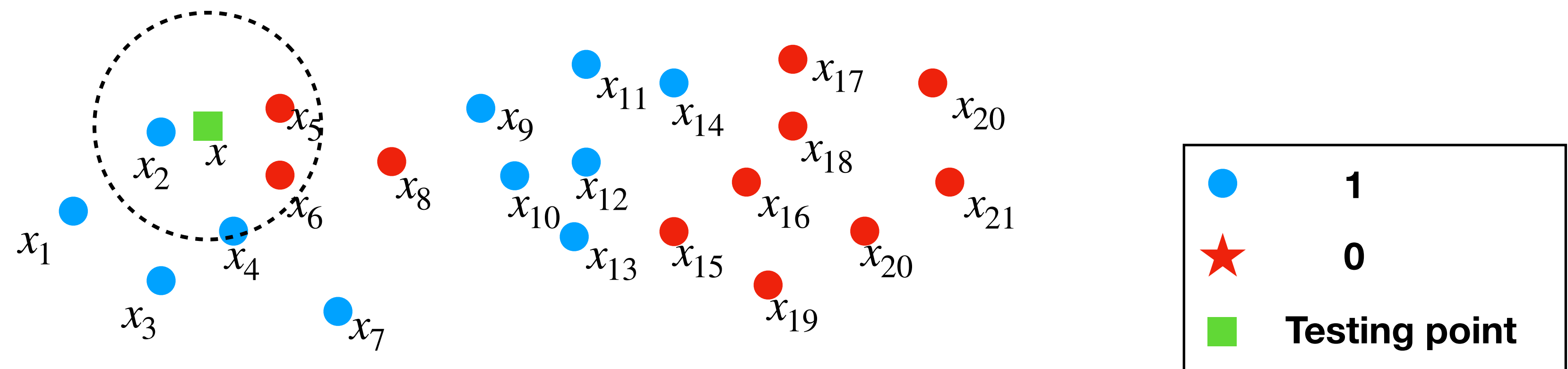
$$f_{S_{train},k}(x) = \text{majority} \{ y_n : x_n \in \text{nbh}_{S_{train},k}(x) \}$$



$$f_{S_{train},3}(x) = 0$$

k-NN can be used for classification ($y \in \{0,1\}$)

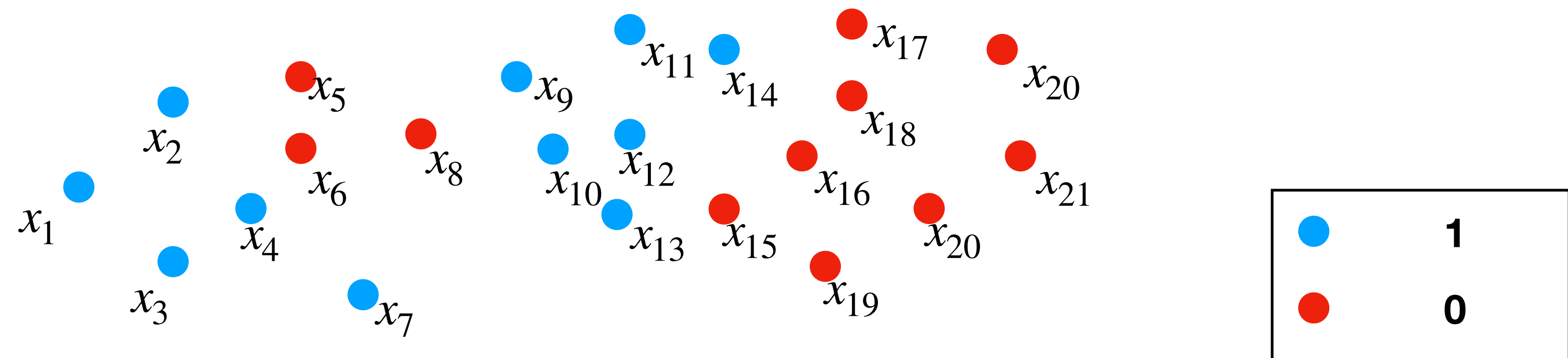
$$f_{S_{train},k}(x) = \text{majority} \{ y_n : x_n \in \text{nbh}_{S_{train},k}(x) \}$$



$$f_{S_{train},4}(x) = ? \quad \textbf{Tie!}$$

k-NN can be used for classification ($y \in \{0,1\}$)

$$f_{S_{train},k}(x) = \text{majority} \{ y_n : x_n \in \text{nbh}_{S_{train},k}(x) \}$$

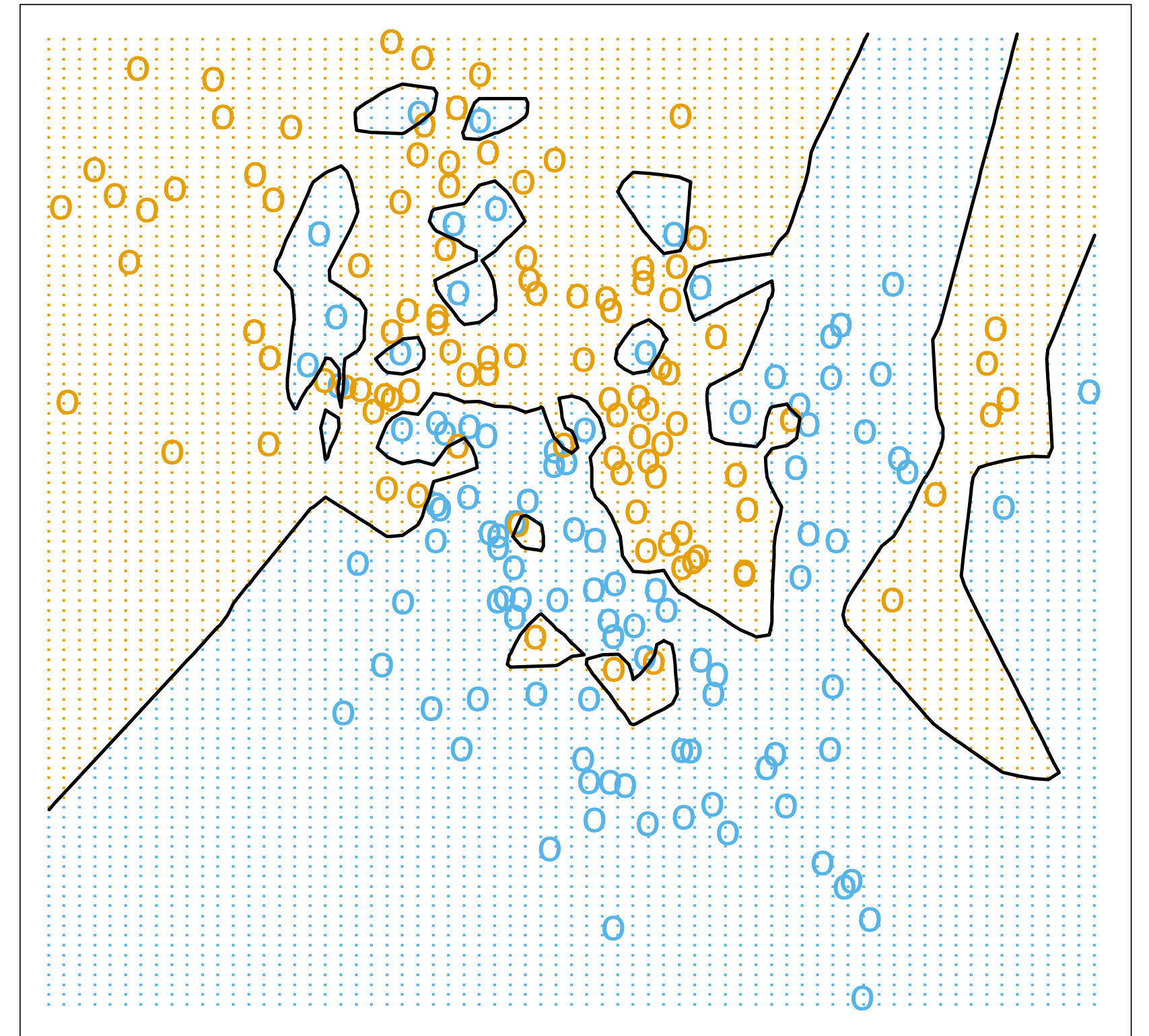


Remarks:

- Choose an odd value for k to prevent ties
- Generalization: smoothing kernels; weighted linear combination of elements

Why does it make sense?

- Relevant in the presence of spatial correlation
- Implicitly models intricate decision boundaries in low-dimensional spaces



Bias-variance tradeoff in k-NN

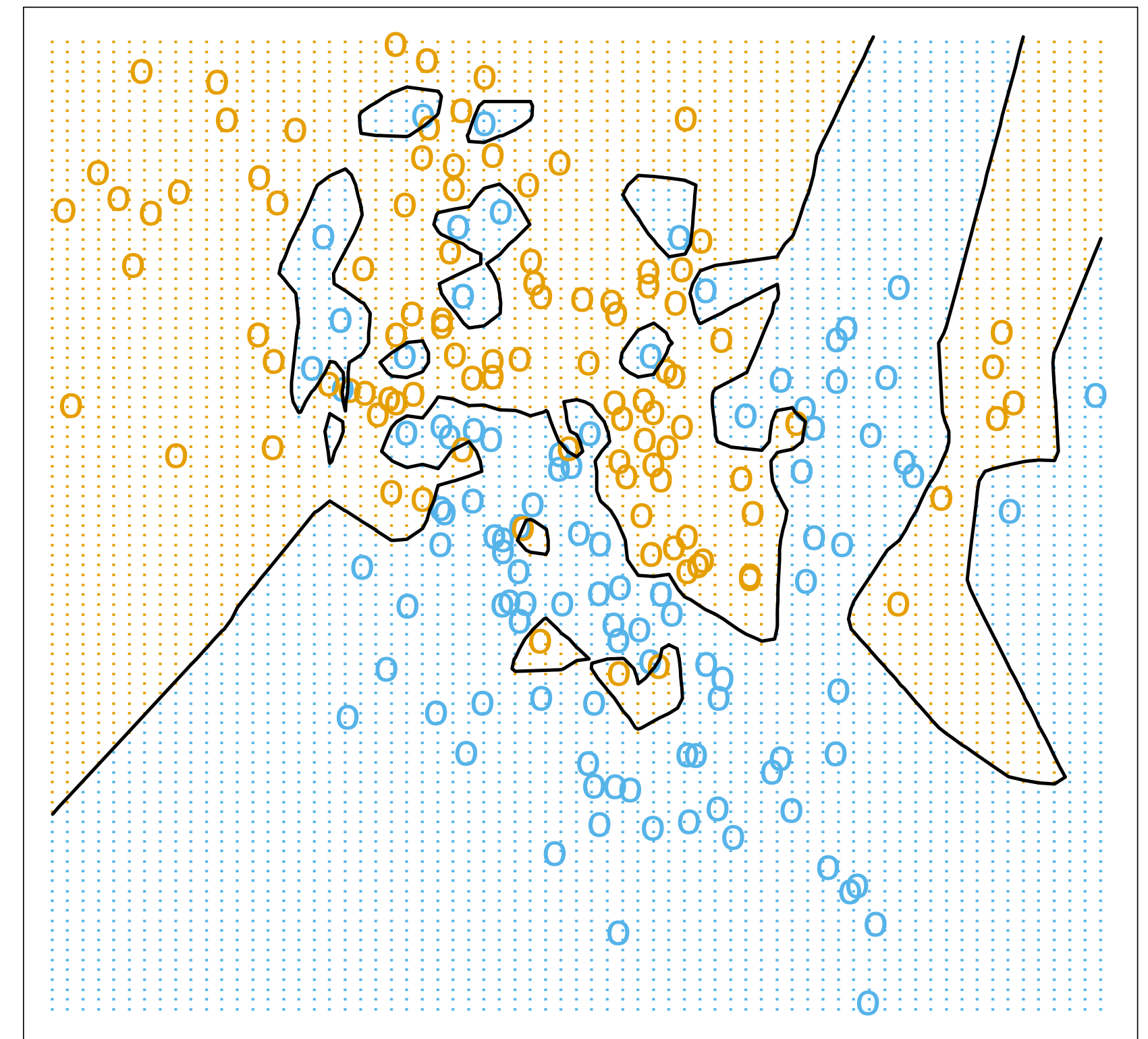
For small k:

