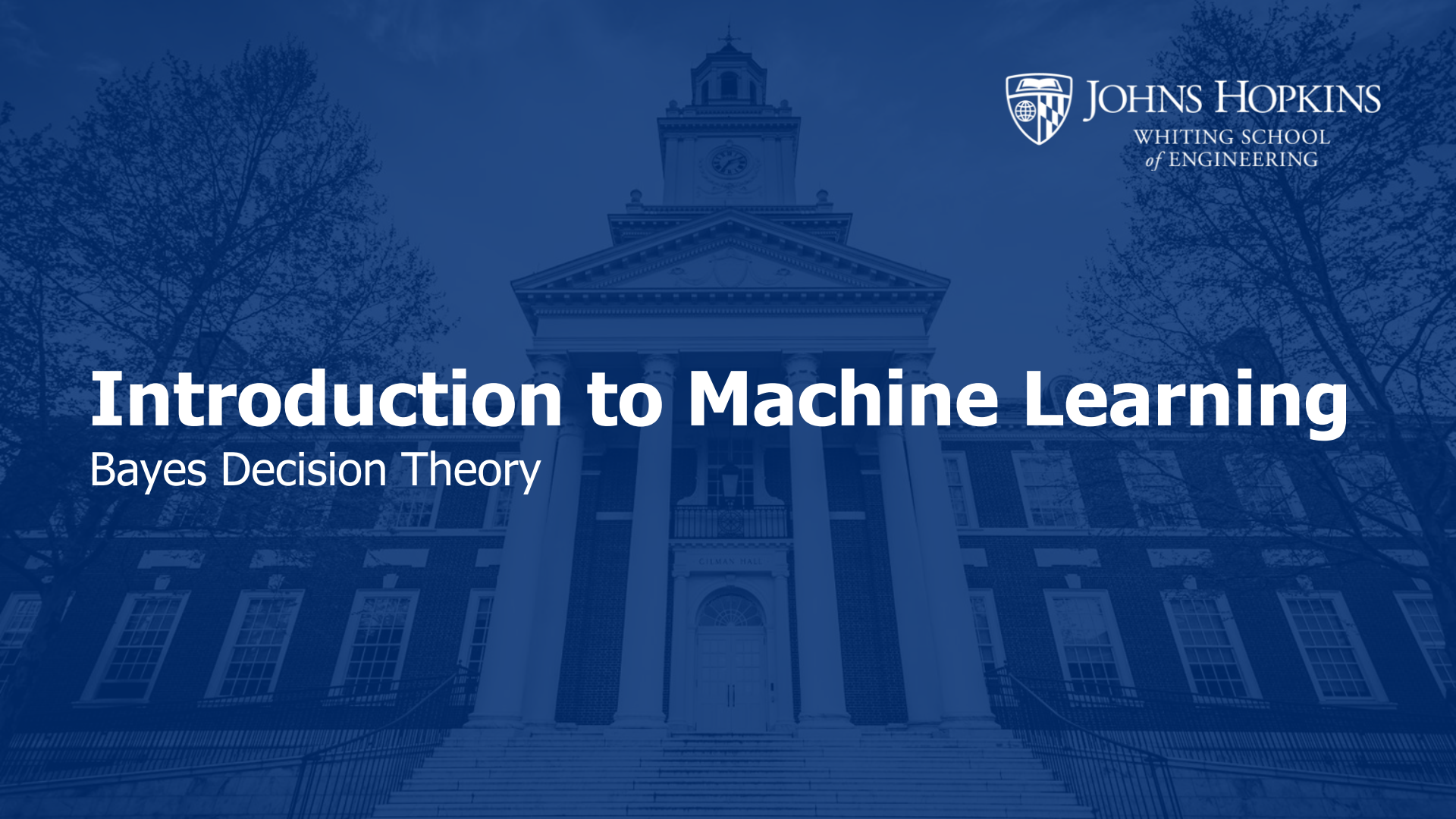


Introduction to Machine Learning

Bayes Decision Theory

Making Decisions

- Much of machine learning is about making decisions.
- Suppose we are trying to decide between two options:
  - Option 1: A_1
  - Option 2: A_2
- This means we have two choices we make make:
  - Choice 1: α_1
  - Choice 2: α_2
- The goal is to use evidence $\mathbf{e}$ to make that decision.

Risk Minimization

- We consider the "cost" of a decision, denoted $\lambda(\alpha_i|A_j)$, where α_i is our choice, and the "correct" choice is A_j.
- Our goal is to minimize expected loss.
- We compute expected loss using the "risk" functional

$$R(\alpha_i|\mathbf{e}) = \sum_{j=1}^{c} \lambda(\alpha_i|A_j)P(A_j|\mathbf{e})\,.$$

- This is called a "risk" functional because it denotes the risk of making choice α_i given the evidence $\mathbf{e}$.

Computing Risk

- We begin by focusing on $P(A_j|\mathbf{e})$ and use Bayes Rule.

$$P(A_j|\mathbf{e}) = \frac{P(\mathbf{e}|A_j)P(A_j)}{P(\mathbf{e})}.$$

- Note that $P(e) = \sum_k P(\mathbf{e}|A_k)P(A_k)$.
- We seek $\alpha(e) = \text{argmin}_{\alpha_i} R(\alpha_i|\mathbf{e})$.
- We denote the loss associated with choice α_i given the better choice is A_j as λ_{ij}.
- Then, assuming two choices (as an example),

$$R(\alpha_1|\mathbf{e}) = \lambda_{11}P(A_1|\mathbf{e}) + \lambda_{12}P(A_2|\mathbf{e})$$
$$R(\alpha_2|\mathbf{e}) = \lambda_{21}P(A_1|\mathbf{e}) + \lambda_{22}P(A_2|\mathbf{e})$$

- We pick A_1 over A_2 if $R(\alpha_1|\mathbf{e}) < R(\alpha_2|\mathbf{e})$.

Computing Risk

- Via some substitutions

$$\lambda_{11}P(A_1|\mathbf{e}) + \lambda_{12}P(A_2|\mathbf{e}) < \lambda_{21}P(A_1|\mathbf{e}) + \lambda_{22}P(A_2|\mathbf{e}).$$

- Rearranging terms

$$(\lambda_{11} - \lambda_{21})P(A_1|\mathbf{e}) < (\lambda_{22} - \lambda_{12})P(A_2|\mathbf{e}).$$

- Let's apply Bayes Rule again.

$$(\lambda_{11} - \lambda_{21})\frac{P(\mathbf{e}|A_1)P(A_1)}{P(\mathbf{e})} < (\lambda_{22} - \lambda_{12})\frac{P(\mathbf{e}|A_2)P(A_2)}{P(\mathbf{e})}.$$

- Which is equivalent to

$$(\lambda_{11} - \lambda_{21})P(\mathbf{e}|A_1)P(A_1) < (\lambda_{22} - \lambda_{12})P(\mathbf{e}|A_2)P(A_2).$$

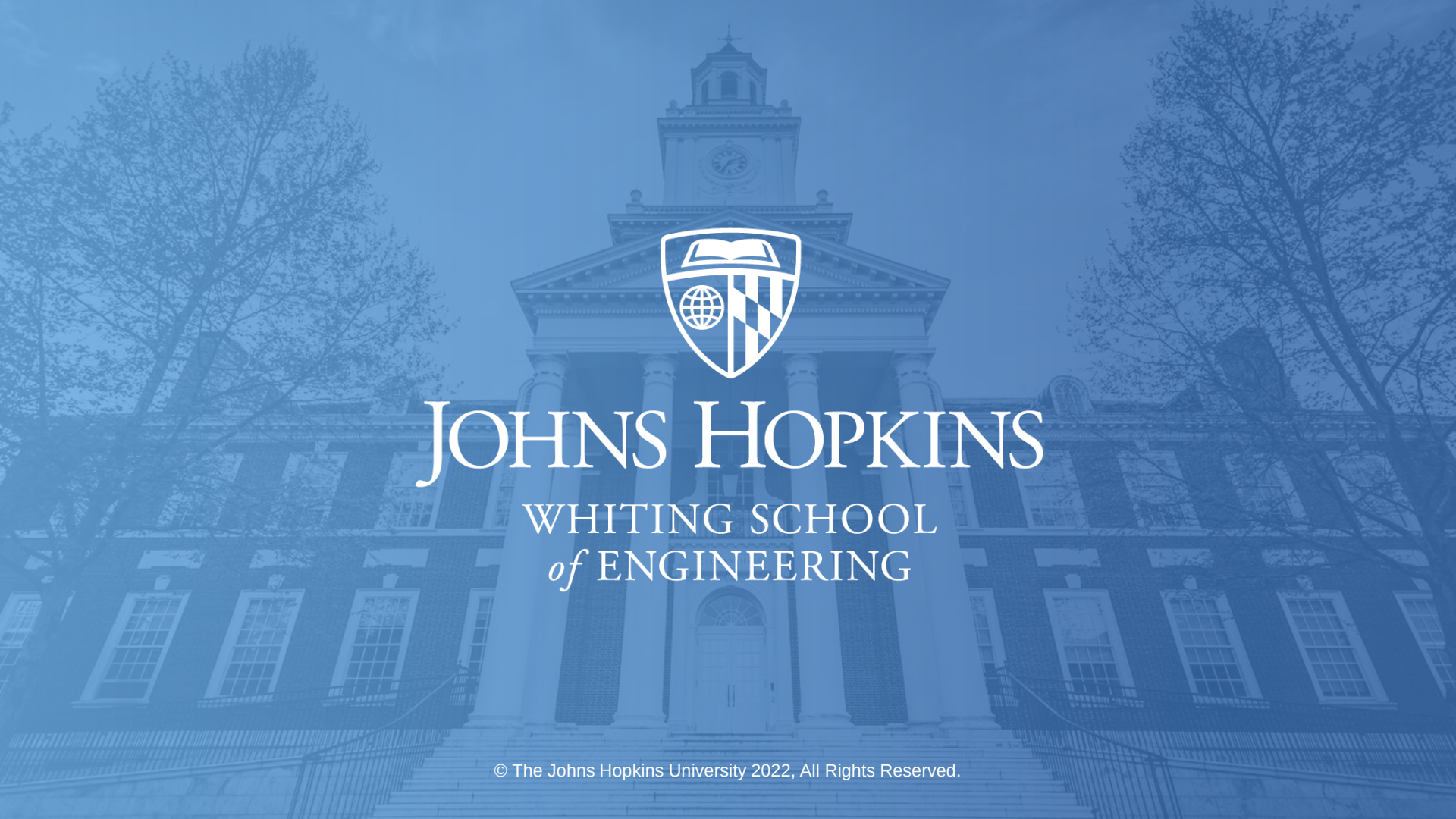
0/1 Loss

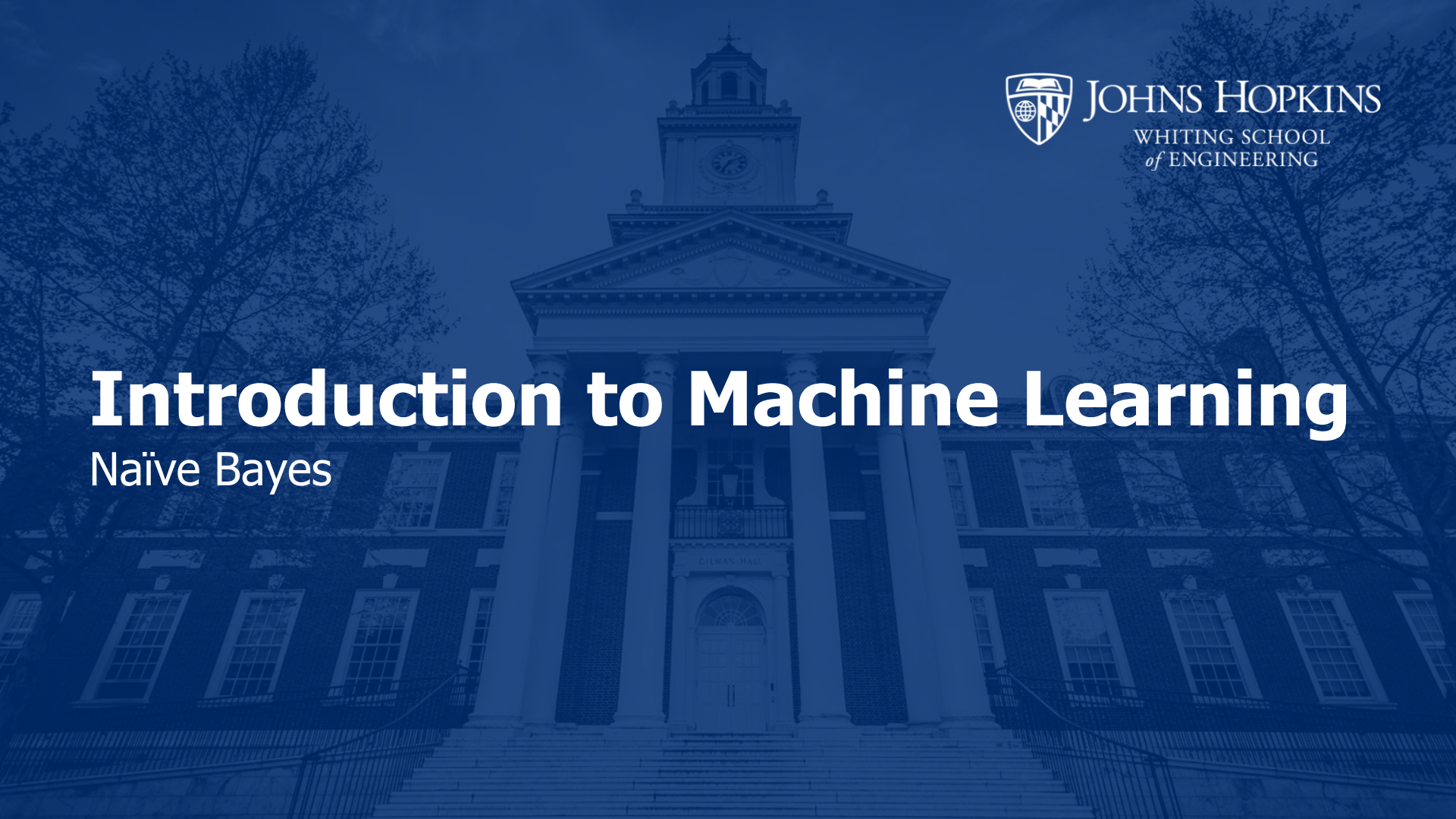
- We apply 0/1 loss, which is among the most basic loss functions. Then
  - $\lambda_{11} = \lambda_{22} = 0$
  - $\lambda_{12} = \lambda_{21} = 1$
- Thus,

