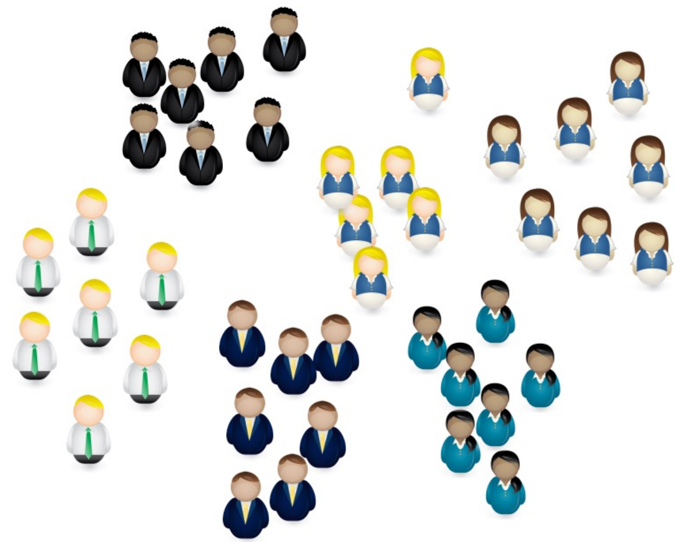
Tufts

CS135
Introduction to Machine Learning

Lecture 7: Nearest Neighbor

KNN: Overview

- $x_i \in \mathbb{R}^d$: features, predictor, attributes, input,…
- $y_i \in \mathbb{R}$: target, label class, output,…
- KNN is **supervised learning**
  - Sampled labeled data (x, y)
  - Goal: learn function $h: X \rightarrow Y$ so prediction $h(x)$ can be confidently made given an unseen observation x



- KNN classifier is **non parametric** and **instance-based**

KNN is non-parametric, instance-based, and used in a supervised learning setting.

nonparametric:
No explicit assumptions about h, avoiding mismodeling the underlying data distribution

instance-based:
No model is explicitly learned, but training instances get used as "knowledge" for testing

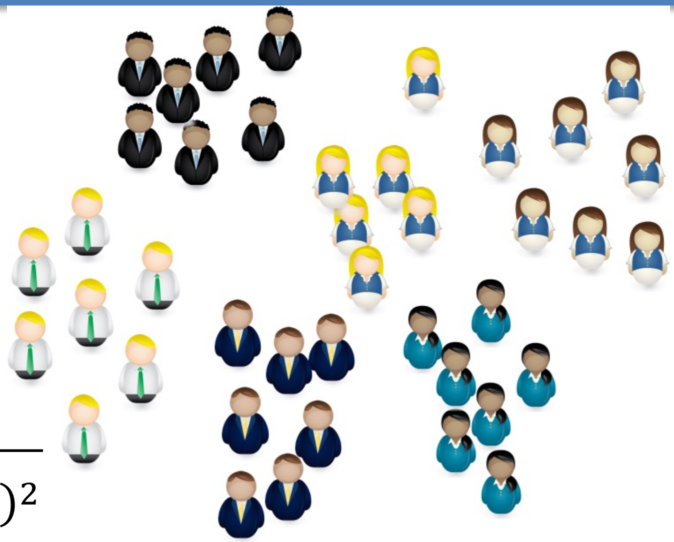
Minimal training but expensive testing

Tufts

KNN: Concept

- In essence, boils down to a majority vote between k most similar instances to a given "unseen" observation

- Similarity is defined according to a distance metric between 2 data points



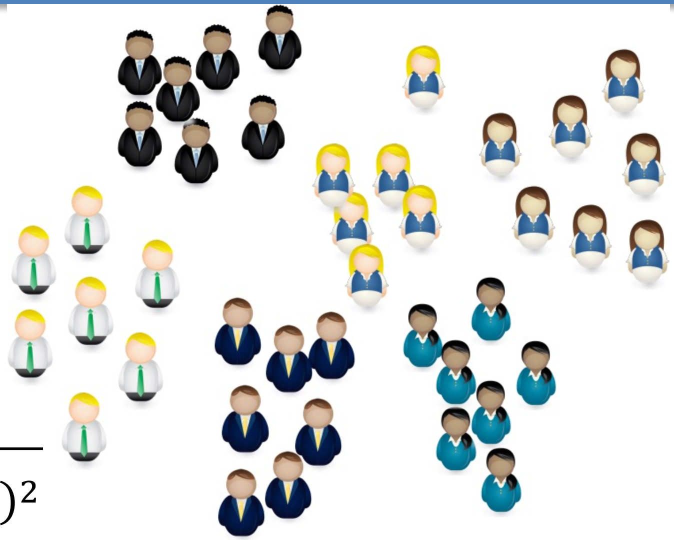
$$d(x, x') = \sqrt{(x_1 - x_1')^2 + (x_2 - x_2')^2 + \cdots + (x_n - x_n')^2}$$

where $d(x, x')$, by definition, is the Euclidean distance– chosen, for example per its popularity. Nonetheless, any metric that measures closeness or similarity works.

KNN: Concept

- In essence, boils down to a majority vote between k most similar instances to a given “unseen” observation

- Similarity is defined according to a distance metric between 2 data points



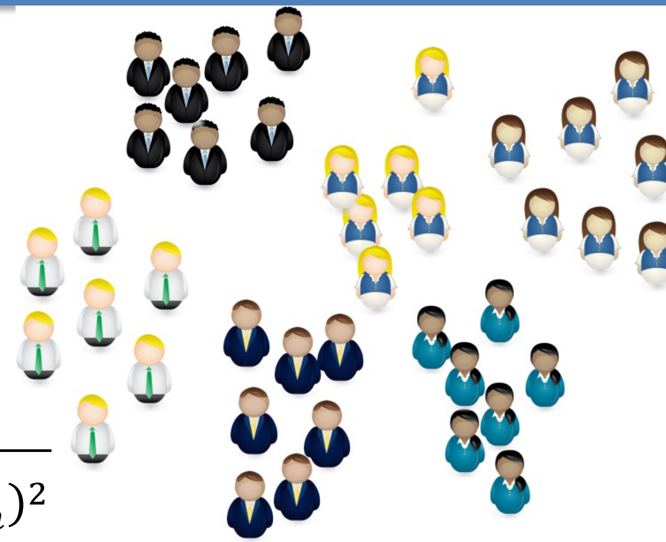
$$d(x, x') = \sqrt{(x_1 - x_1')^2 + (x_2 - x_2')^2 + \cdots + (x_n - x_n')^2}$$

where $d(x, x')$, by definition, is the Euclidean distance– chosen for example per its popularity. Nonetheless, any metric that measures closeness or similarity works.

- Given $k \in \mathbb{R}^+$, $x \in \mathbb{R}^d$, and similarity metric d, KNN simple performs 2 Steps:
  1. Compute d between x and each training observation, with K samples closest to x in set $N_k(x)$. Note that K is typically an odd integer to avoid tie situations
  2. Compute conditional probability for each class as the fraction of points in $N_k(x)$ with the given class label j:

$$P(y = j | X = x) = \frac{1}{K} \sum_{i \in N_k(x)} \mathrm{I}(y^{(i)} = j)\,;\ \mathrm{I}(x) = \begin{cases} 1, & y = j \\ 0, & y \neq j \end{cases}$$

KNN: Concept



- In essence, boils down to a majority vote between k most similar instances to a given "unseen" observation

- Similarity is defined according to a distance metric between 2 data points

$$d(x, x') = \sqrt{(x_1 - x_1')^2 + (x_2 - x_2')^2 + \cdots + (x_n - x_n')^2}$$

where $d(x, x')$, by definition, is the Euclidean distance– chosen for example per its popularity. Nonetheless, any metric that measures closeness or similarity works.

- Given $k \in \mathbb{R}^+$, $x \in \mathbb{R}^d$, and similarity metric d, KNN simple performs <u>2 Steps:</u>

1 … x in

2 … with

KNN searches the memorized training observations for the *K* instances that most Closely resemble the new instance and assigns to it the most common class.