- Low bias - complex decision boundary
- High variance - overfitting

For large k:

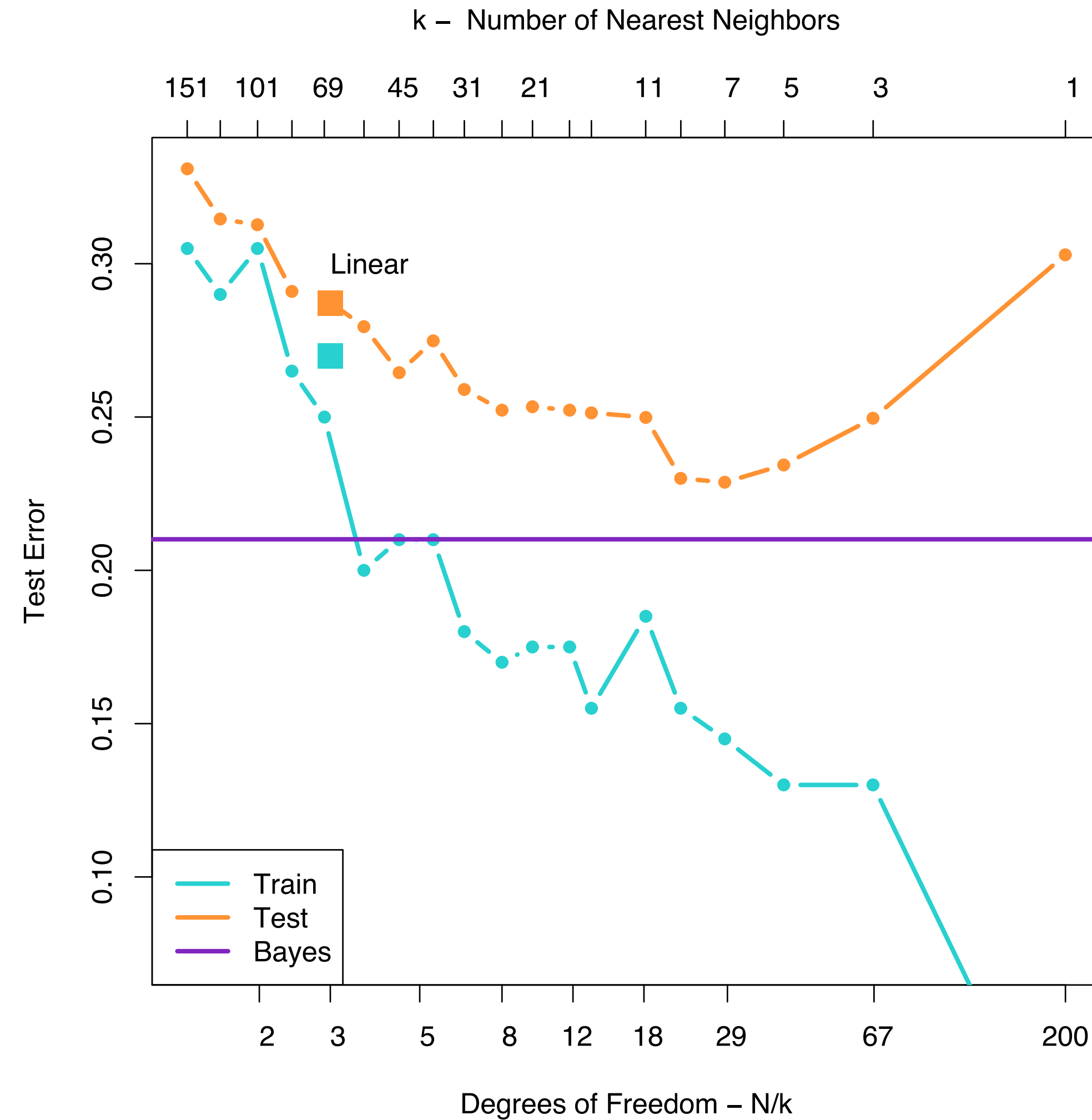
(When $k = N$, prediction is constant)

- High bias
- Low variance



1-nearest neighbor classification

U-shaped curve for k-NN bias-variance tradeoff

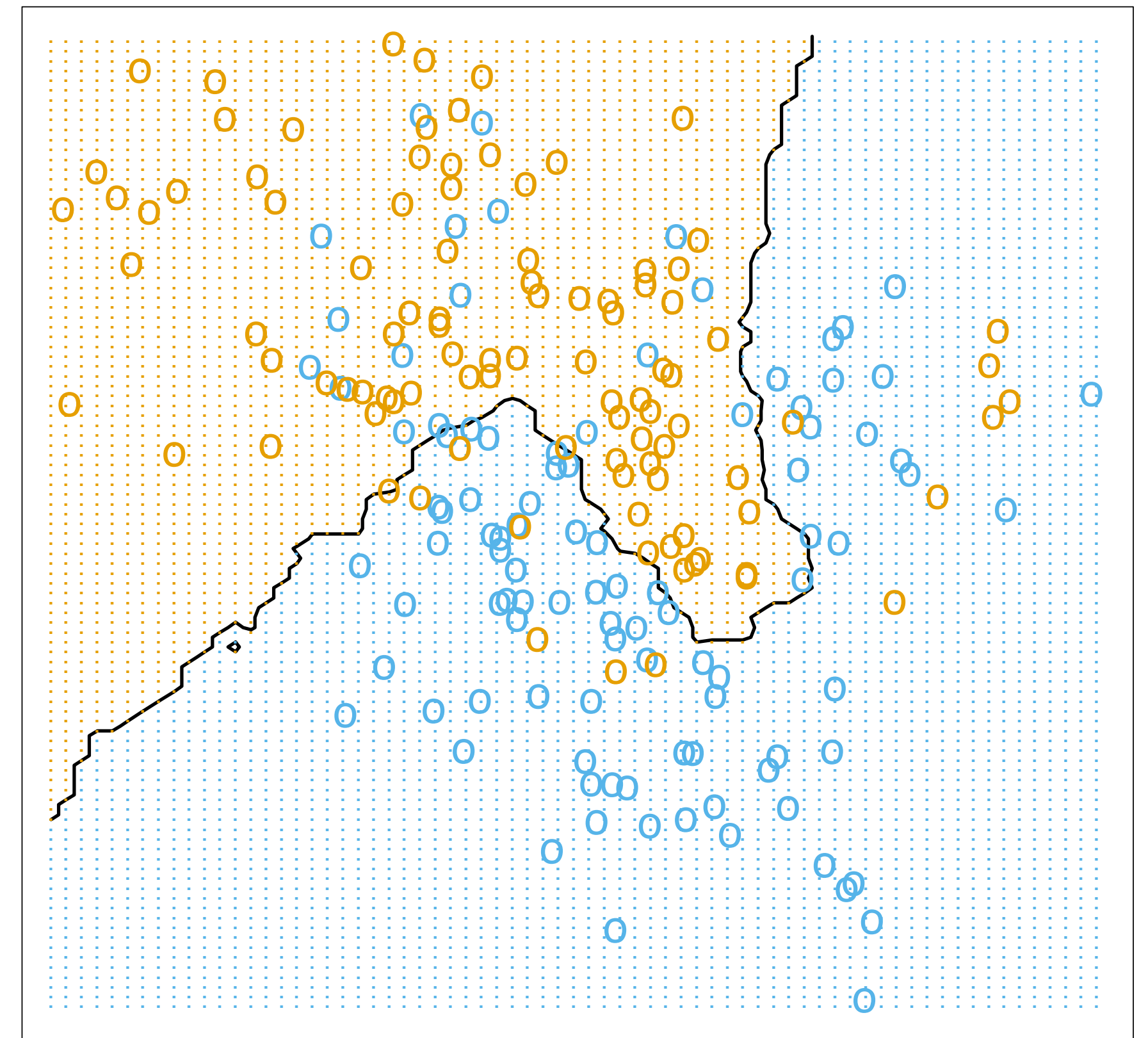


Complexity increases as k decreases

Find a k that balances bias and variance

Characteristics of an optimal k :

- Low bias: Ensures a sufficiently complex decision boundary
- Low variance: Prevents overfitting



