Supervised Learning Intro

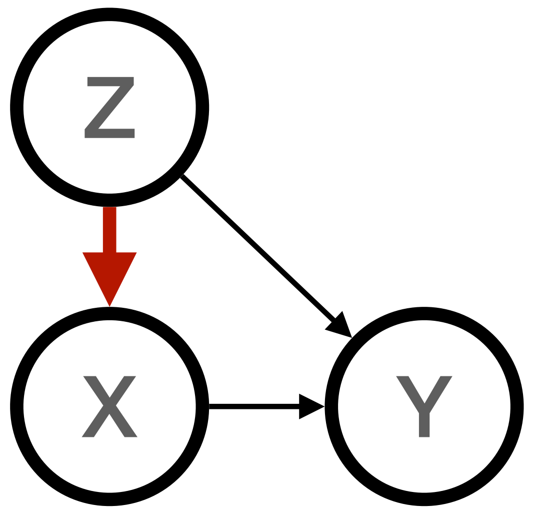
CMPUT 366: Intelligent Systems

P&M §7.1-7.2

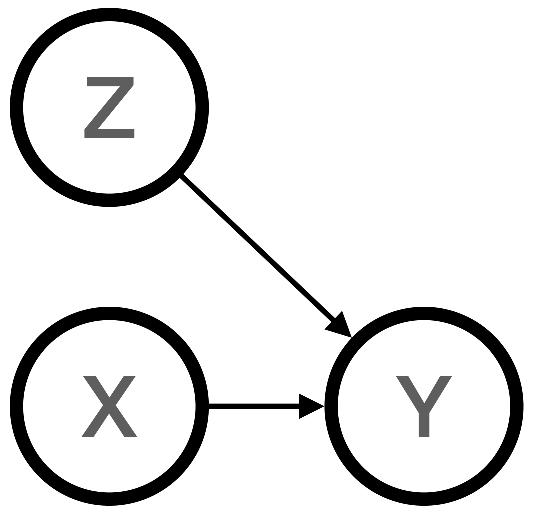
Recap: Causal Inference

- Observational queries $P(Y \mid X = x)$ are different from causal queries $P(Y \mid do(X = x))$
- To evaluate causal query $P(Y \mid do(X = x))$:
  1. Construct post-intervention distribution $\hat{P}$ by removing all links from X's direct parents to X
  2. Evaluate the observational query $\hat{P}(Y \mid X = x)$ in the post-intervention distribution
- Alternative representation: Influence diagrams
  - Causal query in the augmented distribution: $\tilde{P}(Y \mid F_X = x)$
  - Observational query in the augmented distribution: $\tilde{P}(Y \mid X = x, F_X = idle)$
- Not every correct Bayesian network is a valid causal model

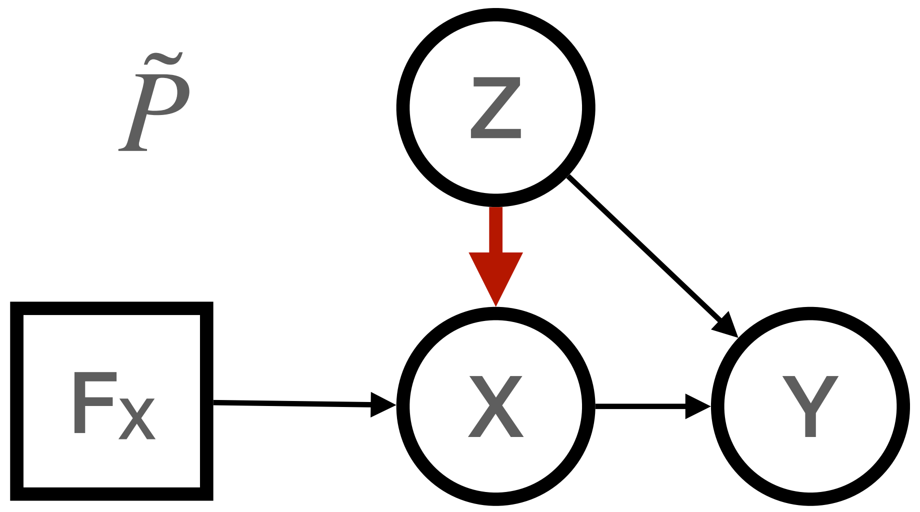
P



$\hat{P}$



$\tilde{P}$



Lecture Outline

1. Recap
2. Supervised Learning Problem
3. Measuring Prediction Quality

Supervised Learning

Definition: A **supervised learning task** consists of

- A set of **input features** $X_1, \ldots, X_n$
- A set of **target features** $Y_1, \ldots, Y_k$
- A set of **training examples**, for which both input and target features are given
- A set of **test examples**, for which only the input features are given

