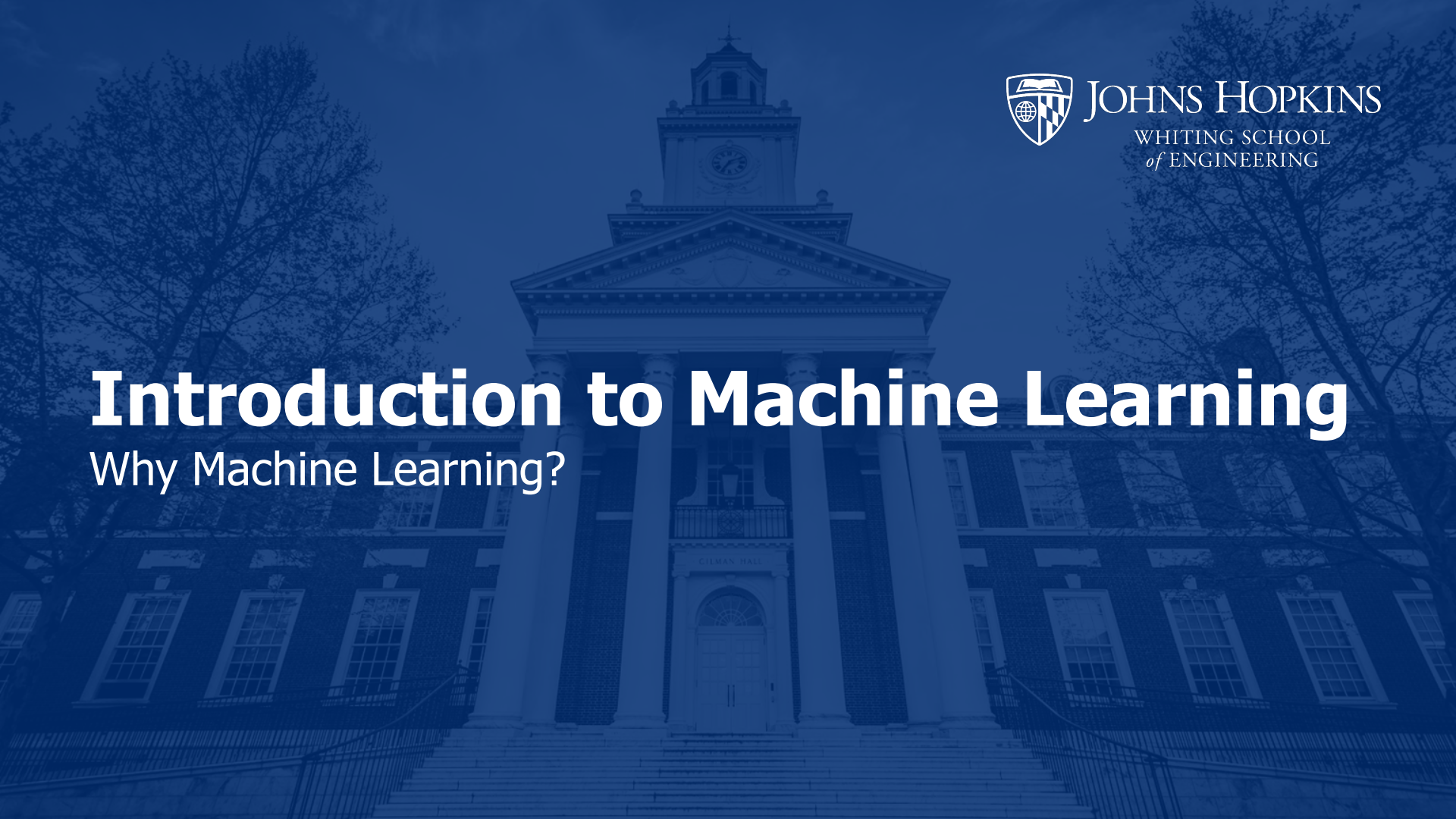


Introduction to Machine Learning

Why Machine Learning?

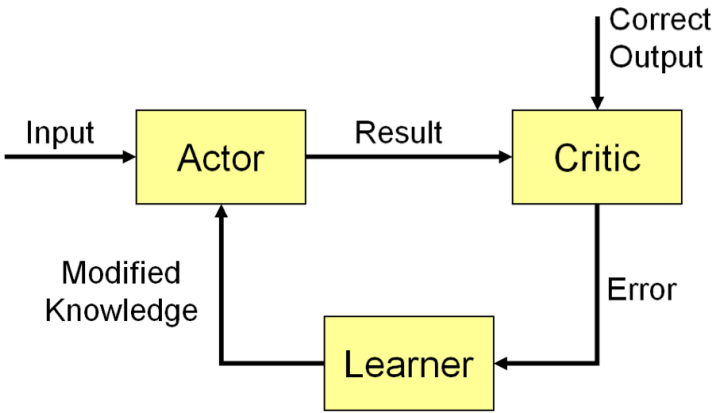
What is Machine Learning?

- A computer program that improves performance over time.
- Mapping of data into a model (Data → Model).
- Learning behavior from data.
- Discovering solutions to hard problems.

Definition: A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance on tasks *T* as measured by *P* improves with *E*.

Types of Machine Learning

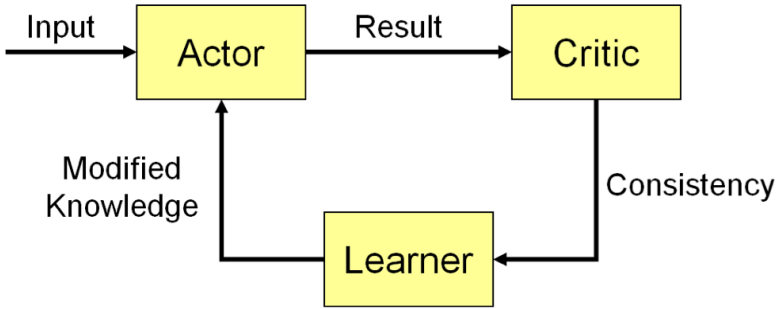
Supervised Learning



- Online learning
- Offline learning

Types of Machine Learning

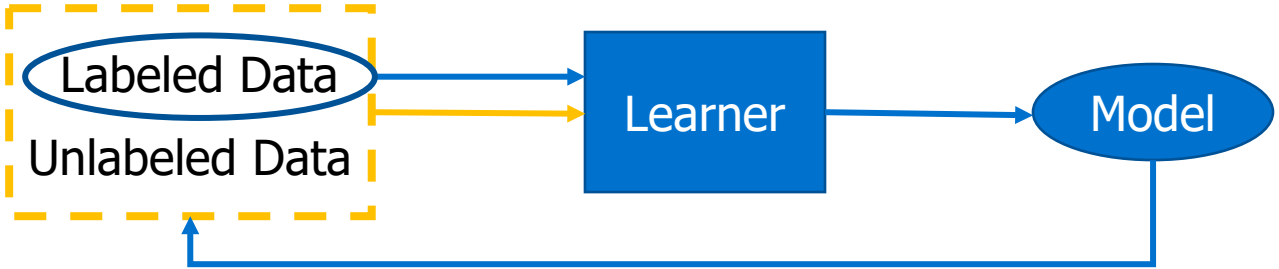
Unsupervised Learning



- Clustering
- Feature extraction

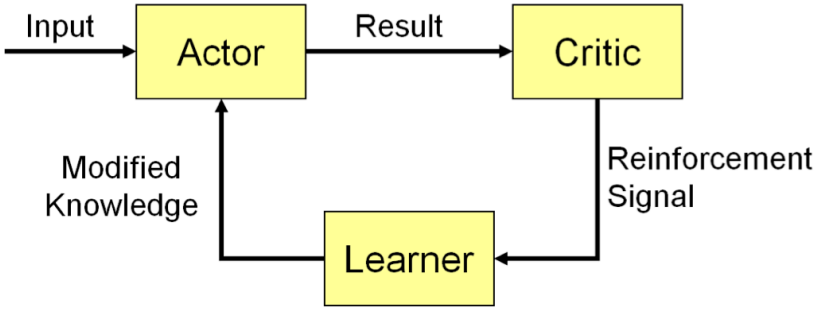
Semi-Supervised Learning

- An approach that mixes labeled and unlabeled data.
- One common approach is called "self training."



Types of Machine Learning

Reinforcement Learning



- Sequential decision problems
- Requires a reinforcement signal rather than ground truth

Example: The Winnow-2 Algorithm

Training

1. Receive a data instance
2. Make a prediction
3. Is the prediction correct? If not,
   - 0 ⇒ 1: Promotion
   - 1 ⇒ 1: Demotion

Classification

- Model:

$$f(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i$$

- For threshold θ,

$$h(\mathbf{x}) = \begin{cases} 1 & f(\mathbf{x}) > \theta \\ 0 & \text{otherwise} \end{cases}$$

Example: The Winnow-2 Algorithm

Promotion

If classify as 0 when correct class is 1, promote:

$$w_i = \begin{cases} \alpha w_i & \text{if } x_i = 1 \\ w_i & \text{if } x_i = 0 \end{cases}$$

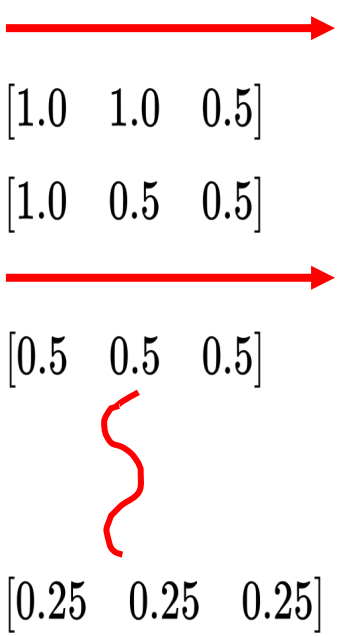
Demotion

If classify as 1 when correct class is 0, demote

$$w_i = \begin{cases} w_i / \alpha & \text{if } x_i = 1 \\ w_i & \text{if } x_i = 0 \end{cases}$$