15-nearest neighbor classification

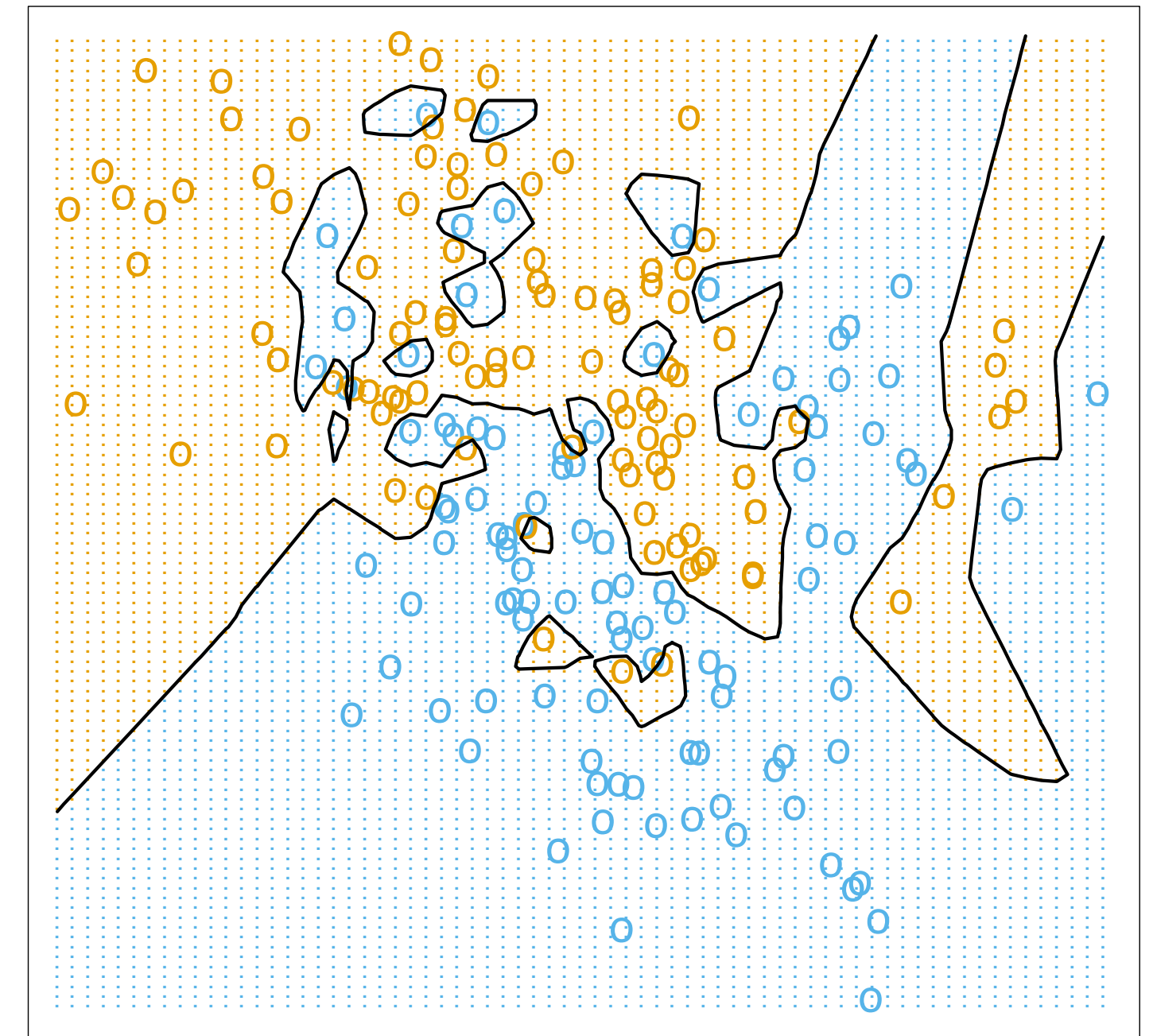
Summary: k-Nearest Neighbor

Pros:

- **No optimization** or training
- **Easy** to implement
- Works well in **low dimensions**, allowing for very complex decision boundaries

Cons:

- **Slow** at query time
- Not suitable for high-dimensional data
- Choosing the right local distance is crucial



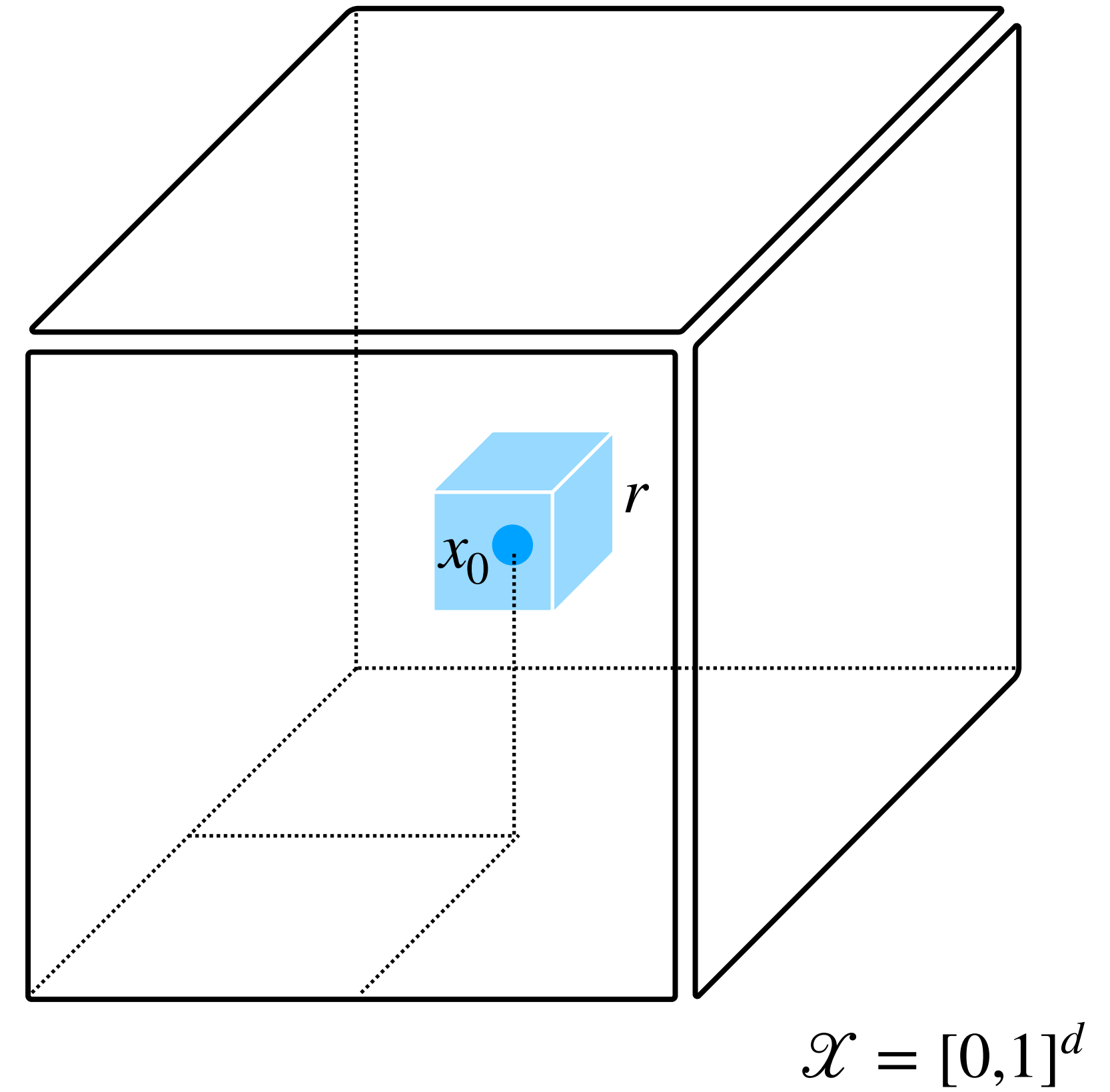
Curse of dimensionality

Claim 1: As the dimensionality grows, fixed-size training sets cover a diminishing fraction of the input space

Assume the data $x \sim \mathcal{U}([0,1]^d)$

Consider a blue box around the center x_0 of size r

$$\mathbb{P}(x \in \text{blue box}) = r^d := \alpha$$



Curse of dimensionality

Claim 1: As the dimensionality grows, fixed-size training sets cover a diminishing fraction of the input space

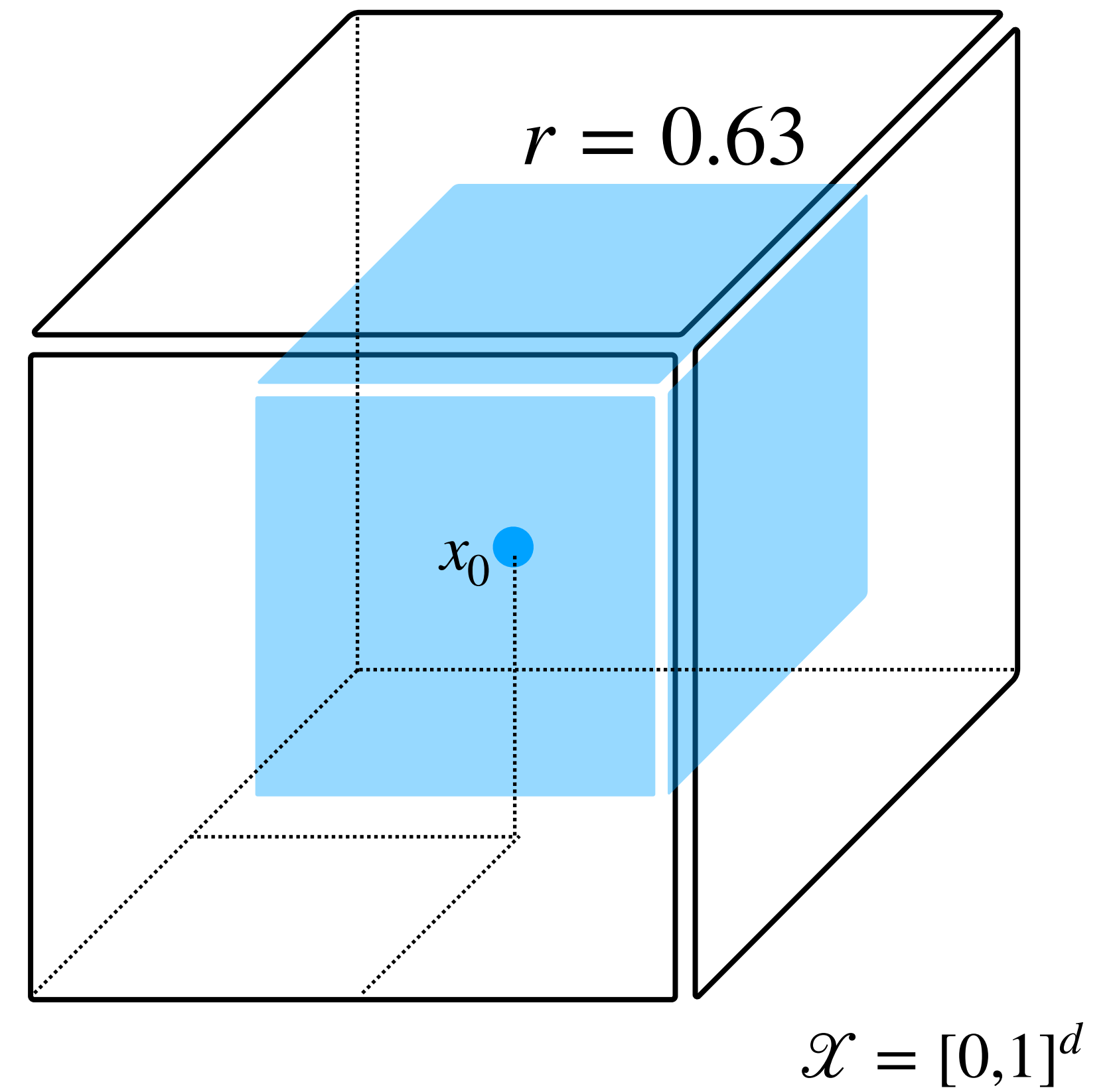
Assume the data $x \sim \mathcal{U}([0,1]^d)$