The goal is to **predict** the values of the **target features** given the **input features**; i.e., **learn** a function $h(x)$ that will map features X to a prediction of Y

- **Classification:** Y_i are **discrete**
- **Regression:** Y_i are **real-valued**

Regression Example

- Aim is to predict the value of **target** Y based on **features** X
- Both X and Y are **real-valued**
  - **Exact values** of both targets and features may not have been in the training set
  - e_8 is an **interpolation** problem: X is **within the range** of the training examples' values
  - e_9 is an **extrapolation** problem: X is **outside** the range of the training examples' values

Ex.	X	Y
e_1	0.7	1.7
e_2	1.1	2.4
e_3	1.3	2.5
e_4	1.9	1.7
e_5	2.6	2.1
e_6	3.1	2.3
e_7	3.9	7

e_8	2.9	?
e_9	5.0	?

Data Representation

- For **real-valued** features, we typically just record the feature values
- For **discrete** features, there are multiple options:
  - **Binary features:** Can code $\{false, true\}$ as $\{0,1\}$ or $\{-1,1\}$
  - Can record **numeric** values for each possible value
    - **Cardinal values: Differences** are meaningful (e.g.,$1,2,7$)
    - **Ordinal values: Order** is meaningful (e.g., $Good, Fair, Poor$)
    - **Categorical values: Neither** differences nor order meaningful (e.g., $Red, Green, Blue$)
  - Vector of **indicator variables**: One per feature value, exactly one is true (sometimes called a "one-hot" encoding) (e.g., Red as $(1,0,0)$, $Green$ as $(0,1,0)$, etc.)

Classification Example: Holiday Preferences

- An agent wants to learn a person's preference for the **length** of holidays
- Holiday can be for 1,2,3,4,5, or 6 days
- Two possible representations:

Ex.	Y
e_1	1
e_2	6
e_3	6
e_4	2
e_5	1

Ex.	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
e_1	1	0	0	0	0	0
e_2	0	0	0	0	0	1
e_3	0	0	0	0	0	1
e_4	0	1	0	0	0	0
e_5	1	0	0	0	0	0

Generalization

- **Question:** What does it mean for a (supervised) learning agent to **perform well**?
- We want to be able to make correct predictions on **unseen** data, not just the training examples
    - We are even willing to sacrifice some **training** accuracy to achieve this
    - We want our learners to **generalize**: to go beyond the given training examples to classify **new examples** well
    - **Problem:** We can't observe performance on unobserved examples!
- We can **estimate** generalization performance by evaluating performance on the **test set** (**Why**?)
    - The learning algorithm doesn't have access to the test data, but we do

Generalization Example

Example: Consider binary two classifiers, **P** and **N**

- **P** classifies all the **positive examples** from the training data as $true$, and all others as $false$
- **N** classifies all of the **negative examples** from the training data as $false$, and all others as $true$

Question: Which classifier performs better on the **training data**?

Question: Which classifier **generalizes** better?

Bias

- The **hypothesis** is the function $h(X)$ that we learn
- The **hypothesis space** is the set of **possible hypotheses**
- A preference for one hypothesis over another is called **bias**
  - Bias is not a bad thing in this context!
  - Preference for "simple" models is a bias
  - Which bias works best for **generalization** is an **empirical** question

Learning as Search

- Given **training data**, a **hypothesis space**, an **error measurement**, and a **bias**, learning can be reduced to **search**
- Learning searches the hypothesis space trying to find the hypothesis that best fits the data given the bias
    - Search space is prohibitively **large** (typically infinite)
    - Almost all machine learning methods are versions of **local search**

Measuring Prediction Error

- We choose our hypothesis partly by measuring its **performance** on training data
  - **Question:** What is the other consideration?
- This is usually described as **minimizing** some quantitative measurement of **error** (or **loss**)
  - **Question:** What might error mean?