$$-P(\mathbf{e}|A_1)P(A_1) < -P(\mathbf{e}|A_2)P(A_2).$$

- From this, we choose A_1 over A_2 if

$$P(\mathbf{e}|A_1)P(A_1) > P(\mathbf{e}|A_2)P(A_2)$$
$$P(A_1|\mathbf{e}) > P(A_2|\mathbf{e}).$$





Introduction to Machine Learning

Naïve Bayes

Bayesian Classification

- In classification, we want to maximize $P(f_h|\mathcal{D})$.
- Using the results from Bayesian decision theory, this means we want

$$class = \text{argmax}_{c \in \mathcal{C}} P(c|f_1, \ldots, f_d).$$

- As we did with Bayes decision theory, we will apply Bayes rule.

$$P(c|f_1, \ldots f_d) = \frac{P(f_1, \ldots f_d|c)P(c)}{P(f_1, \ldots, f_d)}.$$

- This looks like it's expensive!

Complexity

- Assume the random variables $c, f_1, \ldots, f_d$ are all Boolean.
- There are a total of $d + 1$ random variables.
- Consider the distribution, $P(f_1, \ldots, f_d | c)$.
- Fully parameterizing will require 2^{d+1} probabilities.
- Problems:
  - Storage
  - Computational time
  - Excessively large number of parameters to estimate

Conditional Independence

- **Def:** Two random variables X, Y are conditionally independent given a third random variable Z, denoted $(X \perp Y|Z)$, if any of the following hold:

$$P(X, Y|Z) = P(X|Z)P(Y|Z)$$
$$P(X|Y, Z) = P(X|Z)$$
$$P(Y|X, Z) = P(Y|Z)$$

- All of these are equivalent.
- For our classifier, assume $\forall i, j, i \neq j, (f_i \perp f_j|c)$. Then

$$P(c|f_1, \dots f_d) = \frac{P(f_1|c) \times \cdots \times P(f_d|c)P(c)}{P(f_1, \dots, f_d)}$$

$$class = \text{argmax}_{c \in \mathcal{C}}\, P(c) \prod_{i=1}^{d} P(f_i|c)\,.$$

Example

Class	F1	F2	F3	F4	F5
0	1	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	1
0	1	0	0	1	0
0	0	1	1	0	1
1	0	1	0	1	0
1	1	0	0	0	0
1	0	1	0	0	0
1	1	0	1	0	1
1	0	1	0	1	0

Estimate $P(c)$:

$$P(c = 0) = \frac{5}{10} = 0.5$$

$$P(c = 1) = \frac{5}{10} = 0.5$$

Estimate $P(F1 = 0|c = 0)$:

$$P(F1 = 0|c = 0) = \frac{3}{5} = 0.6$$

Example

$P(C=0) = 0.5$	$P(F1=0 \mid C=0) = 0.6$	$P(F2=0 \mid C=0) = 0.4$	$P(F3=0 \mid C=0) = 0.2$	$P(F4=0 \mid C=0) = 0.4$	$P(F5=0 \mid C=0) = 0.6$
	$P(F1=1 \mid C=0) = 0.4$	$P(F2=1 \mid C=0) = 0.6$	$P(F3=1 \mid C=0) = 0.8$	$P(F4=1 \mid C=0) = 0.6$	$P(F5=1 \mid C=0) = 0.4$
$P(C=1) = 0.5$	$P(F1=0 \mid C=1) = 0.6$	$P(F2=0 \mid C=1) = 0.4$	$P(F3=0 \mid C=1) = 0.8$	$P(F4=0 \mid C=1) = 0.6$	$P(F5=0 \mid C=1) = 0.8$
	$P(F1=1 \mid C=1) = 0.4$	$P(F2=1 \mid C=1) = 0.6$	$P(F3=1 \mid C=1) = 0.2$	$P(F4=1 \mid C=1) = 0.4$	$P(F5=1 \mid C=1) = 0.2$

Classify [1 1 0 1 0]

$$C = 0: 0.5 \times 0.4 \times 0.6 \times 0.2 \times 0.6 \times 0.6 = 0.00864$$

$$C = 1: 0.5 \times 0.4 \times 0.6 \times 0.8 \times 0.4 \times 0.8 = \mathbf{0.03072}$$

Decide $C = 1$.

Subtleties

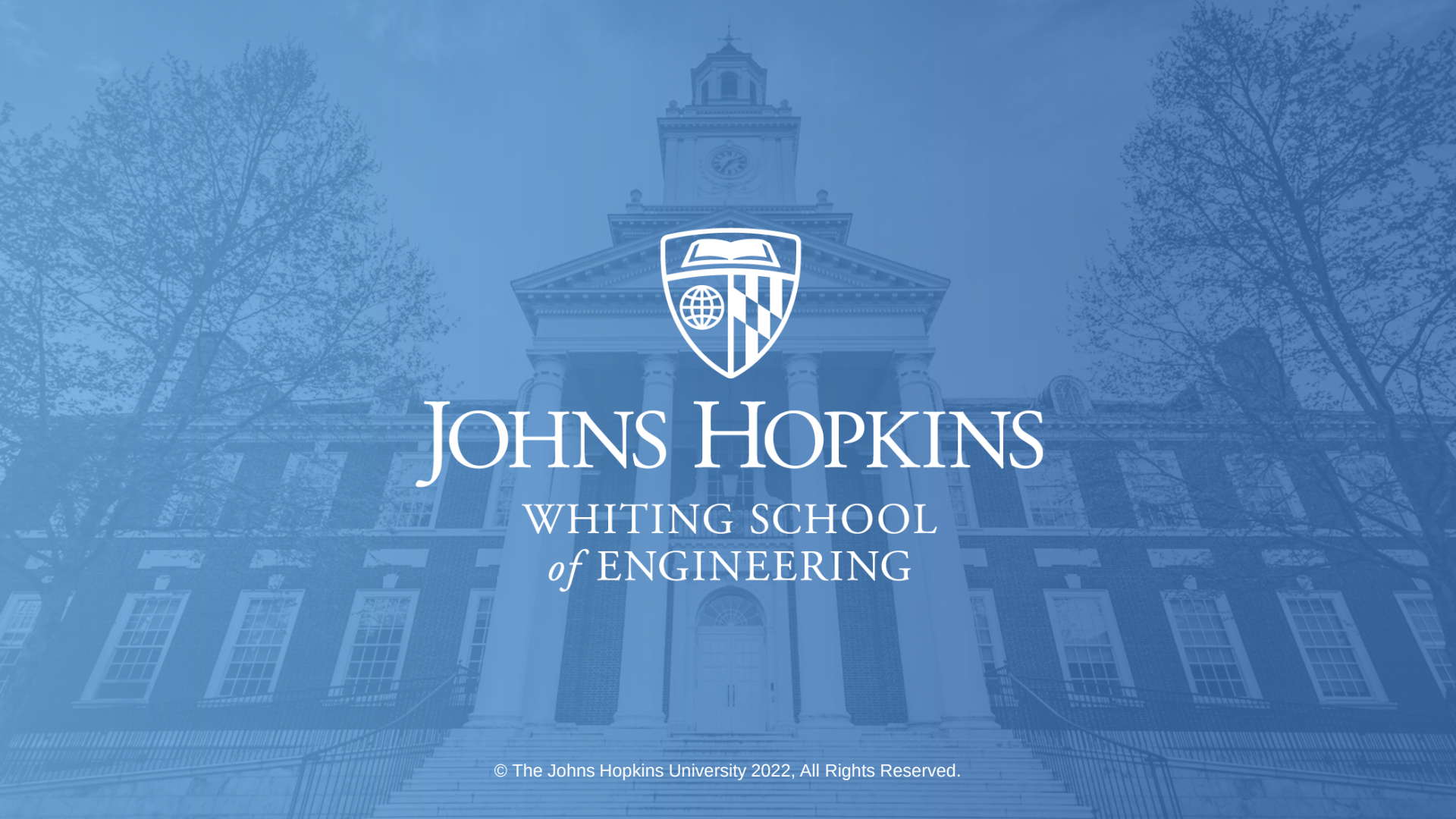
- Naïve Bayes can only solve *linearly separable* problems exactly.
- A problem arises if any of the conditional probabilities equals zero. ⟹ Smooth

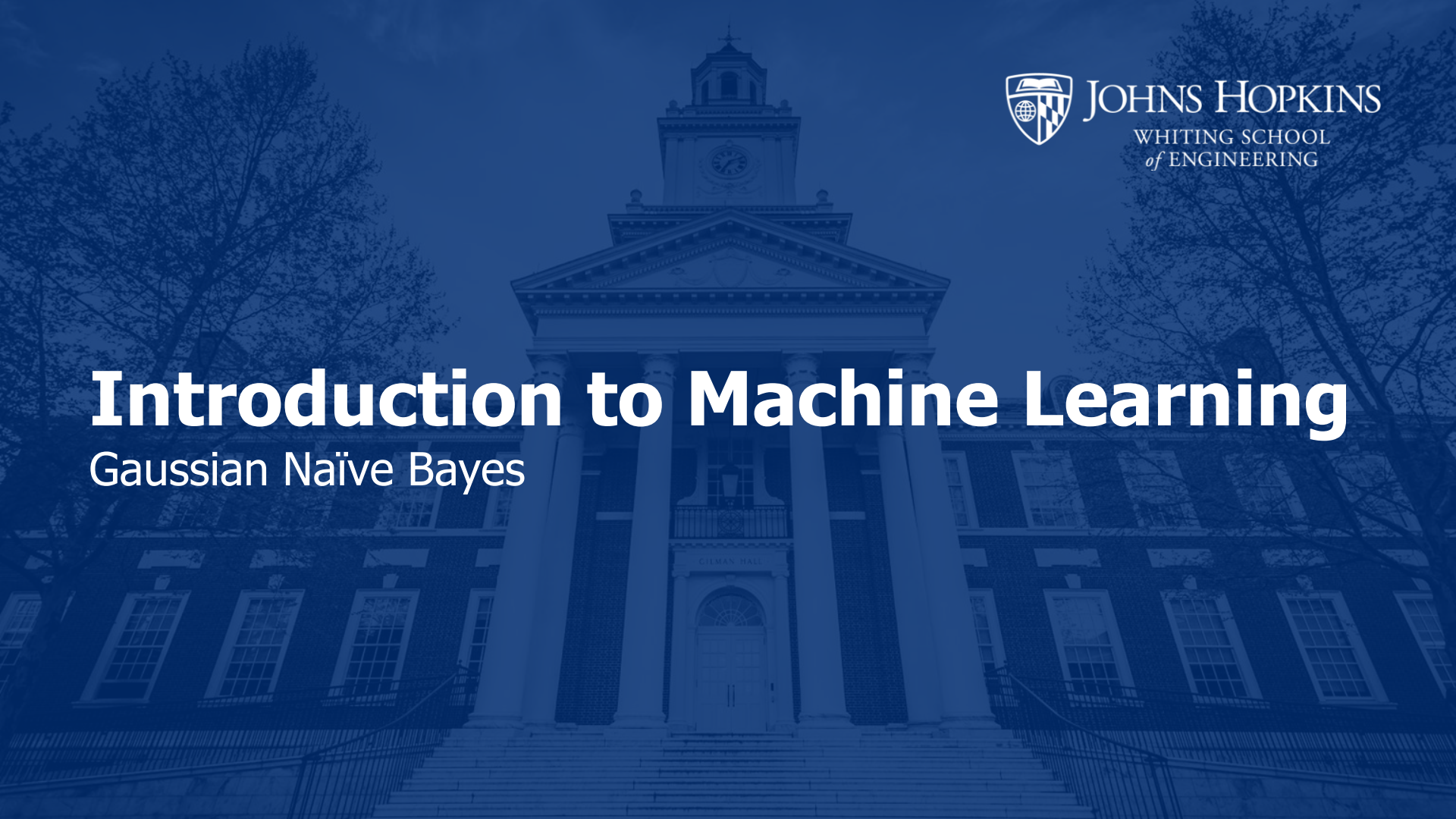
- m-estimate:

$$P(f|c) \approx \frac{n_{f,c} + mp}{n_c + m}$$

- Example: Suppose $m = 1$ and $p = 0.001$.

