AI INSTITUTE #AIISC
UNIVERSITY OF SOUTH CAROLINA

UNIVERSITY OF SOUTH CAROLINA

CSCE 580: Introduction to AI
CSCE 581: Trusted AI

Lecture 14: Machine Learning – Classification Cont.

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

10TH OCT 2023

Carolinian Creed: "I will practice personal and academic integrity."

Credits: **Copyrights of all material reused acknowledged**

Organization of Lecture 14

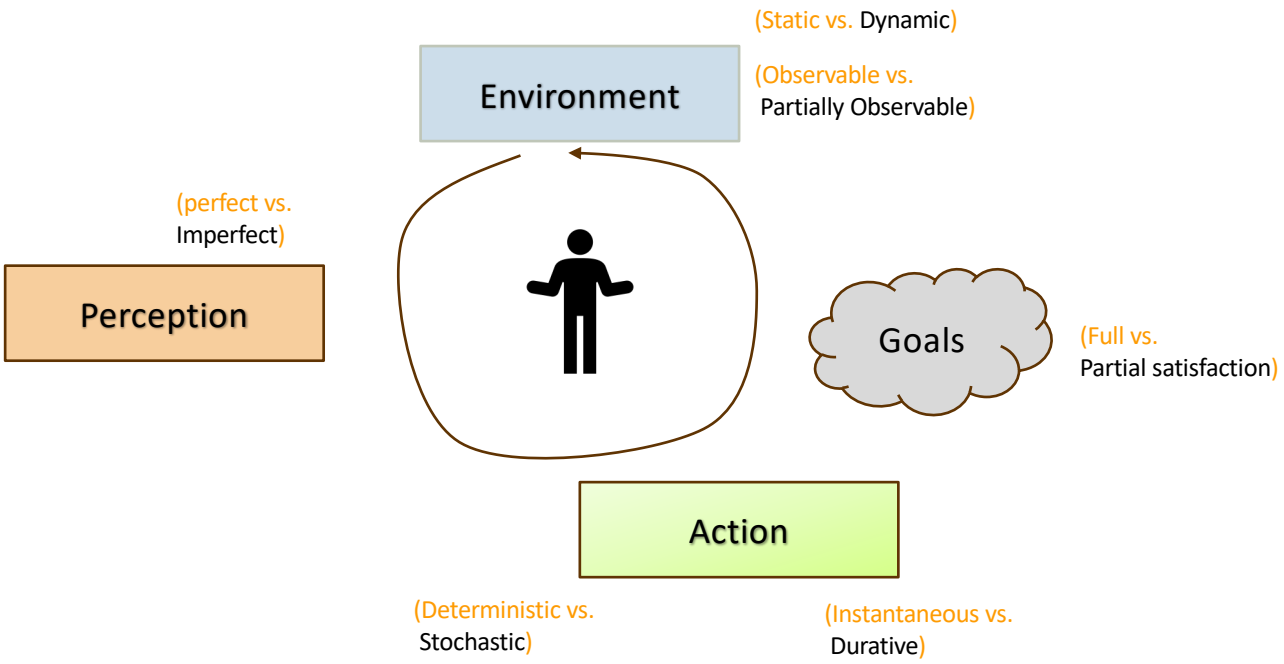
- Introduction Segment
  - Recap of Lecture 13
- Main Segment
  - Naïve Bayes
  - Boosting
  - Explanation
  - Discussion: reading material
    - Choosing a method that works
- Concluding Segment
  - Course Project Discussion
  - About Next Lecture – Lecture 15
  - Ask me anything