0/1 Error

Definition:

The **0/1 error** for a dataset E of examples and hypothesis $\hat{Y}$ is the number of examples for which the prediction was not correct:

$$\sum_{e \in E} 1\left[Y(e) \neq \hat{Y}(e)\right]$$

$1[\,\cdot\,]$ is **indicator function**: value is 1 if the expression in brackets is TRUE, else 0

- Not appropriate for **real-valued** target features (**why**?)
- Does not take into account **how wrong** the answer is
  - e.g., $1\left[2 \neq 1\right] = 1\left[6 \neq 1\right]$
- Most appropriate for **binary** or **categorical** target features

Absolute Error

Definition:

The **absolute error** for a dataset E of examples and hypothesis $\hat{Y}$ is the sum of absolute distances between the predicted target value and the actual target value:

$$\sum_{e \in E} \left| Y(e) - \hat{Y}(e) \right| .$$

- Meaningless for **categorical** variables
- Takes account of **how wrong** the predictions are
- Most appropriate for **cardinal** or *possibly* **ordinal** values

Squared Error

Definition:

The **squared error** (or sum of squares error or mean squared error) for a dataset E of examples and hypothesis $\hat{Y}$ is the sum of squared distances between the predicted target value and the actual target value:

$$\sum_{e \in E} \left(Y(e) - \hat{Y}(e) \right)^2 .$$

- Meaningless for **categorical** variables
- Takes account of **how wrong** the predictions are
  - **Large** errors are much more important than **small** errors
- Most appropriate for **cardinal** values

Worst-Case Error

Definition:
The **worst-case error** for a dataset E of examples and hypothesis $\hat{Y}$ is the maximum absolute difference between the predicted target value and the actual target value:

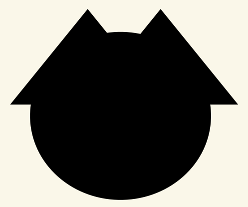
$$\max_{e \in E} \left| Y(e) - \hat{Y}(e) \right| .$$

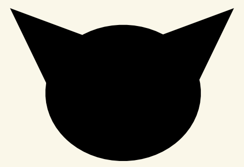
- Meaningless for **categorical** variables
- Takes account of **how wrong** the predictions are
  - but only on **one example**
    (the one whose prediction is furthest from the true target)
- Most appropriate for **cardinal** values

Probabilistic Predictors

- Rather than predicting **exactly** what a target value will be, many common algorithms predict a **probability distribution** over possible values
  - Especially for **classification** tasks
- Vectors of **indicator variables** are the most common data representation for this scheme:
  - Target features of **training** examples have a single 1 for the **true** value
  - **Predicted** target values are **probabilities** that sum to 1

Probabilistic Predictions Example

X	Y_{cat}	Y_{dog}	Y_{panda}
	1	0	0
	0	1	0

X	$\hat{Y}_{cat}$	$\hat{Y}_{dog}$	$\hat{Y}_{panda}$
	0.5	0.45	0.05

Likelihood

- For **probabilistic** predictions, we can use **likelihood** to measure the performance of a learning algorithm

Definition:
The **likelihood** for a dataset E of examples and hypothesis $\hat{Y}$ is the **probability** of independently observing the examples according to the probabilities assigned by the **hypothesis**:

$$\Pr(E) = \prod_{e \in E} \hat{Y}(e = Y(e)).$$

- This has a clear Bayesian interpretation
- **Numerical stability issues:** product of probabilities shrinks **exponentially**!
    - *Example:* Probability of **any** sequence of 1000 coin tosses has probability 2^{-1000}!
    - Floating point underflows almost immediately

Log-Likelihood

Definition:

The **log-likelihood** for a dataset E of examples and hypothesis $\hat{Y}$ is the **log-probability** of independently observing the examples according to the probabilities assigned by the hypothesis:

$$\log \Pr(E) = \log \prod_{e \in E} \hat{Y}(e = Y(e))$$
$$= \sum_{e \in E} \log \hat{Y}(e = Y(e)) \,.$$

- Taking log of the likelihood fixes the underflow issue (**why**?)
- The log function grows **monotonically**, so maximizing log-likelihood is the **same thing** as maximizing likelihood:

$$\left(\Pr(E \mid \hat{Y}_1) > \Pr(E \mid \hat{Y}_2)\right) \iff \left(\log \Pr(E \mid \hat{Y}_1) > \log \Pr(E \mid \hat{Y}_2)\right)$$

Summary

- **Supervised learning** is learning a **hypothesis** function from training examples
  - Maps from **input** features to **target** features
  - **Classification: Discrete** target features
  - **Regression: Real-valued** target features
- **Preferences** among hypotheses are called **bias**
  - An important component of learning!
- Choice of **error measurement** (**loss**) is an important design decision
  - Each loss has its own advantages/disadvantages