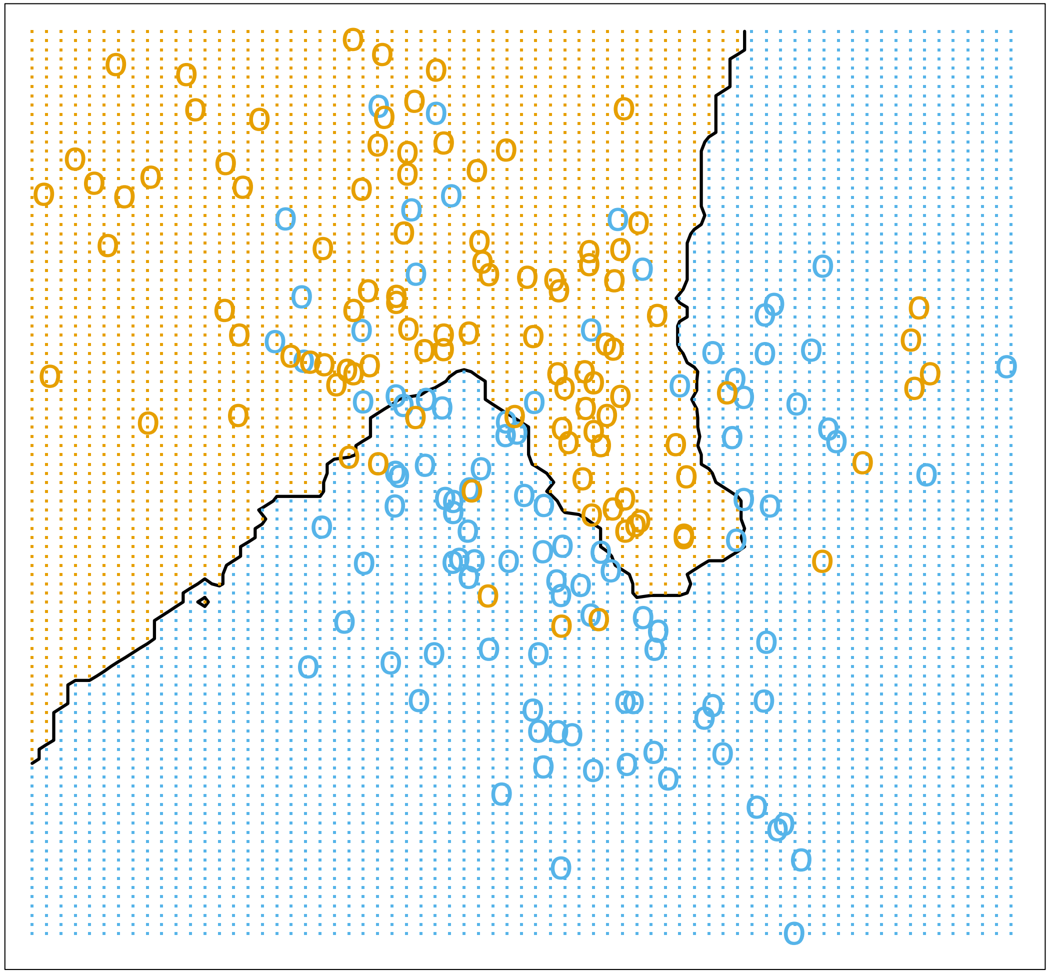
KNN can also be perceived as calculating the decision boundary (or boundaries in multi-class), to then used to classify new observations

Tufts

kNN for Classification

- Labels in discrete set C
- Majority vote:

$$\hat{f}(x) = \arg\max_{c \in C} |\{i \in N_k(x) : y_i = c\}|$$



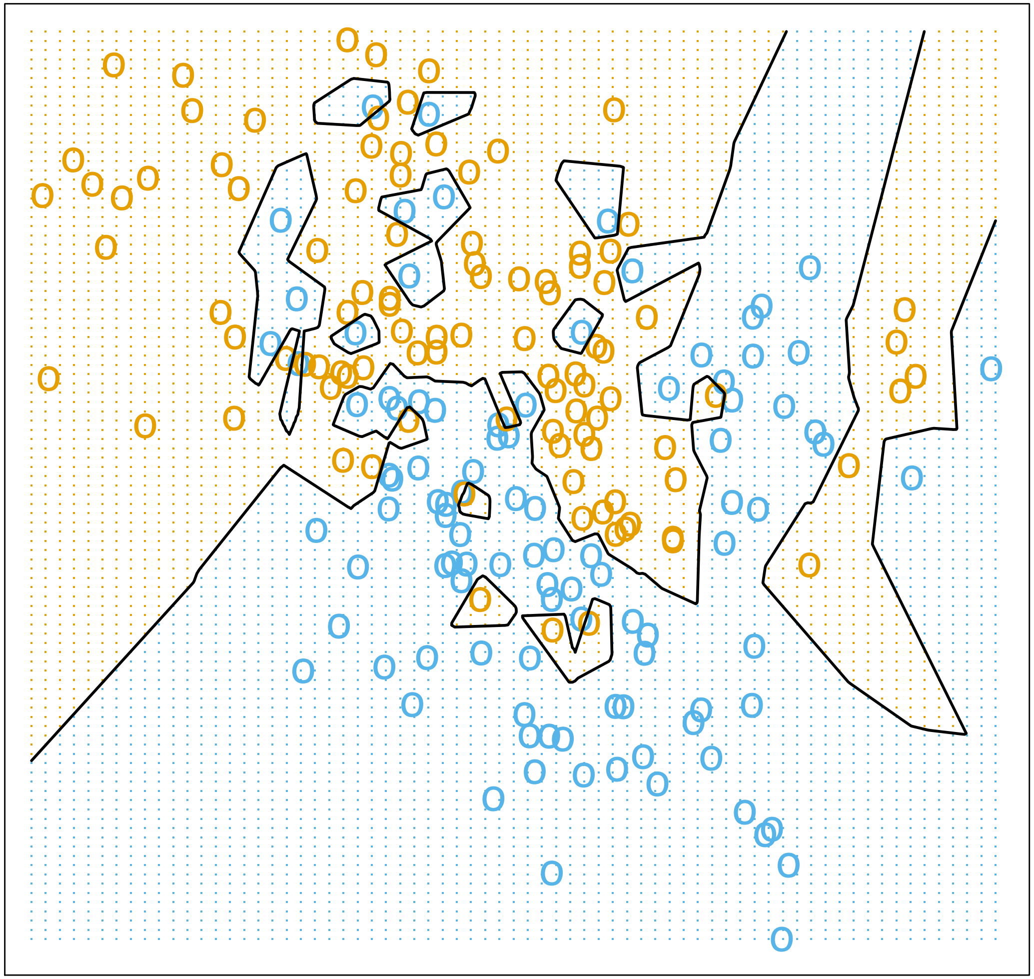
15-NN

kNN for Classification

- Labels in discrete set C

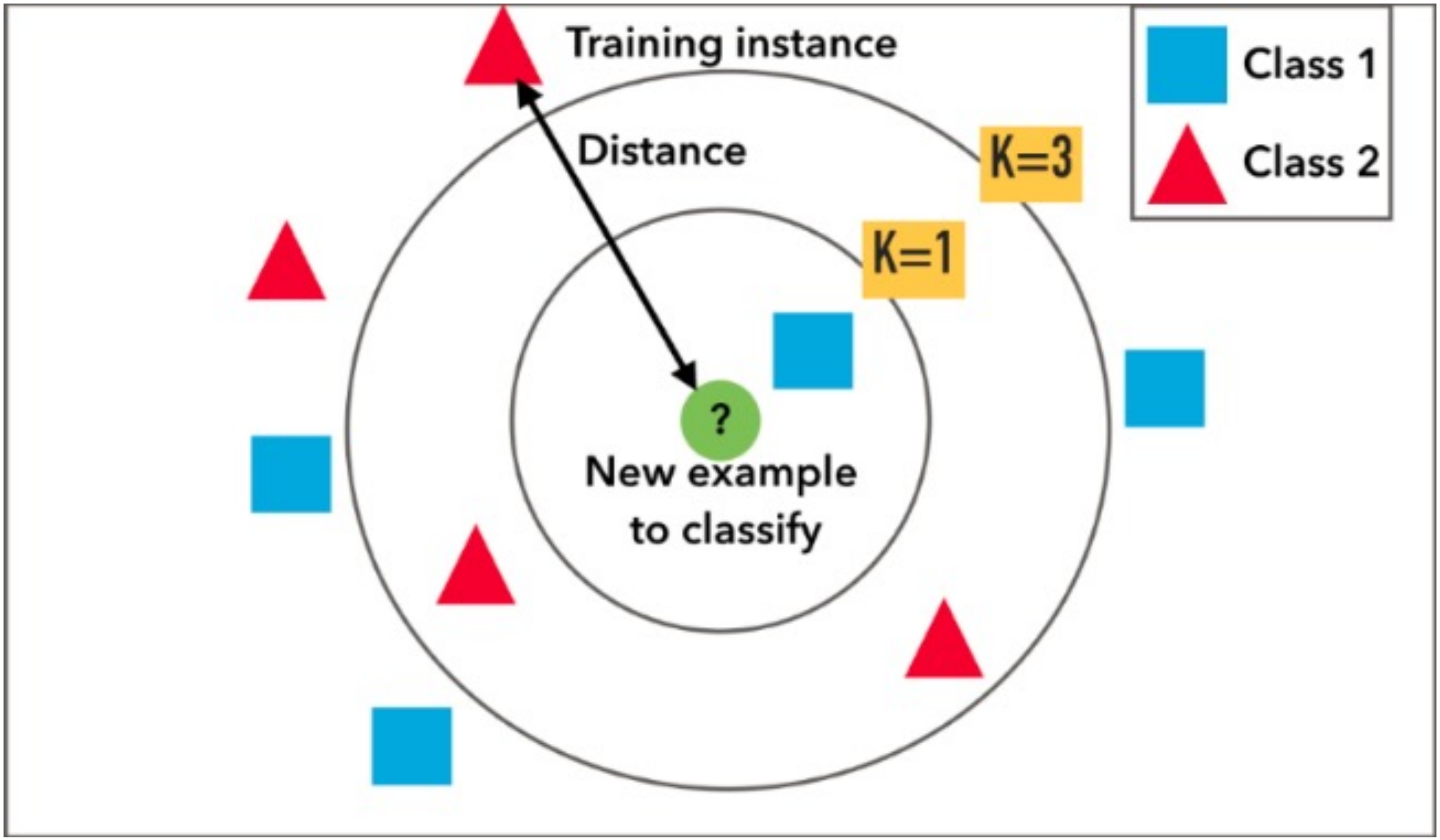
- Majority vote:

