Supervised Learning
Introduction & Framework

CMPUT 261: Introduction to Artificial Intelligence

P&M §7.1-7.3

Assignments

- **Assignment #2** is now available
  - Due **Feb 29/2024** (three weeks from today) at **11:59pm**

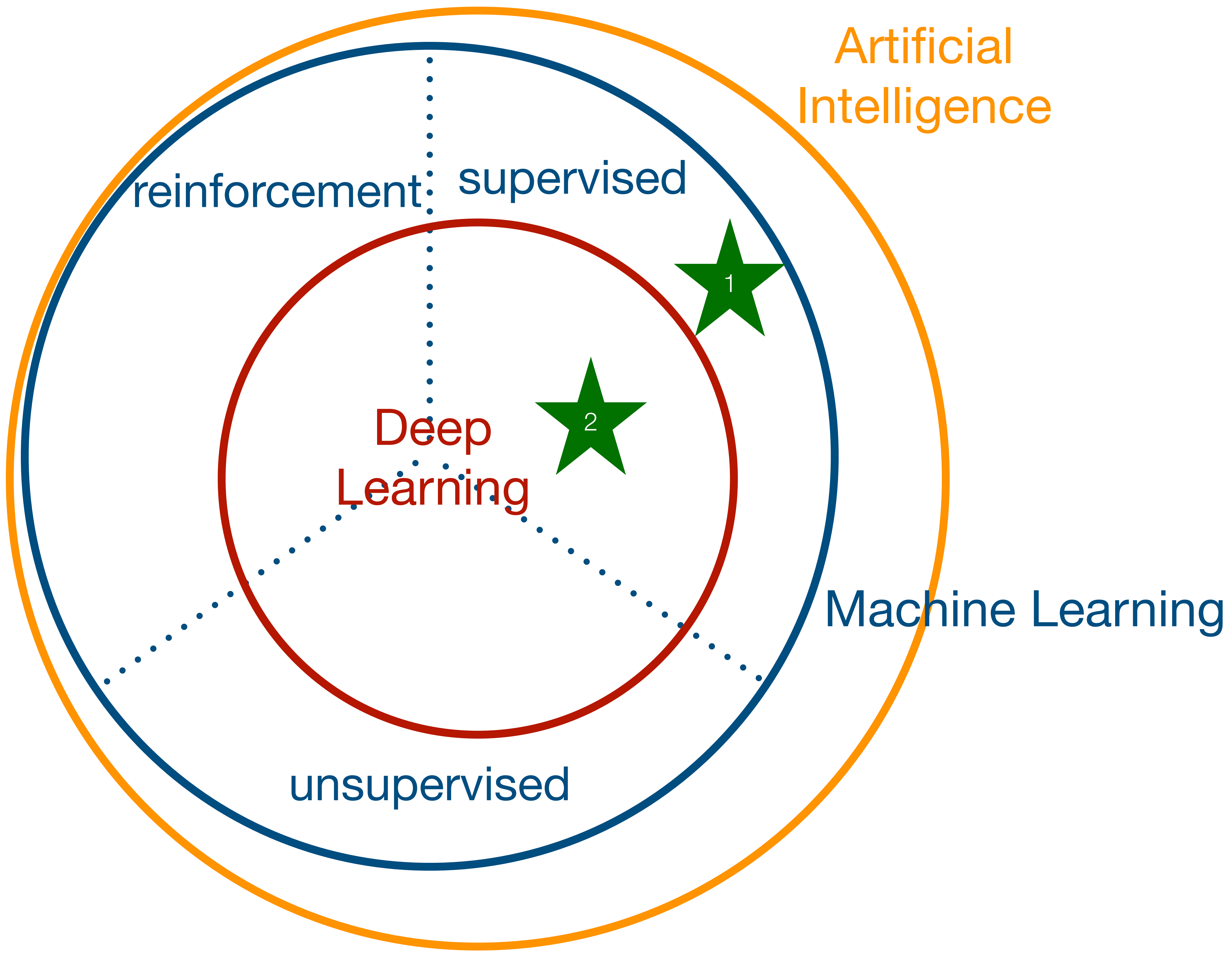
Recap: Uncertainty

- We represent uncertainty about the world by **probabilities**
  - We update our knowledge by **conditioning** on **observations**
  - Observations = learning the value of a **random variable**
- Full, **unstructured** **joint distributions** are intractable to reason about
- **Conditional independence** is a kind of **structure** that is:
  1. widespread
  2. easy to reason about
  3. allows tractable **inference** (computing distribution of unobserved variables)
- **Belief networks** let us compactly represent joint distributions with a lot of conditional independence
  - **Variable elimination** is an algorithm for efficient inference on belief networks