Worked Example

x1	x2	x3	f(x)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

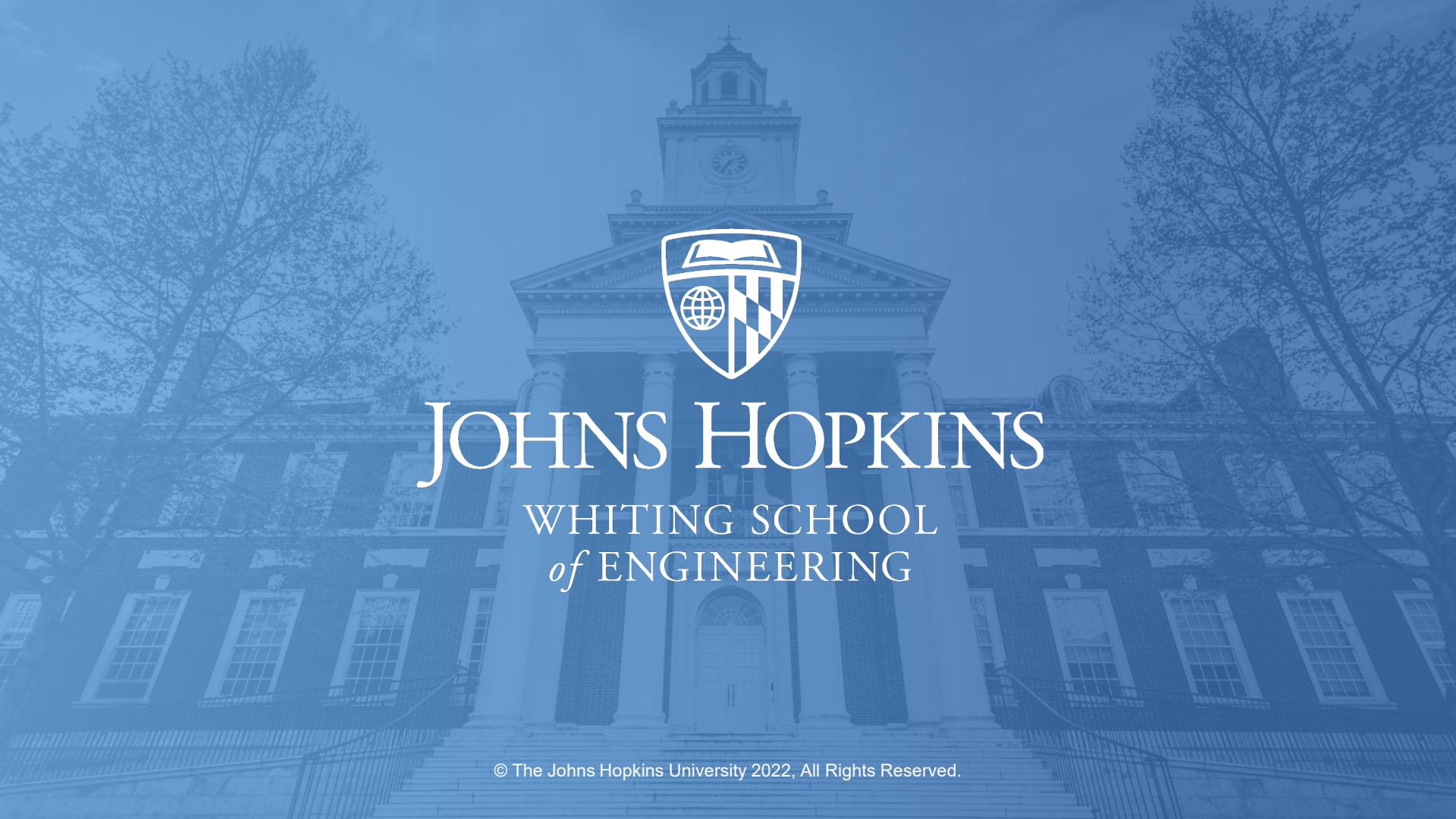


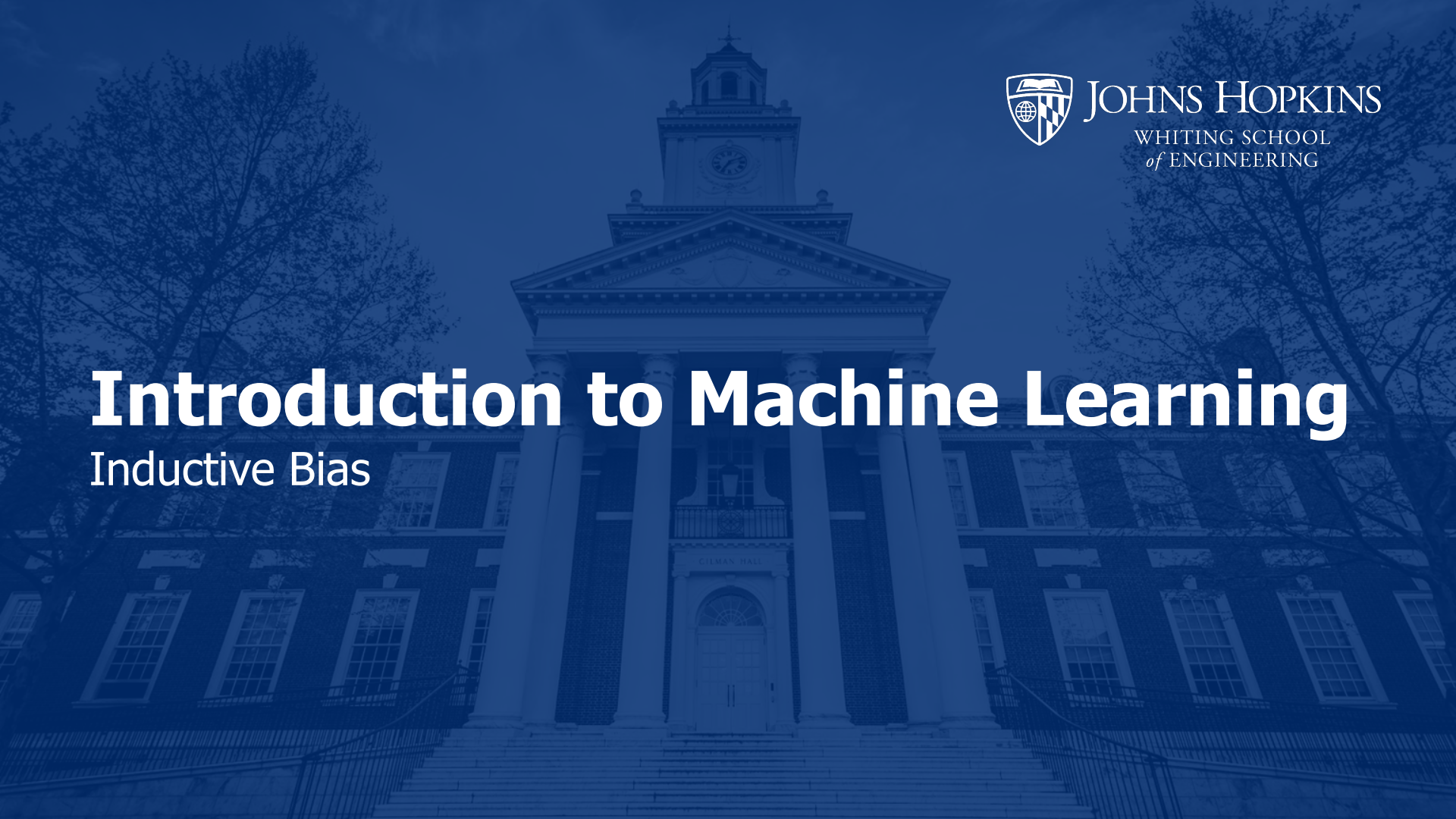
$$(x_1 \vee x_2) \wedge x_3$$

$$[1.0 \quad 1.0 \quad 1.0]$$

$$\alpha = 2$$

$$\theta = 0.5$$





Introduction to Machine Learning

Inductive Bias

Why Does Machine Learning Work?

Definition: The *inductive bias* of a learning system is the set of assumptions that, combined with the observed examples, deductively entail subsequent instance classifications made by the learner

Generalization!

- Tom Mitchell proved – "An unbiased learner cannot learn."
- Inductive bias is *good*!
- Inductive bias is *necessary*!
- Introduces what we call the "bias-variance tradeoff."

Types of Inductive Bias

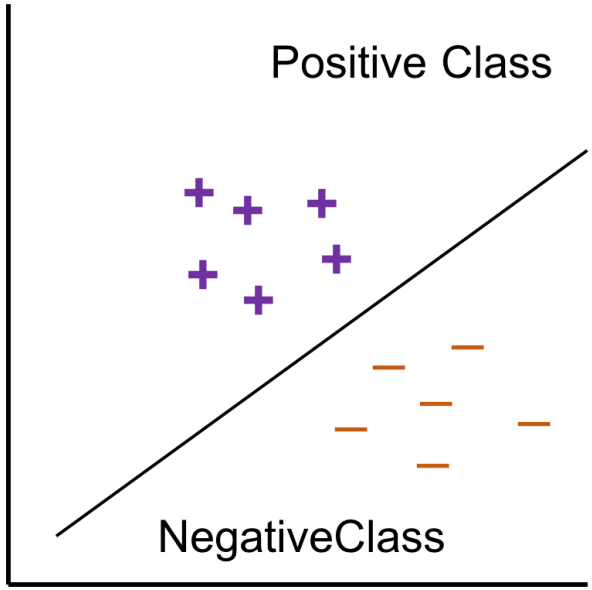
- **Representation Bias** – How the type of model defines the hypothesis (i.e., search) space for the learner.
- **Preference Bias** – How the learner searches the hypothesis space and decides which hypotheses are better than others.

Example: Winnow-2

Winnow-2 has a *linear* representation bias

$$f(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i > \theta$$

Linear Inductive Bias



Linear Inductive Bias

Let weights be [0.5 0.5]

Assume a threshold of $\theta = 0.5$

x_1	x_2	$f(\mathbf{x})$
0	0	0
0	1	0
1	0	0
1	1	1

$f(\mathbf{x}) = (x_1 \wedge x_2)$

Linear Inductive Bias

Exclusive OR has no linear separator.

Concept does not match the linear representation bias.

x_1	x_2	$f(\mathbf{x})$
0	0	0
0	1	1
1	0	1
1	1	0

$f(\mathbf{x}) = (x1 \oplus x_2)$

Preference Bias

- Also sometimes referred to as "selection bias," but this term can be confused with the sampling bias of the same name.
- Emphasizes that machine learning is a *search* process.
- Also emphasizes the search in machine learning involves *optimization*.
- The objective is to maximize performance

Examples

- Maximizing a performance measure / Minimizing a loss function.
- Making the model as simple as possible (Principle of Parsimony, aka Occam's Razor): this avoids "overfitting."

No Free Lunch

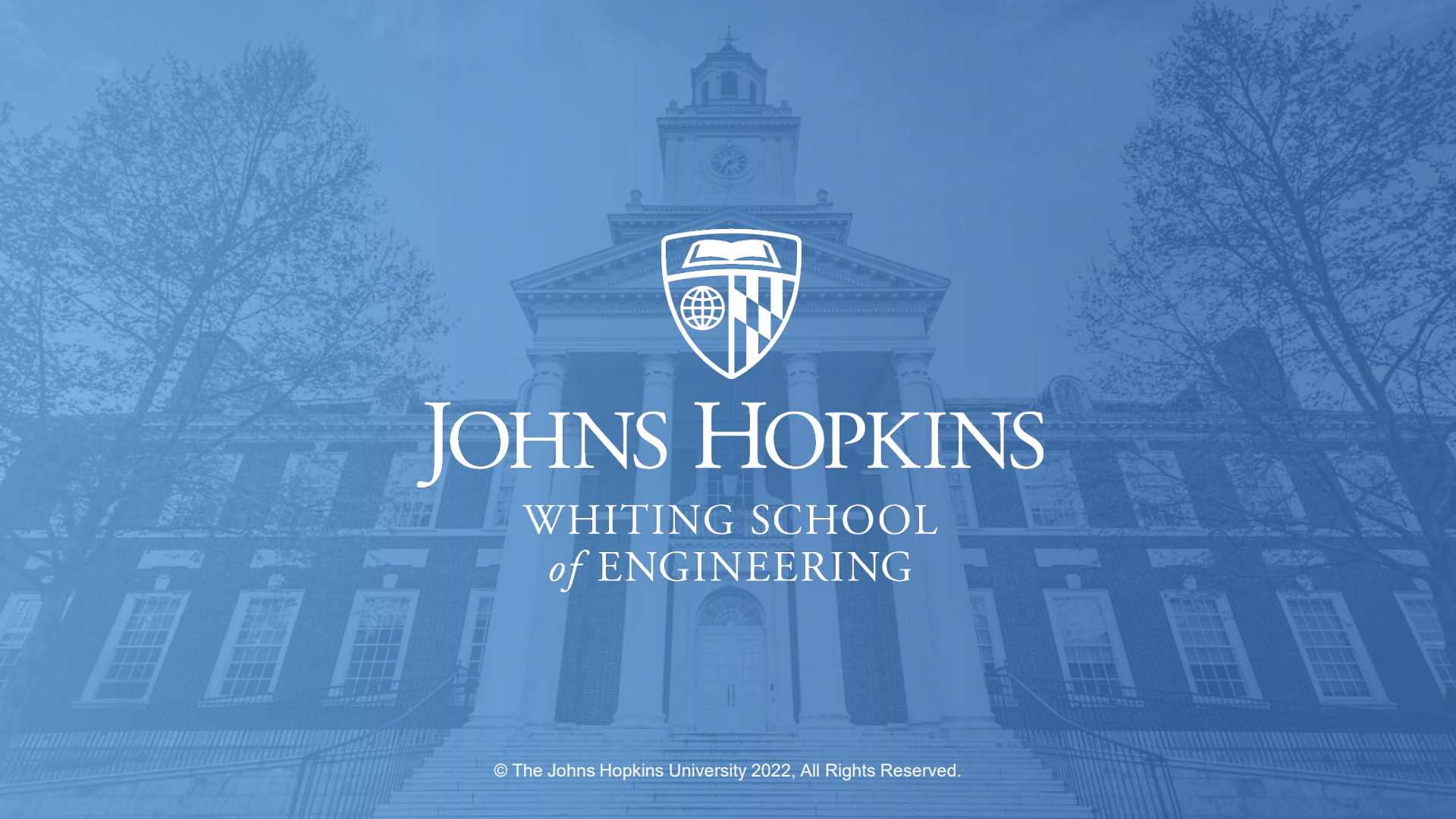
- Consider machine learning as an optimization problem.
- Define $d_c = \{\langle d_c^x(1), d_c^y(1)\rangle, \ldots, \langle d_c^x(m), d_c^y(m)\rangle\}$
- Thus the *i* th mapping is $\langle d_c^x(i), d_c^y(i)\rangle$
- We wish to compare two algorithms: *a*1 and *a*2.
- We compare relative to learning a target function *f*.
- We want to consider $P(d_c^v \mid f, m, a)$