$$\hat{f}(x) = \arg\max_{c \in C} \left| \{ i \in N_k(x) : y_i = c \} \right|$$



1-NN

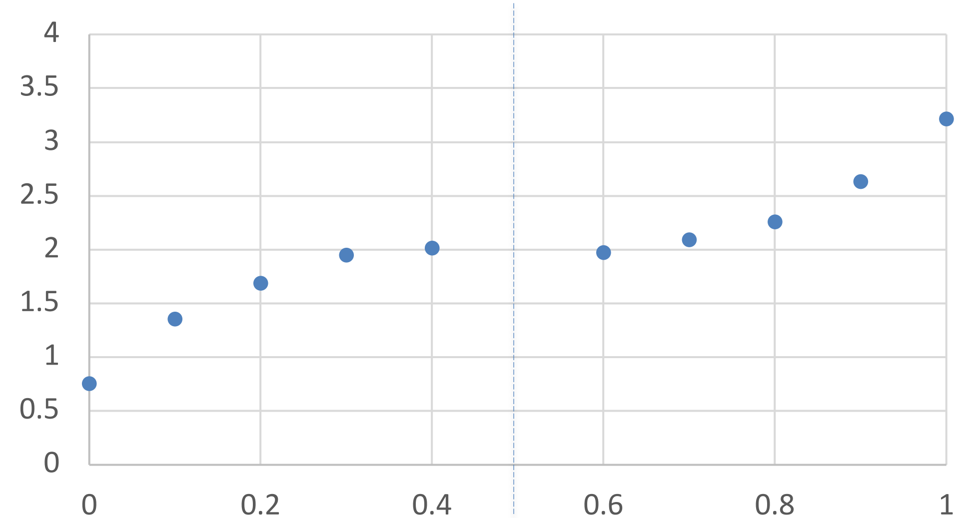
kNN for Classification



Tufts

K-Nearest Neighbor

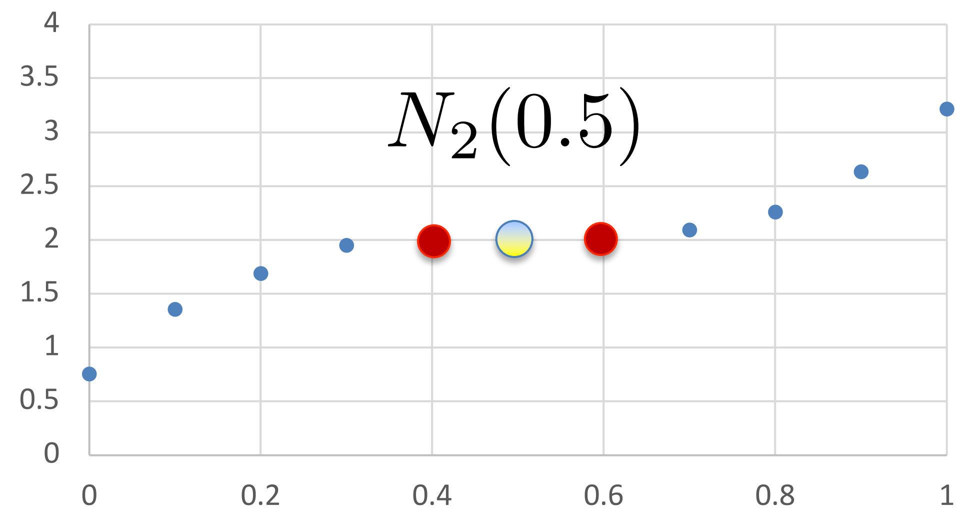
$$\hat{f}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$



where $N_k(x)$ is the set of the k nearest neighbors of x

K-Nearest Neighbor

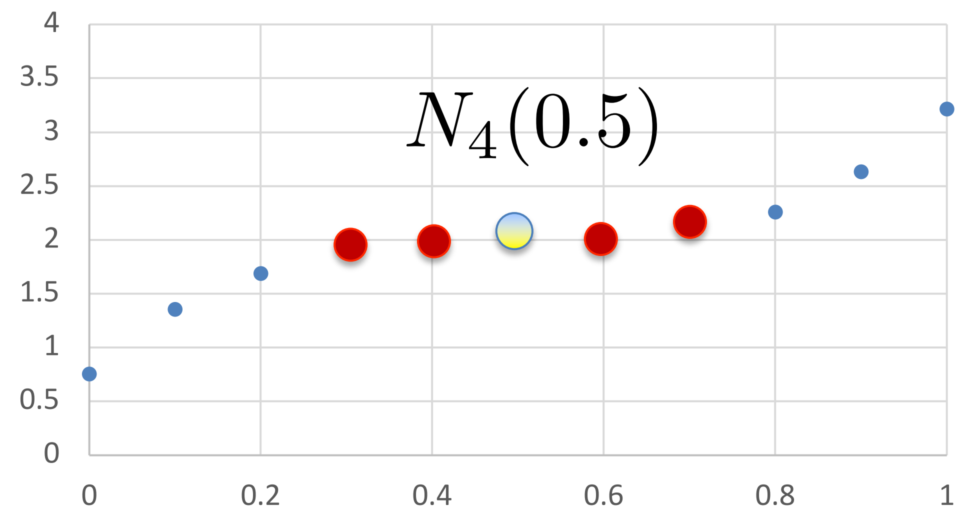
$$\hat{f}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$