Supervised Learning, informally

- In the uncertainty section, we took the probability distribution as **given**
    - Our only problem was to represent and derive distributions
- **Question:** Where do these probabilities **come from**?
- **Supervised learning** is a way to learn probabilities from **examples**
    - Probability of a **target** feature (or **label**) given **input features**
    - i.e., **condition** on input features to get probability of target
- Basic idea:
    - Take a bunch of inputs (e.g., images) and "correct" outputs
    - Learn a model that correctly maps inputs to outputs

Supervised Learning vs. Machine Learning vs. Deep Learning



What is the difference between Supervised Learning, Machine Learning, and Deep Learning?

Lecture Outline

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

After this lecture, you should be able to:

- define supervised learning task, classification, regression, loss function
- represent categorical target values in multiple ways (indicator variables, indexes)
- define generalization performance
- identify an appropriate loss function for different tasks
- explain why a separate test set estimates generalization performance
- define 0/1 error, absolute error, (log-)likelihood loss, mean squared error, worst-case error

Supervised Learning

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

- A set of **input features** $X_1, \ldots, X_d$
- A set of **target features** $Y_1, \ldots, Y_k$
- A set of **training examples** $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$ sampled randomly from some population
- A set of **test examples** $T = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m}$ sampled from the **same population**

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**

Supervised Learning Examples

1. **Computational vision:** Given example images and labels representing objects, output a label for the main object in the image
    - *Input features:* Pixel values of the image
    - *Target features:* One feature for each label (e.g., **dog**, **plane**, etc.)
2. **Precision medicine:** Given examples of symptoms, test results, and treatments, output an estimate of recovery time
    - *Input features:* symptoms, treatment indicators, test results, demographic information
    - *Target features:* recovery time, survival time, etc.
3. **Natural language processing:** Given example sentences and labels representing "sentiment", output how positive or negative the sentence is
    - *Input features:* binary indicators for words or characters (**!)
    - *Target features:* One feature per label (e.g., **positive**, **negative**)

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
  - Input 8 is an **interpolation** problem: X is **within the range** of the training examples' values
  - Input 9 is an **extrapolation** problem: X is **outside** the range of the training examples' values

i	$x^{(i)}$	$y^{(i)}$
1	0.7	1.7
2	1.1	2.4
3	1.3	2.5
4	1.9	1.7
5	2.6	2.1
6	3.1	2.3
7	3.9	7
8	2.9	?
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:

i	$y^{(i)}$
1	1
2	6
3	6
4	2
5	1

i	$y^{(i)}_1$	$y^{(i)}_2$	$y^{(i)}_3$	$y^{(i)}_4$	$y^{(i)}_5$	$y^{(i)}_6$
1	1	0	0	0	0	0
2	0	0	0	0	0	1
3	0	0	0	0	0	1
4	0	1	0	0	0	0
5	1	0	0	0	0	0

Question:

What are the advantages/ disadvantages of each representation?

Generalization

- **Question:** What does it mean for a trained model 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 measure 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**
    - "Training a model" =
      "Choosing a hypothesis from the hypothesis space based on data"
- 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

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 of n examples and hypothesis h is the number of examples for which the prediction was not correct:

$$\sum_{i=1}^{n} \mathbb{1}\left[y^{(i)} \neq h(\mathbf{x}^{(i)})\right]$$

$\mathbb{1}[\,\cdot\,]$ 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., $\mathbb{1}\left[2 \neq 1\right] = \mathbb{1}\left[6 \neq 1\right]$
- Most appropriate for **binary** or **categorical** target features

Absolute Error

Definition:

The **absolute error** for a dataset of n examples and hypothesis h is the sum of absolute distances between the predicted target value and the actual target value:

$$\sum_{i=1}^{n} \left| y^{(i)} - h(\mathbf{x}^{(i)}) \right|$$

- 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 of n examples and hypothesis h is the sum of squared distances between the predicted target value and the actual target value:

$$\sum_{i=1}^{n} \left(y^{(i)} - h(\mathbf{x}^{(i)})\right)^2$$

- 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 of n examples and hypothesis h is the maximum absolute difference between the predicted target value and the actual target value:

$$\max_{1 \le i \le n} \left| y^{(i)} - h(\mathbf{x}^{(i)}) \right|$$

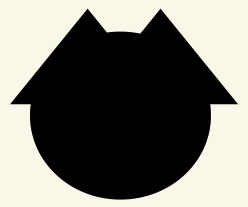
- Meaningless for **categorical** variables
- Takes account of **how wrong** the predictions are
  - but only on **one example**
    (the one whose prediction is furthest from the true target)
- Most appropriate for **cardinal** values

Probabilistic Predictors

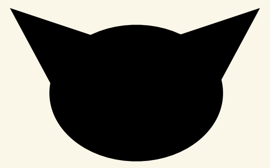
- Rather than predicting **exactly** what a target value will be, many common algorithms predict a **probability distribution** over possible values
    - Especially for **classification** tasks
- Vectors of **indicator variables** are the most common data representation for this scheme:
    - Target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

Probabilistic Predictions Example

Training examples

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

Output on test example

X	$h(X)_{cat}$	$h(X)_{dog}$	$h(X)_{panda}$
	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 S of examples and hypothesis h is the **probability** of independently observing the examples according to the probabilities assigned by the **hypothesis**:

$$\Pr(S \mid h) = \prod_{(\mathbf{x},y)\in S} h(\mathbf{x})_y$$

- This has a clear Bayesian interpretation
- We want to maximize likelihood, so it's not a loss (**why?**)
  - **Question:** What is the corresponding loss?
- **Numerical stability issues:** product of probabilities shrinks **exponentially**!
  - *Example:* Probability of **any** sequence of 5000 coin tosses has probability 2^{-5000}!
  - Floating point underflows almost immediately
    (double-precision floating point can't represent anything smaller than 2^{-1021})

Log-Likelihood

Definition:

The log-likelihood for a dataset S of examples and hypothesis h is the log-probability of independently observing the examples according to the probabilities assigned by the hypothesis:

$$\log \Pr(S \mid h) = \log \prod_{(\mathbf{x},y)\in S} h(\mathbf{x})_y$$

$$= \sum_{(\mathbf{x},y)\in S} \log h(\mathbf{x})_y$$

- 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:

$$\big(\Pr(S \mid h_1) > \Pr(S \mid h_2)\big) \iff \big(\log \Pr(S \mid h_1) > \log \Pr(S \mid h_2)\big)$$

Trivial Predictors

- The simplest possible predictor **ignores all input features** and just predicts the **same value** v for any example
- **Question:** Why would we every want to think about these?

Optimal Trivial Predictors for Binary Data

- Suppose we are predicting a **binary** target
- n_0 **negative** examples
- n_1 **positive** examples
- **Question:** What is the optimal single prediction?

Measure	**Optimal Prediction**
0/1 error	0 if $n_0 > n_1$ else 1
absolute error	0 if $n_0 > n_1$ else 1
squared error	$\frac{n_1}{n_0 + n_1}$
worst case	$\begin{cases} 0 & \text{if } n_1 = 0 \\ 1 & \text{if } n_0 = 0 \\ 0.5 & \text{otherwise} \end{cases}$
likelihood	$\frac{n_1}{n_0 + n_1}$
log-likelihood	$\frac{n_1}{n_0 + n_1}$

Optimal Trivial Predictor Derivations

0/1 error	0 if $n_0 > n_1$ else 1
(negative) log-likelihood	$\frac{n_1}{n_0 + n_1}$

$$L(v) = vn_0 + (1 - v)n_1$$

$$L(v) = -\,n_1 \log v - n_0 \log(1 - v)$$

$$\frac{d}{dv}L(v) = 0$$

$$0 = -\frac{n_1}{v} + \frac{n_0}{1 - v}$$

$$\frac{n_1}{v} = \frac{n_0}{1 - v}$$

$$\frac{n_1}{n_0} = \frac{v}{1 - v} \;\wedge (0 < v < 1) \implies v = \frac{n_1}{n_0 + n_1}$$

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**
- Choice of **error measurement** (**loss**) is an important design decision
- Different losses have different optimal trivial predictors
    - Trivial predictors are a baseline: your real model better outperform the trivial predictor