Consider a blue box around the center x_0 of size r

$$\mathbb{P}(x \in \text{blue box}) = r^d := \alpha$$

If $\alpha = 0.01$, to have:

$d = 10$, we need $r = 0.63$



Curse of dimensionality

Claim 1: As the dimensionality grows, fixed-size training sets cover a diminishing fraction of the input space

Assume the data $x \sim \mathcal{U}([0,1]^d)$

Consider a blue box around the center x_0 of size r

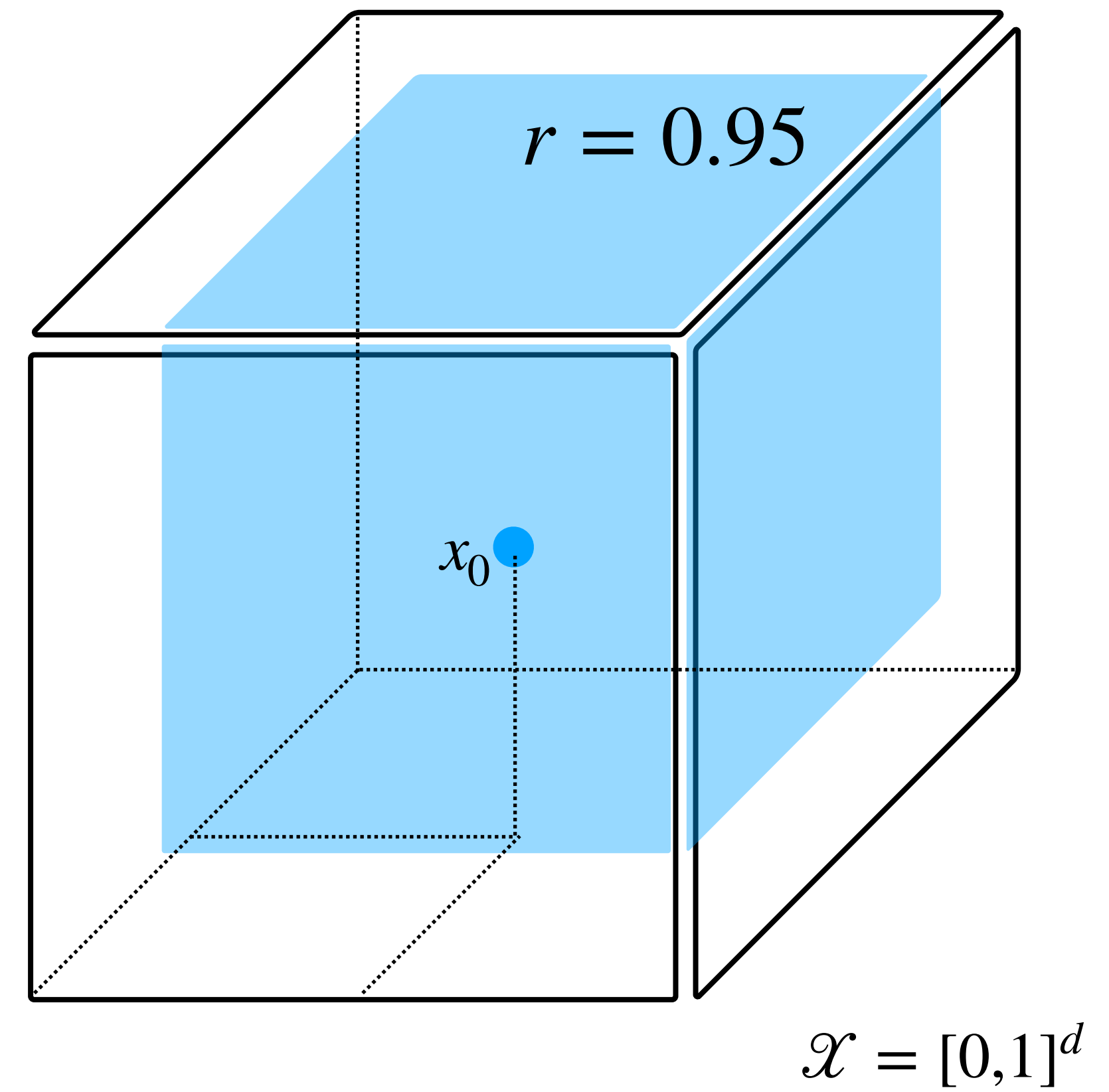
$$\mathbb{P}(x \in \text{blue box}) = r^d := \alpha$$

If $\alpha = 0.01$, to have:

$d = 10$, we need $r = 0.63$

$d = 100$, we need $r = 0.95$

We need to explore almost the whole box



Curse of dimensionality

Claim 2: In high-dimension, data-points are far from each other.

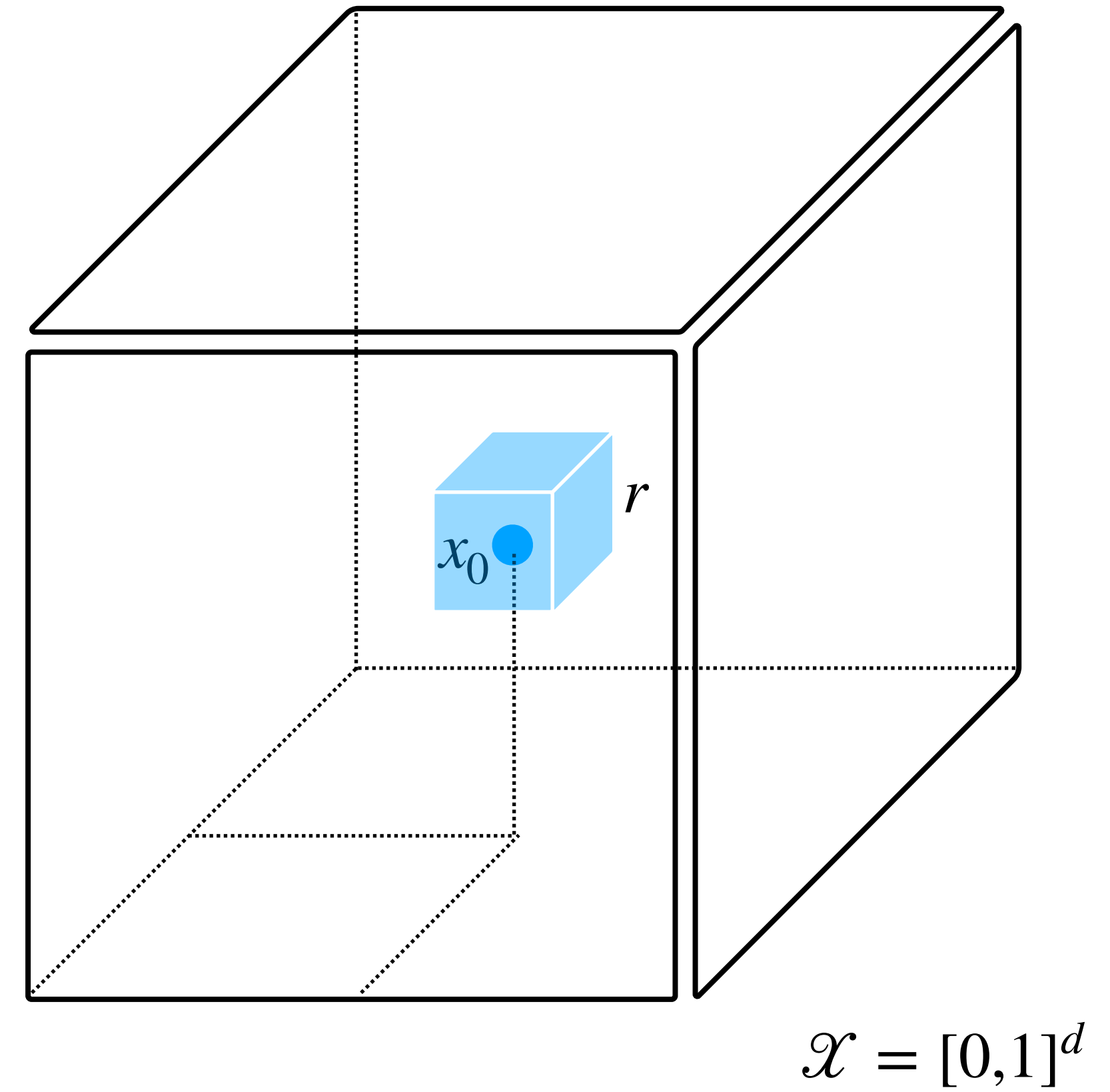
Consider N i.i.d. points uniform in the $[0,1]^d$

$$\mathbb{P}(\exists x_i \in \text{cube}) \geq 1/2 \implies r \geq \left(1 - \frac{1}{2^{1/N}}\right)^{1/d}$$

Proof: $\mathbb{P}(x \notin \text{cube}) = 1 - r^d$

$$\mathbb{P}(x_i \notin \text{cube}, \forall i \leq N) = (1 - r^d)^N$$

$$\mathbb{P}(\exists x_i \in \text{cube}) = 1 - (1 - r^d)^N$$



Curse of dimensionality

Claim 2: In high-dimension, data-points are far from each other.