$$\frac{0 + 0.001}{n_c + 1} \neq 0$$





Introduction to Machine Learning

Gaussian Naïve Bayes

Fundamental of Gaussian Naïve Bayes

- Focused on working with continuous (real-valued) features.
- Still assumes discrete classes (i.e., not used for regression).
- Prior class probability:

$$P(class)$$

- Variable distribution is Gaussian:

$$X \sim \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- Assume d random variables: $X_1, \ldots, X_d$ and class variable Y.
- Our goal is to estimate $\mu_{ik} = E[X_i|Y = y_k]$ and $\sigma_{ik}^2 = E[(X_i - \mu_{ik})^2|Y = y_k]$

Maximum Likelihood Estimation

- MLE estimate for the mean:

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j X_i^j \delta(Y^j = y_k)$$

where

$$\left(Y^j = y_k\right) = \begin{cases} 1 & Y^j = y_k \\ 0 & \text{otherwise} \end{cases}$$

- MLE estimate for the variance:

$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j \left(X_i^j - \hat{\mu}_{ik}\right)^2 \delta(Y^j = y_k)$$

unbiased:

$$\hat{\sigma}_{ik}^2 = \frac{1}{\left(\sum_j \delta(Y^j = y_k)\right) - 1} \sum_j \left(X_i^j - \hat{\mu}_{ik}\right)^2 \delta(Y^j = y_k)$$

GNB Rule

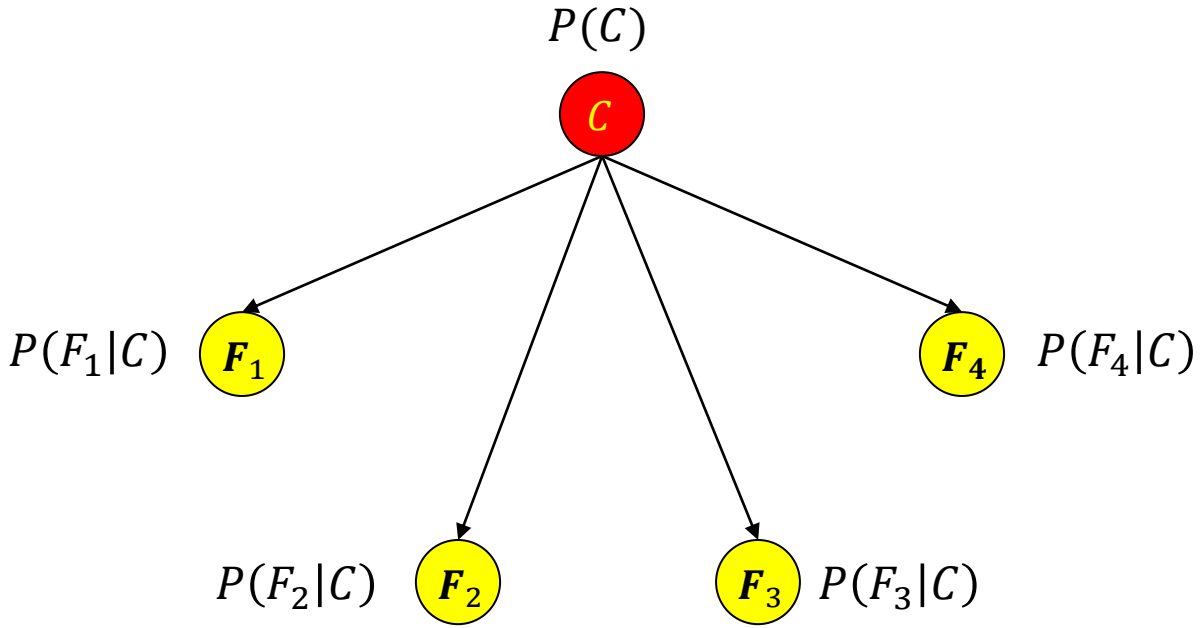
- The Gaussian Naïve Bayes rule then becomes

$$class = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i=1}^{d} P(f_i|c)$$

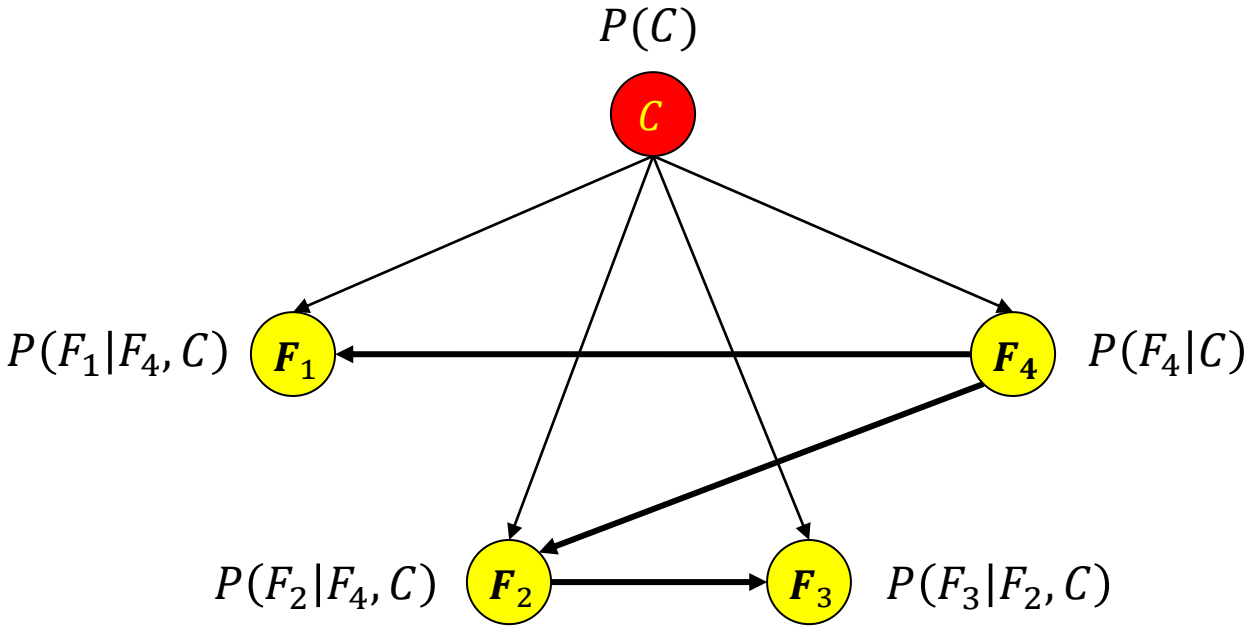
- Substituting in the Gaussians gives us

$$class = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i=1}^{d} \mathcal{N}(\hat{\mu}_{ic}, \hat{\sigma}_{ic}) .$$

Tree Augmented Naïve Bayes



Tree Augmented Naïve Bayes



Classification Rule

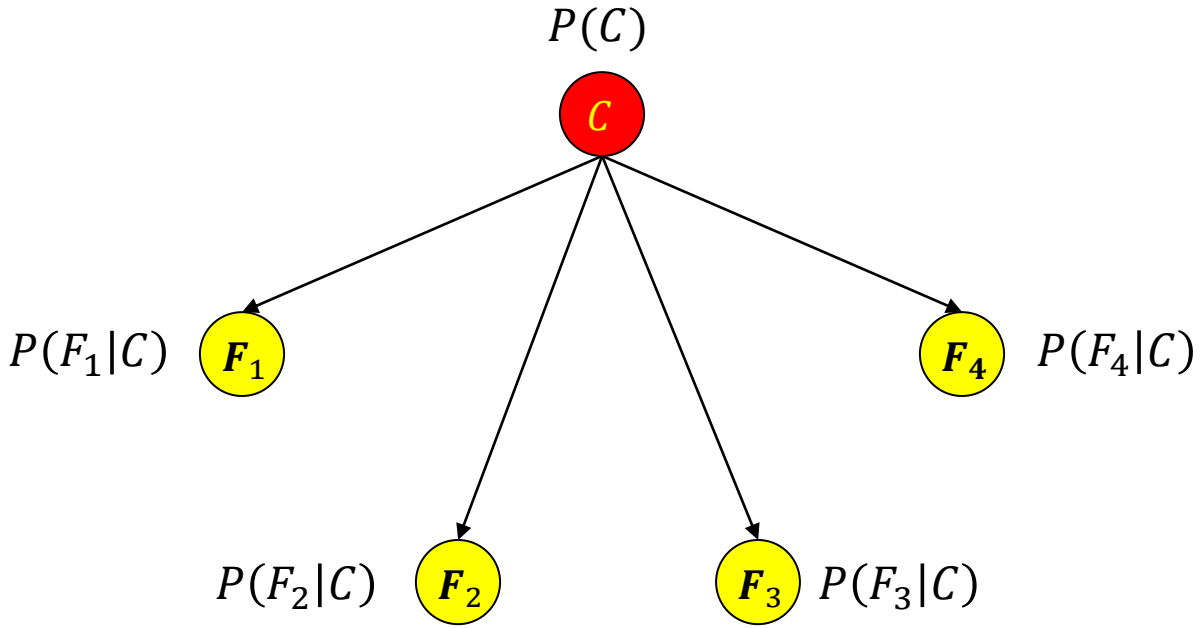
- To classify using the augmented structure.

$$class = \underset{c \in C}{\operatorname{argmax}} P(c) P(f_{\text{root}} | c) \prod_{(f_i, f_j) \in TAN} P(f_i | f_j, c)$$

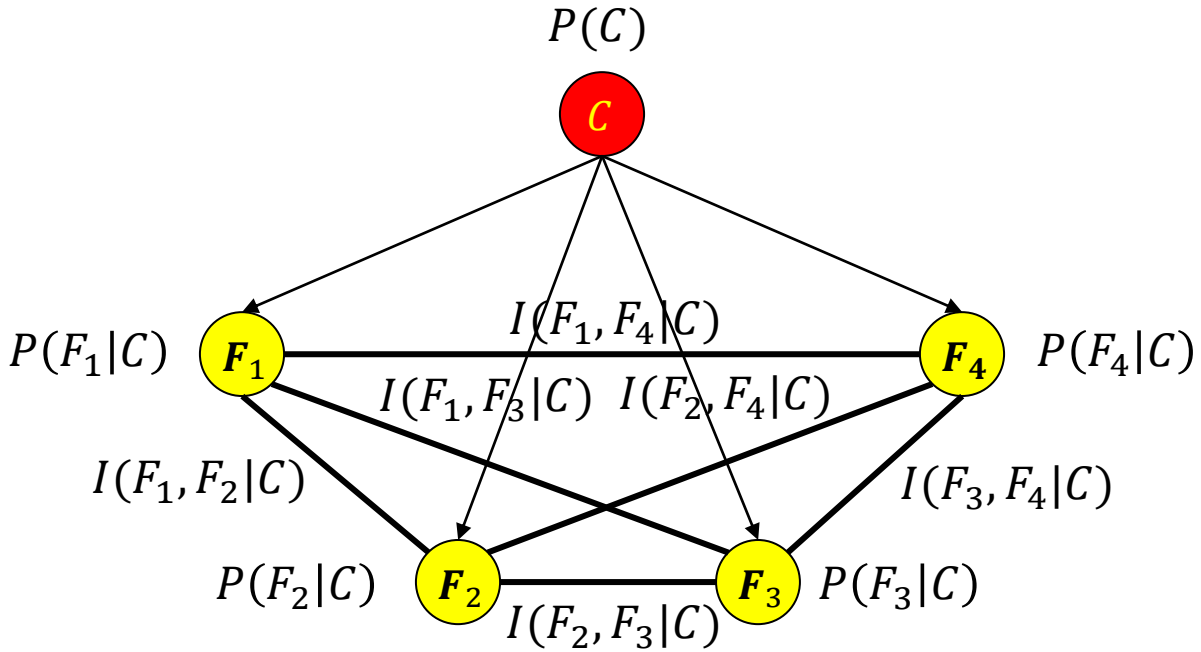
- Use conditional mutual information.

$$(F_i, F_j | C) = \sum_{F_i, F_j, C} P(F_i, F_j, C) \log \frac{P(F_i, F_j | C)}{P(F_i | C) P(F_j | C)}$$