where $N_k(x)$ is the set of the k nearest neighbors of x

K-Nearest Neighbor

$$\hat{f}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$

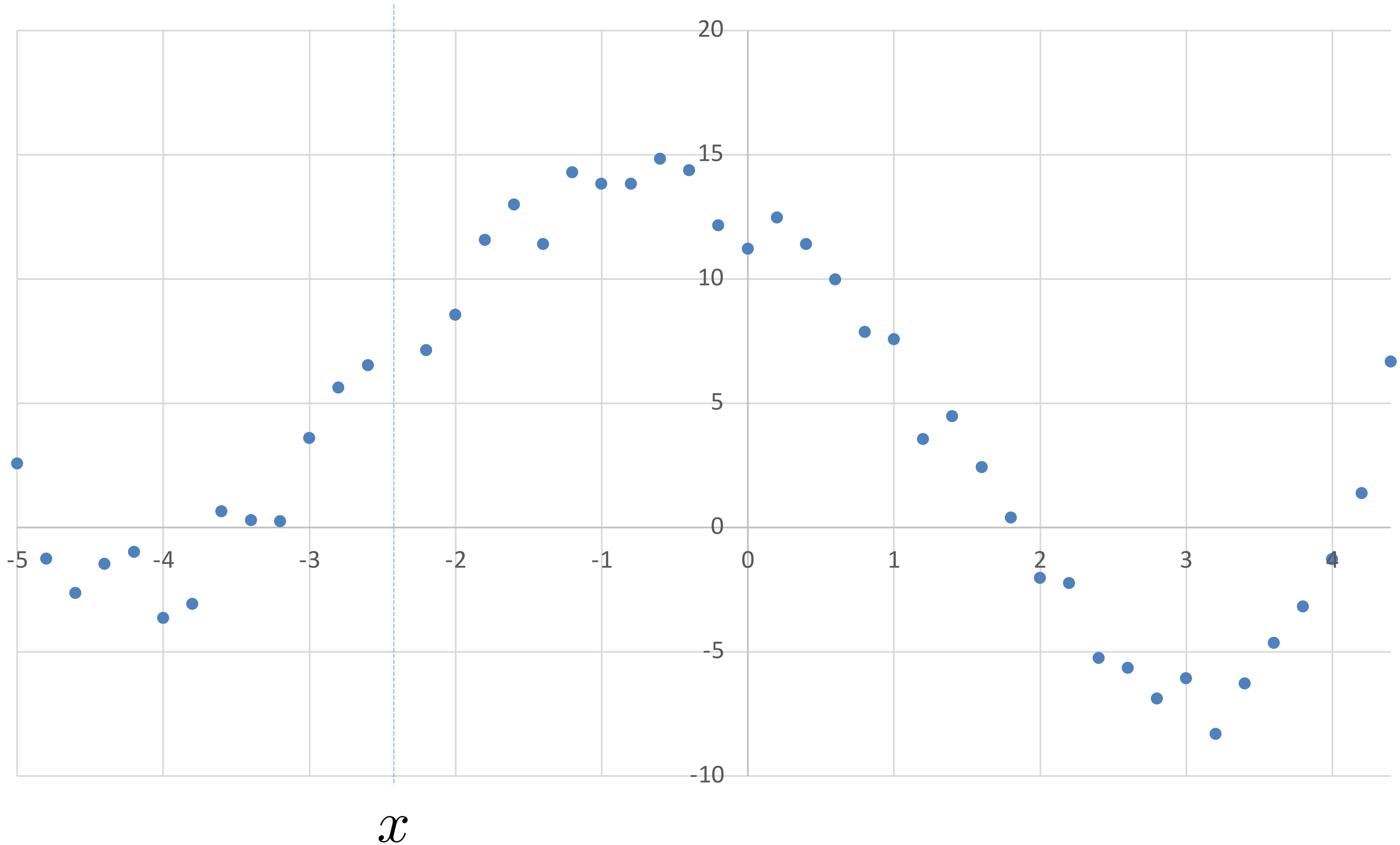


where $N_k(x)$ is the set of the k nearest neighbors of x

- ❑ $|N_k(x)| = k$
- ❑ For all $i \in N_k(x)$ and $j \notin N_k(x)$

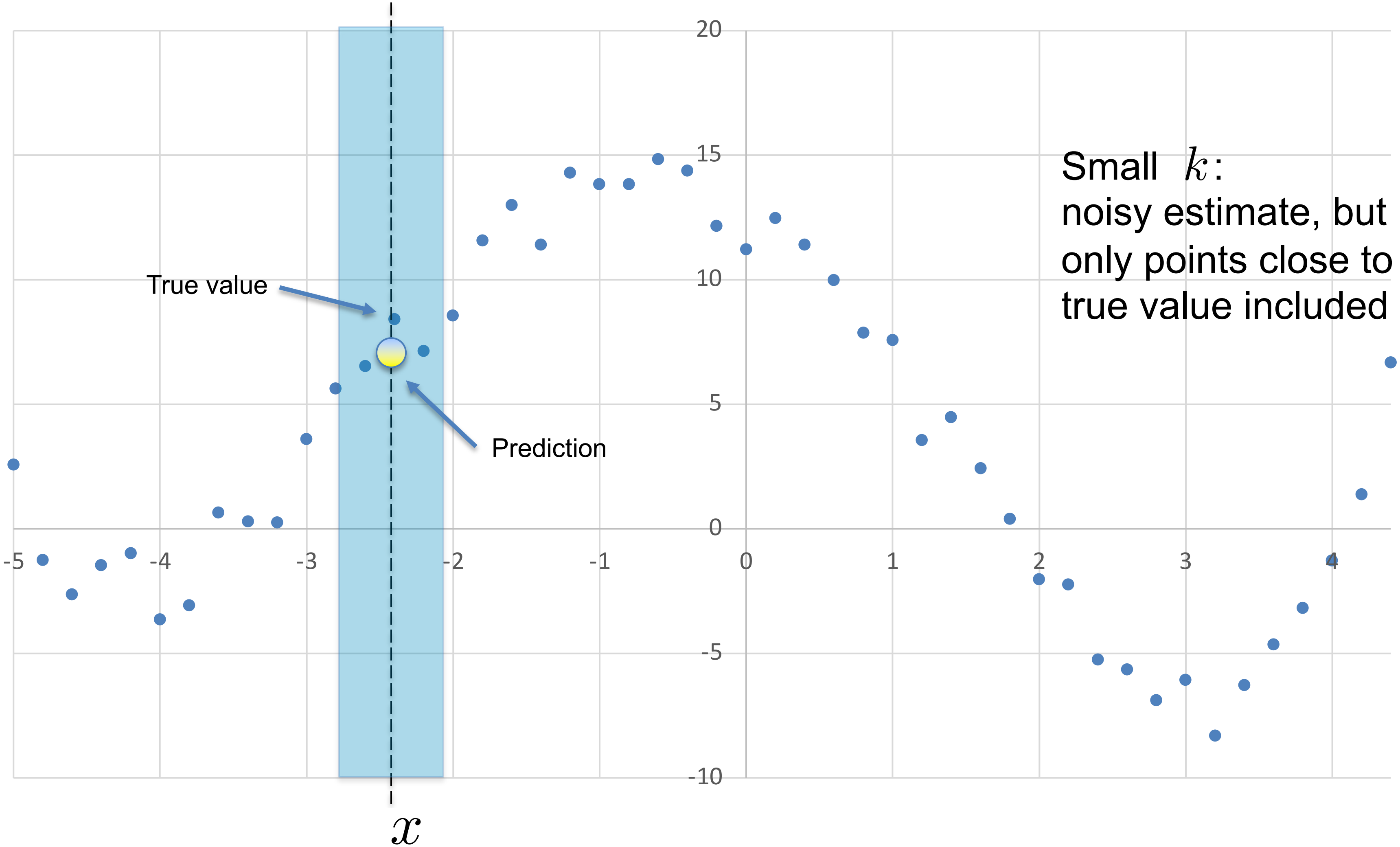
$$\|x - x_i\| \leq \|x - x_j\|$$

How big should k be?



Tufts

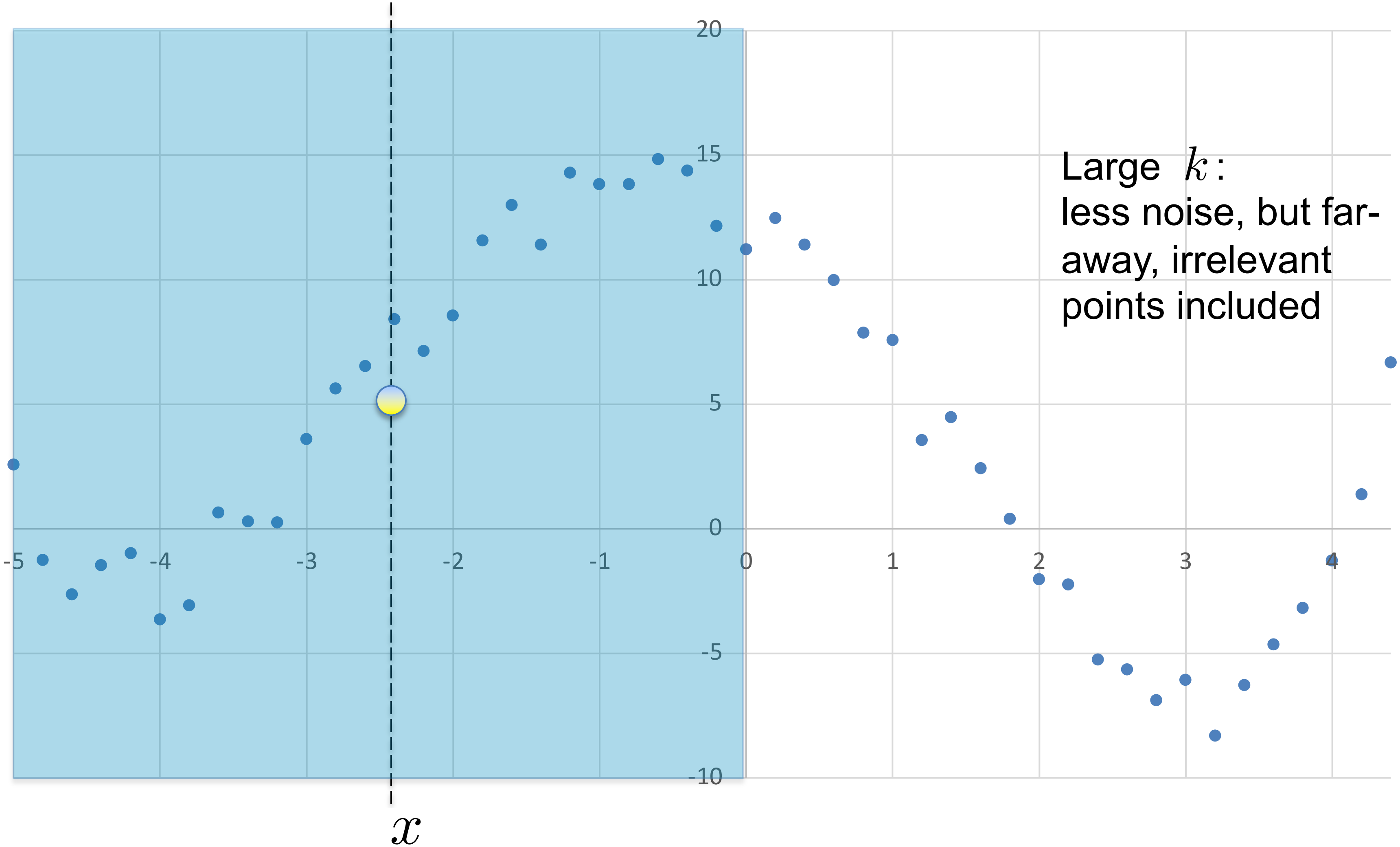
How big should k be?



Small k:
noisy estimate, but only points close to true value included

Tufts

How big should k be?