Introduction Section

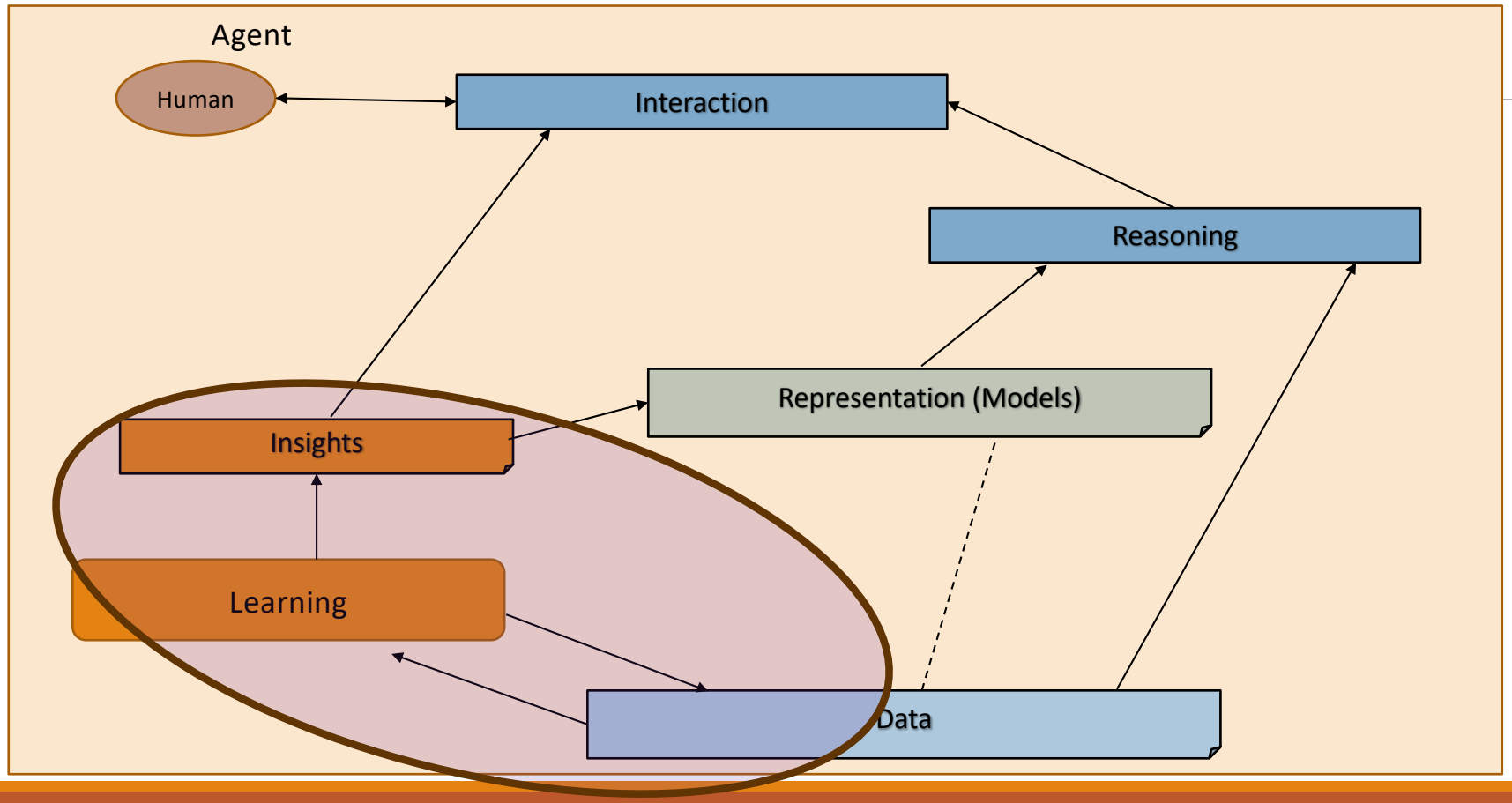
Recap of Lecture 13

- The variety of methods for classification
- Logistic Regression
- Decision trees
- Random forest

Intelligent Agent Model



Relationship Between Main AI Topics



Where We Are in the Course

CSCE 580/ 581 – In This Course

- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2-3: Data: Formats, Representation and the Trust Problem
- Week 4-5: Search, Heuristics - Decision Making
- Week 6: Constraints, Optimization – Decision Making
- Week 7: Classical Machine Learning – Decision Making, Explanation
- Week 8: Machine Learning - Classification
- Week 9: Machine Learning - Classification – Trust Issues and Mitigation Methods
- Topic 10: Learning neural network, deep learning, Adversarial attacks
- Week 11: Large Language Models – Representation, Issues
- Topic 12: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 13: Planning, Reinforcement Learning – Sequential decision making
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