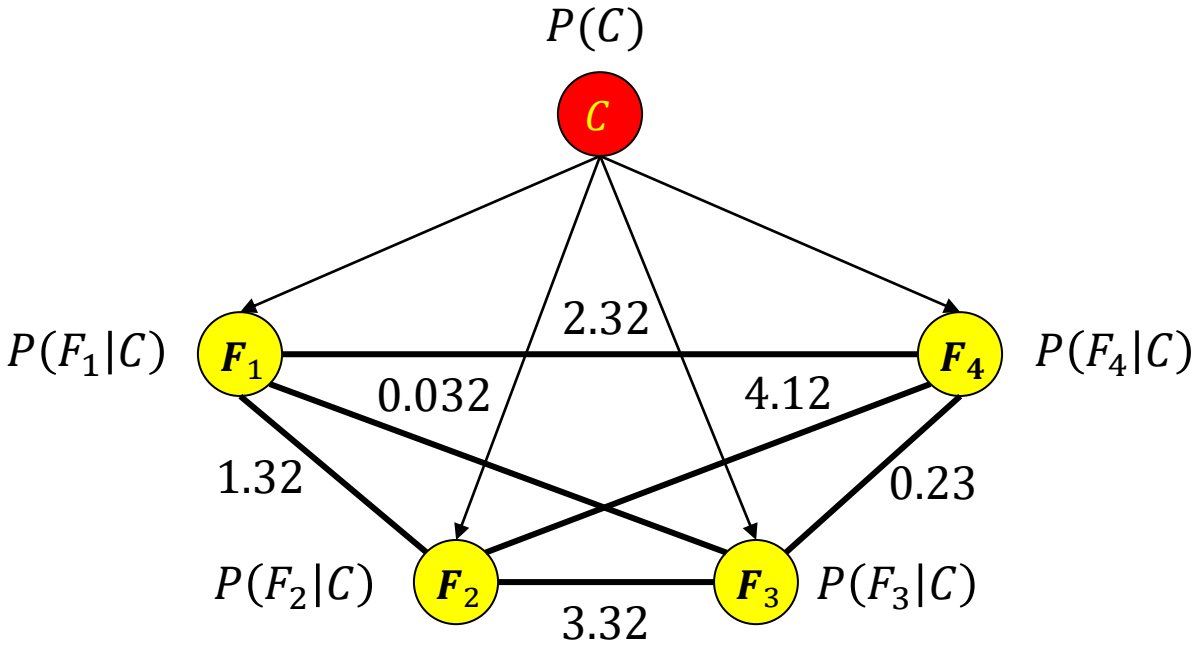
Building a TAN



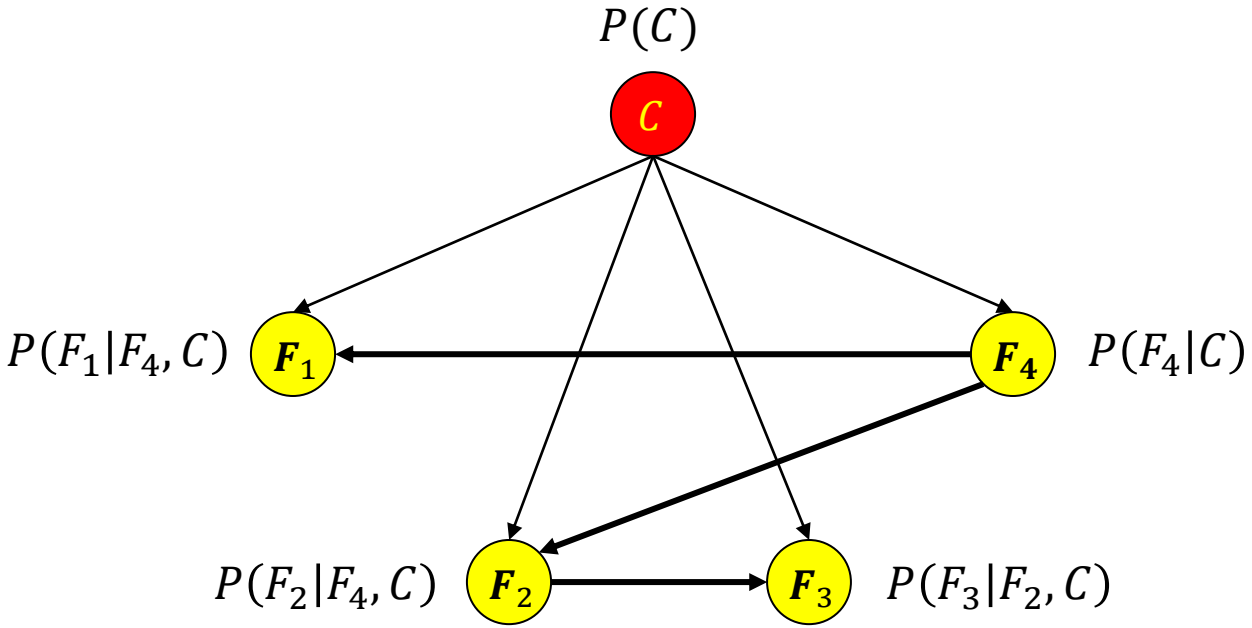
Building a TAN

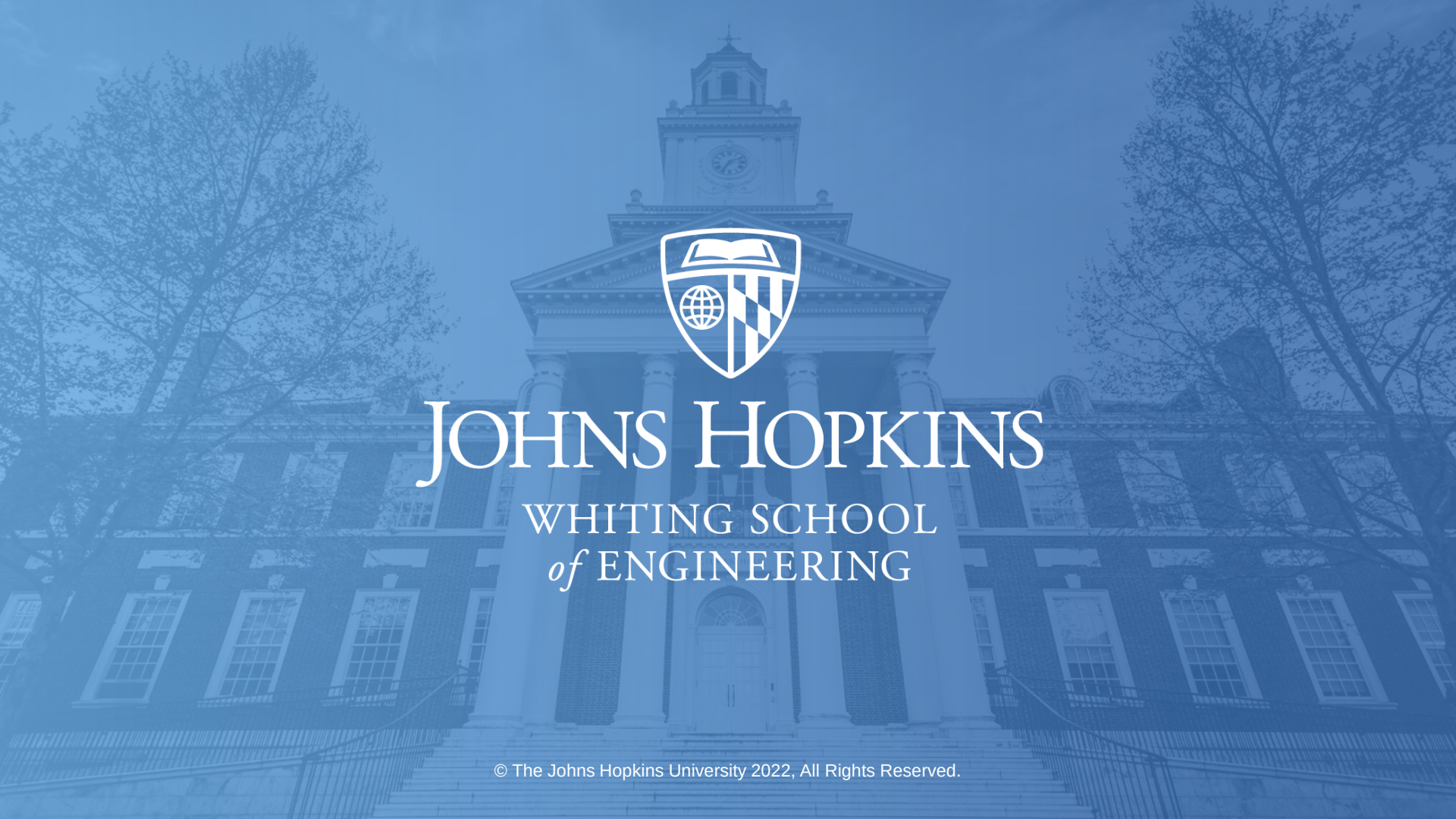


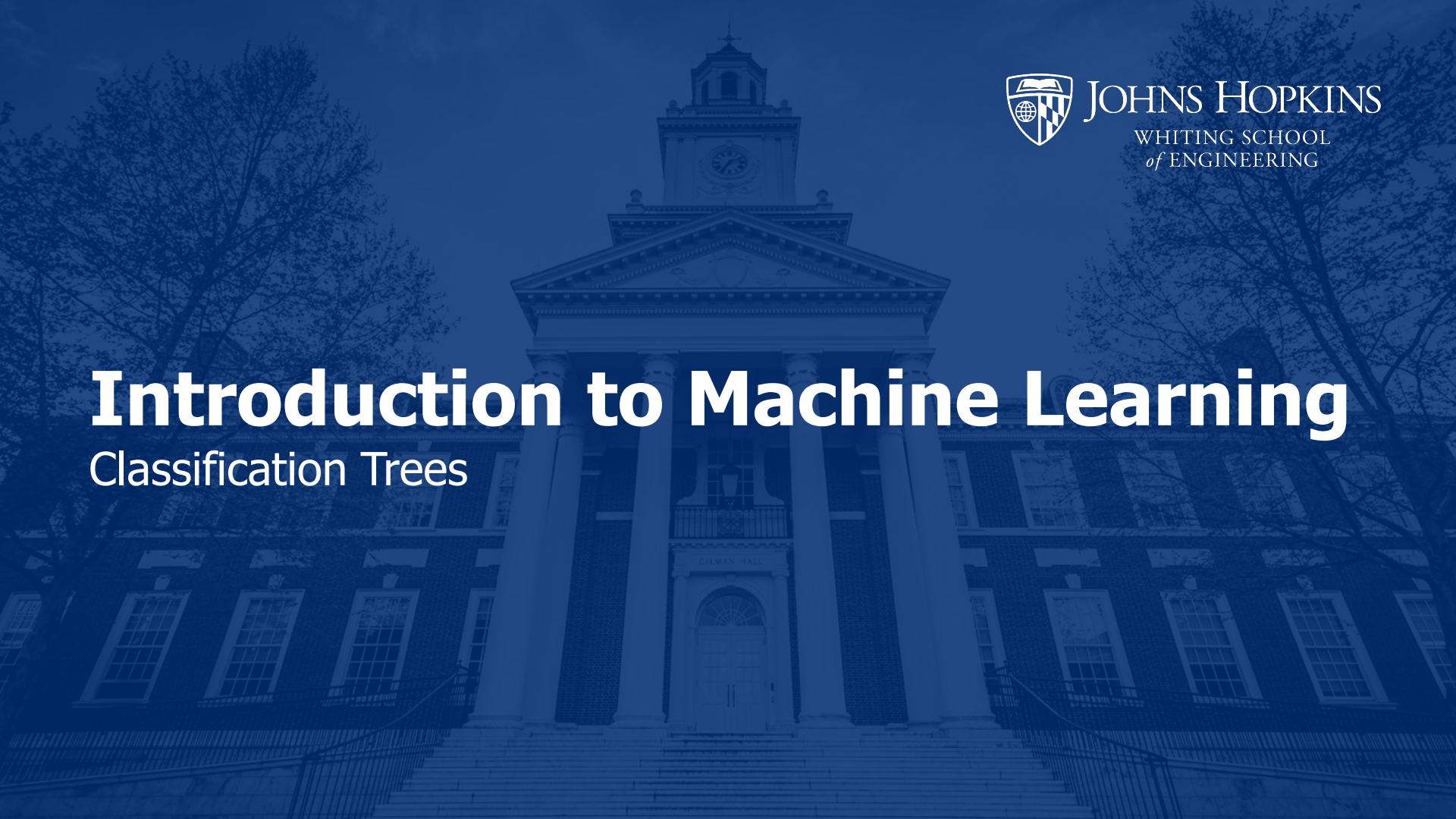
Building a TAN



Building a TAN







Introduction to Machine Learning

Classification Trees

Decision Tree Learning

- Rule Induction
- Explanation Based Learning
  - Speed up classification
  - Domain theory → Explains examples
- Sequential Covering

- Iterative Dichotomizer 3 (ID3) – J. R. Quinlan
- Classification and Regression Tree (CART) – L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone

Classification Tree

- A tree structure where
  - Interior nodes are decision variables (need not be binary)
  - Leaf nodes assign class labels
- Assume
  - Data set $\mathcal{D}$
  - Feature set $\mathcal{F}$, where $f_i \in \mathcal{F} \rightarrow \text{domain } d(f_i)$
- When we choose a feature $f_i \in \mathcal{F}$ we partition $\mathcal{D}$ into $|d(f_i)|$ subsets (again, need not just be two)

Example

- Data characterizing the weather
- Features: Outlook, Temperature, Humidity, Wind
- Domains:

$$d(\text{Outlook}) = \{\text{sunny}, \text{overcast}, \text{rainy}\}$$
$$d(\text{Temperature}) = \{\text{cold}, \text{mild}, \text{hot}\}$$
$$d(\text{Humidity}) = \{\text{high}, \text{normal}\}$$
$$d(\text{Wind}) = \{\text{true}, \text{false}\}$$

- Assume binary (two-class) classification. Note, this need not be the case.