Consider N i.i.d. points uniform in the $[0,1]^d$

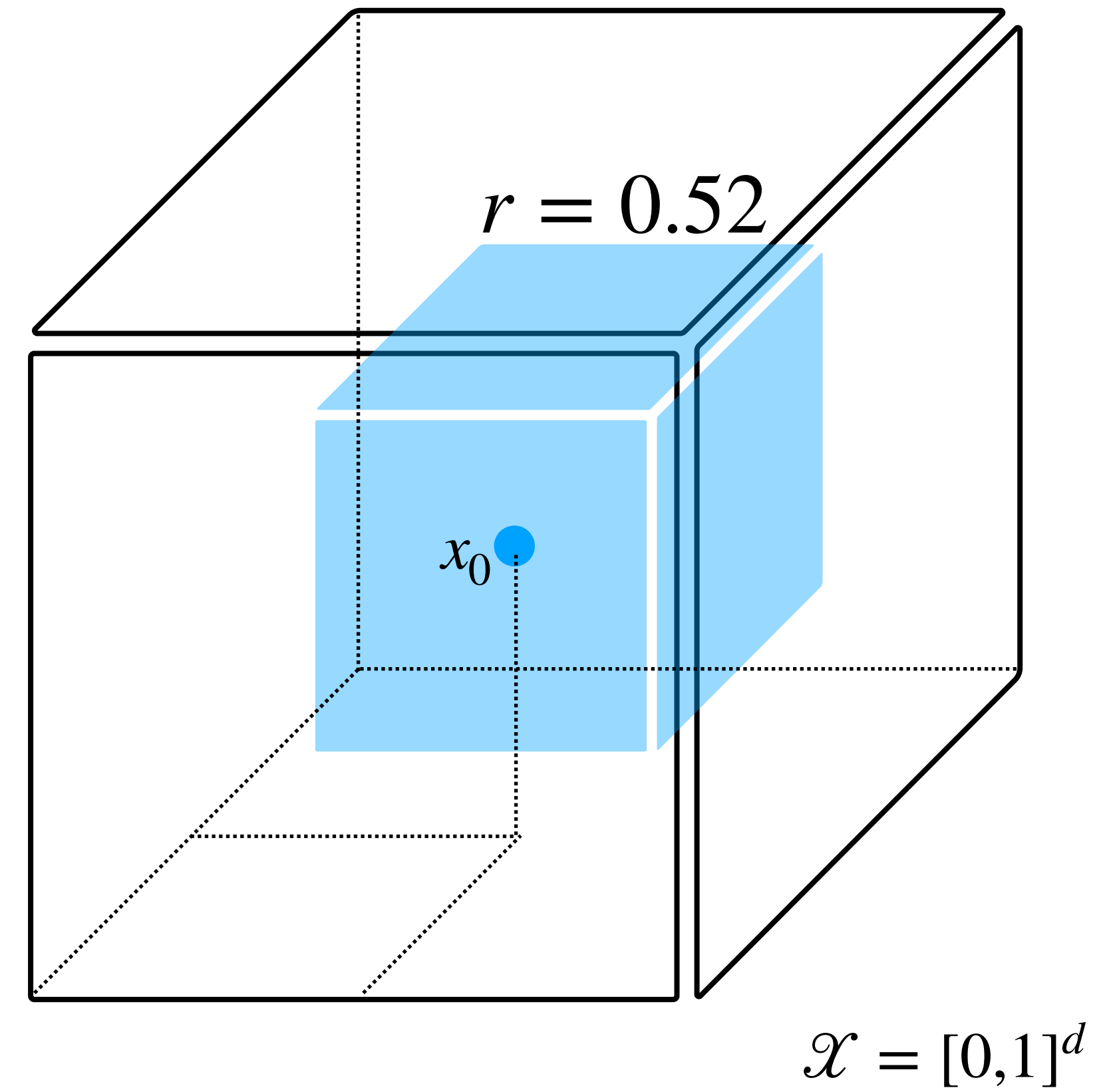
$$\mathbb{P}(\exists x_i \in \text{cube}) \geq 1/2 \implies r \geq \left(1 - \frac{1}{2^{1/N}}\right)^{1/d}$$

Proof: $\mathbb{P}(x \notin \text{cube}) = 1 - r^d$

$$\mathbb{P}(x_i \notin \text{cube}, \forall i \leq N) = (1 - r^d)^N$$

$$\mathbb{P}(\exists x_i \in \text{cube}) = 1 - (1 - r^d)^N$$

For $d = 10$, $N = 500$, we have $r \geq 0.52$



Generalization bound for 1-NN

Setup: $(X, Y) \sim \mathcal{D}$ over $\mathcal{X} \times \mathcal{Y} = [0,1]^d \times \{0,1\}$

Goal: Bound the classification error:

$$L(f) = \mathbb{P}_{(X,Y) \sim \mathcal{D}}(Y \neq f(X))$$

Baseline:

- Bayes classifier: minimizes L over all classifiers

$$f_*(x) = 1_{\eta(x) \geq 1/2} \text{ where } \eta(x) = \mathbb{P}(Y = 1 \mid X = x)$$

- Bayes risk: represents the minimum probability of misclassification

$$L(f_*) = \mathbb{P}(f_*(X) \neq Y) = \mathbb{E}_{X \sim \mathcal{D}_X}[\min\{\eta(X), 1 - \eta(X)\}]$$

Generalization bound for 1-NN

Setup: $(X, Y) \sim \mathcal{D}$ over

Goal: Bound the classification

Baseline:

Proof 1:

$$\eta(x) \geq 1/2 \iff \mathbb{P}(Y = 1 | X = x) \geq 1/2$$

$$\iff \mathbb{P}(Y = 1 | X = x) \geq \mathbb{P}(Y = 0 | X = x)$$

$$\iff 1 \in \arg \max_{y \in \{0,1\}} \mathbb{P}(Y = y | X = x)$$

$$\text{Thus } 1_{\eta(x) \geq 1/2} = \arg \max_{y \in \{0,1\}} \mathbb{P}(Y = y | X = x) = f_*(x)$$

- Bayes classifier: minimizes L over all classifiers

$$f_*(x) = 1_{\eta(x) \geq 1/2} \text{ where } \eta(x) = \mathbb{P}(Y = 1 | X = x)$$

- Bayes risk: represents the minimum probability of misclassification

$$L(f_*) = \mathbb{P}(f_*(X) \neq Y) = \mathbb{E}_{X \sim \mathcal{D}_X}[\min\{\eta(X), 1 - \eta(X)\}]$$

Generalization bound for 1-NN

Proof 2:

$$\begin{aligned} L(f_*) &= \mathbb{E}_{(X,Y) \sim \mathcal{D}}[1_{f_*(X) \neq Y}] \\ &= \mathbb{E}_{X \sim \mathcal{D}_X}[\mathbb{E}_{Y \sim \mathcal{D}_{Y|X}}[1_{f_*(X) \neq Y} | X]] \\ &= \mathbb{E}_{X \sim \mathcal{D}_X}[\mathbb{E}_{Y \sim \mathcal{D}_{Y|X}}[1_{f_*(X) \neq Y} | X] 1_{\eta(X) \geq 1/2} + \mathbb{E}_{Y \sim \mathcal{D}_{Y|X}}[1_{f_*(X) \neq Y} | X] 1_{\eta(X) < 1/2}] \\ &= \mathbb{E}_{X \sim \mathcal{D}_X}[\mathbb{E}_{Y \sim \mathcal{D}_{Y|X}}[1_{1 \neq Y} | X] 1_{\eta(X) \geq 1/2} + \mathbb{E}_{Y \sim \mathcal{D}_{Y|X}}[1_{0 \neq Y} | X] 1_{\eta(X) < 1/2}] \\ &= \mathbb{E}_{X \sim \mathcal{D}_X}[\mathbb{P}(Y = 0 | X) 1_{\eta(X) \geq 1/2} + \mathbb{P}(Y = 1 | X) 1_{\eta(X) < 1/2}] \\ &= \mathbb{E}_{X \sim \mathcal{D}_X}[\min\{\eta(X), 1 - \eta(X)\}] \end{aligned}$$

- Bayes risk: represents the minimum probability of misclassification

$$L(f_*) = \mathbb{P}(f_*(X) \neq Y) = \mathbb{E}_{X \sim \mathcal{D}_X}[\min\{\eta(X), 1 - \eta(X)\}]$$

Generalization bound for 1-NN

Assumption: $\exists c \geq 0, \forall x, x' \in \mathcal{X}$:

$$|\eta(x) - \eta(x')| \leq c \|x - x'\|_2$$

➡ Nearby points are likely to share the same label

Generalization bound for 1-NN

Assumption: $\exists c \geq 0, \forall x, x' \in \mathcal{X}$:

$$|\eta(x) - \eta(x')| \leq c \|x - x'\|_2$$

➡ Nearby points are likely to share the same label

Claim:

$$\mathbb{E}_{S_{train}}[L(f_{S_{train}})] \leq 2L(f_*) + c \mathbb{E}_{S_{train}, X \sim \mathcal{D}_X}[\|X - \text{nbh}_{S_{train}, 1}(X)\|]$$

← geometric term: average distance between a random point and its closest neighbor

Generalization bound for 1-NN

Assumption: $\exists c \geq 0, \forall x, x' \in \mathcal{X}$:

$$|\eta(x) - \eta(x')| \leq c \|x - x'\|_2$$

➡ Nearby points are likely to share the same label

Claim:

$$\begin{aligned} \mathbb{E}_{S_{train}}[L(f_{S_{train}})] &\leq 2L(f_*) + c \mathbb{E}_{S_{train}, X \sim \mathcal{D}_X}[\|X - \text{nbh}_{S_{train},1}(X)\|] \\ &\leq 2L(f_*) + 4c\sqrt{d}N^{-\frac{1}{d+1}} \end{aligned}$$

Interpretation:

For constant d and $N \rightarrow \infty$: $\mathbb{E}_{S_{train}}[L(f_{S_{train}})] \leq 2L(f_*)$

To achieve a constant error, we need $N \propto d^{(d+1)/2}$ - curse of dimensionality

Despite common belief: Interpolation method can generalize well

Proof

We want to bound

$$\mathbb{E}_{S_{train}}[L(f_{S_{train}})] = \mathbb{E}_{S_{train}}[\mathbb{P}_{(X,Y) \sim \mathcal{D}}[f_{S_{train}}(X) \neq Y]]$$

We first sample N unlabeled examples $S_{train,X} = (X_1, \dots, X_N) \sim \mathcal{D}_X$, an unlabeled example $X \sim \mathcal{D}_X$ and define $X' = \text{nbh}_{S_{train},1}(X)$

Finally we sample $Y \sim \eta(X)$ and $Y' \sim \eta(X')$

We have:

$$\begin{aligned} \mathbb{E}_{S_{train}}[L(f_{S_{train}})] &= \mathbb{E}_{S_{train,X}, X \sim \mathcal{D}_X, Y \sim \eta(X), Y' \sim \eta(X')}[\mathbb{1}_{Y \neq f_{S_{train}}(X)}] \\ &= \mathbb{E}_{S_{train,X}, X \sim \mathcal{D}_X, Y \sim \eta(X), Y' \sim \eta(X')}[\mathbb{1}_{Y \neq Y'}] \\ &= \mathbb{E}_{S_{train,X}, X \sim \mathcal{D}_X}[\mathbb{P}_{Y \sim \eta(X), Y' \sim \eta(X')}(Y \neq Y')] \end{aligned}$$

Proof

Consider two points $x, x' \in [0,1]^d$.