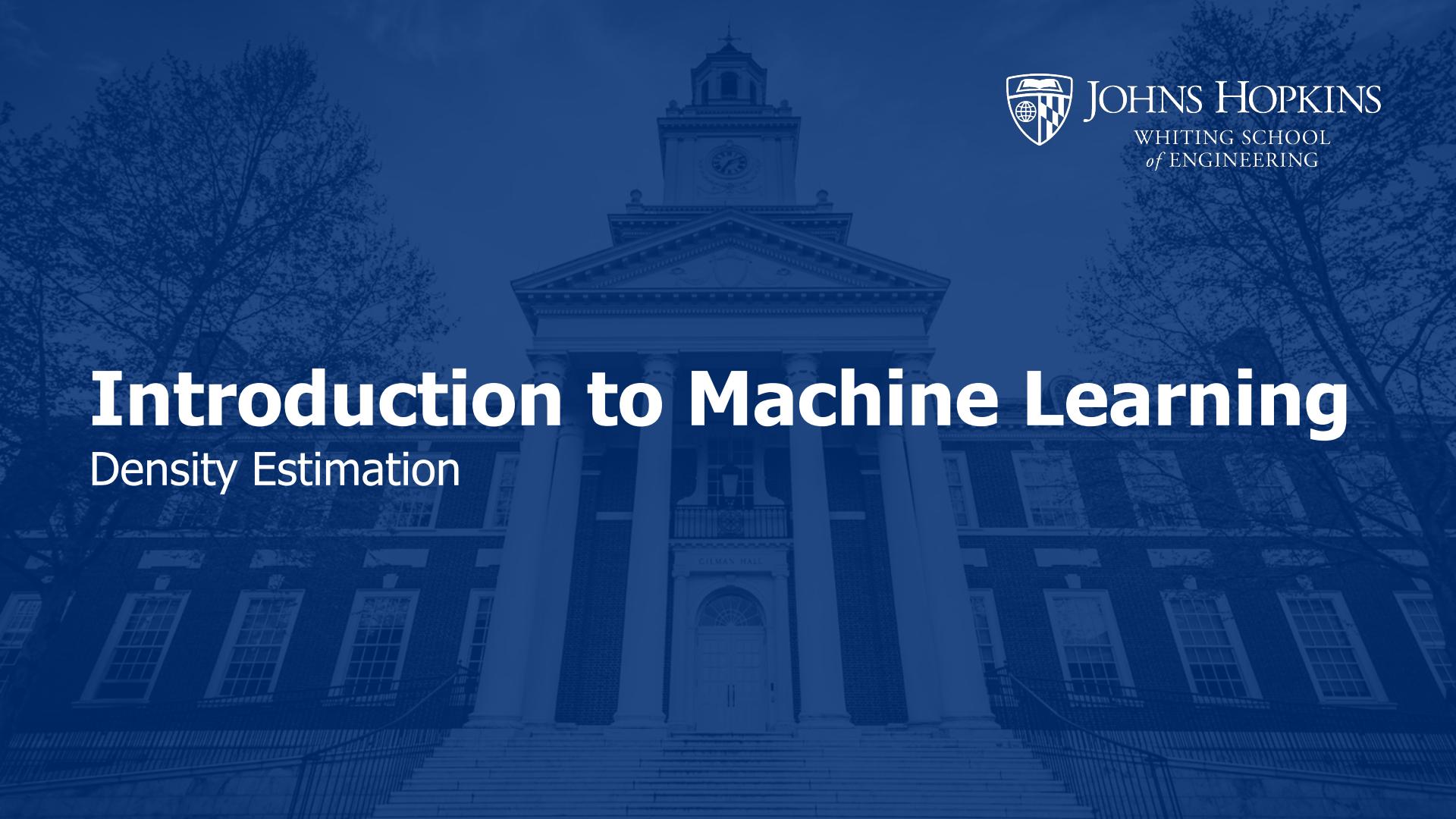
Theorem (Wolpert & Macready): For any pair of algorithms *a*1 and *a*2,

$$\sum_f P(d_c^y \mid f, m, a1) = \sum_f P(d_c^y \mid f, m, a2)$$

Take Aways

- Don’t look for a panacea of learning algorithms.
- Understand your problem and your data.
- Consider the effect of the performance function on the behavior of the learning algorithm.
- Pick an algorithm that is well-suited to optimize that performance function.





Introduction to Machine Learning

Density Estimation

The Problem

- Given a data set, we want to estimate the probability distribution for that data.
- For discrete random variables, this involves estimating the probability *mass* function.
- For continuous random variables, this involves estimating the probability *density* function.

$$P(\mathbf{X}) = \int_{\mathcal{R}} p(\mathbf{x}')d\mathbf{x}'$$

Deriving an Approach

- We work off of *n* data samples, $\mathbf{x}_1, \ldots, \mathbf{x}_n$
- We begin with a nonparametric assumption – we do not know a parametric form for the underlying distribution.
- We focus on answering the question, "What is the probability that k of the samples fall in some region $\mathcal{R}$.

$$P_k(\mathbf{x}) = \binom{n}{k} P^k (1-P)^{n-k}$$

$$E[k] = nP$$

$$\frac{k}{n} \approx P$$

Deriving an Approach

- Combining:

$$P(\mathbf{X}) = \int_{\mathcal{R}} p(\mathbf{x}')d\mathbf{x}' \approx p(\mathbf{x})V \approx \frac{k}{n}$$

$$p(\mathbf{x}) \approx \frac{k/n}{V} = \frac{k}{nV}$$

- Assume a hypercube with side length equal to h. Then $V \approx h^d$ so

$$P(x) \approx \frac{k}{nh^d}$$

Histogram Estimator

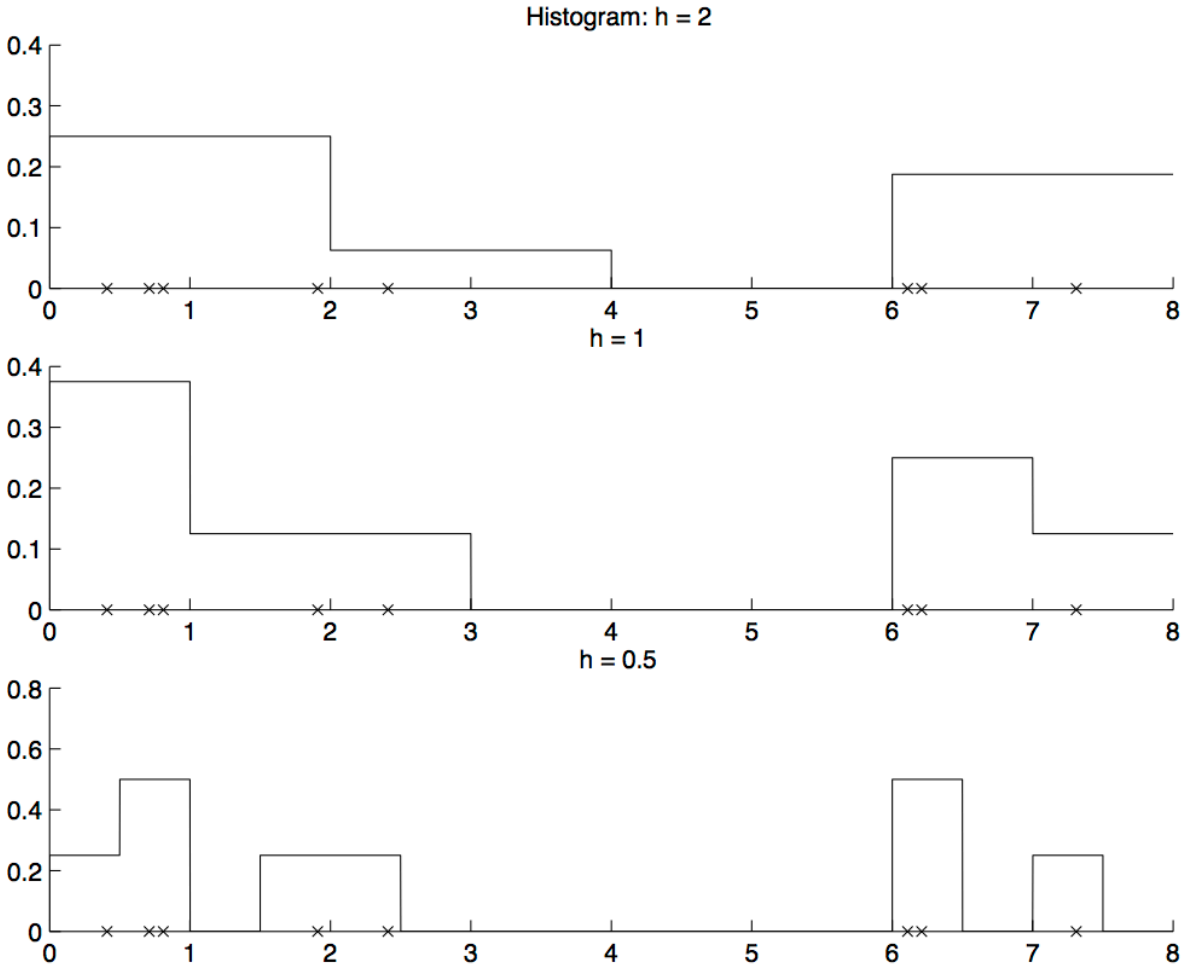
$$\mathcal{X} = \{\mathbf{x}^t\}_{t=1}^n$$

- Set an origin at $\mathbf{x}_0$
- Define a bin width *h*.
- Specify a bin as

$$[\mathbf{x}_0 + mh, \mathbf{x}_0 + (m+1)h)$$

- Then

$$\tilde{p}(\mathbf{x}) = \frac{1}{nh} \sum_{t=1}^{n} \mathbb{I}\{\mathbf{x}^t \in \text{bin}(\mathbf{x})\}$$



(Alpaydin, 2019)