Large k:
less noise, but far-away, irrelevant points included

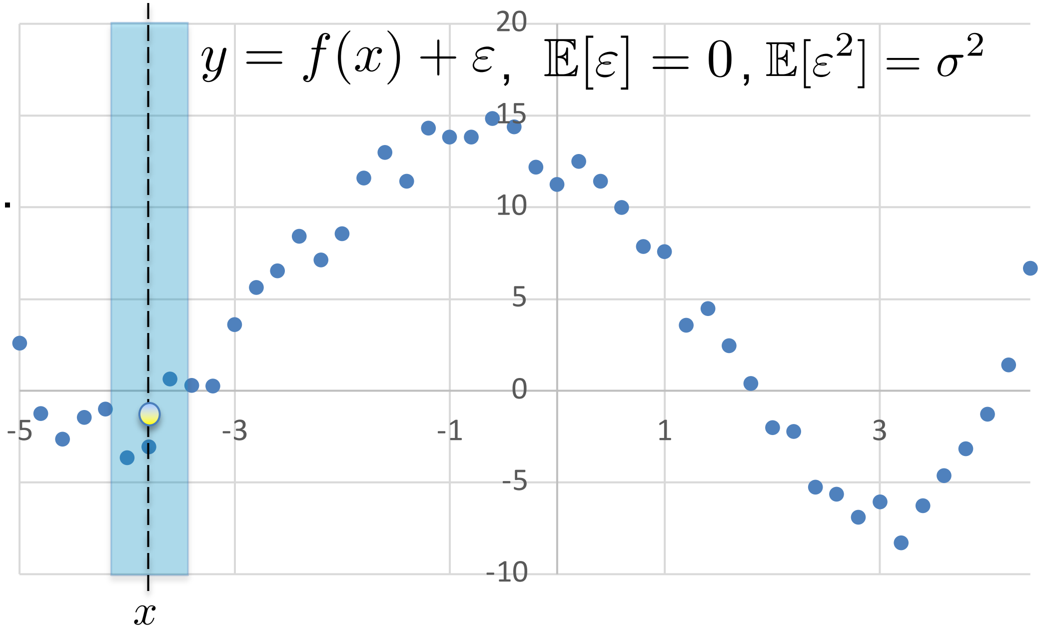
Tufts

Bias vs. Variance Trade-off

$$y_i = f(x_i) + \varepsilon_i, \qquad i = 1, \ldots, n.$$

ε_i i.i.d., $\mathbb{E}[\varepsilon_i] = 0$, $\mathbb{E}[\varepsilon_i^2] = \sigma^2 < \infty$.

$$\hat{f}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$



Expected Prediction Error (EPE):

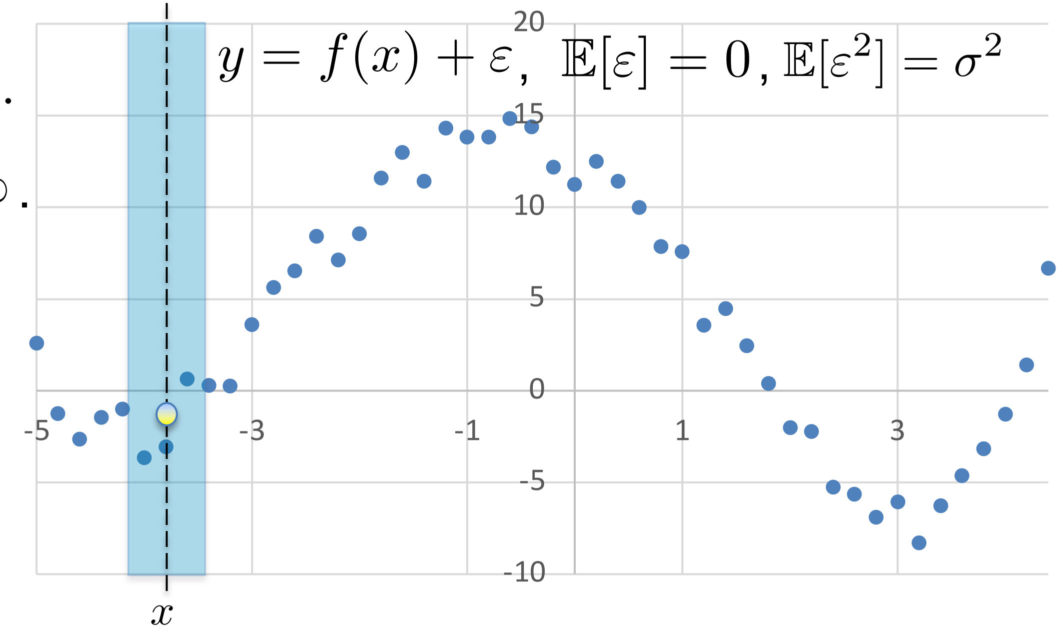
$$\mathbb{E}\left[\left(y - \hat{f}(x)\right)^2\right] = \mathbb{E}\left[(y - \mathbb{E}[y])^2\right] + \left(\mathbb{E}[y] - \mathbb{E}[\hat{f}(x)]\right)^2 + \mathbb{E}\left[\left(\mathbb{E}[\hat{f}(x)] - \hat{f}(x)\right)^2\right]$$

Bias vs. Variance Trade-off

$$y_i = f(x_i) + \varepsilon_i, \qquad i = 1, \ldots, n.$$

ε_i i.i.d., $\mathbb{E}[\varepsilon_i] = 0\,, \mathbb{E}[\varepsilon_i^2] = \sigma^2 < \infty\,.$

$$\hat{f}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$



Expected Prediction Error (EPE):

$$\mathbb{E}\left[\left(y - \hat{f}(x)\right)^2\right] = \underbrace{\mathbb{E}\left[(y - \mathbb{E}[y])^2\right]}_{\text{inherent noise}} + \left(\mathbb{E}[y] - \mathbb{E}[\hat{f}(x)]\right)^2 + \mathbb{E}\left[\left(\mathbb{E}[\hat{f}(x)] - \hat{f}(x)\right)^2\right]$$

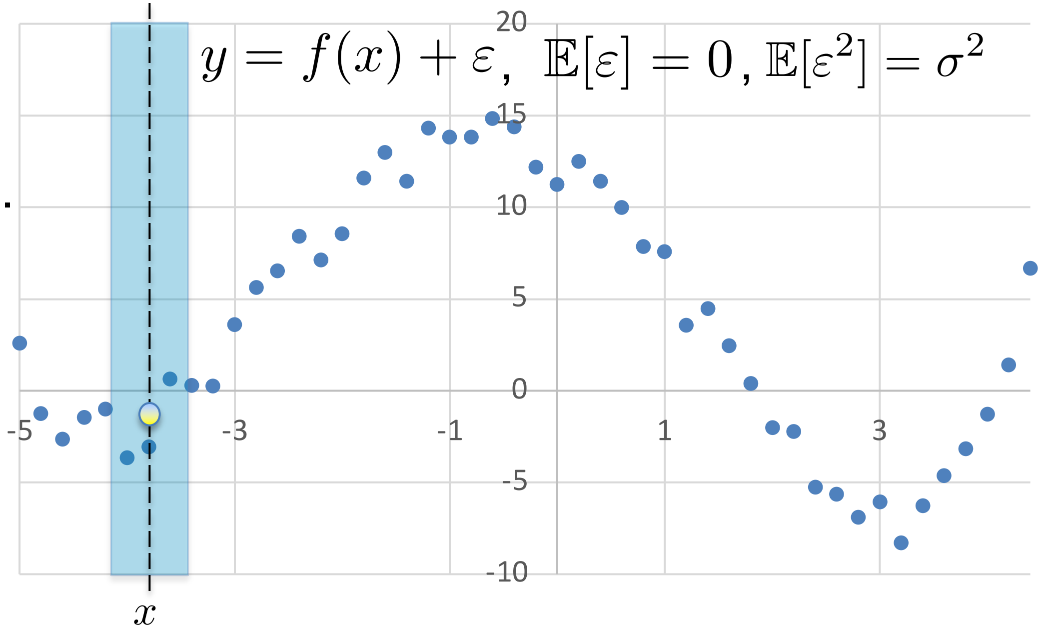
Tufts

Bias vs. Variance Trade-off

$$y_i = f(x_i) + \varepsilon_i, \qquad i = 1, \ldots, n.$$

ε_i i.i.d., $\mathbb{E}[\varepsilon_i] = 0\,, \mathbb{E}[\varepsilon_i^2] = \sigma^2 < \infty\,.$

$$\hat{f}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$



Expected Prediction Error (EPE):

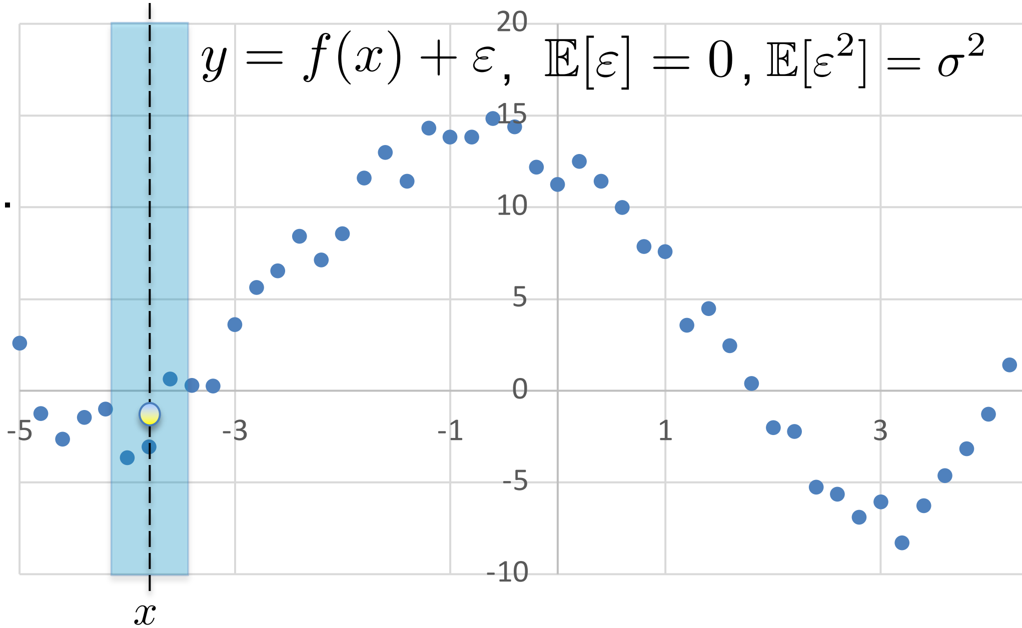
$$\mathbb{E}\left[\left(y - \hat{f}(x)\right)^2\right] = \mathbb{E}\left[(y - \mathbb{E}[y])^2\right] + \underbrace{\left(\mathbb{E}[y] - \mathbb{E}[\hat{f}(x)]\right)^2}_{\text{estimator } \textbf{bias}} + \mathbb{E}\left[\left(\mathbb{E}[\hat{f}(x)] - \hat{f}(x)\right)^2\right]$$