Sample their labels $Y \sim \eta(x)$ and $Y' \sim \eta(x')$

Claim:

$$\mathbb{P}(Y' \neq Y) \leq 2 \min\{\eta(x), 1 - \eta(x)\} + c\|x - x'\|$$

- Simple case: $x = x'$

$$\begin{aligned}\mathbb{P}(Y' \neq Y) &= \mathbb{E}[1_{Y' \neq Y} 1_{Y'=1} + 1_{Y' \neq Y} 1_{Y'=0}] \\ &= \mathbb{P}(Y' = 1)\mathbb{P}(Y = 0) + \mathbb{P}(Y' = 0)\mathbb{P}(Y = 1) \\ &= 2\eta(x)(1 - \eta(x)) \\ &\leq 2 \min\{\eta(x), 1 - \eta(x)\}\end{aligned}$$

Case 1:

Y=0 $(1 - \eta(x))$

Y'=1 $\eta(x)$

Case 2:

Y=1 $\eta(x)$

Y'=0 $(1 - \eta(x))$

Proof

- General case:

$$\begin{aligned}\mathbb{P}(Y \neq Y') &= \eta(x)(1 - \eta(x')) + \eta(x')(1 - \eta(x)) \\ &= \eta(x)(1 - \eta(x)) + \eta(x)(\eta(x) - \eta(x')) \\ &\quad + \eta(x)(1 - \eta(x)) + (\eta(x') - \eta(x))(1 - \eta(x)) \\ &= 2\eta(x)(1 - \eta(x)) + (2\eta(x) - 1)(\eta(x) - \eta(x')) \\ &\leq 2\eta(x)(1 - \eta(x)) + |(2\eta(x) - 1)| |\eta(x) - \eta(x')| \\ &\leq 2\eta(x)(1 - \eta(x)) + |\eta(x) - \eta(x')| \\ &\leq 2\eta(x)(1 - \eta(x)) + c\|x - x'\| \\ &\leq 2 \min\{\eta(x), 1 - \eta(x)\} + c\|x - x'\|\end{aligned}$$

Proof

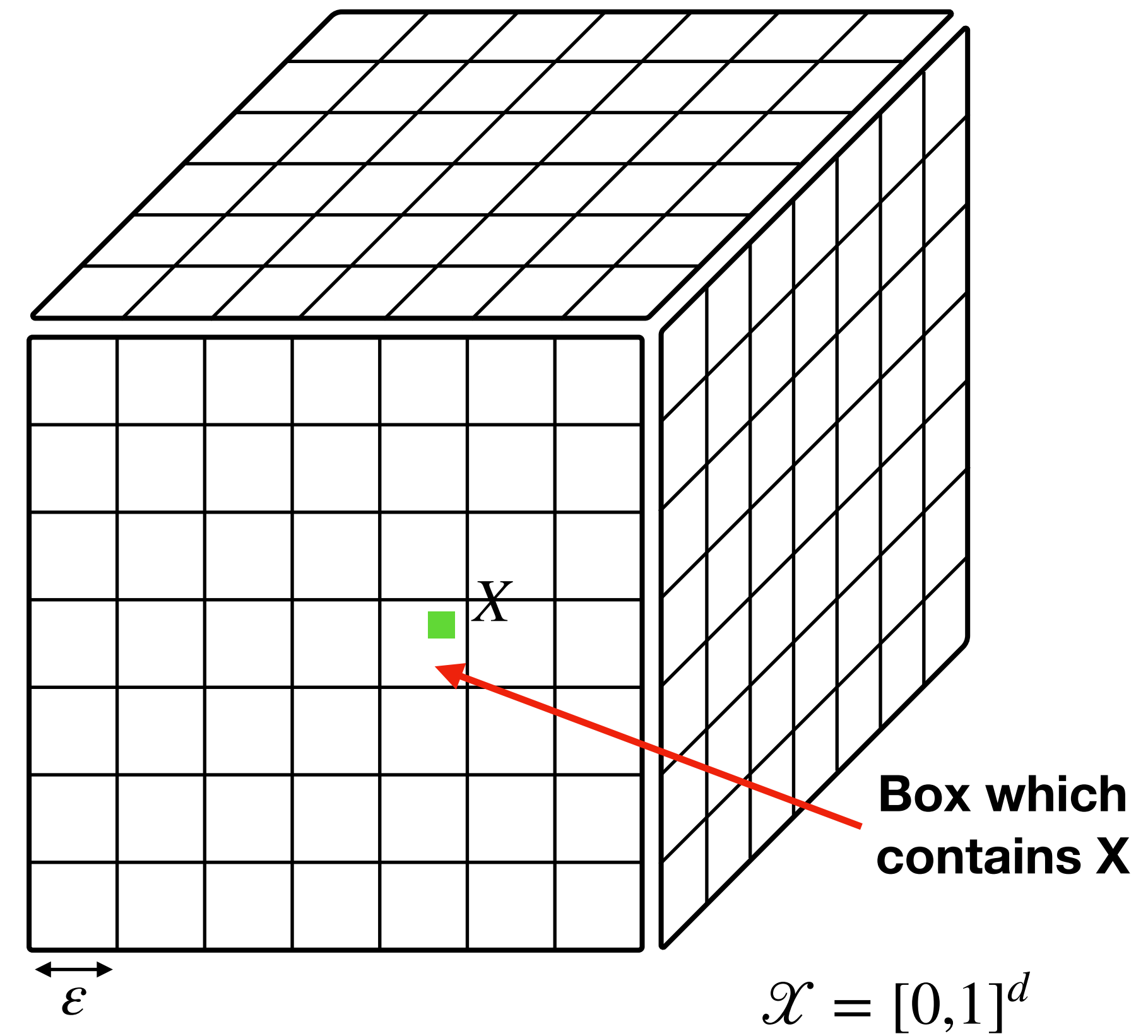
$$\begin{aligned}\mathbb{E}_{S_{train}}[L(f_{S_{train}})] &= \mathbb{E}_{S_{train}, X, X' \sim \mathcal{D}_X, Y \sim \eta(X), Y' \sim \eta(X')}[\mathbb{1}_{Y \neq f_{S_{train}}(X)}] \\ &= \mathbb{E}_{S_{train}, X, X' \sim \mathcal{D}_X, Y \sim \eta(X), Y' \sim \eta(X')}[\mathbb{1}_{Y \neq Y'}] \\ &= \mathbb{E}_{S_{train}, X, X' \sim \mathcal{D}_X}[\mathbb{P}_{Y \sim \eta(X), Y' \sim \eta(X')}(Y \neq Y')] \\ &\leq \mathbb{E}_{S_{train}, X, X' \sim \mathcal{D}_X}[2 \min\{\eta(X), 1 - \eta(X)\} + c\|X - X'\|] \\ &\leq 2L(f_*) + c\mathbb{E}_{S_{train}, X \sim \mathcal{D}_X}[\|X - \text{nbh}_{S_{train}, 1}(X)\|]\end{aligned}$$

Bound on the geometric term

Consider a fresh sample $X \sim \mathcal{D}_X$ and denote by

$$p_k = \mathbb{P}(X \in \text{Box}_k)$$

Consider the box which contains X . Two options:



ε - cover of the Hypercube

Bound on the geometric term

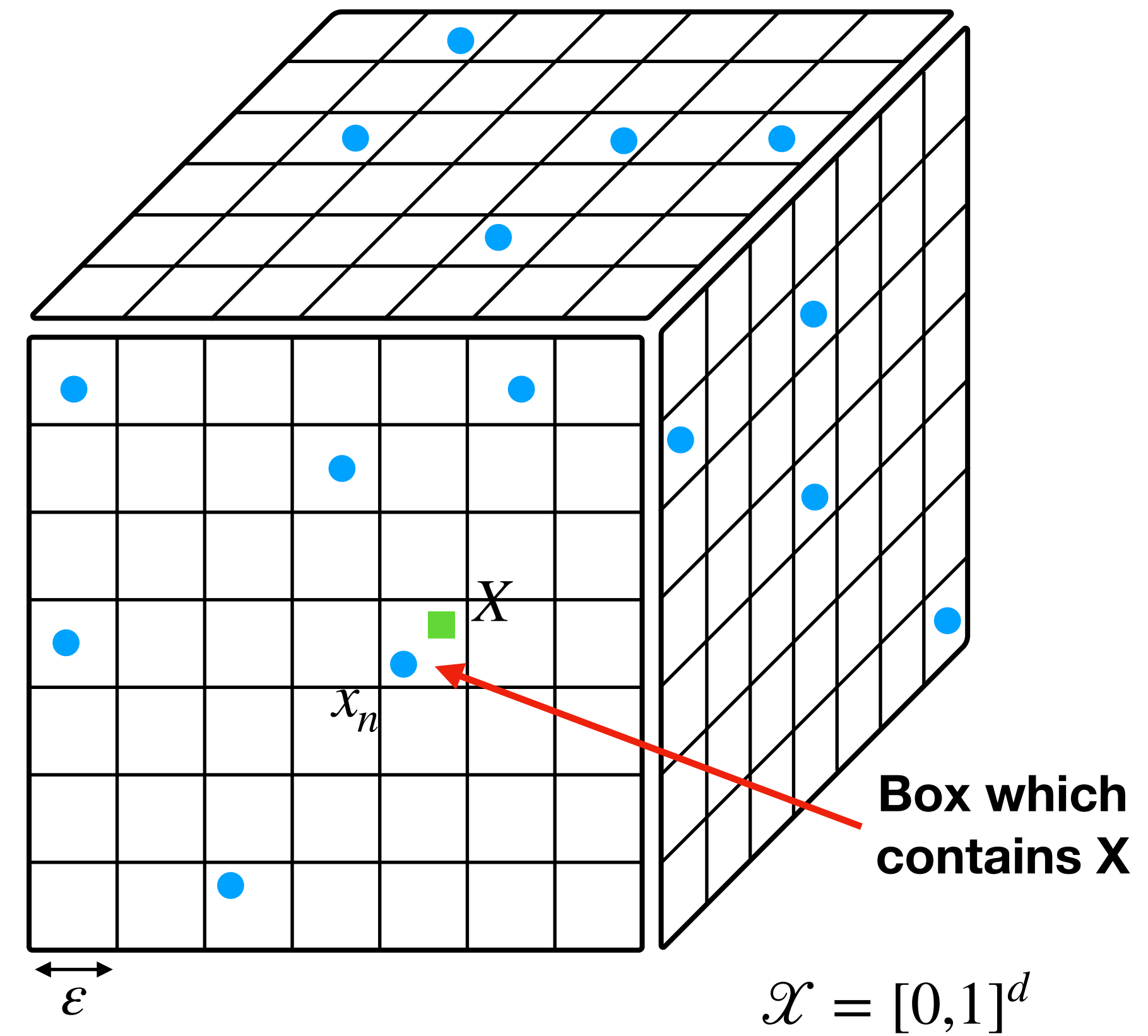
Consider a fresh sample $X \sim \mathcal{D}_X$ and denote by

$$p_k = \mathbb{P}(X \in \text{Box}_k)$$

Consider the box which contains X . Two options:

- The box contains an element of S_{train} . X has a neighbor in S_{train} at distance at most $\sqrt{d}\epsilon$