Main Section

Credit: Retrieved from internet

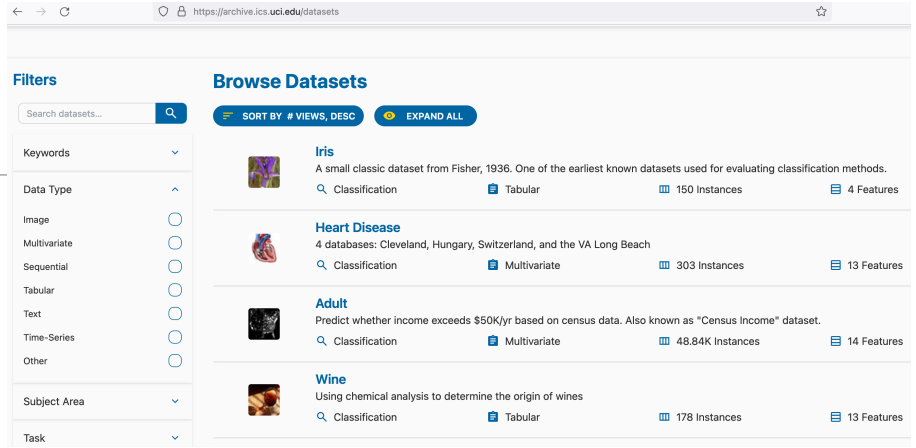
Machine Learning – Insights from Data

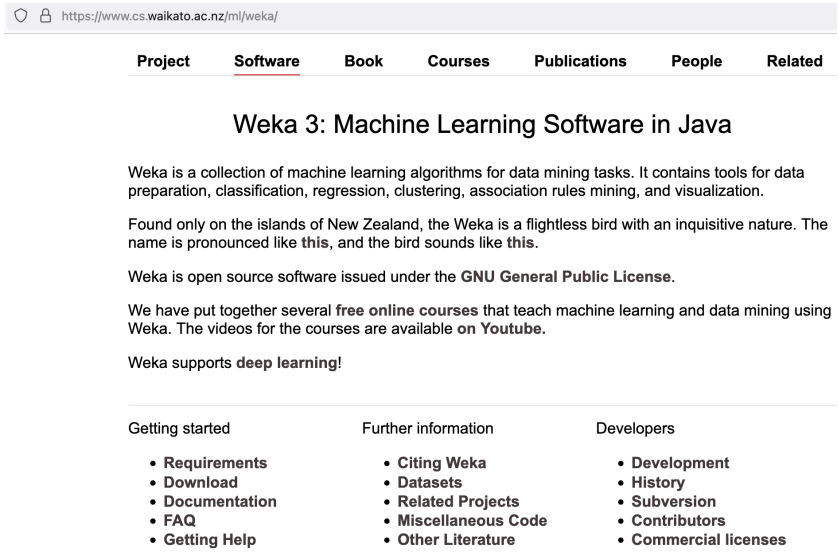
- Descriptive analysis
  - Describe a past phenomenon
  - **Methods**: classification (feedback from label), clustering, dimensionality reduction, anomaly detection, neural methods, reinforcement learning (feedback from hint/ reward)
- Predictive analysis
  - Predict about a new situation
  - **Methods**: time-series, neural networks
- Prescriptive analysis
  - What an agent should do
  - **Methods**: simulation, reinforcement learning, reasoning

- New areas
  - Counterfactual analysis
  - Causal Inferencing
  - Scenario planning

Reference and Demo

- Data: UCI Datasets
  - https://archive.ics.uci.edu/datasets
  - Browse or search

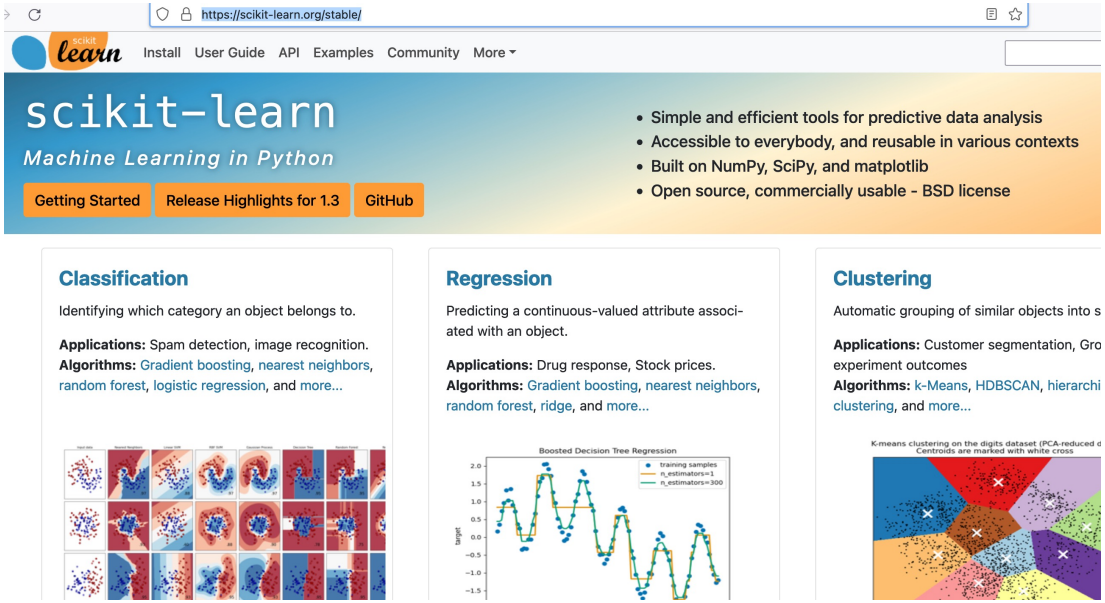




- Tools:
  - Weka - https://www.cs.waikato.ac.nz/ml/weka/
  - Download tool and dataset
- Libraries
  - Scikit - https://scikit-learn.org/stable/

Reference and Demo

- Data: UCI Datasets
  - https://archive.ics.uci.edu/datasets
  - Browse or search
- Tools:
  - Weka - https://www.cs.waikato.ac.nz/ml/weka/
  - Download tool and dataset
- Libraries
  - Scikit - https://scikit-learn.org/stable/