Bias vs. Variance Trade-off

$$y_i = f(x_i) + \varepsilon_i, \qquad i = 1, \dots, n.$$

ε_i i.i.d., $\mathbb{E}[\varepsilon_i] = 0\,, \mathbb{E}[\varepsilon_i^2] = \sigma^2 < \infty\,.$

$$\hat{f}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$



Expected Prediction Error (EPE):

$$\mathbb{E}\left[\left(y - \hat{f}(x)\right)^2\right] = \mathbb{E}\left[(y - \mathbb{E}[y])^2\right] + \left(\mathbb{E}[y] - \mathbb{E}[\hat{f}(x)]\right)^2 + \underbrace{\mathbb{E}\left[\left(\mathbb{E}[\hat{f}(x)] - \hat{f}(x)\right)^2\right]}_{\text{estimator } \textbf{variance}}$$

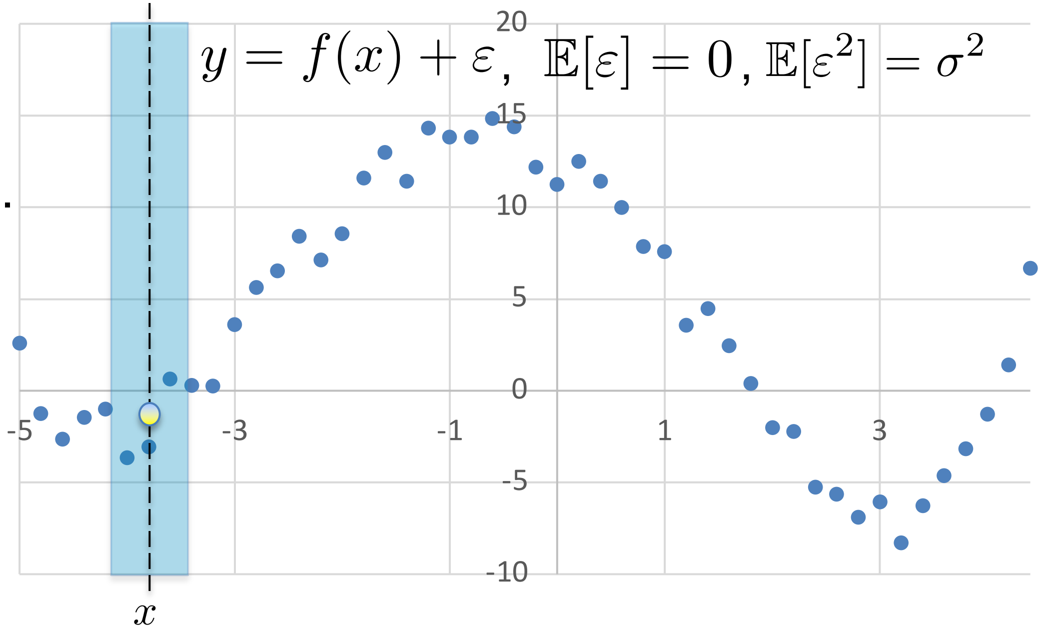
Tufts

Bias vs. Variance Trade-off

$$y_i = f(x_i) + \varepsilon_i, \qquad i = 1, \ldots, n.$$

ε_i i.i.d., $\mathbb{E}[\varepsilon_i] = 0\,, \mathbb{E}[\varepsilon_i^2] = \sigma^2 < \infty\,.$

$$\hat{f}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$



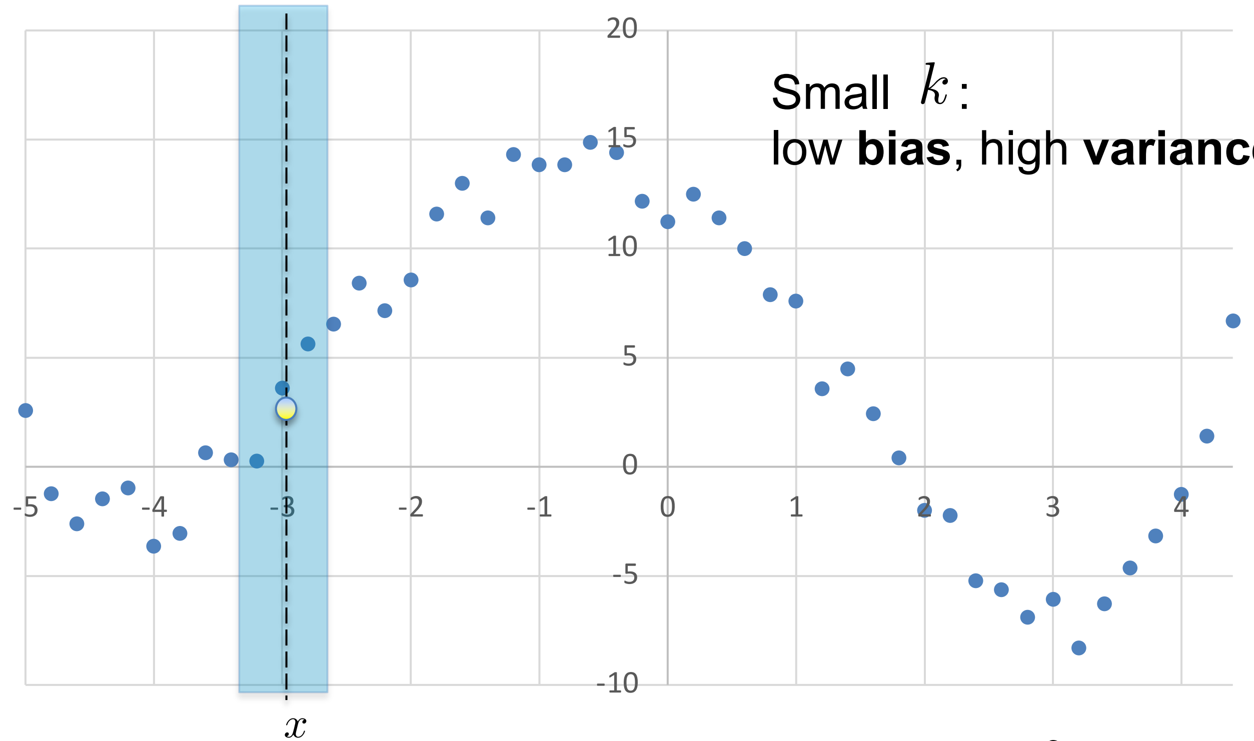
Expected Prediction Error (EPE):

$$\mathbb{E}\left[\left(y - \hat{f}(x)\right)^2\right] = \mathbb{E}\left[(y - \mathbb{E}[y])^2\right] + \left(\mathbb{E}[y] - \mathbb{E}[\hat{f}(x)]\right)^2 + \mathbb{E}\left[\left(\mathbb{E}[\hat{f}(x)] - \hat{f}(x)\right)^2\right]$$

$$= \sigma^2 + \left(f(x) - \frac{1}{k} \sum_{i \in N_k(x)} f(x_i)\right)^2 + \frac{\sigma^2}{k}$$