Building the Tree - Complexity

- Begin by considering the complexity of learning a decision tree
  - Sample complexity
  - Algorithmic complexity
- How do we build the tree?
- Could choose features at random
  - Not efficient
  - Could lead to overfitting
- Goal: Build a tree that uses the fewest attributes possible

The Decision Tree Problem

- Let $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a finite set of training examples
- Let $\mathcal{F} = \{f_1, \ldots, f_k\}$ be a finite set of features
- Let *w* be the cost of some tree $\mathcal{T}$ = expected number of features required to classify example $\mathbf{x}_i$

$DT(\mathcal{D}, \mathcal{F}, w)$: Given example set $\mathcal{D}$ and feature set $\mathcal{F}$,

does there exist a decision tree using only features from $\mathcal{F}$ for classifying examples in $\mathcal{D}$ such that the cost of the decision tree is less than *w*?

Theorem: $DT(\mathcal{D}, \mathcal{F}, w)$ is NP-complete (Hyafil & Rivest, 1976).

Proof Sketch

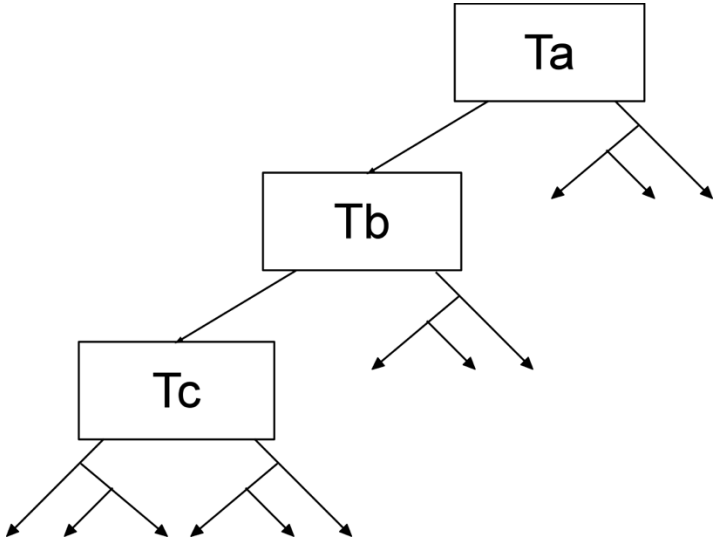
First, we show that $DT(\mathcal{D}, \mathcal{F}, w)$ is in $\mathcal{NP}$

- Compute the length of each branch of the tree
- Multiply each branch/path length by the fraction of training examples at the leaves
- Add up all the weighted branch lengths
- Complexity: Linear in the size of the tree via depth-first search
  (technically, $O(n|\mathcal{F}|)$ if each leaf has one example,
  and each branch uses all $|\mathcal{F}|$ features; still polynomial)

Proof Sketch

Second, we need a polynomial-time reduction from another $\mathcal{NP}$-hard problem

- Use the Exact Cover 3 problem
- Given a set of $3q$ elements and a collection of subsets, finding a grouping of these elements such that each element belongs to exactly one of these subsets and each of these subsets has exactly three elements.



The ID3 Algorithm

- Employ greedy search
- Start by assuming only two classes (will generalize next)
- Based on Shannon's Information Theory (reduce entropy)
  - Let $I(p,n)$ be the entropy (amount of uncertainty) in the data

$$I(p,n) = -\frac{p}{p+n}\lg\frac{p}{p+n} - \frac{n}{p+n}\lg\frac{n}{p+n}$$

- Consider $f_i \in \mathcal{F}$ generating a partition $\mathcal{D}_\pi \subseteq \mathcal{D}$

$$E_\pi(f_i) = \sum_{j=1}^{m_i} \frac{p_\pi^j + n_\pi^j}{p_\pi + n_\pi} I\left(p_\pi^j, n_\pi^j\right)$$

  where $m_i = |d(f_i)|, p_\pi, n_\pi \sim \mathcal{D}_\pi$

- Information gain: $\text{gain}_\pi(f_i) = I(p_\pi, n_\pi) - E_\pi(f_i)$

More than Two Classes

- Need to consider the entropy of a class structure with $k > 2$ classes

$$I(c_1, \ldots, c_k) = -\sum_{\ell=1}^{k} \frac{c_\ell}{c_1 + \cdots + c_k} \lg \frac{c_\ell}{c_1 + \cdots + c_k}$$

- Also need expected entropy

$$E_\pi(f_i) = \sum_{j=1}^{m_i} \frac{c_{\pi,1}^j + \cdots + c_{\pi,k}^k}{c_{\pi,1} + \cdots + c_{\pi,k}} I\left(c_{\pi,1}^j + \cdots + c_{\pi,k}^k\right)$$

Example

- Recall:

$$d(\text{Outlook}) = \{\text{sunny}, \text{overcast}, \text{rainy}\}$$
$$d(\text{Temperature}) = \{\text{cold}, \text{mild}, \text{hot}\}$$
$$d(\text{Humidity}) = \{\text{high}, \text{normal}\}$$
$$d(\text{Wind}) = \{\text{true}, \text{false}\}$$

- Let **P** denote the positive class
- Let **N** denote the negative class

$$I(p,n) = -\frac{9}{14}\lg\frac{9}{14} - \frac{5}{14}\lg\frac{5}{14} \approx 0.94$$

Example	Outlook	Temperature	Humidity	Wind	Class
1	Sunny	Hot	High	False	N
2	Sunny	Hot	High	True	N
3	Overcast	Hot	High	False	P
4	Rainy	Mild	High	False	P
5	Rainy	Cool	Normal	False	P
6	Rainy	Cool	Normal	True	N
7	Overcast	Cool	Normal	True	P
8	Sunny	Mild	High	False	N
9	Sunny	Cool	Normal	False	P
10	Rainy	Mild	Normal	False	P
11	Sunny	Mild	Normal	True	P
12	Overcast	Mild	High	True	P
13	Overcast	Hot	Normal	False	P
14	Rainy	Mild	High	True	N

Example

- Consider the feature Outlook

$$E(\text{ Outlook }) = \frac{5}{14} I_{\text{sunny}}(p,n) + \frac{4}{14} I_{\text{overcast}}(p,n) + \frac{5}{14} I_{\text{rainy}}(p,n)$$

- Consider the three partitions

$$I_{\text{sunny}}(p,n) = -\frac{2}{5}\lg\frac{2}{5} - \frac{3}{5}\lg\frac{3}{5} \approx 0.971$$
$$I_{\text{overcast}}(p,n) = -\frac{4}{4}\lg\frac{4}{4} - \frac{0}{4}\lg\frac{0}{4} \approx 0.0$$
$$I_{\text{rainy}}(p,n) = -\frac{3}{5}\lg\frac{3}{5} - \frac{2}{5}\lg\frac{2}{5} \approx 0.971$$

- Then

$$E(\text{ Outlook }) = \frac{5}{14}0.971 + \frac{4}{14}0.0 + \frac{5}{14}0.971 \approx 0.694$$

- So the gain is $\text{gain}(\text{ Outlook }) = 0.94 - 0.694 = 0.246$

Example

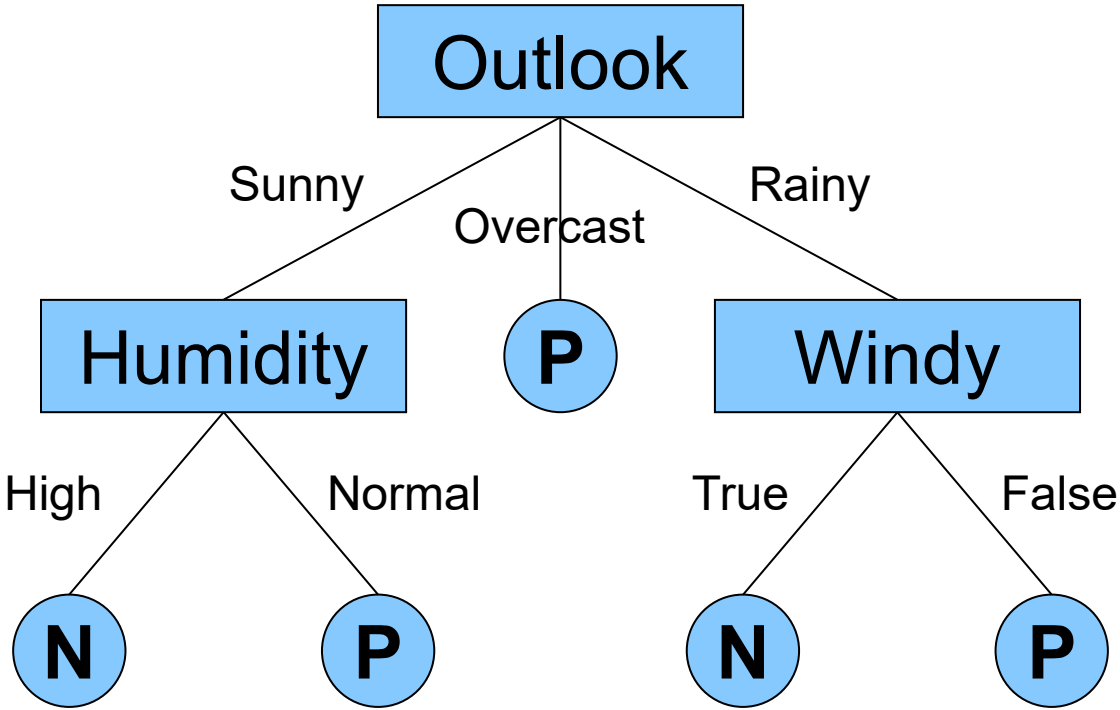
- Need the expected entropies for all four features

$$E(\text{Outlook}) \approx 0.694$$
$$E(\text{Temperature}) \approx 0.911$$
$$E(\text{Humidity}) \approx 0.789$$
$$E(\text{Wind}) \approx 0.892$$

- From this, we calculate the gain for all four features

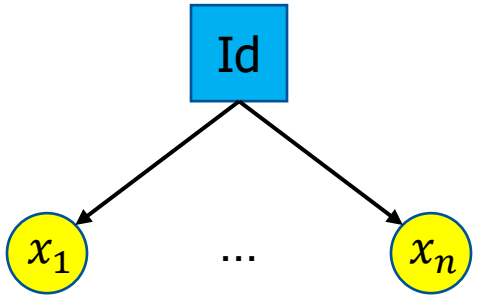
$$\text{gain}(\text{Outlook}) = 0.94 - 0.694 = 0.246$$
$$\text{gain}(\text{Temperature}) = 0.94 - 0.911 = 0.029$$
$$\text{gain}(\text{Humidity}) = 0.94 - 0.789 = 0.151$$
$$\text{gain}(\text{Wind}) = 0.94 - 0.892 = 0.048$$

Example


Outlook
Sunny
Overcast
Rainy
Humidity
P
Windy
High
Normal
True
False
N
P
N
P

Gain Ratio

We need to be careful not to use "features" that correspond to unique identifiers as decision variables in the tree

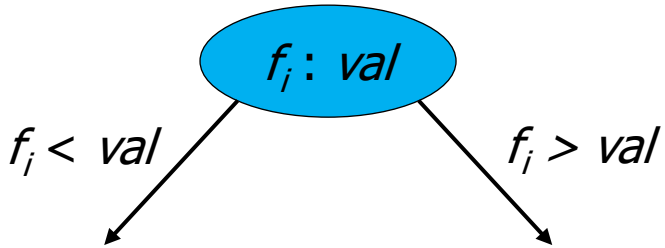


$$IV(f_i) = -\sum_{j=1}^{m_i} \frac{(p_j + n_j)}{(p+n)} \lg \frac{(p_j + n_j)}{(p+n)}$$