Naïve Estimator

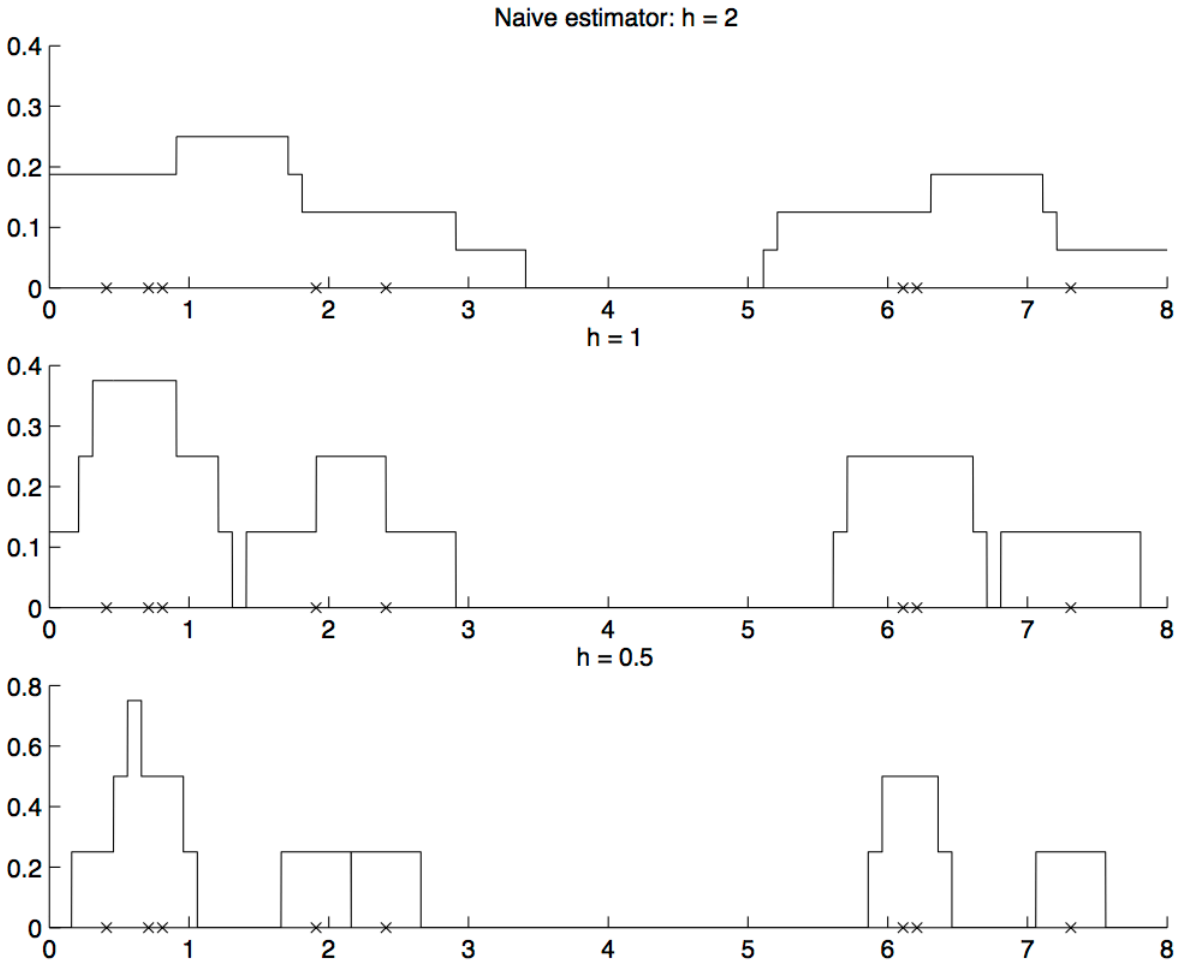
$$\mathcal{X} = \{\mathbf{x}^t\}_{t=1}^n$$

- No origin needed
- Define a bin width *h*.
- Specify a bin around query point **x** as

$$\left(\mathbf{x}^t - h, \mathbf{x}^t + h\right]$$

- Then

$$\tilde{p}(\mathbf{x}) = \frac{1}{2nh} \sum_{t=1}^{n} \mathbb{I}\left\{\left(\mathbf{x}^t - h, \mathbf{x}^t + h\right]\right\}$$



(Alpaydin, 2019)

Weighted Estimators

- We can generalize this idea.

$$\tilde{p}(\mathbf{x}) \approx \frac{1}{nh} \sum_{t=1}^{n} w\left(\frac{\mathbf{x} - \mathbf{x}^t}{h}\right)$$

- where

$$w(\mathbf{u}) = \begin{cases} \frac{1}{2} & \text{if } |\mathbf{u}| < 1 \\ 0 & \text{otherwise} \end{cases}$$

Kernel Functions

- This weighting "function" forms the basis for what we call *kernels*, denoted $K(\mathbf{u})$
- The weight is determined based on similarity to a data point, where

$$K(\mathbf{u}) \geq 0, \forall \mathbf{u}$$
$$K(-\mathbf{u}) = K(\mathbf{u}), \forall \mathbf{u}$$
$$\int_{-\infty}^{+\infty} K(\mathbf{u}) d\mathbf{u} = 1$$

- This defines a "feature map" $\phi : X \to \mathcal{V}$ such that

$$K\left(\mathbf{x}, \mathbf{x}'\right) = \left\langle \phi(\mathbf{x})^{\top} \phi(\mathbf{x}) \right\rangle_{\mathcal{V}}$$

Kernel Estimator

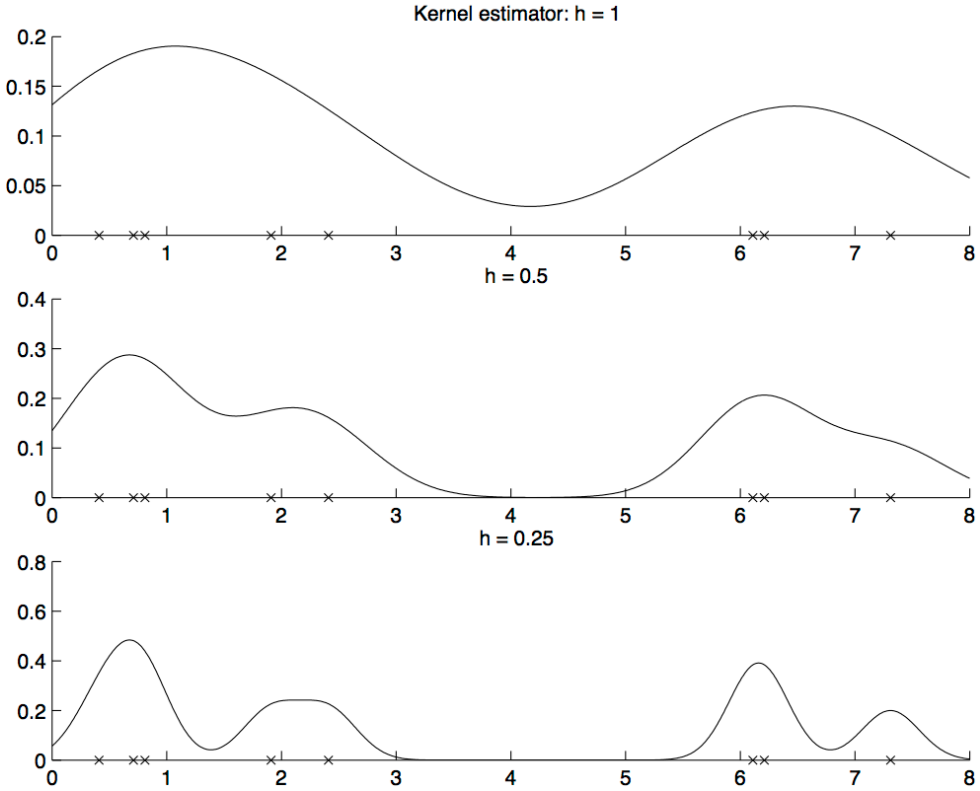
- This leads to what is known as "kernel density estimation" (KDE).

$$\tilde{p}(\mathbf{x}) = \frac{1}{nh} \sum_{t=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}^t}{h} \right)$$

$$K(\mathbf{u}) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{\mathbf{u}^2}{2} \right]$$

$$\tilde{p}(\mathbf{x}) = \frac{1}{\sqrt{2\pi} nh} \sum_{t=1}^{n} \exp\left[-\frac{\left(\mathbf{x} - \mathbf{x}^t\right)^2}{2h} \right]$$

Kernel Density Estimator



(Alpaydin, 2019)

Parzen Window Estimator

- An alternative builds off of the naïve estimation idea.

$$\tilde{p}(x) = \frac{1}{nh} \sum_{t=1}^{n} \mathbb{I}\left\{\left|x - x^t\right| \leq h\right\} K\left(\frac{x - x^t}{h}\right)$$

$$\tilde{p}(\mathbf{x}) = \frac{1}{nh^d} \sum_{t=1}^{n} K\left(\frac{\left\|\mathbf{x} - \mathbf{x}^t\right\|}{h}\right)$$

$$\tilde{p}(\mathbf{x}) = \frac{1}{nV} \sum_{t=1}^{n} K\left(\frac{\left\|\mathbf{x} - \mathbf{x}^t\right\|}{h}\right)$$

where

$$K(\mathbf{u}) = \left(\frac{1}{\sqrt{2\pi}}\right)^d \exp\left[-\frac{\|\mathbf{u}\|^2}{2}\right]$$

with covariance:

$$K(\mathbf{u}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}\mathbf{u}^\top \Sigma^{-1}\mathbf{u}\right]$$

K-Nearest Neighbor Estimator

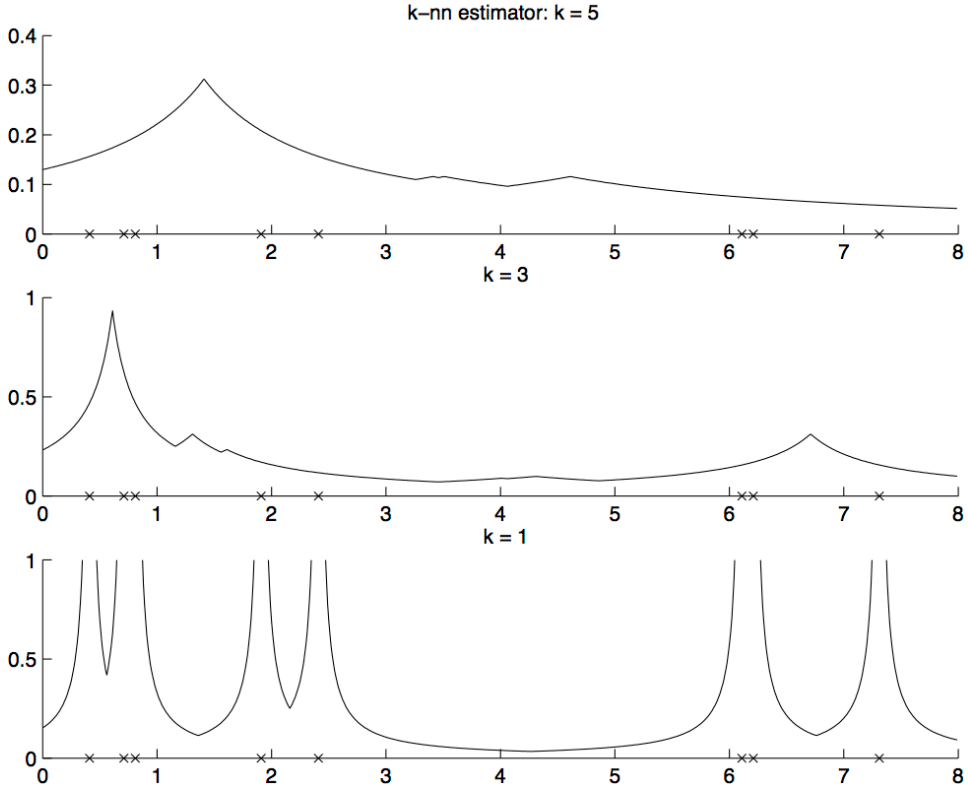
- Assumed we have a distance function $D(\mathbf{x}, \mathbf{x}')$
- Then

