CS486/686: Introduction to Artificial Intelligence
Lecture 6a - Supervised Machine Learning: Foundations

Jesse Hoey & Victor Zhong

School of Computer Science, University of Waterloo

January 27, 2025

Readings: Poole & Mackworth Chap. 7.1-7.2

Learning

Learning is the ability to **improve behavior based on experience**

- The **range** of behaviors is expanded: the agent can do more
- The **accuracy** on tasks is improved: the agent can do things better
- The **speed** is improved: the agent can do things faster

Components of a learning problem

The following components are part of any learning problem:

- **Task**: The behavior or task that's being improved
  For example: classification, acting in an environment
- **Data**: The experiences that are being used to improve performance in the task
- **Measure of improvement**: How can the improvement be measured?
  For example: increasing accuracy in prediction, new skills that were not present initially, improved speed

Common Learning Tasks

- **Supervised classification**: Given a set of pre-classified training examples, classify a new instance
- **Unsupervised learning**: Find *natural classes* for examples
- **Reinforcement learning**: Determine what to do based on rewards and punishments
- **Transfer Learning**: Learning from an expert
- **Active Learning**: Learner actively seeks to learn
- **Inductive logic programming**: Build richer models in terms of logic programs

Feedback

Learning tasks can be characterized by the feedback given to the learner

- **Supervised learning**
  What has to be learned is specified for each example
- **Unsupervised learning**
  No classifications are given; the learner has to discover categories and regularities in the dat
- **Reinforcement learning** Feedback occurs after a sequence of actions; a form of supervised learning

Measuring Success

- The measure of success is not how well the agent performs on the training examples, but **how well the agent performs for new (unseen) examples**
- Consider two agents solving a binary classification task:
  - P claims the negative examples seen are the only negative examples
    Every other instance is positive
  - N claims the positive examples seen are the only positive examples
    Every other instance is negative
- Both agents **correctly classify every training example**, but **disagree on every other example**

Implementing P/N agents

Inputs:
e is the test example
$X(e)$ are the input variables of example e
$Y(e)$ is the output variable of example e (T/F)
data $[i = 1 \ldots N]$: training data, list of examples like e
Output: estimated Y value for the test example e

$y \leftarrow P(e, \text{data})$
if $X(e)$ is the same as some $X(\text{data}[i])$ **then**
    return $Y(\text{data}[i])$
else
    return True

P/N agents use training data as their model
they are "exemplar-based" agents need an exact match

Bias

- The tendency to prefer one hypothesis over another is called a **bias**
- A bias is **necessary** to make predictions on unseen data
- Saying a hypothesis is better than N's or P's hypothesis **isn't something that's obtained from the data**
- To have any inductive process make predictions on unseen data, you **need a bias**
- What constitutes a good bias is an empirical question about which **biases work best in practice**

Learning as search

- Given a representation and a bias, the problem of learning can be reduced to one of **search**
- Learning is search through the space of possible representations looking for the representation or representations that best fits the data, given the bias

Learning as search

- Given a representation and a bias, the problem of learning can be reduced to one of **search**
- Learning is search through the space of possible representations looking for the representation or representations that best fits the data, given the bias
- These search spaces are typically prohibitively large for systematic search
- A learning algorithm is made of a **search space**, an **evaluation function**, and a **search method**

Supervised Learning

Given:

- 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** where the values for the input features and the target features are given for each example
- a set of **test examples**, where only the values for the input features are given

Predict the values for the target features for the test examples

Supervised Learning

Given:

- 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** where the values for the input features and the target features are given for each example
- a set of **test examples**, where only the values for the input features are given

Predict the values for the target features for the test examples

- **Classification** when the Y_i are discrete
- **Regression** when the Y_i are continuous

Supervised Learning

Given:

- 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** where the values for the input features and the target features are given for each example
- a set of **test examples**, where only the values for the input features are given

Predict the values for the target features for the test examples

- **Classification** when the Y_i are discrete
- **Regression** when the Y_i are continuous

Very important: keep training and test sets separate!
(see "N and P" agents slide)

Noise

- Data isn't perfect:
  - some of the features are assigned the **wrong value**
  - the features given are **inadequate** to predict the classification
  - there are examples with **missing features**

Noise

- Data isn't perfect:
  - some of the features are assigned the **wrong value**
  - the features given are **inadequate** to predict the classification
  - there are examples with **missing features**
- **Overfitting** occurs when a distinction appears in the data, but doesn't appear in the unseen examples
  This happens because of **random correlations** in the training set