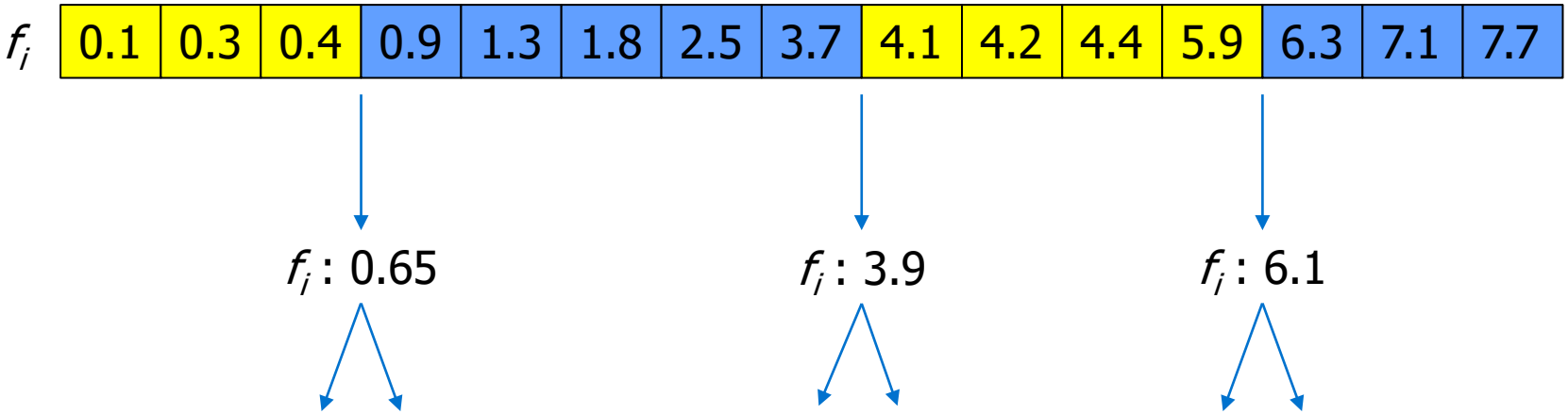
$$\text{gainRatio}(f_i) = \frac{\text{gain}(f_i)}{\text{IV}(f_i)}$$

Continuous Attributes

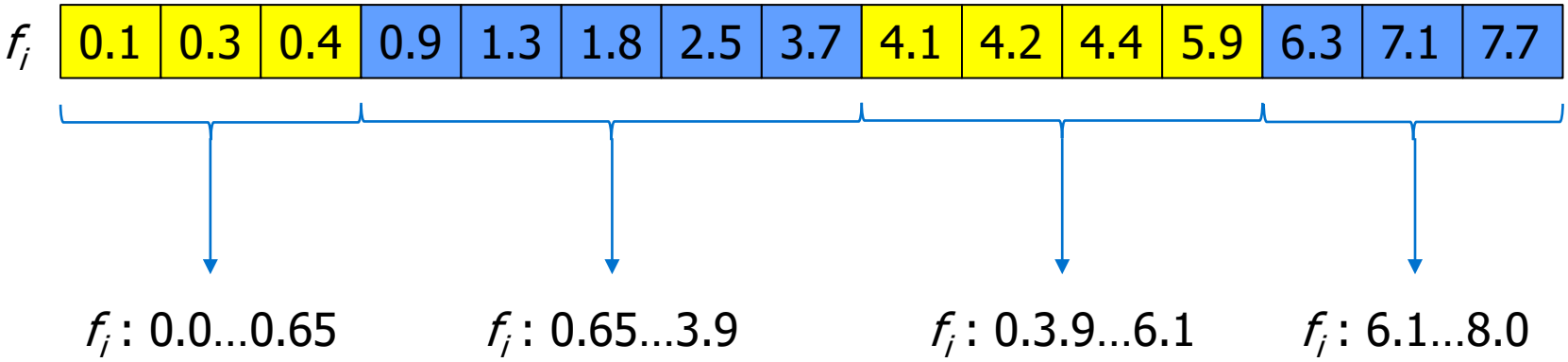
- Standard decision trees assume discrete attributes.
- Decision trees are able to handle continuous attributes as well.
- Generally, when dealing with continuous attributes, binary splits are generated for those attributes.

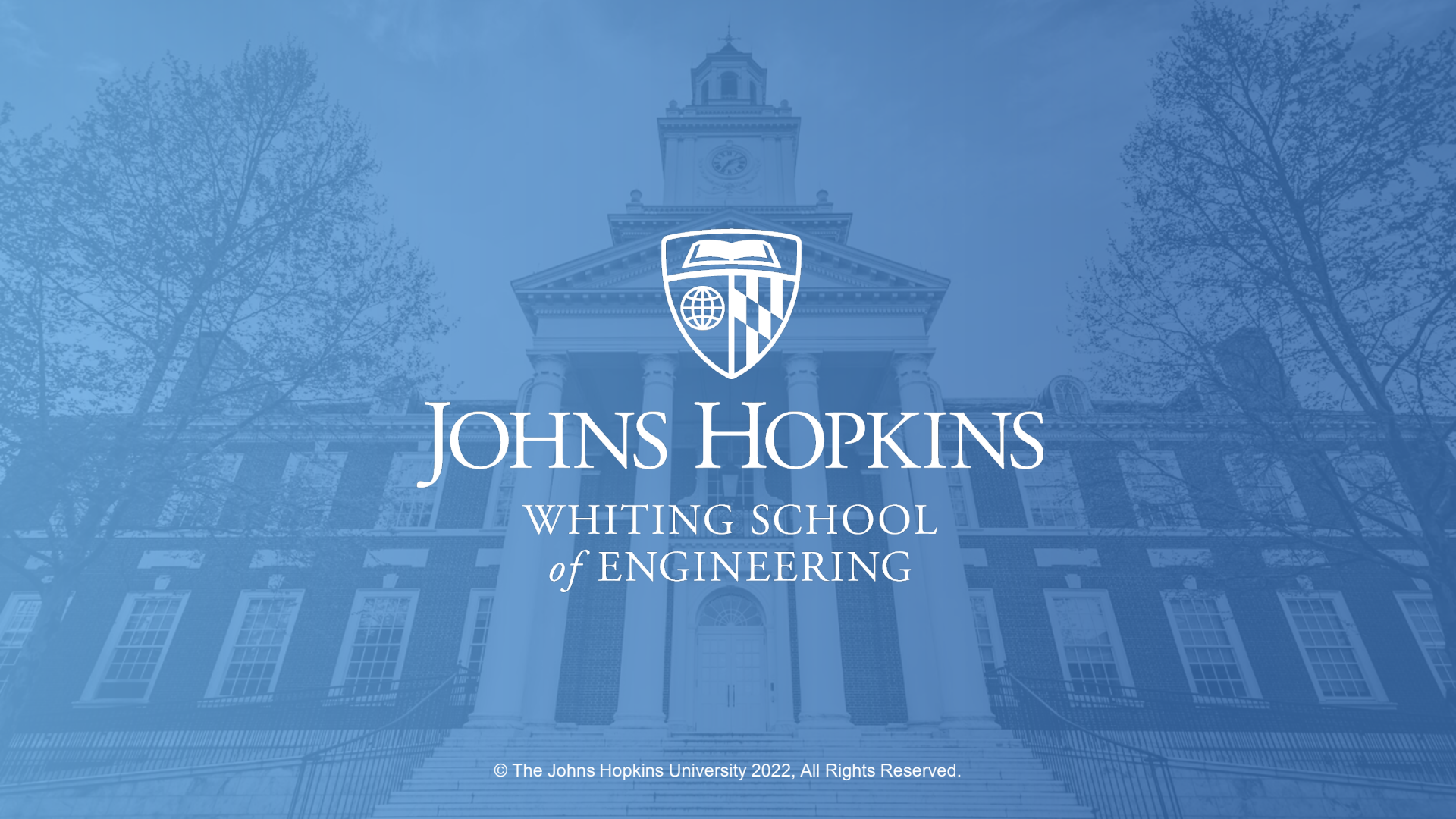


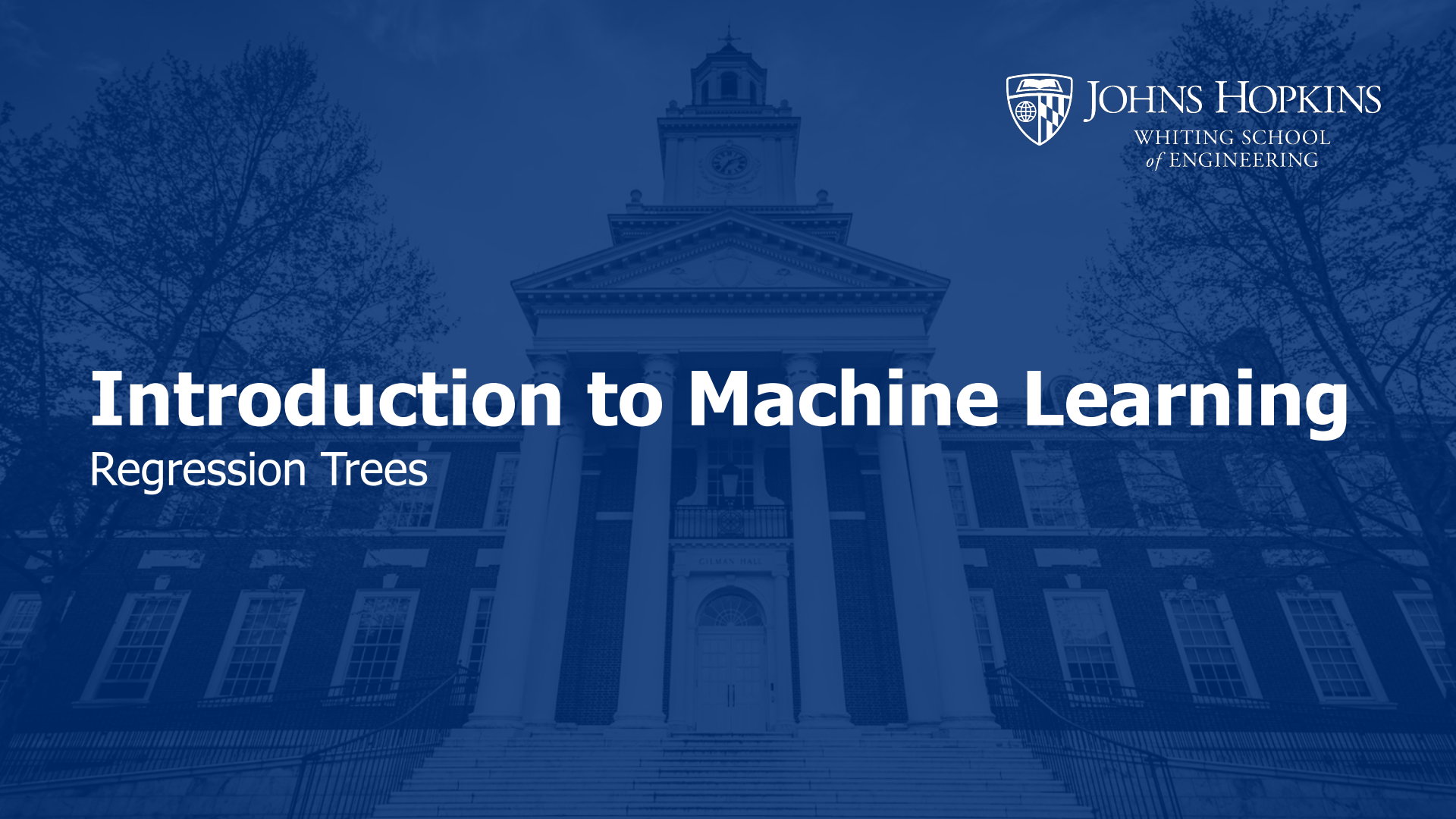
Continuous Attributes – Classification



Continuous Attributes – Classification







Introduction to Machine Learning

Regression Trees

Regression Tree Learning

- The task is to learn a tree that predicts real-valued function $f(\mathbf{x})$
- The primary change over classification trees is the splitting criterion
- Linear Regression
  - Ordinary Linear Regression (OLR)
  - Ordinary Least Squares (OLS)
- Training set $\mathcal{D} = \left\{\mathbf{x}^t, r^t\right\}_{t=1}^{n}$ where $r^t \in \Re$
- Response function is assumed to have form $r^t = f\left(\mathbf{x}^t\right) + \epsilon$
- Functional form is linear: $f\left(\mathbf{x}^t\right) = \alpha_0 + \sum_{i=1}^{d} \alpha_i x_i^t$
- Find suitable values for $\{\alpha_0, \alpha_1, \ldots, \alpha_d\}$
- Loss function: $\mathrm{MSE}(\boldsymbol{\alpha}) = \frac{1}{n} \sum_{t=1}^{n} \left(f\left(\mathbf{x}^t\right) - r^t\right)^2$

Maximum Likelihood Estimation

- Recall assumption that $r^t = f\left(\mathbf{x}^t\right) + \epsilon$
- Maximum likelihood hypothesis says

$$
\begin{aligned}
h_{ML} &= \\
&= \arg\max_{h \in \mathcal{H}} \prod_{t=1}^{n} P(\mathbf{x}^t | h) \\
&= \arg\max_{h \in \mathcal{H}} \prod_{t=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{f(\mathbf{x}^t) - r^t}{\sigma}\right)^2\right]
\end{aligned}
$$

