UCLA Samueli Computer Science

CS145 Discussion: Week 2

Decision Tree, Nearest Neighbors, ML Pipeline, Programming Prep

Junheng Hao

Friday, 01/15/2021

Roadmap

- Announcement
- Lecture Review
- Programming Prep for Problem Sets

Announcements

- 5:00pm PST, Jan. 15: Weekly quiz 2 released on Gradescope.
- **11:59pm PST, Jan. 17 (Sunday):** Weekly quiz 2 closed on Gradescope!
  - Start the quiz before **11:00pm PST, Jan. 17** to have the full 60-minute time
- 5:00pm, Jan. 15: Problem set 1 released on campuswire/CCLE, submission on Gradescope.
  - Please assign pages of your submission with corresponding problem set outline items on GradeScope.
  - You do not need to submit code, only the results required by the problem set
  - Due on **11:59pm PST, Jan. 29 (Friday)**
- There is no class on **Jan. 18 (Monday)**, in observance of Martin Luther King Jr. Day.

About Quiz 2

- Quiz release date and time: **Jan 08, 2021 (Friday) 05:00 PM PST**
- Quiz due/close date and time: **Jan 10, 2021 (Sunday) 11:59 PM PST**
- You will have up to **60 minutes** to take this exam. → Start before **11:00 PM** Sunday
- You can find the exam entry named "Week 2 Quiz" on GradeScope.
- Topics: Decision Tree, Nearest Neighbors, General machine learning basics and pipeline
- Question Types
  - True/false, multiple choices, and auto-graded short answers (fill blanks)
  - Some questions may include several subquestions.
- Some light calculations are expected. Some scratch paper and one scientific calculator (physical or online) are recommended for preparation.
- More Info: https://campuswire.com/c/GB5E561C3/feed/57

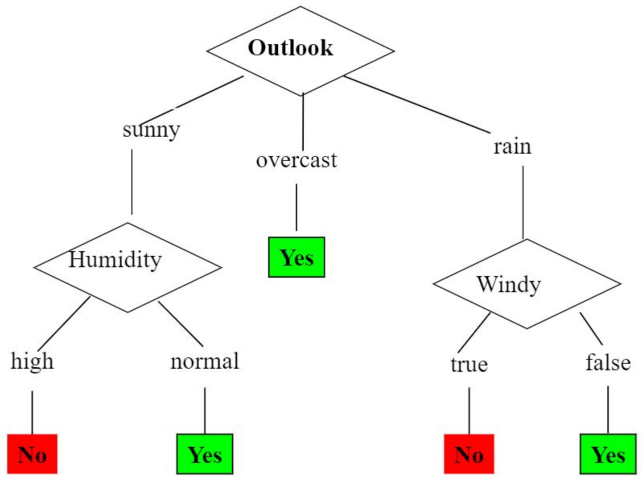
Lecture Review

Decision Tree, Nearest Neighbors, ML Pipelines

Decision Tree

- Decision Tree Classification: From data to model

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No



Decision Tree: Takeaway

- Choosing the Splitting Attribute
- At each node, available attributes are evaluated on the basis of separating the classes of the training examples.
- A goodness function (information measurement) is used for this purpose:
  - **Information Gain**
  - Gain Ratio*
  - Gini Index*

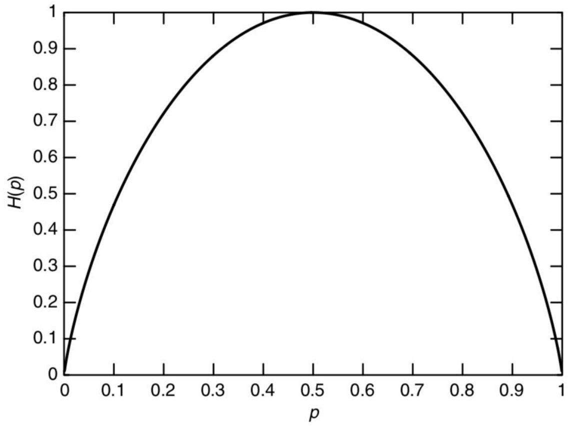
Decision Tree: Attribute Selection

- Which is the best attribute?
  - The one which will result in the smallest tree
  - Heuristic: choose the attribute that produces the “purest” nodes
- Popular *impurity criterion*: *information gain*
  - Information gain increases with the average purity of the subsets that an attribute produces
- Strategy: choose attribute that results in greatest information gain