Evaluating Predictions

Suppose Y is a feature and e is an example:

- $Y(e)$ is the value of feature Y for example e
- $\widehat{Y}(e)$ is the predicted value of feature Y for example e
- The **error** of the prediction is a measure of how close $\widehat{Y}(e)$ is to $Y(e)$
- There are many possible errors that could be measured

Measures of Error

E is the set of examples

$\mathbf{T}$ is the set of target features

- **absolute error**

$$\sum_{e \in E} \sum_{Y \in \mathbf{T}} \left| Y(e) - \widehat{Y}(e) \right|$$

Measures of Error

E is the set of examples

$\mathbf{T}$ is the set of target features

- **absolute error**

$$\sum_{e \in E} \sum_{Y \in \mathbf{T}} \left| Y(e) - \widehat{Y}(e) \right|$$

- **sum of squares error**

$$\sum_{e \in E} \sum_{Y \in \mathbf{T}} (Y(e) - \widehat{Y}(e))^2$$

Measures of Error

E is the set of examples

$\mathbf{T}$ is the set of target features

- **absolute error**

$$\sum_{e \in E} \sum_{Y \in \mathbf{T}} \left| Y(e) - \widehat{Y}(e) \right|$$

- **sum of squares error**

$$\sum_{e \in E} \sum_{Y \in \mathbf{T}} (Y(e) - \widehat{Y}(e))^2$$

- **worst-case error**

$$\max_{e \in E} \max_{Y \in \mathbf{T}} \left| Y(e) - \widehat{Y}(e) \right|$$

Measures of Error

E is the set of examples

$\mathbf{T}$ is the set of target features

- **absolute error**

$$\sum_{e \in E} \sum_{Y \in \mathbf{T}} \left| Y(e) - \widehat{Y}(e) \right|$$

- **sum of squares error**

$$\sum_{e \in E} \sum_{Y \in \mathbf{T}} \left(Y(e) - \widehat{Y}(e) \right)^2$$

- **worst-case error**

$$\max_{e \in E} \max_{Y \in \mathbf{T}} \left| Y(e) - \widehat{Y}(e) \right|$$

- A **cost-based error** takes into account costs of various errors

Precision and Recall

- Not all errors are equal, e.g. predict:
  - a patient has a disease when they do not
  - a patient doesn't have a disease when they do
- need to map out both kinds of errors to find the best trade-off

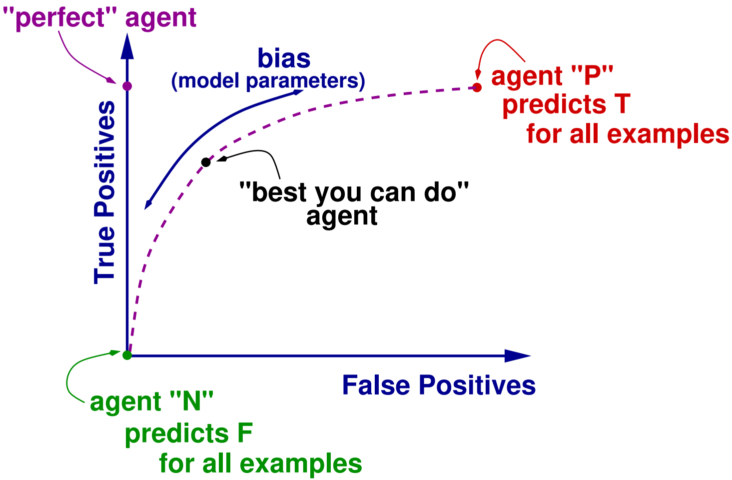
		predicted	
		T	F
actual	T	true positive (**TP**)	false negative (**FN**)
	F	false positive (**FP**)	true negative (**TN**)

- **recall** = **sensitivity** = TP/(TP+FN)
- **specificity** = TN/(TN+FP)
- **precision** = TP/(TP+FP)
- **F1-measure** =

$$\frac{2 \times \mathit{Precision} \times \mathit{Recall}}{\mathit{Precision} + \mathit{Recall}}$$

gives even weight to precision and recall

Receiver Operating Curve (ROC)



The ROC gives **full range of performance of an algorithm across different biases**

Basic Models for Supervised Learning

Many learning algorithms can be seen as deriving from:

- **decision trees**
- **linear classifiers** (generalizes to neural networks)
- **Bayesian classifiers**

Next

- Supervised learning: decision trees and learning strategies (Poole & Mackworth chapter 7.1-7.3.1, 7.4-7.4.1)