Classifier Method Types

- Individual methods
  - Logistic Regression
  - Decision Tree
  - Naïve Bayes
- Ensemble
  - Bagging: Aggregate classifiers ("bootstrap aggregation" => bagging)
    - Random Forest
    - Samples are chosen with replacement (bootstrapping), and combined (aggregated) by taking their average
  - Gradient Boosting: aggregate to turn weak learners into strong learners
    - Boosters (aggregators) turn weak learners into strong learners by focusing on where the individual weak models (decision trees, linear regressors) went wrong
    - Gradient Boosting

Source:

- Data Mining: Concepts and Techniques, by Jiawei Han and Micheline Kamber
- https://towardsdatascience.com/getting-started-with-xgboost-in-scikit-learn-f69f5f470a97

Random Forest in Action

- Code examples:
  - https://github.com/biplav-s/course-d2d-ai/blob/main/sample-code/l6-l7-l8-supervised-ml/Supervised-RandomForest-Classification.ipynb
  - Scikit Library: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Naïve Bayes Classifier

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Notation:
Class variable y and
dependent feature vector x_1 through x_n

Using the naive conditional independence assumption that

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y),$$

for all i, this relationship is simplified to

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Bayes assumption: given the value of the class variable, every pair of features are conditionally independent

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Source: https://scikit-learn.org/stable/modules/naive_bayes.html

Boosting Methods

- Concepts
  - **Weak learner**: a classifier that is only slightly correlated with the true classification
    - label examples better than random guessing
  - **Strong learner**: a classifier that is (arbitrarily) well-correlated with the true classification.
- Boosting
  - "Convert weak learners to strong learners"
  - Adapt[at]ive Resampling and Combining algorithm

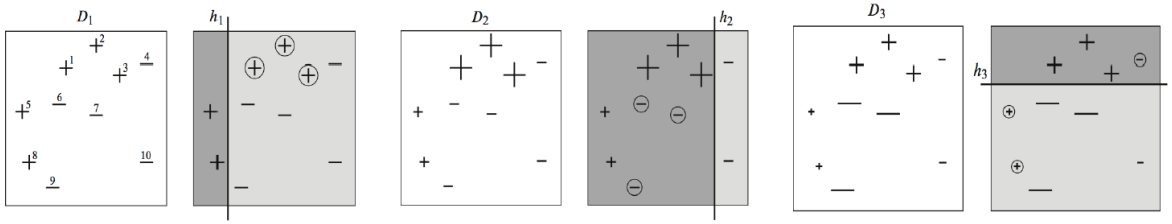


Figure: AdaBoost. Source: Figure 1.1 of [Schapire and Freund, 2012]

Source: https://en.wikipedia.org/wiki/Boosting_(machine_learning)

Image Courtesy: Prof. Cheng Li

Boosting Methods

Gradient Boosting = Gradient Descent + Boosting

Adaboost

$$H(x) = \sum_t \rho_t h_t(x)$$

- Fit an additive model (ensemble) $\sum_t \rho_t h_t(x)$ in a forward stage-wise manner.
- In each stage, introduce a weak learner to compensate the shortcomings of existing weak learners.
- In Adaboost, "shortcomings" are identified by high-weight data points.

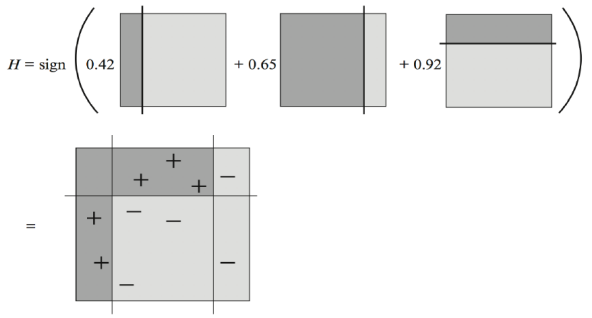


Figure: AdaBoost. Source: Figure 1.2 of [Schapire and Freund, 2012]

Content and Image Courtesy: Prof. Cheng Li
https://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

Boosting Methods

Gradient Boosting = Gradient Descent + Boosting

Adaboost

$$H(x) = \sum_t \rho_t h_t(x)$$

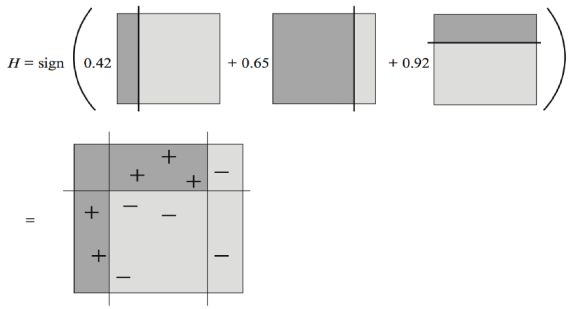


Figure: AdaBoost. Source: Figure 1.2 of [Schapire and Freund, 2012]