$$\tilde{p}(\mathbf{x}) = \frac{k}{2nD(\mathbf{x}, \mathbf{x}^k)}$$

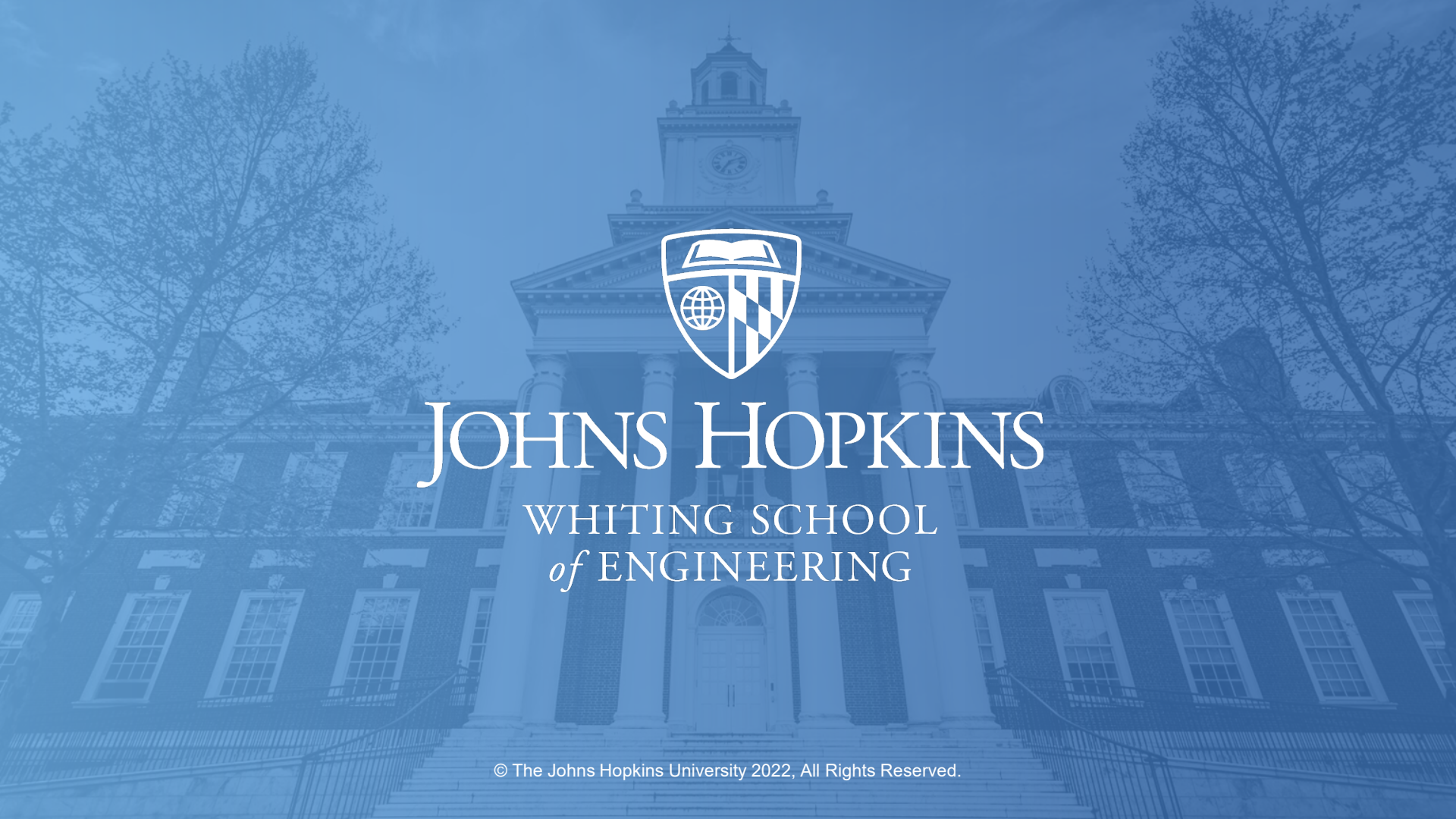
$$h = D(\mathbf{x}, \mathbf{x}^k)$$

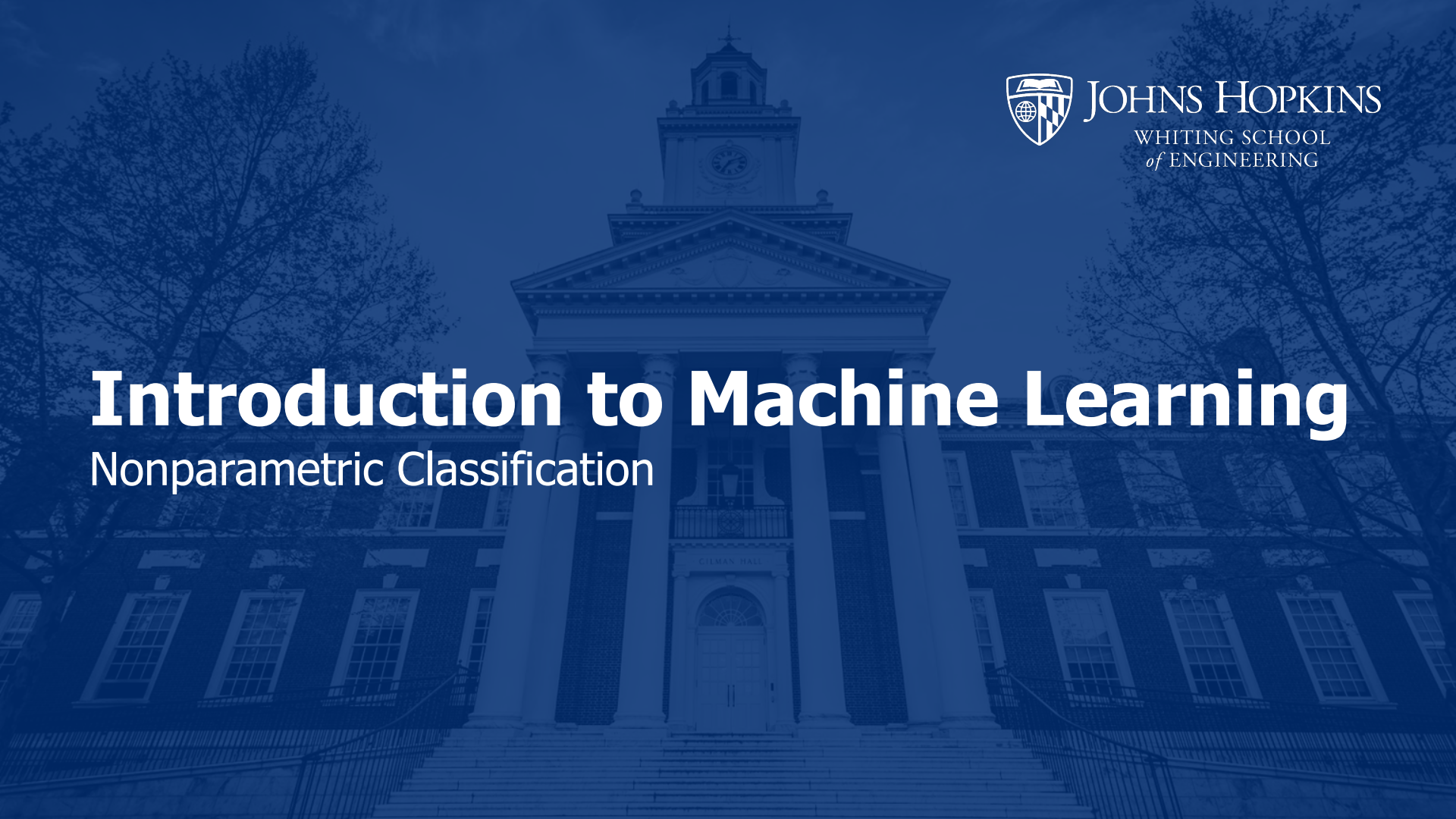
$$\tilde{p}(\mathbf{x}) = \frac{1}{nD(\mathbf{x}, \mathbf{x}^k)} \sum_{\mathbf{x}^t \in \mathcal{N}(\mathbf{x})} K\left(\frac{\mathbf{x} - \mathbf{x}^t}{D(\mathbf{x}, \mathbf{x}^k)}\right)$$

K-Nearest Neighbor Estimator



(Alpaydin, 2019)





Introduction to Machine Learning

Nonparametric Classification

Classification

- Suppose we are presented with a query point $\mathbf{x}_q$ and want to know the class to which it belongs.
- A probabilistic approach:

$$c = \arg\max_{c \in \mathcal{C}} P(c \mid \mathbf{x}_q)$$

where

$$P(c \mid \mathbf{x}_q) = \frac{P(\mathbf{x}_q \mid c)P(c)}{\sum_{c' \in \mathcal{C}} P(\mathbf{x}_q \mid c')P(c')}$$

- Thus we have a density estimation task where we need to estimate $P(\mathbf{x}_q \mid c)$ and $P(c)$

Parzen Window Classification

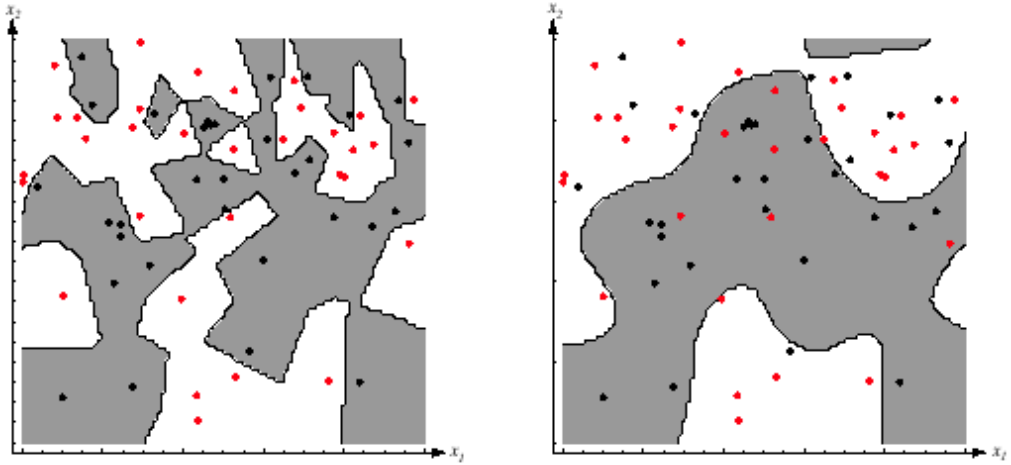
- Given $x = \left\langle \left\{ x_i^1, \ldots, x_i^d \right\}, c_i \right\rangle_{i=1}^n$
- Find the class to which query point $\mathbf{x}_q$ belongs.

$$H(\mathbf{u}) = \begin{cases} 1 & |u_j| < \frac{1}{2}, j = 1, \ldots, d \\ 0 & \text{otherwise} \end{cases}$$

where

$$\mathbf{u} = \frac{\mathbf{x} - \mathbf{x}_q}{h^d}, \quad K_c(\mathbf{x}_q) = \sum_{t=1}^{n} H\left(\frac{\mathbf{x}_q - \mathbf{x}^t}{n} \right) \Bigg|_c, \quad c = \arg\max_{\mathbf{c}} K_c(\mathbf{x}_q)$$