Tufts

Bias vs. Variance Trade-off

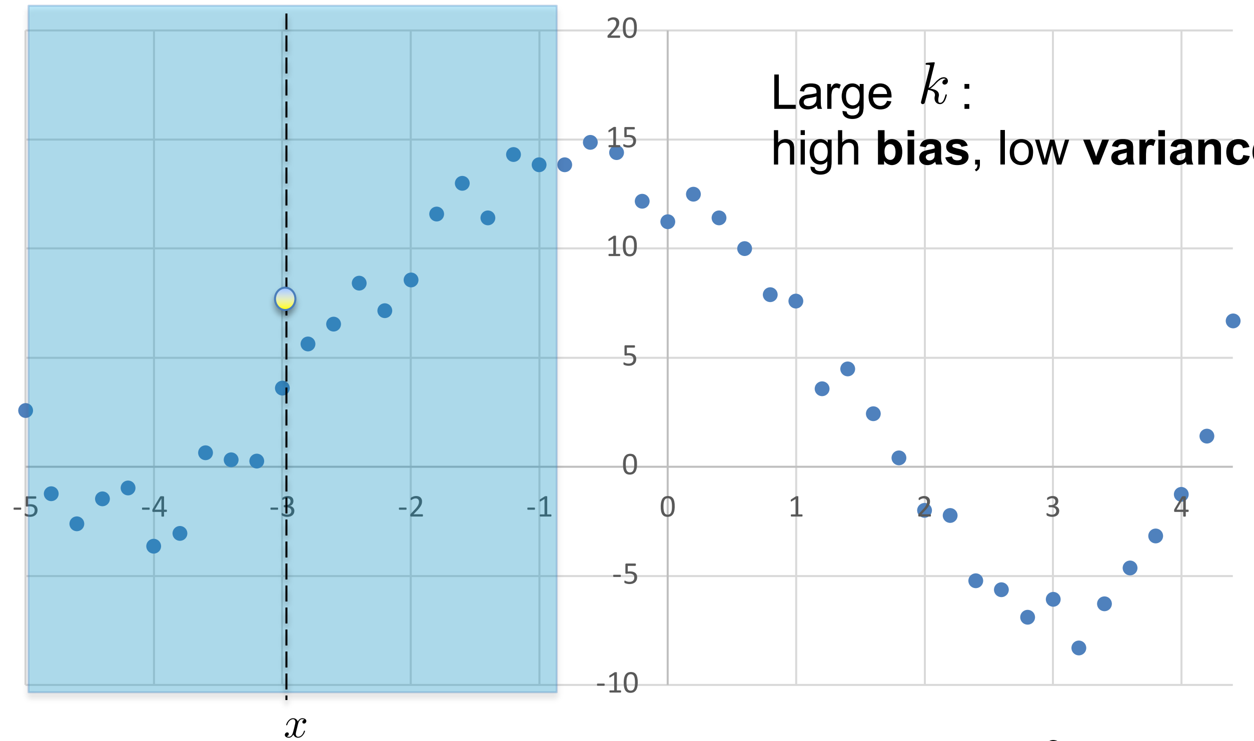


Small k:
low **bias**, high **variance**

EPE: $\mathbb{E}\left[\left(y-\hat{f}(x)\right)^2\right] = \sigma^2 + \underbrace{\left(f(x)-\frac{1}{k}\sum_{i\in N_k(x)} f(x_i)\right)^2}_{\text{estimator bias}} + \underbrace{\frac{\sigma^2}{k}}_{\text{estimator variance}}$

Tufts

Bias vs. Variance Trade-off



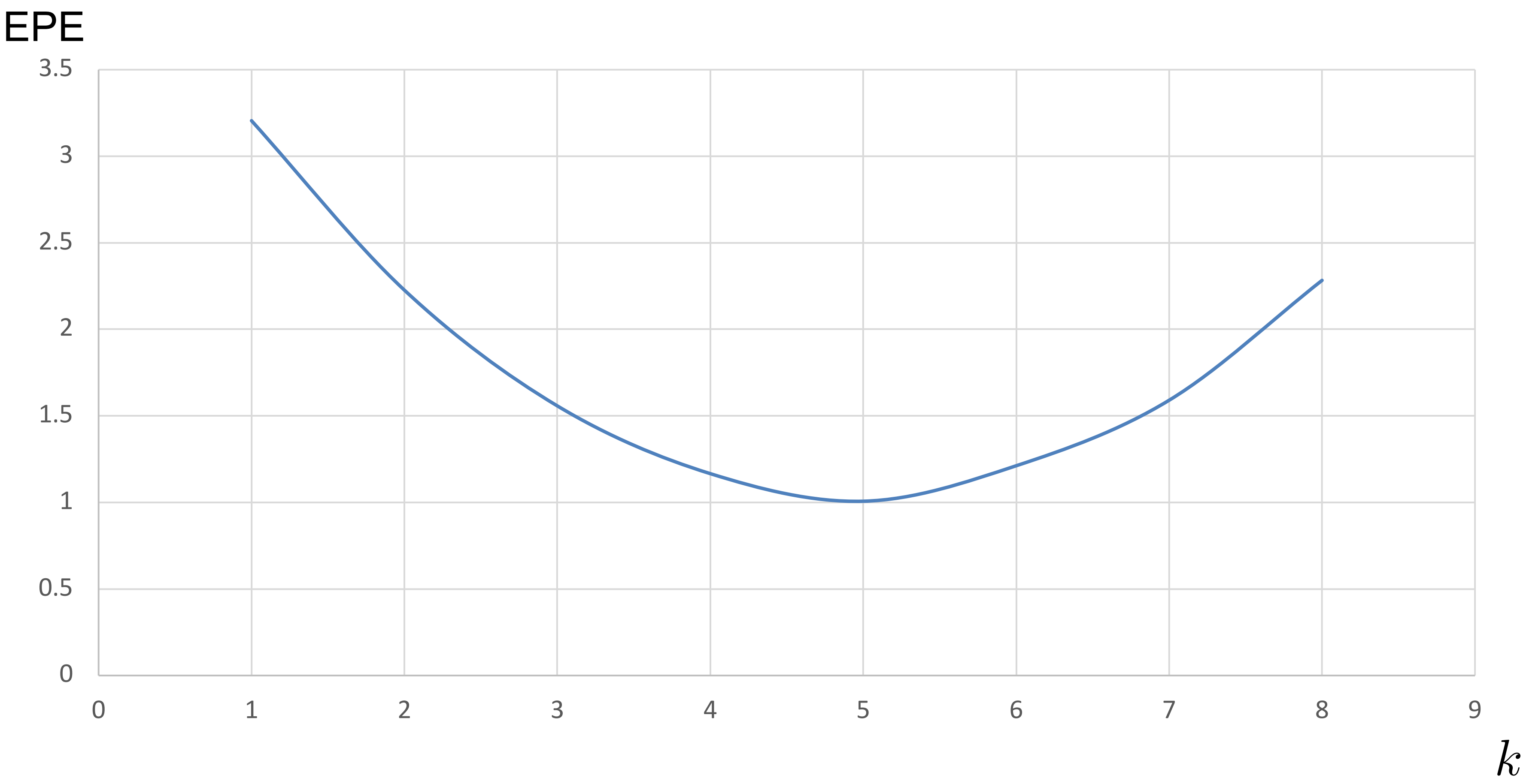
Large k :
high **bias**, low **variance**

EPE: $$\mathbb{E}\left[\left(y - \hat{f}(x)\right)^2\right] = \sigma^2 + \underbrace{\left(f(x) - \frac{1}{k}\sum_{i \in N_k(x)} f(x_i)\right)^2}_{\text{estimator bias}} + \underbrace{\frac{\sigma^2}{k}}_{\text{estimator variance}}$$

estimator **bias** estimator **variance**

Tufts

Bias vs. Variance Tradeoff



Why k-NN?

- Almost no statistical assumption about data other than continuity (though smoothness helps)
- Insensitive to outliers– accuracy can be affected from noise or irrelevant features
- Very simple to code!
- Works well in many cases!
- Versatile– useful for classification or regression

Why Not k-NN?

- ❑ Computationally expensive– base algorithm stores all training data
- ❑ High memory requirement
- ❑ Stores all (or almost all) of the training data
- ❑ Prediction stage might be slow (i.e., with big N)
- ❑ Sensitive to irrelevant features and the scale of the data

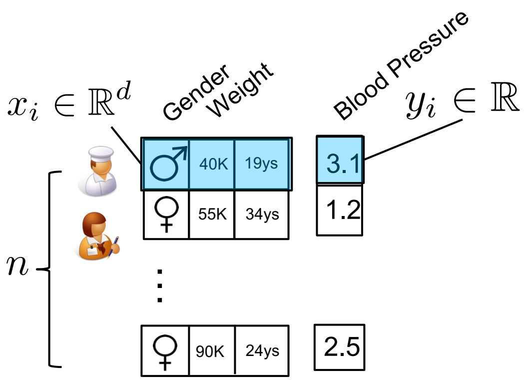
k-NN: Summary

- ❑ A positive integer *k* is specified, along with a new sample
- ❑ *K* entries in our database closest to new sample are selected
- ❑ Most common class is used to classify (i.e., majority voting)

Key KNN Features:

- ❑ KNN stores entire training dataset as representation
- ❑ KNN does not learn a model
- ❑ KNN makes prediction just-in-time by calculating similarity between an input sample and each training instance

Implementation in Practice



$$\hat{f}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$

- Use specifically designed **data structure** for nearest-neighbor queries

Tufts

Why **not** k-NN?