Maximum Likelihood Estimation

Simplifying

$$
\begin{aligned}
h_{ml} &= \arg\max_{h\in\mathcal{H}} \sum_{t=1}^{n} \ln \frac{1}{\sigma\sqrt{2\pi}} - \frac{1}{2}\left(\frac{f(\mathbf{x}^t) - r^t}{\sigma}\right)^2 \\
&= \arg\max_{h\in\mathcal{H}} \sum_{t=1}^{n} -\frac{1}{2}\left(\frac{f(\mathbf{x}^t) - r^t}{\sigma}\right)^2 \\
&= \arg\max_{h\in\mathcal{H}} \sum_{t=1}^{n} -(f(\mathbf{x}^t) - r^t)^2 \\
&= \arg\min_{h\in\mathcal{H}} \sum_{t=1}^{n} (f(\mathbf{x}^t) - r^t)^2
\end{aligned}
$$

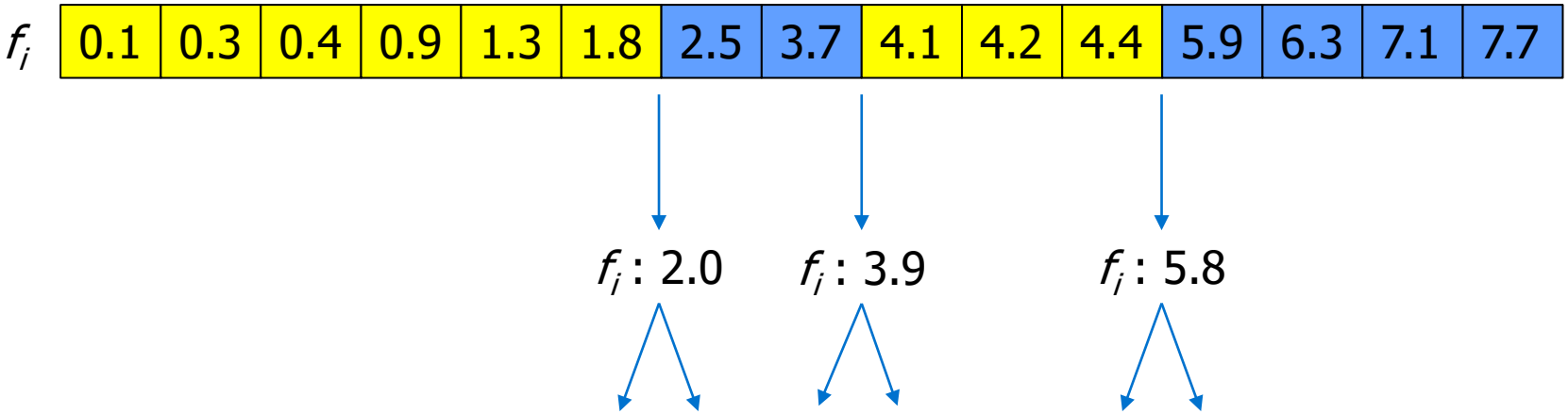
Gradient descent rule: $\nabla_\alpha MSE(\boldsymbol{\alpha}) = \frac{2}{n}\sum_{t=1}^{n}\left(\boldsymbol{\alpha}^\top \mathbf{x}^t - r^t\right)\mathbf{x}^t$

Real-Valued Features

- Binary splits only
  - Sort data by the feature in question
  - Consider successive midpoints
- Can still lead to excessive splits to evaluation
- One strategy is to consider quantiles
- Another strategy is to consider the mean, mean minus one standard deviation, and mean plus one standard deviation

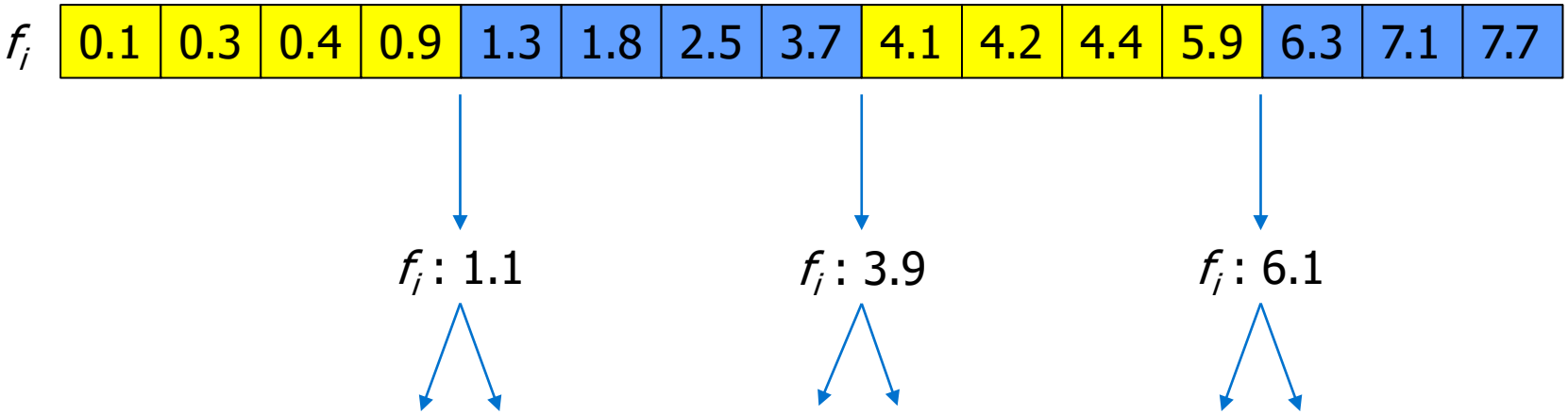
Continuous Attributes – Regression

Equal width: four bins, every 1.9 from baseline



Continuous Attributes – Regression

Equal frequency: four bins, every 4 points from baseline



Using a Regression Tree

- Piecewise constant regressogram
- Local least squares
  - Small disjuncts problem
  - Typically use the "mean" of the points at the leaves

where

$$\hat{r} = \frac{\sum_t b_m\left(\mathbf{x}^t\right) r^t}{\sum_t b_m\left(\mathbf{x}^t\right)}$$

$$b_m(\mathbf{x}) = \begin{cases} 1 & \text{if } x \in \mathcal{D}_m \\ 0 & \text{otherwise} \end{cases}$$

Splitting Criterion

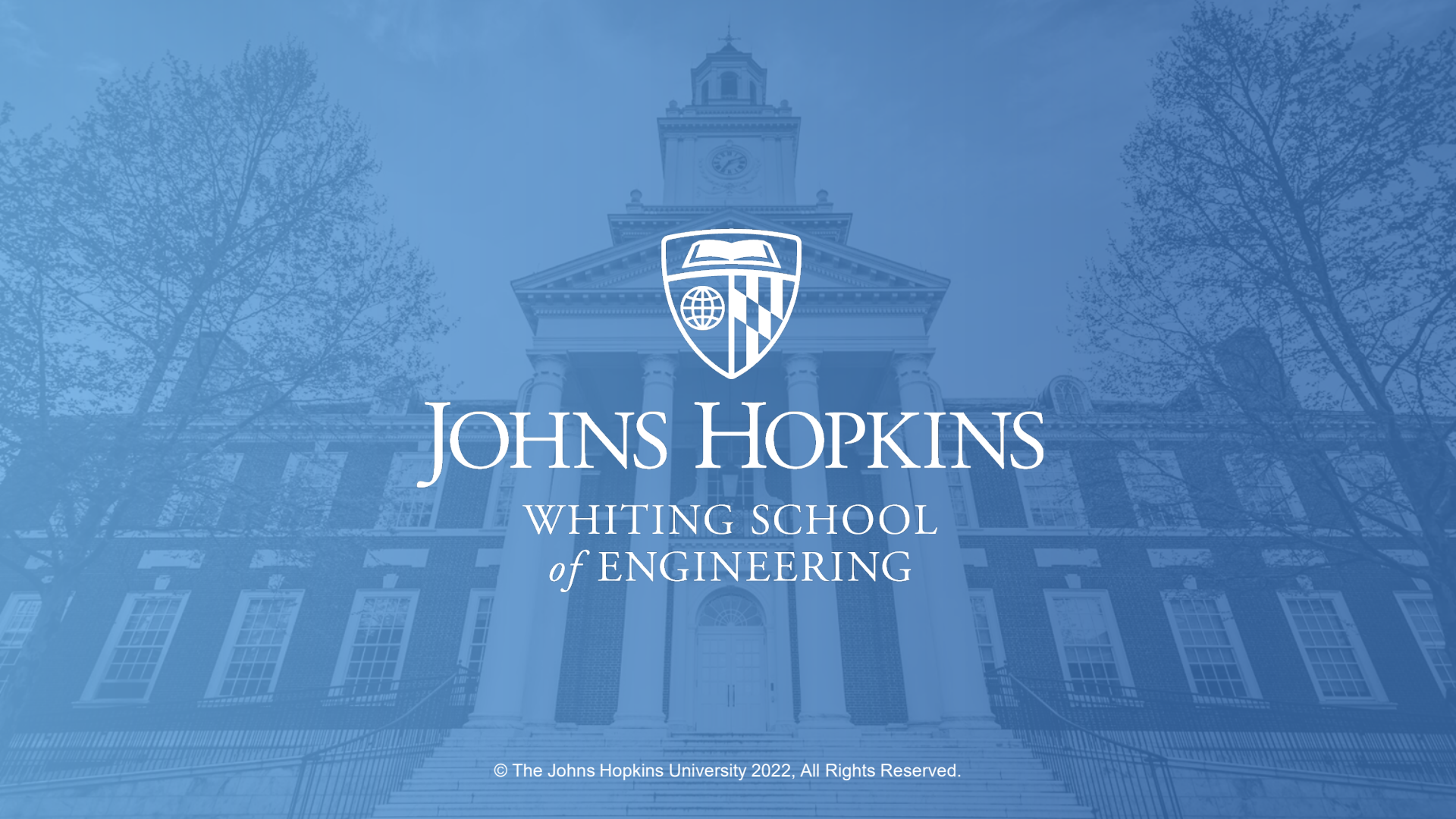
- Use mean squared error (MSE) with "early stopping"
- Let $\mathcal{D}_{mj} \subseteq \mathcal{D}_m$ be a subset of the points at node *m* in the tree when taking branch *j*.
- Then

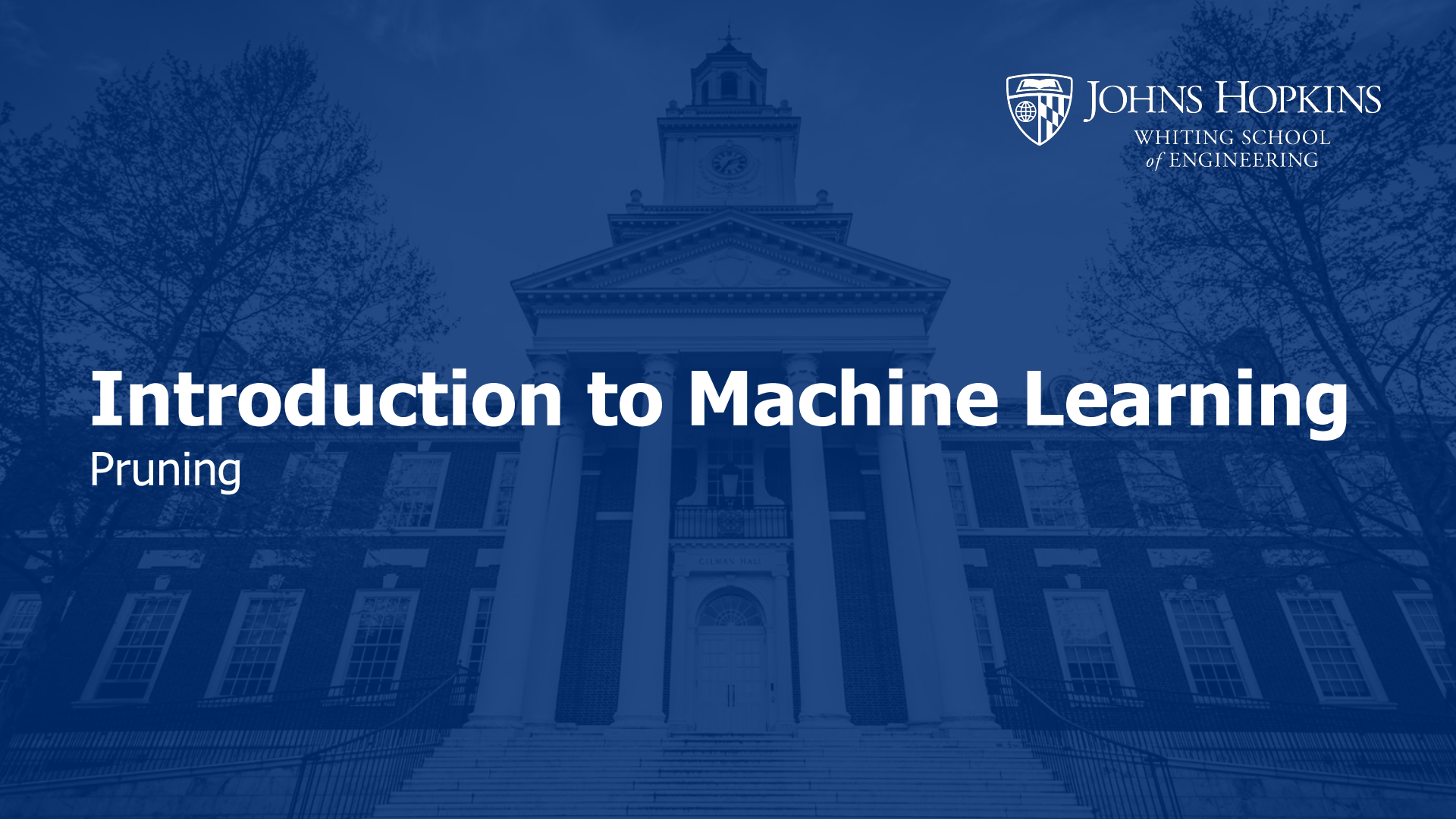
$$\bigcup_{j=1}^{p} \mathcal{D}_{mj} = \mathcal{D}_m$$