Example Parzen Window



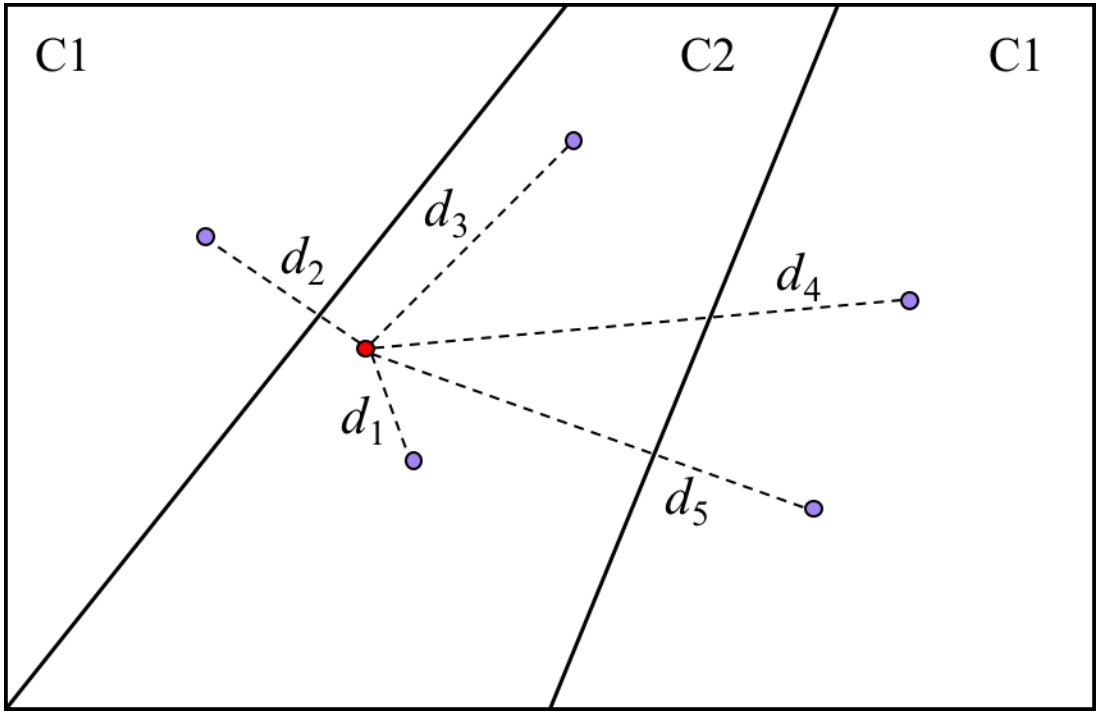
(Duda, Hart, & Stork, 2001)

Nearest Neighbor Classification

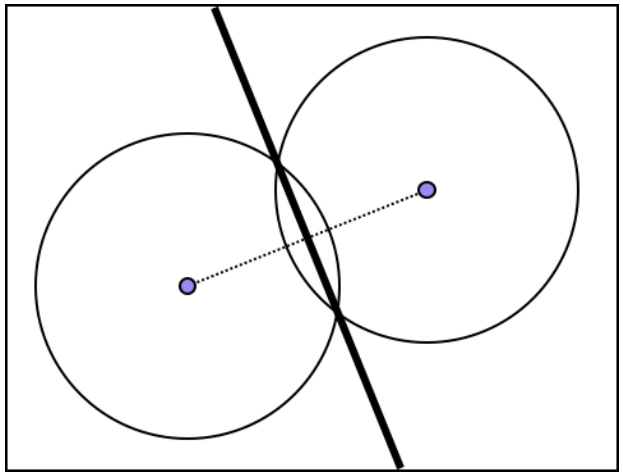
- Assume that $k = 1$
- Given $x = \left\langle \left\{ x_i^1, \ldots, x_i^d \right\}, c_i \right\rangle_{i=1}^n$
- Find the class to which query point $\mathbf{x}_q$ belongs.
- The nearest neighbor rule states,

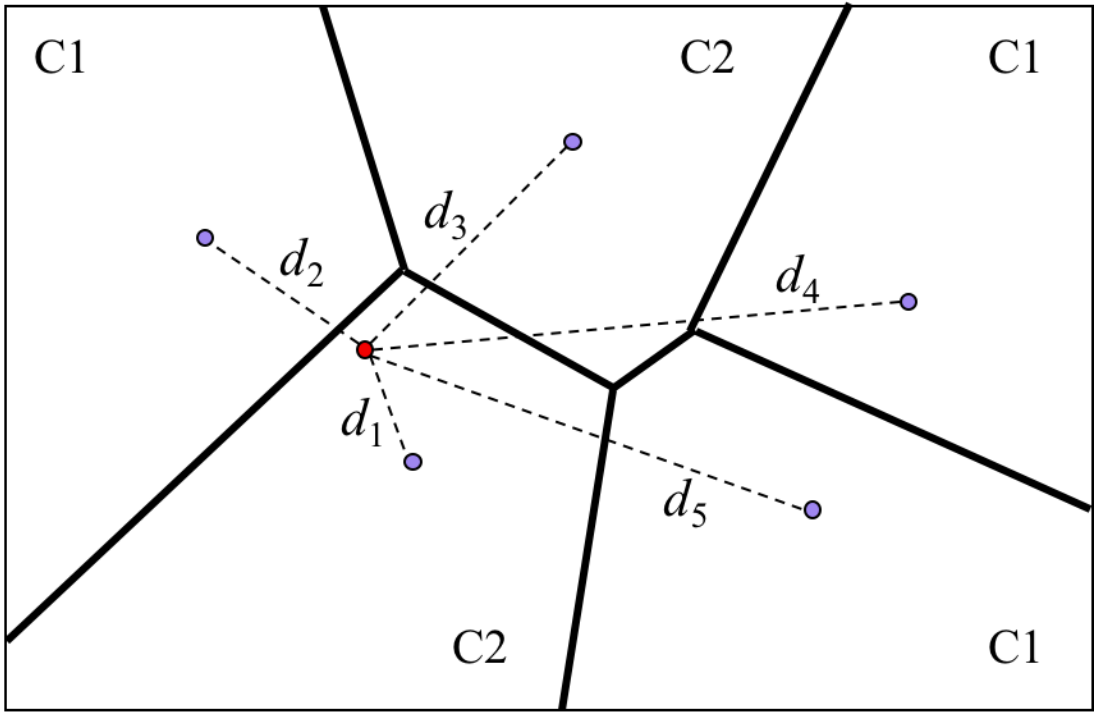
> Find $\mathbf{r} \in \mathcal{X}$ such that $\forall \mathbf{x} \in \mathcal{X}, \mathbf{x} \neq \mathbf{r}, D(\mathbf{x}_q, \mathbf{r}) < D(\mathbf{x}_q, \mathbf{x})$, and return the class label $c \in \mathcal{C}$ associated with example $\mathbf{r}$.

Example



Voronoi Diagrams





Distance Metrics

- Metric Spaces
  1. Non-negative: $D(\mathbf{x}, \mathbf{y}) \geq 0$
  2. Reflexive: $D(\mathbf{x}, \mathbf{y}) = 0$ if and only if **x** = **y**
  3. Symmetric: $D(\mathbf{x}, \mathbf{y}) = D(\mathbf{y}, \mathbf{x})$
  4. Triangle Inequality: $D(\mathbf{x}, \mathbf{y}) + D(\mathbf{y}, \mathbf{z}) \geq D(\mathbf{x}, \mathbf{z})$
- Minkowski Metrics: $D_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p}$
  - Manhattan Distance: $D_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} |x_i - y_i|$
  - Euclidean Distance: $D_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$
  - Max-coordinate Distance: $D_\infty(\mathbf{x}, \mathbf{y}) = \max_i |x_i - y_i|$

Edited Nearest Neighbor

- Nearest neighbor methods are expensive.
- They are also susceptible to noisy data and irrelevant features.
- Can draw inspiration from what we call “Helpful Teacher” methods.
  - Improves computational performance
  - Focuses on questionable regions in data
- ”Editing” is similar to stepwise backward elimination in feature selection.

Edited Nearest Neighbor – v1

The first approach removes misclassified examples.

1. For each $\mathbf{x}^t \in \mathcal{X}$ do
   a) Classify $\mathbf{x}^t$ using one-nearest-neighbor on $X \setminus \{\mathbf{x}^t\}$
   b) Compare class returned to true class of $\mathbf{x}^t$
   c) If disagree, remove $\mathbf{x}^t$ from $\mathcal{X}$
2. Repeat until performance degrades on separate validation data set

Edited Nearest Neighbor – v2

The first approach removes misclassified examples.

1. For each $\mathbf{x}^t \in \mathcal{X}$ do
   a) Classify $\mathbf{x}^t$ using one-nearest-neighbor on $X \setminus \{\mathbf{x}^t\}$
   b) Compare class returned to true class of $\mathbf{x}^t$
   c) If agree, remove $\mathbf{x}^t$ from $\mathcal{X}$
2. Repeat until performance degrades on separate validation data set

Condensed Nearest Neighbor

Draws inspiration from stepwise forward selection.

We seek to find a subset of data points $z \subset x$

1. Initialize $z \leftarrow \text{rand}(x)$
2. Repeat
   a) For each $x^t \in X$
      - Find $x' = \arg\min_{x'' \in \mathcal{Z}} D\left(\mathbf{x}'', \mathbf{x}^t\right)$
      - If $\text{class}\left(\mathbf{x}'\right) \neq \text{class}\left(\mathbf{x}^t\right)$ add $\mathbf{x}'$ to $\mathcal{Z}$
3. Until $\mathcal{Z}$ does not change

The Value Difference Metric

- Let $f(\mathbf{x})$ be a feature of data instance $\mathbf{x}$. We assume $f(\mathbf{x}) \in \{v_1, \ldots, v_k\}$
- Let $C_i = \#\{f(\mathbf{x}) = v_i\}$ be the count of the number of times in data set $\mathcal{D}$ that $f(\mathbf{x})$ takes on the value of v_i
- Let $C_{i,a} = \#\{f(\mathbf{x}) = v_i \wedge \text{class} = a\}$
  be the count of the number of times in data set $\mathcal{D}$ that $\mathbf{x}$ is a member of class *a* and $f(\mathbf{x})$ takes on the value of v_i
- For this feature *f*, define the distance between feature values v_i and v_j as follows:

$$\delta(v_i, v_j) = \sum_{a=1}^{\text{num classes}} \left| \frac{C_{i,a}}{C_i} - \frac{C_{j,a}}{C_j} \right|^p$$

- Then the distance between two data instances $\mathbf{x}$ and $\mathbf{y}$ is defined to be

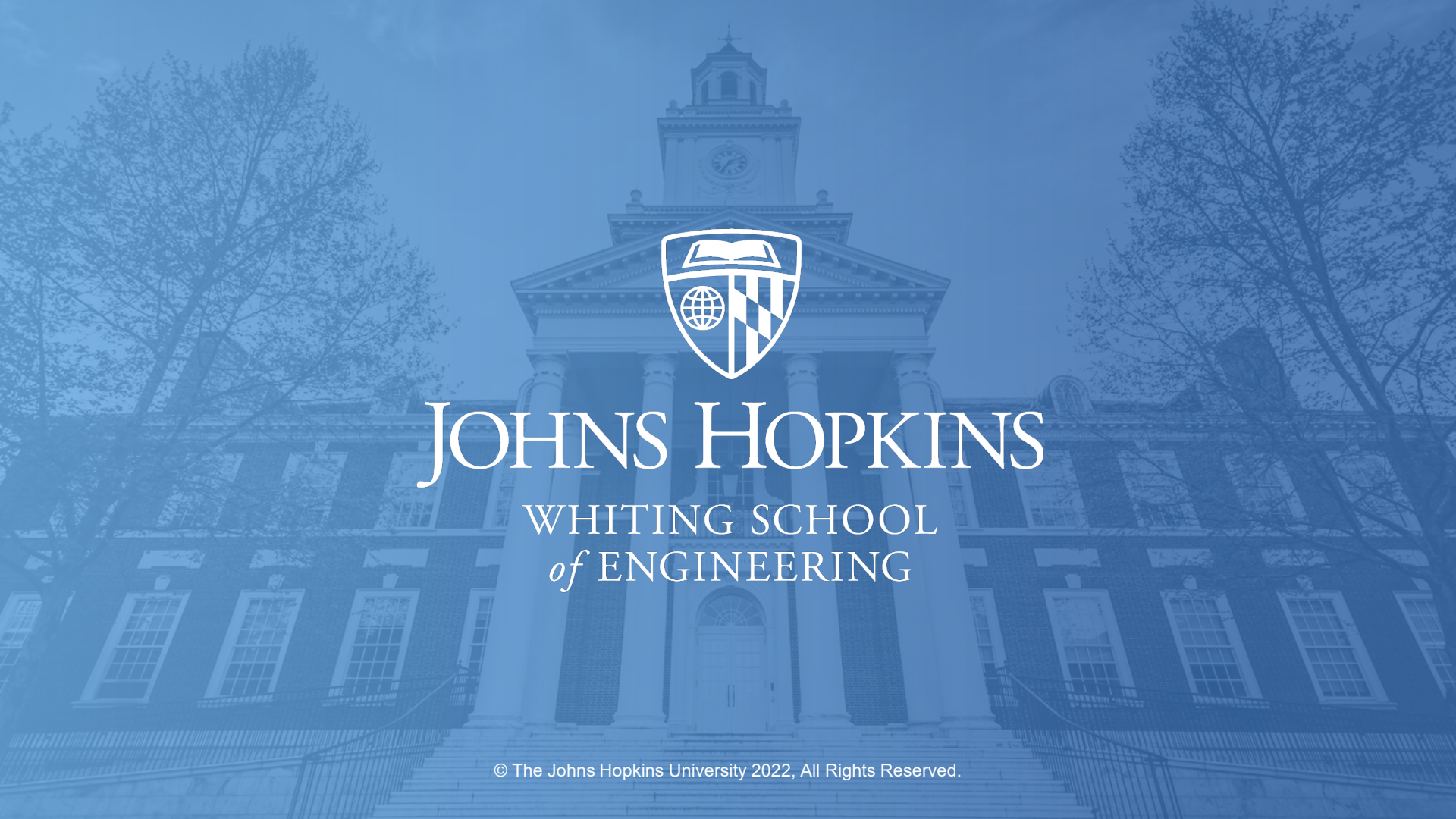
$$D_{\text{cat}}(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^{d} \delta(\mathbf{x}_k, \mathbf{y}_k) \right)^{1/p}$$