It happens with probability $1 - (1 - p_k)^N$



ϵ - cover of the Hypercube

Bound on the geometric term

Consider a fresh sample $X \sim \mathcal{D}_X$ and denote by

$$p_k = \mathbb{P}(X \in \text{Box}_k)$$

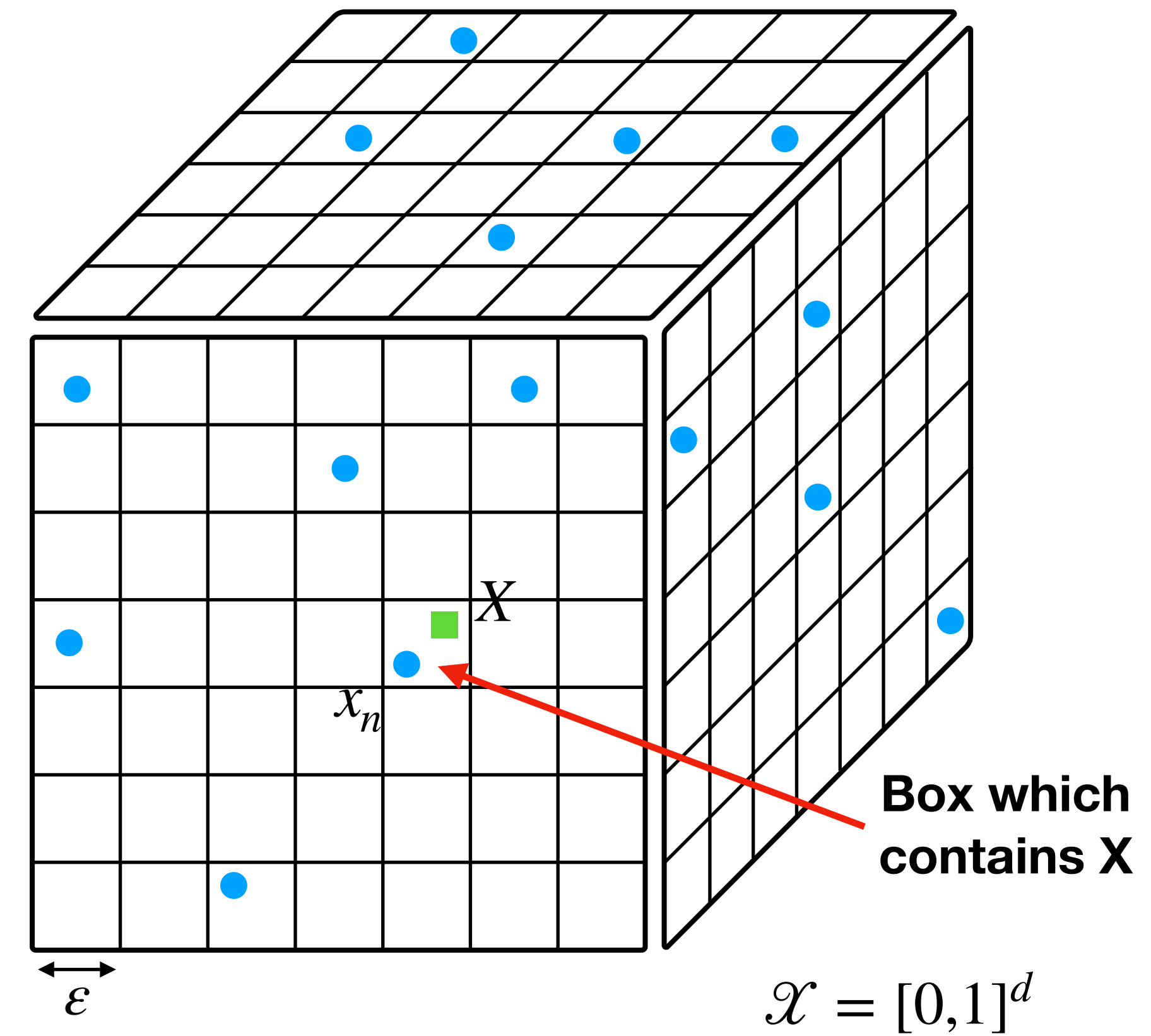
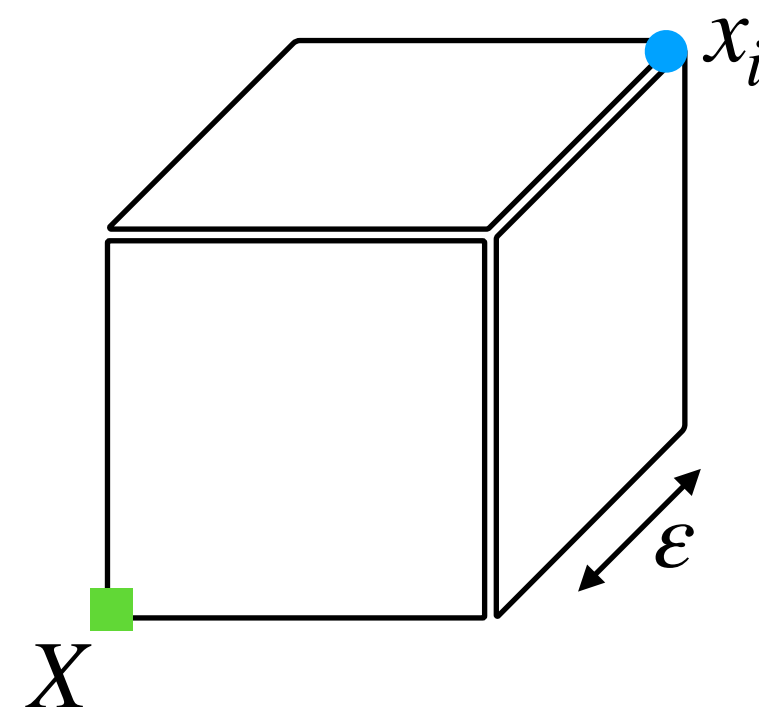
Consider the box which contains X . Two options:

- The box contains an element of S_{train} . X has a neighbor in S_{train} at distance at most $\sqrt{d}\epsilon$

It happens with probability $1 - (1 - p_k)^N$

Proof: Consider the worst case:

$$\|X - x_i\| = \sqrt{\sum_{i=1}^d \epsilon^2} = \sqrt{d}\epsilon$$



ϵ - cover of the Hypercube

Bound on the geometric term

Consider a fresh sample $X \sim \mathcal{D}_X$ and denote by

$$p_k = \mathbb{P}(X \in \text{Box}_k)$$

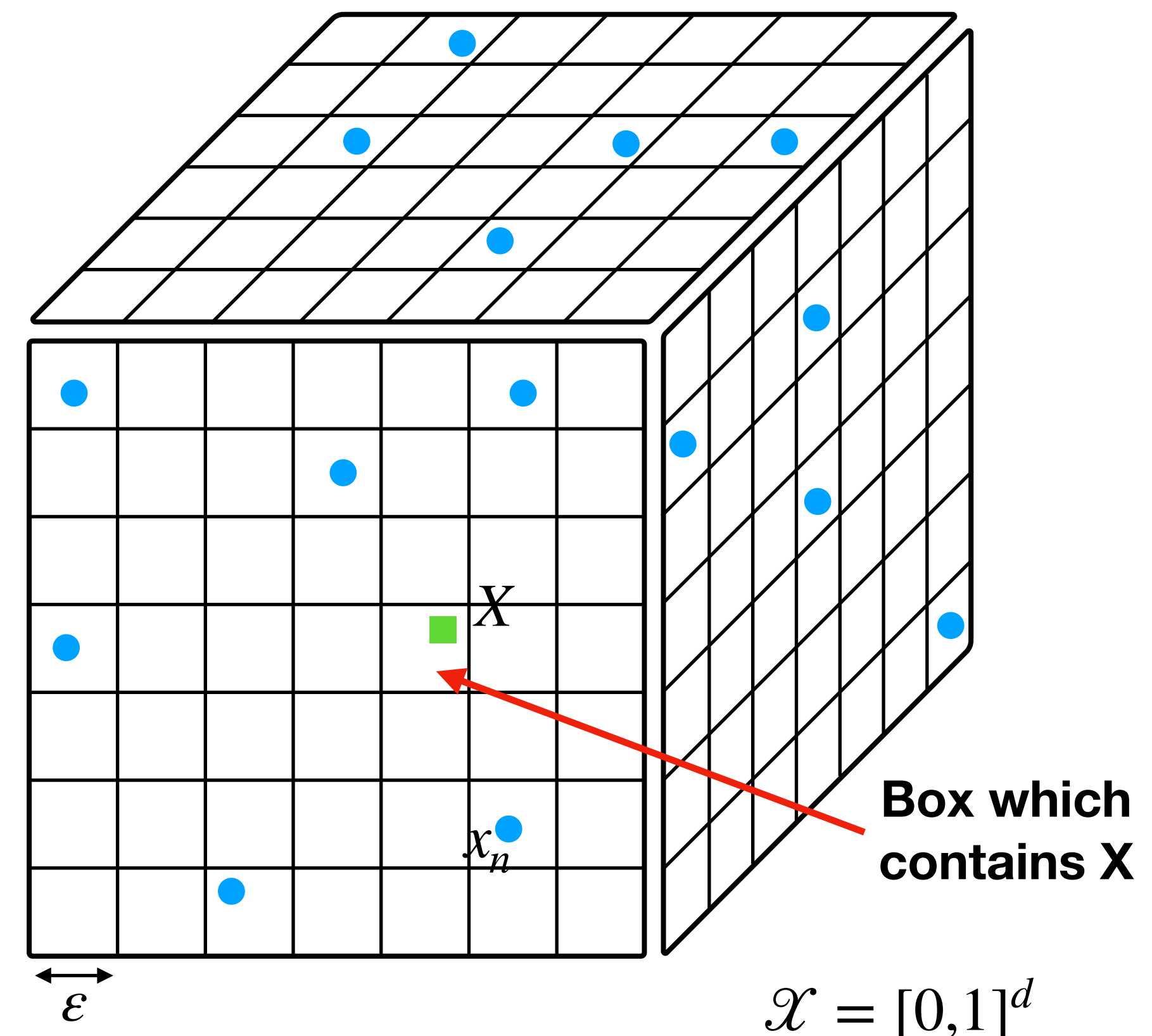
Consider the box which contains X . Two options:

- The box contains an element of S_{train} . X has a neighbor in S_{train} at distance at most $\sqrt{d}\epsilon$

It happens with probability $1 - (1 - p_k)^N$

- There is no element of S_{train} . The nearest neighbor of X can be at worst at a distance $\sqrt{d}$