AdaBoost,
Illustration: for binary classification, images

1.Form a large set of simple features
2. Initialize weights for training images
3. For T rounds
    1. Normalize the weights
    2. For available features from the set, train a classifier using a single feature and evaluate the training error
    3. Choose the classifier with the lowest error
    4. Update the weights of the training images: increase if classified wrongly by this classifier, decrease if correctly
4. Form the final strong classifier as the linear combination of the T classifiers (coefficient larger if training error is small)

Source: https://en.wikipedia.org/wiki/Boosting_(machine_learning)

Image Courtesy: Prof. Cheng Li
https://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

Boosting Methods

Gradient Boosting = Gradient Descent + Boosting

Gradient Boosting

- Fit an additive model (ensemble) $\sum_t \rho_t h_t(x)$ in a forward stage-wise manner.
- In each stage, introduce a weak learner to compensate the shortcomings of existing weak learners.
- In Gradient Boosting, "shortcomings" are identified by gradients.
- Recall that, in Adaboost, "shortcomings" are identified by high-weight data points.
- Both high-weight data points and gradients tell us how to improve our model.

Gradient Boosting = Gradient Descent + Boosting

Adaboost

$$H(x) = \sum_t \rho_t h_t(x)$$

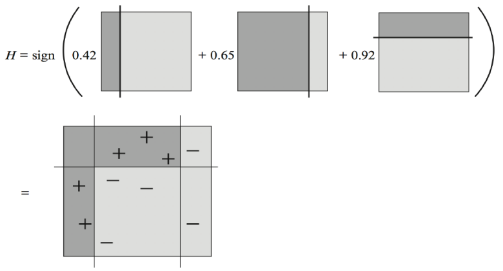


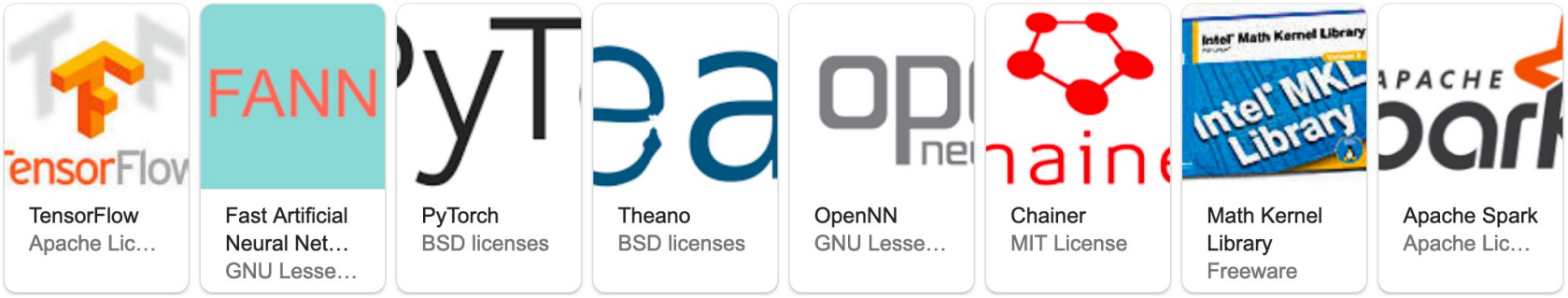
Figure: AdaBoost. Source: Figure 1.2 of [Schapire and Freund, 2012]

Content and Image Courtesy: Prof. Cheng Li
https://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

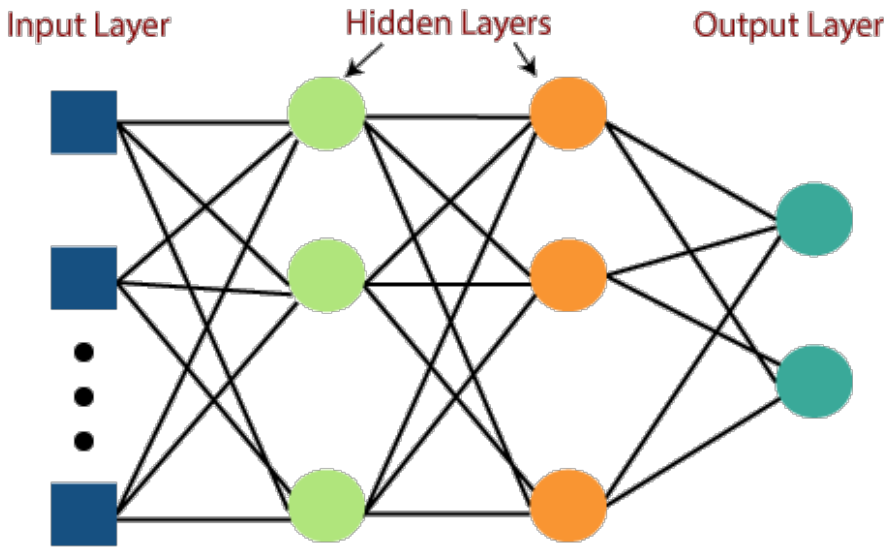
NBC and Boosting in Action

- Code examples:
  - https://github.com/biplav-s/course-d2d-ai/blob/main/sample-code/l6-l7-l8-supervised-ml/Supervised-NaiveBayes-GradientBoost-NN-Classification.ipynb
- Scikit Library:
  - NBC: https://scikit-learn.org/stable/modules/naive_bayes.html
  - GradientBoost: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