- Let

$$b_{mj}(\mathbf{x}) = \begin{cases} 1 & \text{if } x \in \mathcal{D}_{mj} \\ 0 & \text{otherwise} \end{cases}$$

- Then estimate the response as $\hat{r} = \frac{\sum_t b_m\left(\mathbf{x}^t\right) r^t}{\sum_t b_m\left(\mathbf{x}^t\right)}$
- Make split that minimizes $MSE'_m = \frac{1}{n_m} \sum_j \sum_t \left(r^t - \hat{r}^t_{mj}\right)^2 b\_mj\left(\mathbf{x}^t\right)$





Introduction to Machine Learning

Pruning

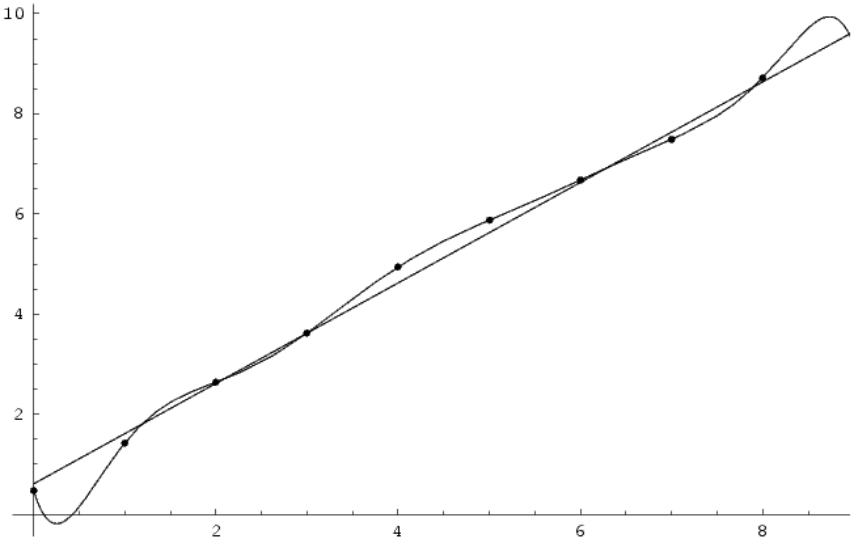
Overfitting

- We say a tree $\mathcal{T}$ overfits to data $\mathcal{D}$ if $\exists \mathcal{T}'$ such that

$$\text{error}_{\mathcal{D}}(\mathcal{T}) < \text{error}_{\mathcal{D}}(\mathcal{T}') \text{ but}$$

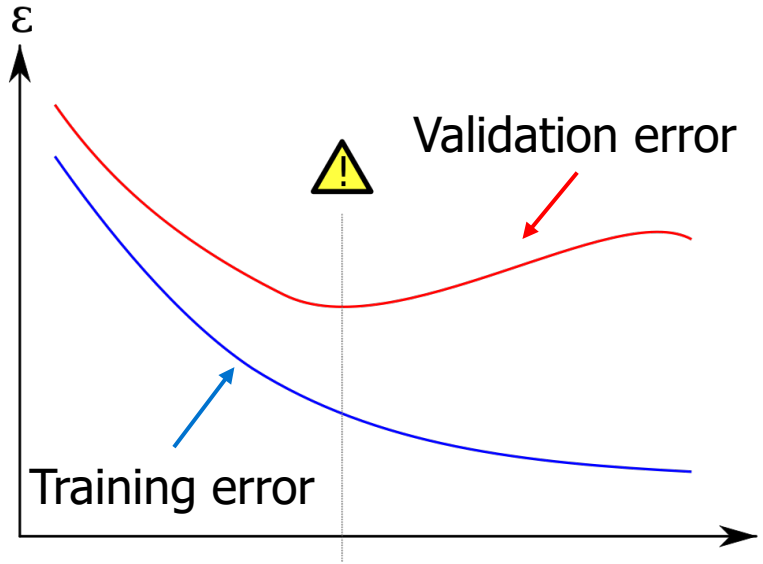
$$\text{error}_{X}(\mathcal{T}) > \text{error}_{X}(\mathcal{T}')$$

- In other words, while error on the training data is low, generalization error is high
- Results in fitting the ”noise” in the data
- Often tied to the number of parameters in the model



Using a Validation Set

- To avoid overfitting, one approach is to maintain a separate "validation" set
- Track training error, $\text{error}_{\mathcal{D}}(\mathcal{T})$
  relative to the validation error, $\text{error}_{\mathcal{X}}(\mathcal{T})$
- When curves diverge, overfitting is starting to occur
- Issues:
  - If data is limited, validation set makes matters worse
  - Comparing algorithms where one uses a validation set and the other does not is unfair
- Alternative: Regularization



Reduced Error Pruning

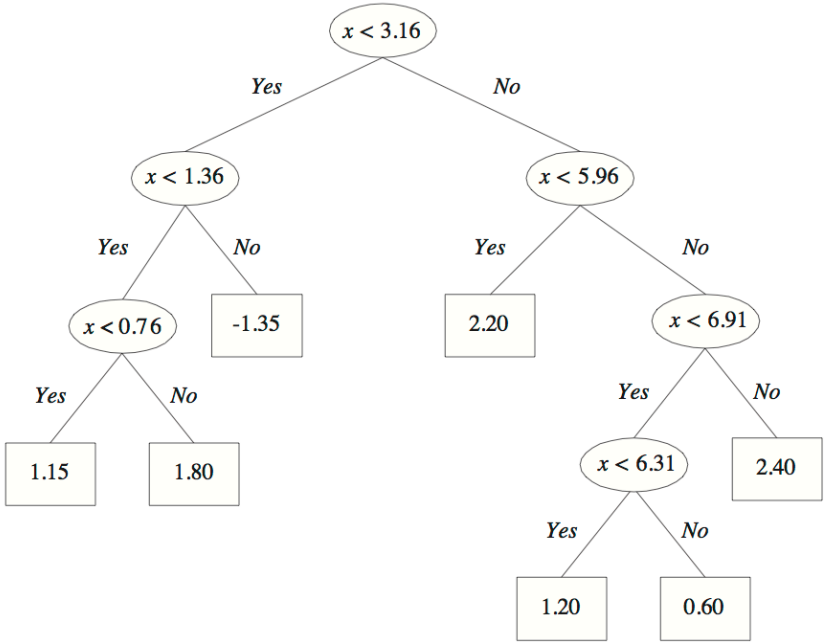
- Also known as “post-pruning”
- Grow tree until it cannot be grown any further (overfits)
- Tag vertices for possible pruning
- Test possible pruned trees by removing tagged vertices one at a time
- Test the best against the original on validation set
- Accept pruned tree if no worse
- Repeat until performance on validation set starts to degrade

Early Stopping – Regression

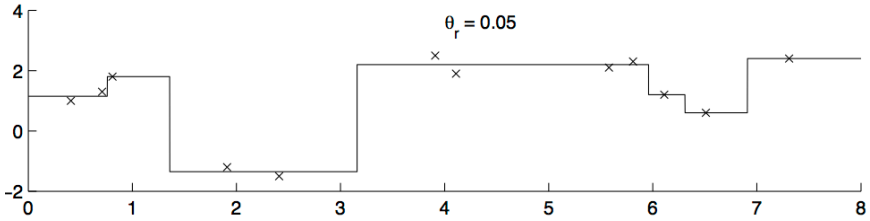
- Also known as "pre-pruning"
- Approach commonly used on regression trees, but can be done with classification trees too
- Can be done without a validation set (but can be done with a validation set too)
- For regression, loss function is $MSE(\boldsymbol{\alpha})$
- Set a threshold (tuned) θ_r to be used as a stopping criterion
- Once $MSE(\boldsymbol{\alpha})$ drops below θ_r stop

Example

Let $\theta_r = 0.5$

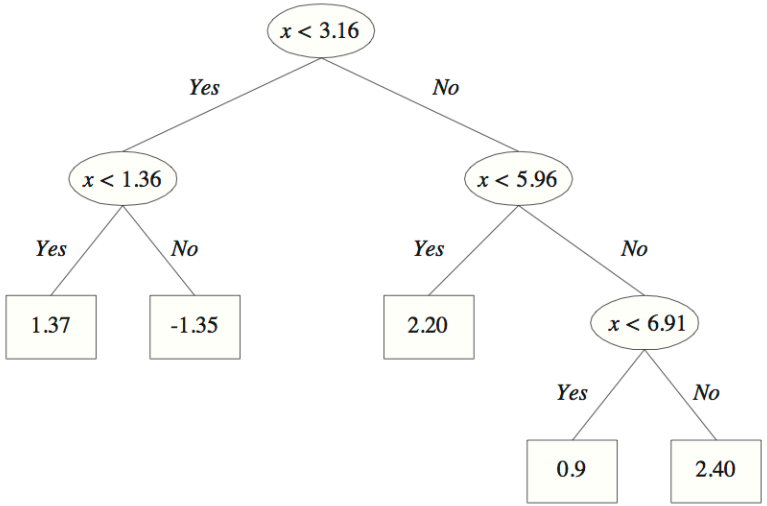


Regressogram

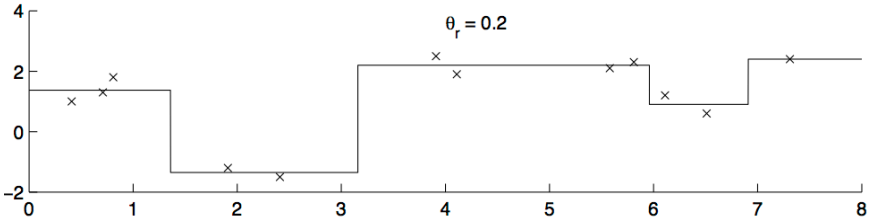


Example

Let $\theta_r = 0.2$

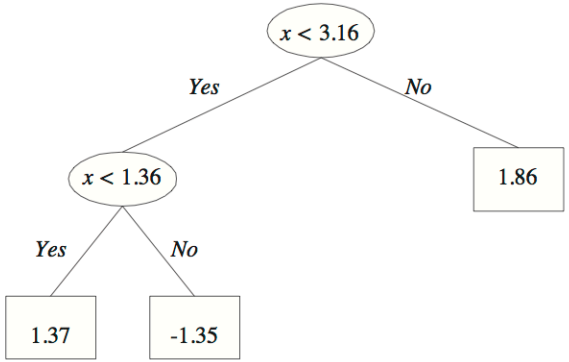


Regressogram

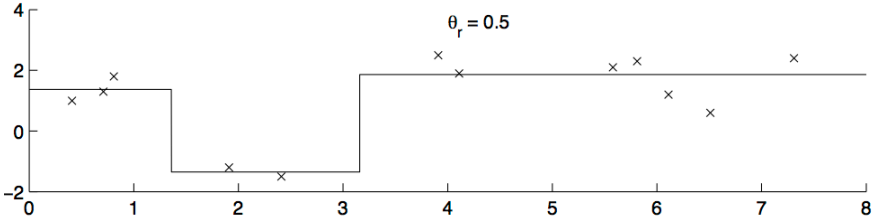


Example

Let $\theta_r = 0.5$



Regressogram



Early Stopping – Classification

- Quinlan proposed an early stopping rule for ID3
- Proposed using a χ^2 test

  H_0 : A is independent of class of objects C being split by A
- Suppose we test at a 99% confidence level

$$p_i' = p \cdot \frac{p_i + n_i}{p + n}$$

$$n_i' = n \cdot \frac{p_i + n_i}{p + n}$$

$$\chi^2 = \sum_{i=1}^{v} \left[\frac{(p_i - p_i')^2}{p_i'} + \frac{(n_i - n_i')^2}{n_i'} \right]$$

- Assumes $v - 1$ degrees of freedom, where v is the number of values A can take on