It happens with probability $(1 - p_k)^N$



ϵ - cover of the Hypercube

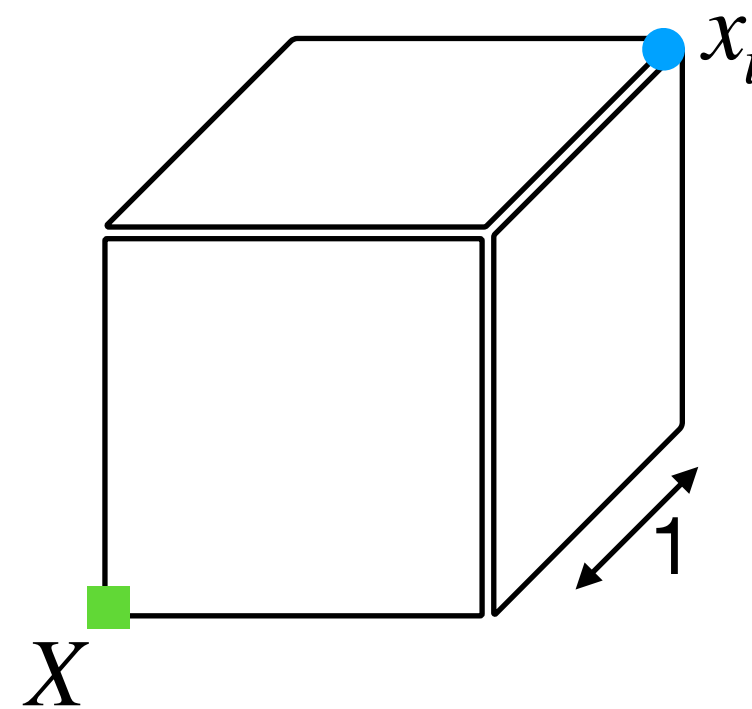
Bound on the geometric term

Consider a fresh sample $X \sim \mathcal{D}_X$ and denote by
 $p_k = \mathbb{P}(X \in \text{Box}_k)$

C

Proof: Consider the worst case:

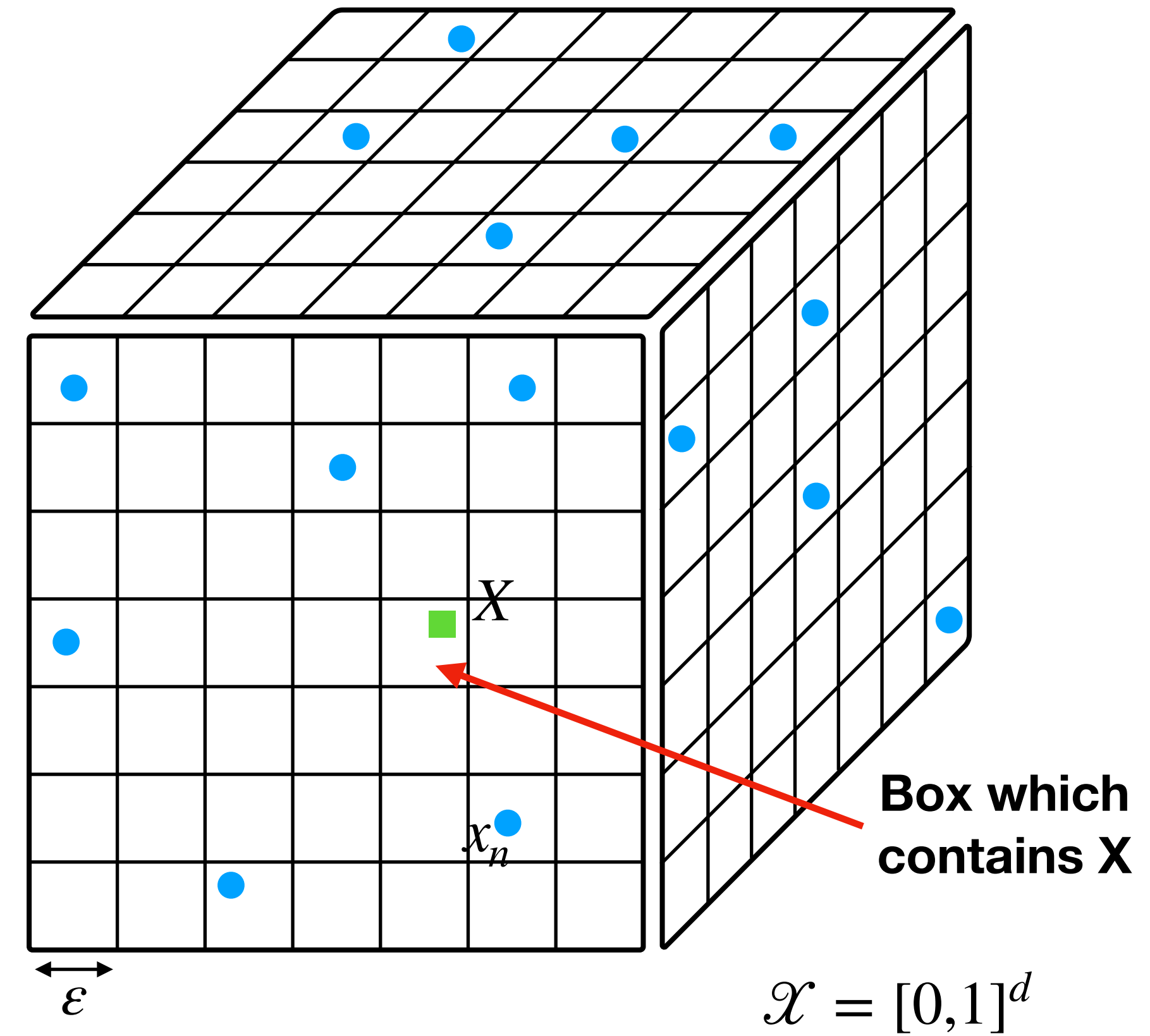
$$\|X - x_i\| = \sqrt{\sum_{i=1}^d 1} = \sqrt{d}$$



of

X can be at worst at a distance $\sqrt{d}$

It happens with probability $(1 - p_k)^N$



ϵ - cover of the Hypercube

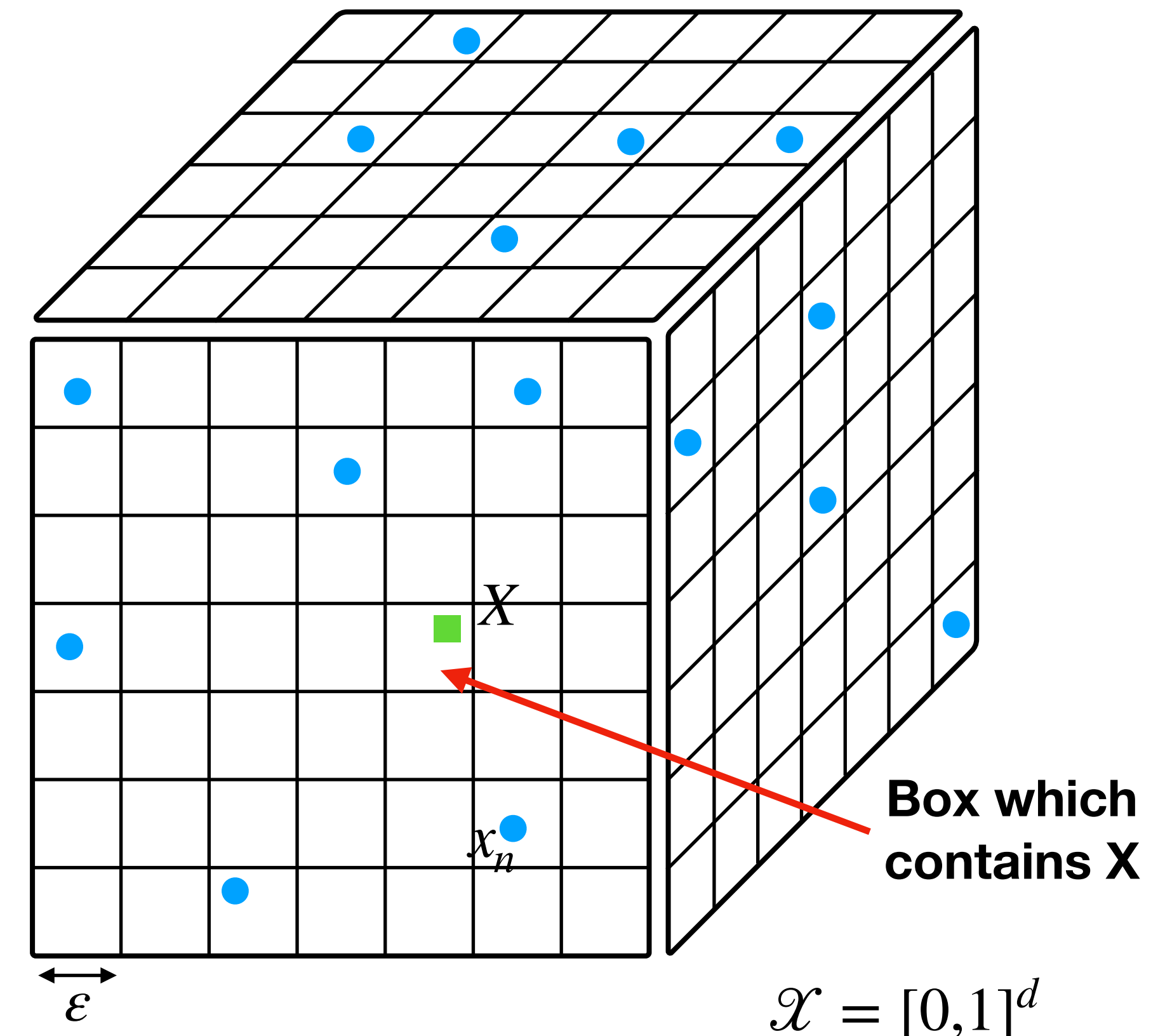
Bound on the geometric term

$$\mathbb{E}[\|X - \text{nbh}(X)\|] \leq \sum_k p_k [(1 - p_k)^N \sqrt{d} + (1 - (1 - p_k)^N) \sqrt{d} \varepsilon]$$

Claim: The bound is derived by optimizing over p_k and ε

Intuition:

- If p_k is large: it is likely that we pick that box but it is also likely that we find a training point in that box
- If p_k is small, we are generally safe, as by its definition, this scenario occurs infrequently



ε - cover of the Hypercube

Nearest Neighbors is a local averaging method

Local averaging methods aim to approximate the Bayes predictor directly - without the need for optimization

This is achieved by approximating the conditional distribution $p(y | x)$ by some $\hat{p}(y | x)$

These “plug-in” estimators are:

- $f(x) \in \arg \max_{y \in \mathcal{Y}} \hat{\mathbb{P}}(Y = y | x)$ for classification with the 0-1 loss
- $f(x) = \hat{\mathbb{E}}[Y | x] = \int_{\mathcal{Y}} y \hat{p}(y | x) dy$ for regression with the square loss

In the case of nearest neighbors:

$$\hat{p}(y | x) = \sum_{n=1}^N \hat{w}_n(x) 1_{y=y_n}$$

where $\hat{w}_n(x) = 1/k$ for the k nearest neighbors (0 otherwise)

Recap

- k-NN: a local averaging method for regression and classification
 - use a notion of distance to define *neighborhoods* ($= k$ nearest neighbors)
 - the prediction is a function of these neighborhoods
 - e.g., majority selection for classification, weighted sum for regression
- Bias-variance: small/large k leads to low/high bias and high/low variance
- Curse of dimensionality: as $d \nearrow \infty$, it is harder to define local neighborhoods
- For $N \rightarrow \infty$, 1-NN is competitive with Bayes classifier
- N needs to scale exponentially in d to achieve the same error