Neural Network Methods



NN – Multi Layer Perceptron



Content and Image Courtesy:
https://github.com/Thanasis1101/MLP-from-scratch

Logistic Regression in a Slide

Function estimate (linear)
W: weight, b: bias

$$f(X_j) = X_j W + b$$

Update Weight

$$W^* = W - \eta \frac{dL}{dW}$$

Error Term (mean squared error)

$$MSE = \frac{1}{n} \sum_{j=1}^{n} \left[f(X_{j\cdot}) - y_j \right]^2$$

Common Code Pattern

```
y = tf.matmul(x, W) + b
loss = tf.reduce_mean(tf.square(y - y_label))
```

Keras and TensorFlow

- By Example:
  - https://github.com/biplav-s/course-nl/blob/master/l9-ml-review/Basic%20TensorFlow%20and%20Keras.ipynb
- TensorFlow's NMIST tutorial
  - https://www.tensorflow.org/tutorials/quickstart/beginner
- More examples
  - Number Addition by sequence learning: https://keras.io/examples/nlp/addition_rnn/
  - AutoEncoder: https://machinelearningmastery.com/lstm-autoencoders/

NN/ MLP

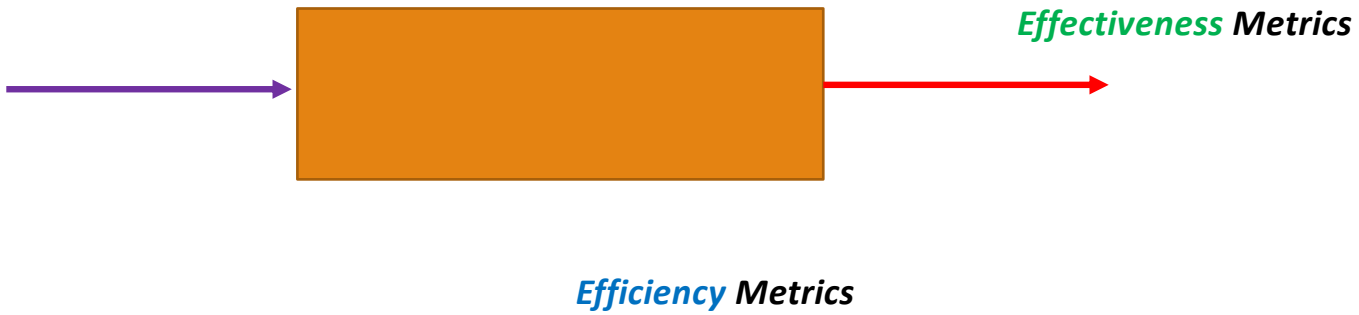
- Code examples:
  - https://github.com/biplav-s/course-d2d-ai/blob/main/sample-code/l6-l7-l8-supervised-ml/Supervised-NaiveBayes-GradientBoost-NN-Classification.ipynb
- Scikit Library:
  - MLP: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Activity: Try Weka and Classifiers

- Naïve Bayes Method
- Gradient Tree Boosting
- Neural Network – MLP

Metric Types

- **Effectiveness**: what the **user** of a system sees, primarily cares about
- **Efficiency**: what the **executor** in a system sees, primarily cares about



Metrics: Accuracy, Precision, Recall

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Accuracy =
(TP+TN)/
(TP+FP+FN+TN)

Precision =
(TP)/
(TP+FP)

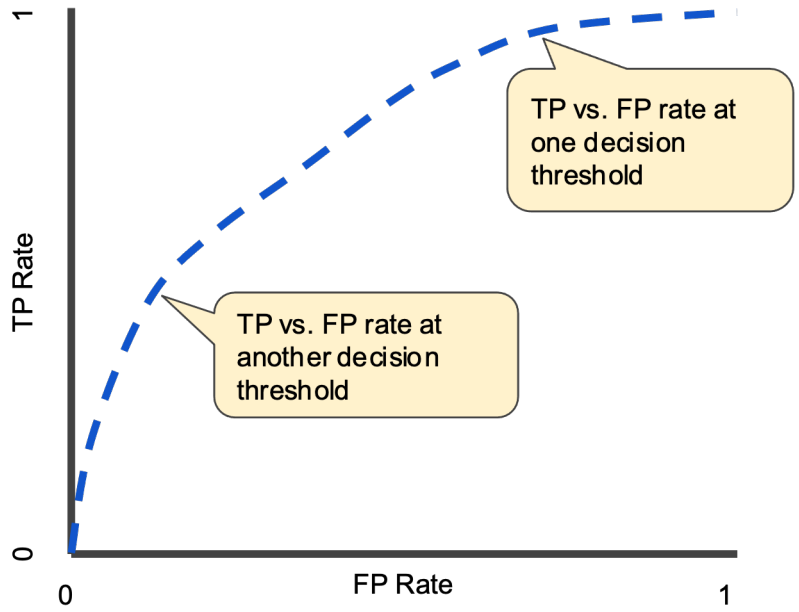
Recall =
(TP)/
(TP+FN)