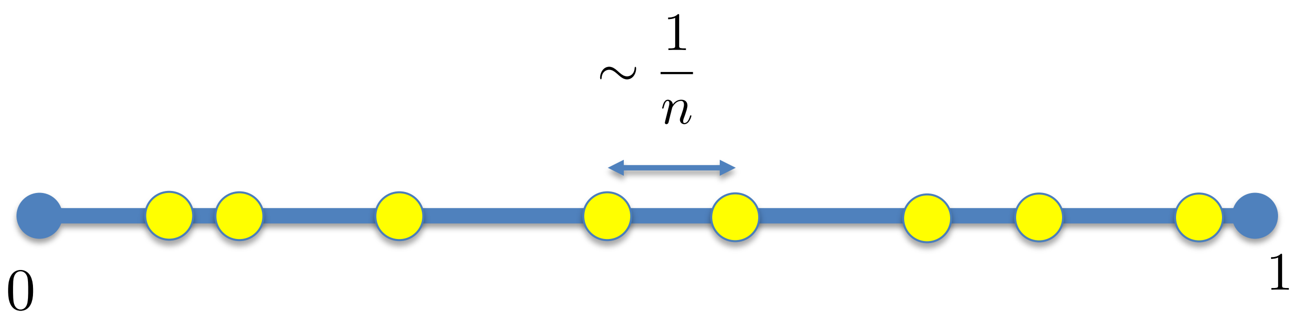
… the curse of dimensionality

Curse of Dimensionality

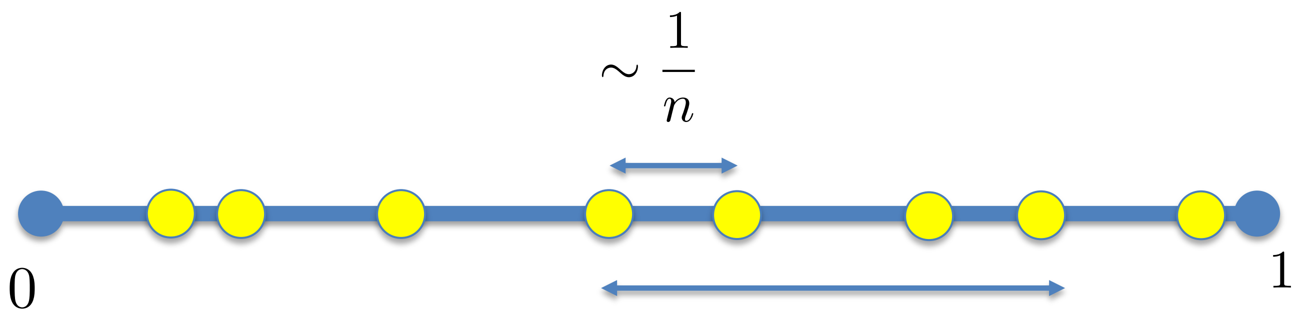
- Suppose that $d = 1$, and that you have a dataset of n samples, where

$$x_i \in [0, 1], \quad i = 1, \ldots, n.$$

- Suppose that points are distributed over $[0, 1]$



Curse of Dimensionality



k -th neighbor at distance at most $\sim \frac{k}{n}$

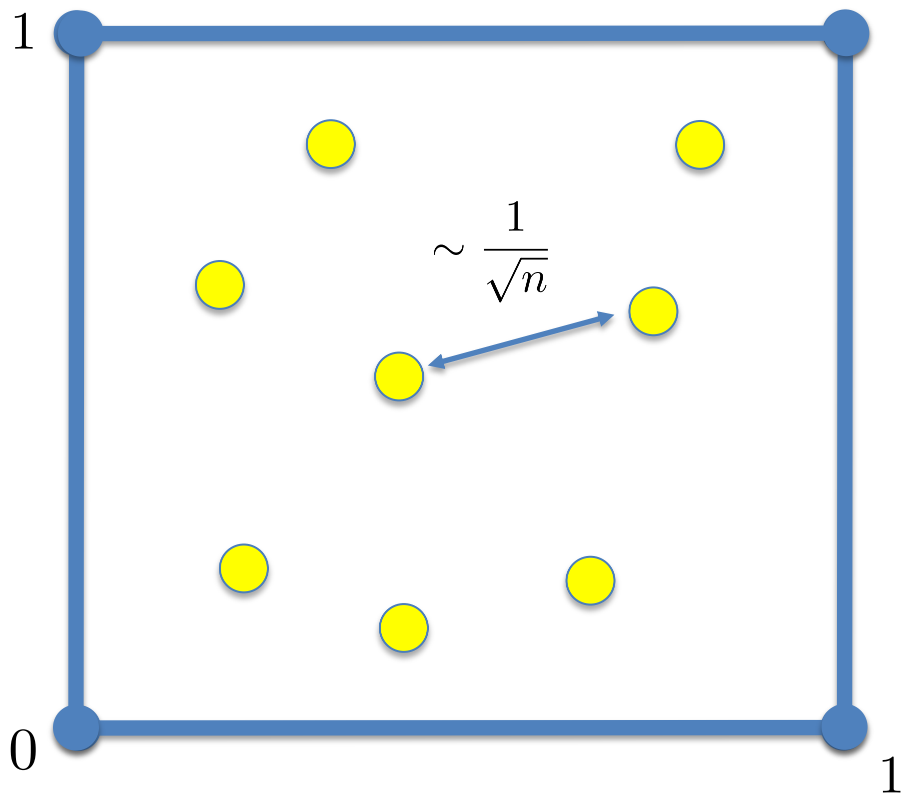
As you increase the number of samples n, k-NN becomes **less biased!**

If you increase k slowly enough with n, e.g., $k = \log n$,
both bias **and** variance will **go to zero!**

Curse of Dimensionality

- What if $d = 2$?

$$x_i \in [0,1]^2, \quad i = 1, \dots, n.$$

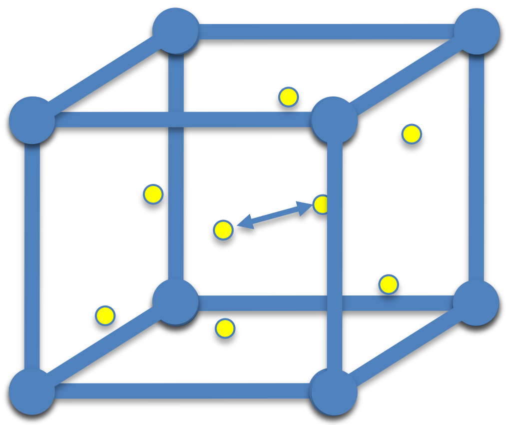


Tufts

Curse of Dimensionality

- ❑ What if $d = 3$?

$$x_i \in [0,1]^3, \quad i = 1, \dots, n.$$



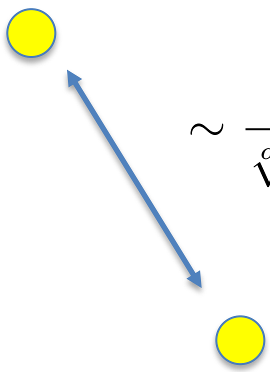
$$\sim \frac{1}{\sqrt[3]{n}}$$

Tufts

Curse of Dimensionality

- ❑ For arbitrary d

$$x_i \in [0,1]^d, \quad i = 1, \ldots, n.$$



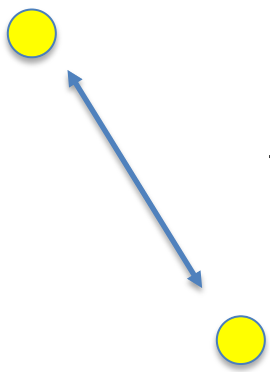
$$\sim \frac{1}{\sqrt[d]{n}}$$

$$n = 100, d = 100$$

$$\frac{1}{\sqrt[d]{n}} \approx 0.954992586021436$$

- ❑ Extremely **low density**
- ❑ Points are lie on **opposite boundaries**!

Curse of Dimensionality

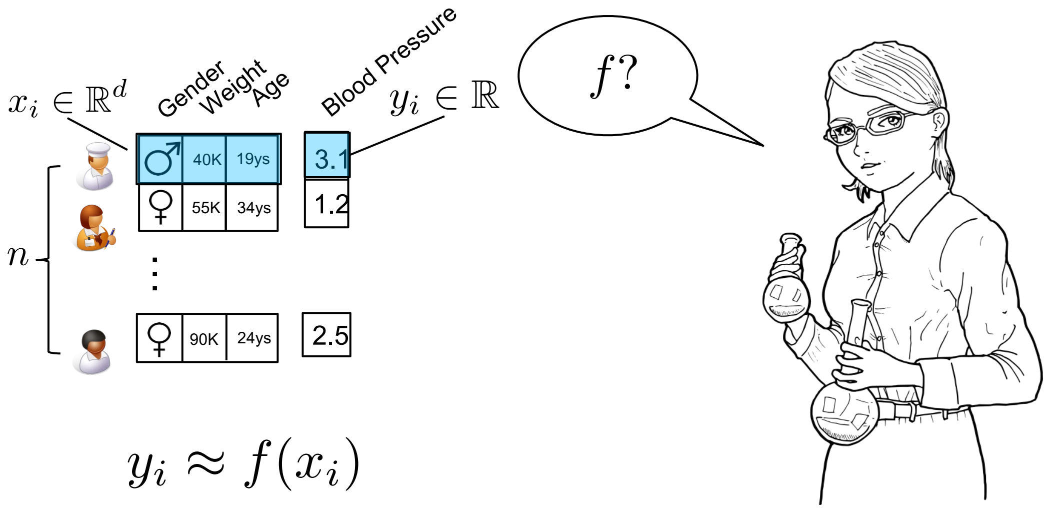


$$\frac{1}{\sqrt[d]{n}} \le \varepsilon \qquad \Leftrightarrow \qquad n \ge \left(\frac{1}{\epsilon}\right)^{d}$$

$$\left.\begin{array}{l}\varepsilon = 0.1\\ d = 100\end{array}\right\} \Rightarrow n \ge 10^{100}$$

- **Curse of Dimensionality**: To maintain an unbiased estimate with k-NN, the size of the dataset needs to grow **exponentially** with the dimension size!!!

Summary



- To regress f from data, we *need* to make *some assumption* on f …
- The assumption of *continuity* led us to k-NN…
- k-NN suffers from curse of dimensionality…