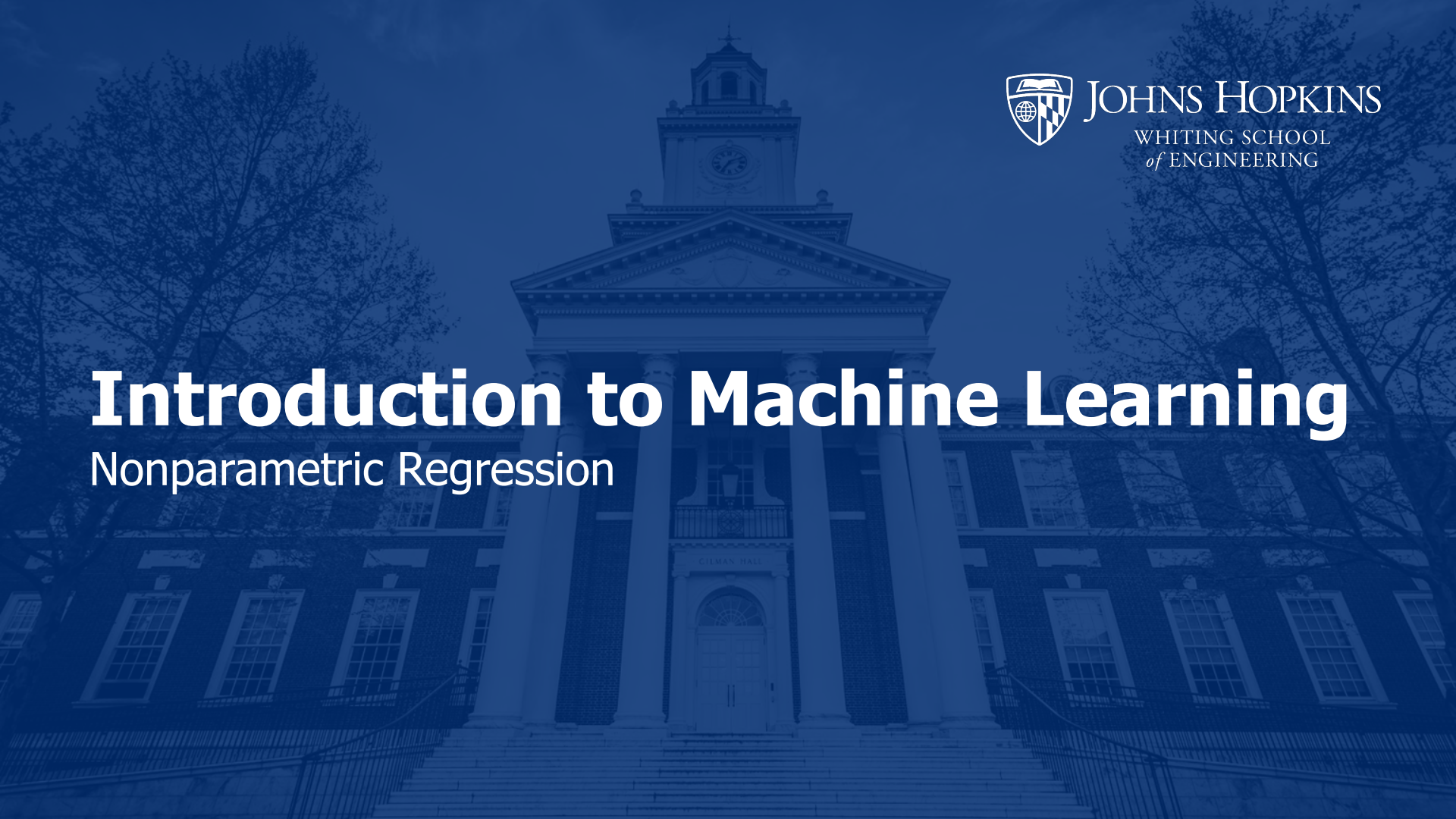
Mixed Data Types

- Suppose we have data instances $\mathbf{x}, \mathbf{y}$ consisting of a combination of categorical and numerical features.
- Remember to normalize the numerical features first (to range 0 … 1).
- Let $D_{\text{num}}(\mathbf{x}, \mathbf{y})$ be the distance between the numerical part of the two instances.
- Let $D_{cat}(\mathbf{x}, \mathbf{y})$ be the distance between the categorical part of the two instances.
- Calculate a combined distance as follows:

$$D(\mathbf{x}, \mathbf{y}) = \sqrt[p]{(D_{\text{num}}(\mathbf{x}, \mathbf{y}))^p + (D_{\text{cat}}(\mathbf{x}, \mathbf{y}))^p}$$

- Since the features are "normalized," this effectively combines the base components of the distance calculations. (Note the *p*-th root isn't necessary!)





Introduction to Machine Learning

Nonparametric Regression

Regression

- Suppose we are presented with a query point $\mathbf{x}_q$ and want to compute the function value at that point.

$$X = \left\{\mathbf{x}^t, r^t\right\}_{t=1}^{n}, \quad r^t = g\left(\mathbf{x}^t\right) + \epsilon$$

- Parametric Regression
  - Need to find coefficients in a predefined functional form
  - Minimize loss such as sum squared error
  - Leads to "least squares" regression
- Nonparametric Regression
  - Examples from data define the "shape" of the regression function
  - Compute regressed output $\mathbf{x}_q$ at query time
  - Results in what is called a "smoother"

Regressogram Smoother

First specify an origin $\mathbf{x}_0$

and a bin width *h*

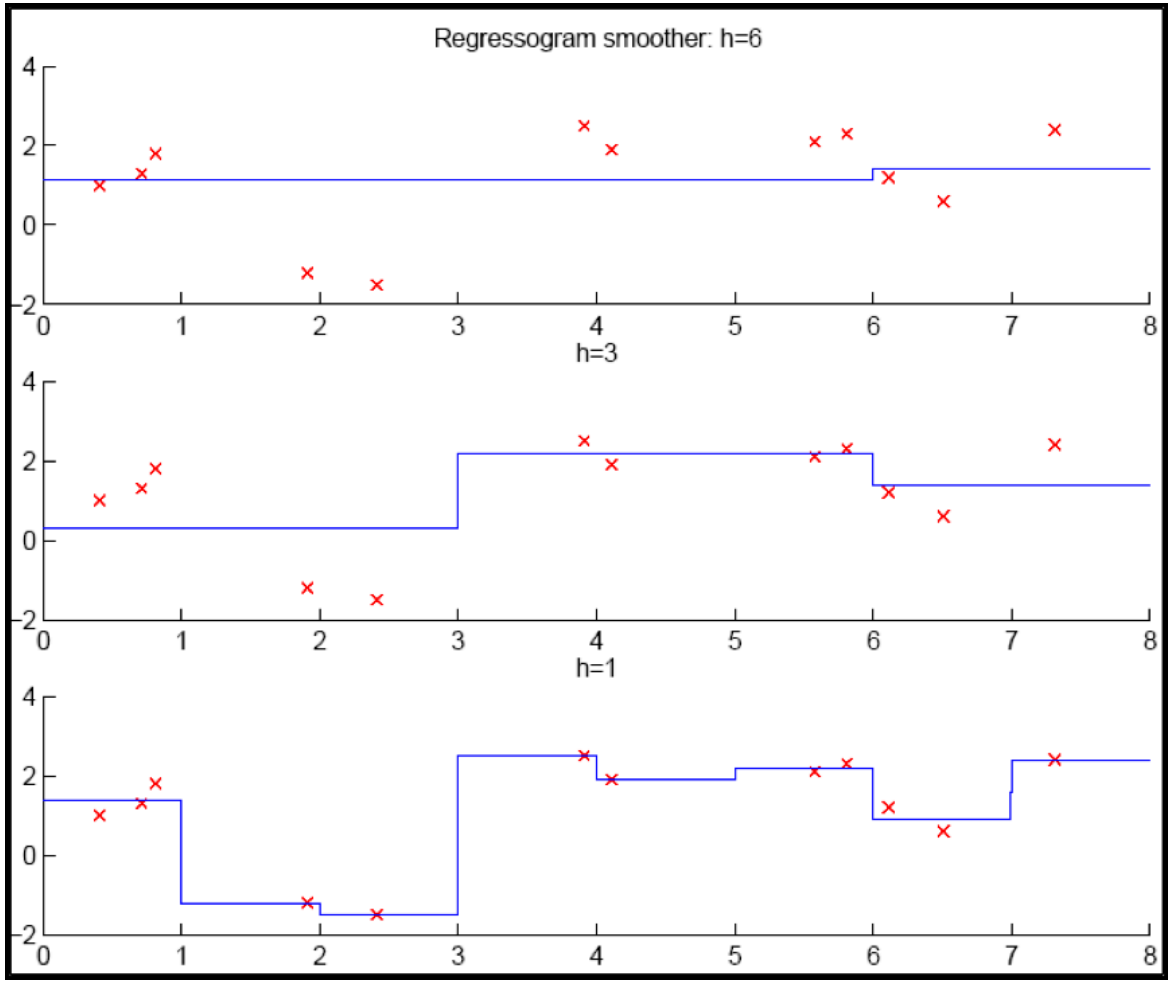
Then estimate the function at a query point

$\mathbf{x}_q$ as

$$\hat{g}(\mathbf{x}_q) = \frac{\sum_{t=1}^{n} b(\mathbf{x}_q, \mathbf{x}^t) r^t}{\sum_{t=1}^{n} b(\mathbf{x}_q, \mathbf{x}^t)}$$