F1 Score: *Harmonic Mean*

1/F1 = 1/Precision + 1/Recall

F1 = 2*(Recall * Precision) /
(Recall + Precision)

ROC – Receiver Operating Characteristic curve



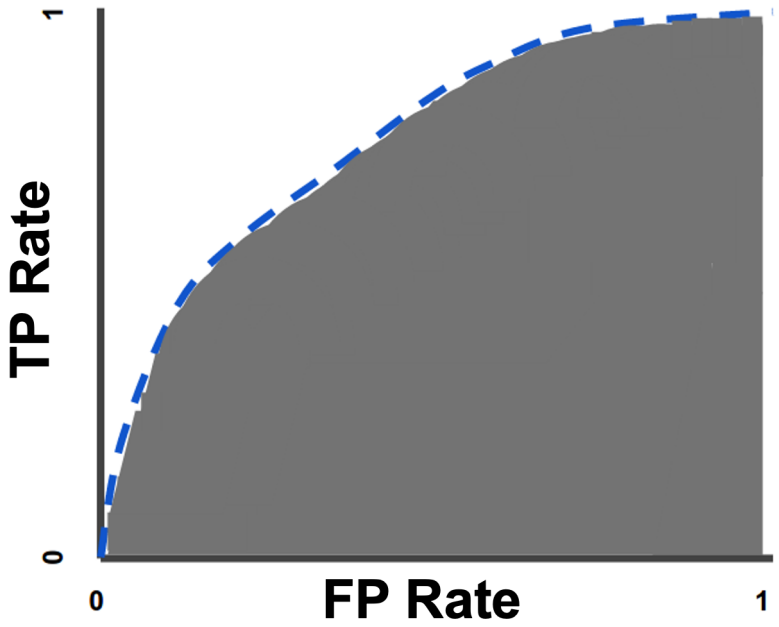
True Positive Rate = Recall =
(TP)/
(TP+FN)

False Positive Rate =
(FP)/
(FP+TN)

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Source: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

AUC – Area Under the ROC Curve



- Aggregate measure of performance across all possible classification thresholds.
- Interpretation: probability that the model ranks a random positive example more highly than a random negative example

Source: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

References

- Blogs: https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/
- Google: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Discussion: 10 Tips Paper

- Access: https://biodatamining.biomedcentral.com/articles/10.1186/s13040-017-0155-3
- Chicco, D. Ten quick tips for machine learning in computational biology. *BioData Mining* **10,** 35 (2017). https://doi.org/10.1186/s13040-017-0155-3

The Tips

- Tip 1: Check and arrange your input dataset properly
- Tip 2: Split your input dataset into three independent subsets (training set, validation set, test set), and use the test set only once you complete training and optimization phases
- Tip 3: Frame your biological problem into the right algorithm category
- Tip 4: Which algorithm should you choose to start? The simplest one!
- Tip 5: Take care of the imbalanced data problem
- Tip 6: Optimize each hyper-parameter
- Tip 7: Minimize overfitting
- Tip 8: Evaluate your algorithm performance with the Matthews correlation coefficient (MCC) or the Precision-Recall curve
- Tip 9: Program your software with open source code and platforms
- Tip 10: Ask for feedback and help to computer science experts, or to collaborative Q&A online communities

Course Project

Project Discussion: What Problem Fascinates You ?

- Data
  - Water
  - Finance
  - …
- Analytics
  - Search, Optimization, Learning, Planning, …
- Application
  - Building chatbot
- Users
  - Diverse demographics
  - Diverse abilities
  - Multiple human languages

Project execution in sprints

- Sprint 1: (Sep 12 – Oct 5)
  - Solving: Choose a decision problem, identify data, work on solution methods
  - Human interaction: Develop a basic chatbot (no AI), no problem focus
- Sprint 2: (Oct 10 – Nov 9)
  - Solving: Evaluate your solution on problem
  - Human interaction: Integrated your choice of chatbot (rule-based or learning-based) and methods
- Sprint 3: (Nov 14 – 30)
  - **Evaluation:** Comparison of your solver chatbot with an LLM-based alternative, like ChatGPT