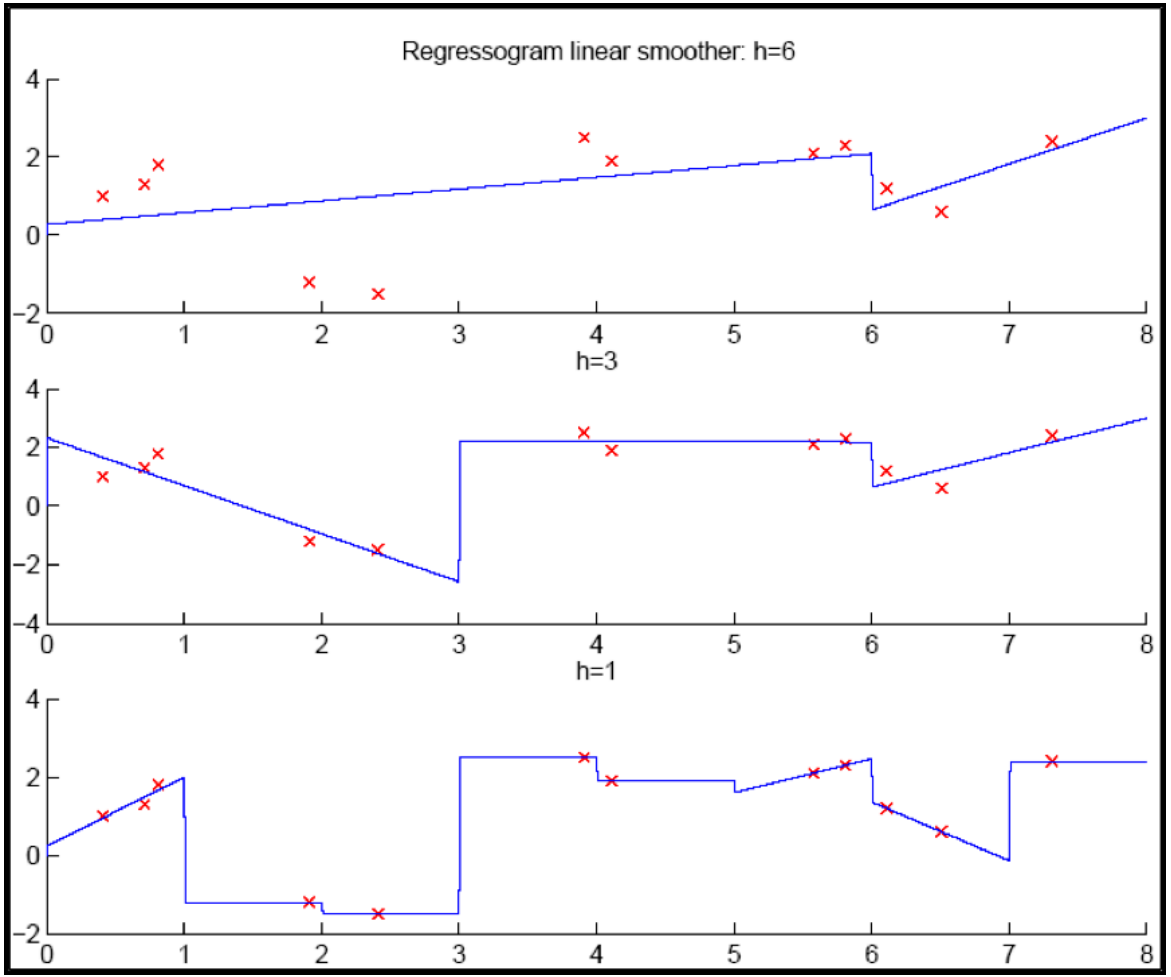
where

$$b(\mathbf{x}_q, \mathbf{x}^t) = \begin{cases} 1 & \text{if } \mathbf{x}^t \in \text{bin}(\mathbf{x}_q) \\ 0 & \text{otherwise} \end{cases}$$



(Alpaydin, 2019)

Regressogram Smoother



(Alpaydin, 2019)

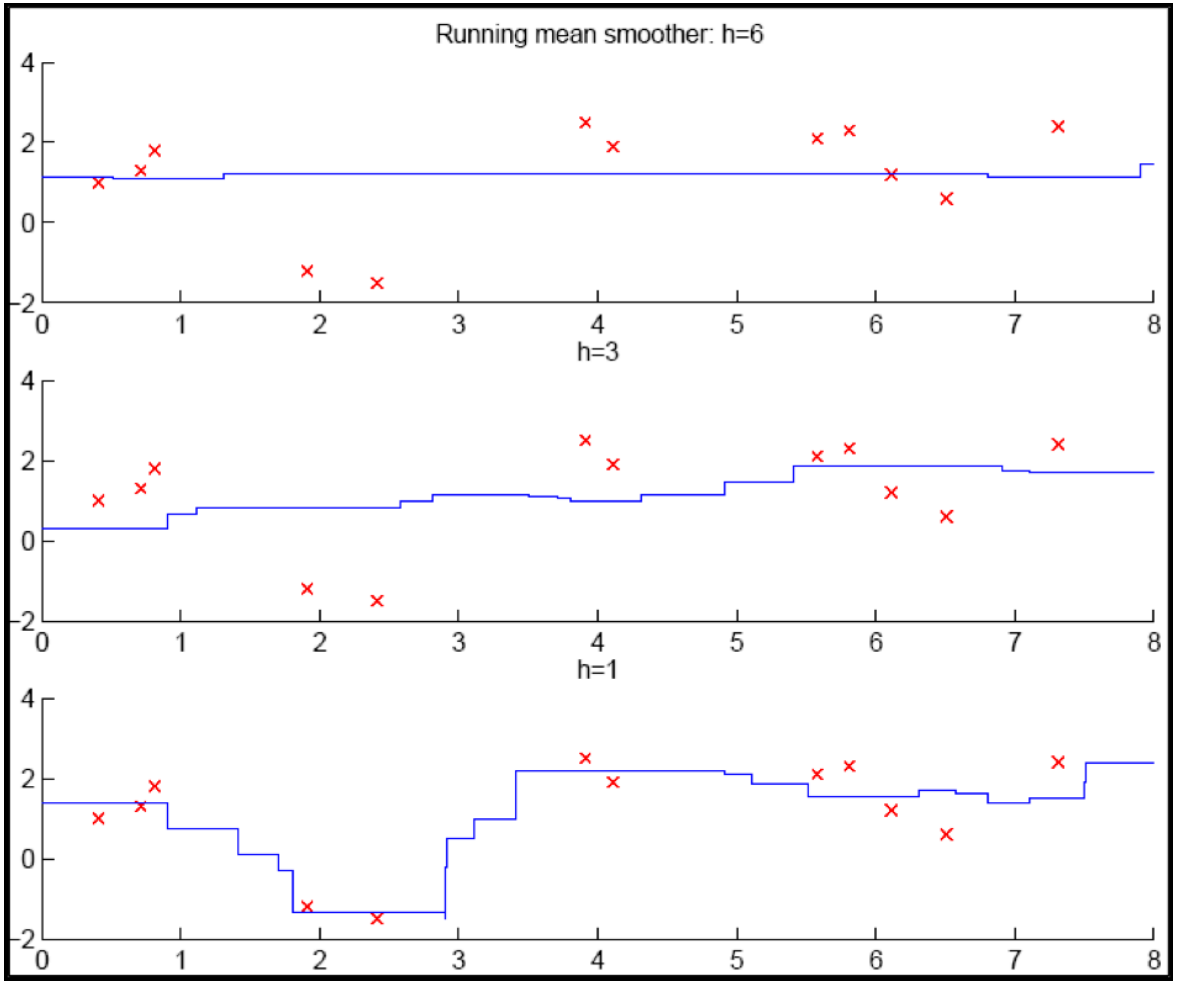
Running Mean Smoother

Estimate the function at a query point $\mathbf{x}_q$ as

$$\hat{g}(\mathbf{x}_q) = \frac{\sum_{t=1}^{n} w\left(\frac{\mathbf{x}_q - \mathbf{x}^t}{h}\right) r^t}{\sum_{t=1}^{n} w\left(\frac{\mathbf{x}_q - \mathbf{x}^t}{h}\right)}$$

where

$$w(\mathbf{u}) = \begin{cases} 1 & \text{if } |\mathbf{u}| < 1 \\ 0 & \text{otherwise} \end{cases}$$



(Alpaydin, 2019)

Kernel Smoother

- Implements a form of locally weighted regression
- Options:
  - Select *k* neighbors of $\mathbf{x}_q$ and fit a linear function
  - Select *k* neighbors of $\mathbf{x}_q$ and fit a polynomial function
  - Select *k* neighbors of $\mathbf{x}_q$ and fit a piece-wise linear function
- A better option is to employ a kernel function and apply a weighted average

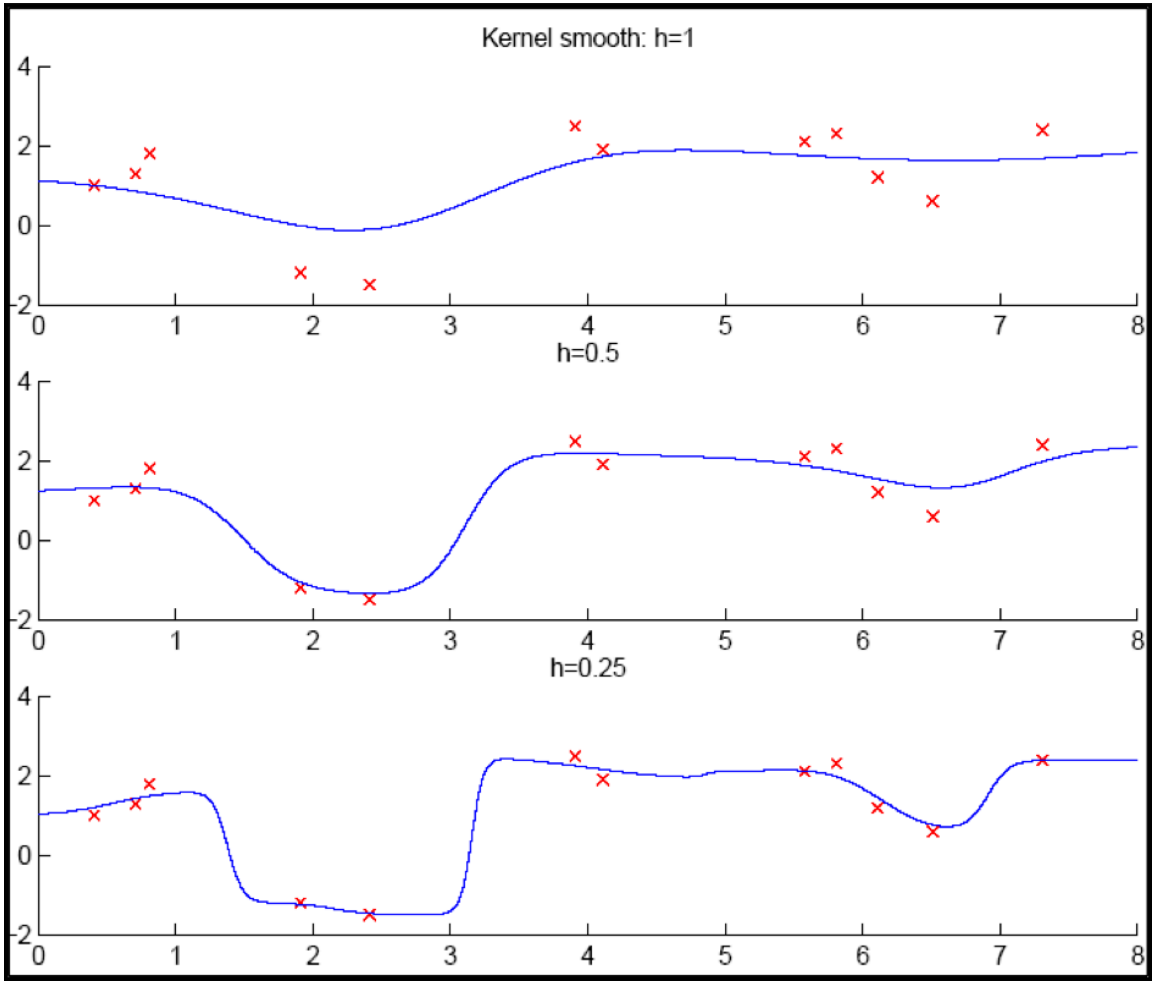
Kernel Smoother

Estimate the function at a query point $\mathbf{x}_q$ as

$$\hat{g}\left(\mathbf{x}_q\right)=\frac{\sum_{t=1}^{n} K\left(\mathbf{x}_q, \mathbf{x}^t\right) r^t}{\sum_{t=1}^{n} K\left(\mathbf{x}_q, \mathbf{x}^t\right)}$$

where

$$K\left(\mathbf{x}_q, \mathbf{x}^t\right)=\exp\left[\frac{1}{2\sigma} D\left(\mathbf{x}_q, \mathbf{x}^t\right)\right]$$



(Alpaydin, 2019)