Project Discussion: Dates and Deliverables

Project execution in sprints

- Sprint 1: (Sep 12 – Oct 5)
  - Solving: Choose a decision problem, identify data, work on solution methods
  - Human interaction: Develop a basic chatbot (no AI), no problem focus
- Sprint 2: (Oct 10 – Nov 9)
  - Solving: Evaluate your solution on problem
  - Human interaction: Integrated your choice of chatbot (rule-based or learning-based) and methods
- Sprint 3: (Nov 14 – 30)
  - **Evaluation:** Comparison of your solver chatbot with an LLM-based alternative, like ChatGPT

- Oct 12, 2023
  - Project checkpoint
  - In-class presentation
- Nov 30, 2023
  - Project report due
- Dec 5 / 7, 2023
  - In-class presentation

Skeleton: A Basic Chatbot

- Run in an infinite loop until the user wants to quit
- Handle any user response
  - User can quit by typing “Quit” or “quit” or just “q”
  - User can enter any other text and the program has to handle it. The program should write back what the user entered and say – “I do not know this information”.
- Handle known user query types // Depends on your project
  - “Tell me about N-queens”, “What is N ?”
  - “Solve for N=4?”
  - “Why is this a solution? ”
- Handle chitchat // Support at least 5, extensible from a file
  - “Hi” => **“Hello”**
  - …
- ***Store session details in a file***

Illustrative Project

1. **Title**: Solve and explain solving of n-queens puzzle
2. **Key idea**: Show students how a course project will look like
3. **Who will care when done**: students of the course, prospective AI students and teachers
4. **Data need**: n: the size of game; interaction
5. **Methods**: search
6. **Evaluation**: correctness of solution, quality of explanation, appropriateness of chat
7. **Users**: with and without AI background; with and without chess background
8. **Trust issue**: user may not believe in the solution, may find interaction offensive (why queens, not kings? …)

Project Discussion: Illustration

1. Create a private Github repository called "CSCE58x-Fall2023-<studentname>-Repo". Share with Instructor (biplav-s) and TA (kausik-l)
2. Create Google folder called "CSCE58x-Fall2023-<studentname>-SharedInfo". Share with Instructor (prof.biplav@gmail.com) and TA (lakkarajukausik90@gmail.com)
3. Create a Google doc in your Google repo called "Project Plan" and have the following by next class (Sep 5, 2023)

1. **Title**: Solve and explain solving of n-queens puzzle
2. **Key idea**: Show students how a course project will look like
3. **Who will care when done**: students of the course, prospective AI students and teachers
4. **Data need**: n: the size of game; interaction
5. **Methods**: search
6. **Evaluation**: correctness of solution, quality of explanation, appropriateness of chat
7. **Users**: with and without AI background; with and without chess background
8. **Trust issue**: user may not believe in the solution, may find interaction offensive (why queens, not kings? …)

Project Illustration: N-Queens

- Sprint 1: (Sep 12 – Oct 5)
  - Solving: Choose a decision problem, identify data, work on solution methods
    - Method 1: Random solution
    - Method 2: Search – BFS
    - Method 3: Search - …
  - Human interaction: Develop a basic chatbot (no AI) as outlined
  - Deliverable
    - Code structure in Github
      - ./data
      - ./code
      - ./docs
      - ./test
    - Presentation: Make sprint presentation on Oct 12, 2023

Reference: Project Rubric

- **Project results** – 60%
  - Working system ? – 30%
  - Evaluation with results superior to baseline? – 20%
  - Considered related work? – 10%
- **Project effort**s – 40%
  - Project report – 20%
  - Project presentation (updates, final) – 20%
- **Bonus**
  - Challenge level of problem – 10%
  - Instructor discretion – 10%
- **Penalty**
  - Lack of timeliness as per announced policy (right) - up to 30%

Milestones and **Penalties**

- Oct 12, 2023
  - Project checkpoint
  - In-class presentation
  - **Penalty: presentation not ready by Oct 10, 2023 [-10%]**
- Nov 30, 2023
  - Project report due
  - **Project report not ready by date [-10%]**
- Dec 5 / 7, 2023
  - In-class presentation
  - **Project presentations not ready by Dec 4, 2023 [-10%]**

<Project Title> - <Your Name>

Format for Interim Presentation on Oct 12, 2023

Project Context

1. Problem
2. Who will care/ users
3. Data needs:
4. Methods:
5. Evaluation:
6. Trust issue:

Achievement

- Status
- Test Case
  - E.g., <input, correct output>
- Sample Result
- Discuss others points:
  - Challenges faced
  - Any help needed

1 min context, 1 min achievement, 1 min Q/A

Lecture 13: Summary

- We talked about
  - Decision trees/ random forest
  - The variety of methods
  - Choosing a method that works

Concluding Section

About Next Lecture – Lecture 14

Lecture 15: Student Project Presentations

- Put presentations in Google drive by end of day
- Stick to 1+1+1 minutes rule while presenting