Decision Tree: Entropy of Random Variable

$$X = \begin{cases} 1 & \text{with probability} \quad p \\ 0 & \text{with probability} \quad 1-p \end{cases}$$

$$H(X) = -p \log p - (1-p)\log(1-p) \stackrel{\text{def}}{=} H(p)$$



Decision Tree: Attribute Selection

- Information in a split with ***x*** items of one class, ***y*** items of the second class

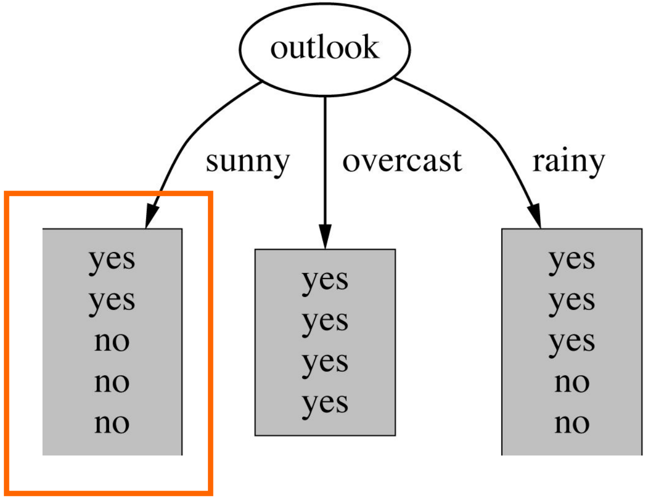
$$\mathrm{info}([\mathrm{x},\mathrm{y}]) = \mathrm{entropy}\left(\frac{\mathrm{x}}{\mathrm{x}+\mathrm{y}}, \frac{\mathrm{y}}{\mathrm{x}+\mathrm{y}}\right)$$

$$= -\frac{x}{x+y}\log\left(\frac{x}{x+y}\right) - \frac{y}{x+y}\log\left(\frac{y}{x+y}\right)$$

Attribute: “Outlook” = “Sunny”

- “Outlook” = “Sunny”: 2 and 3 split

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -\frac{2}{5}\log(\frac{2}{5}) - \frac{3}{5}\log(\frac{3}{5}) = 0.971 \text{ bits}$$

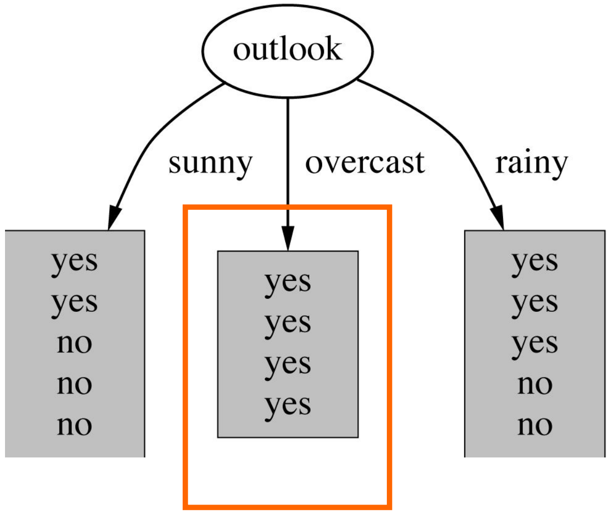


Attribute: “Outlook” = “Overcast”

- “Outlook” = “Overcast”: 4/0 split

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1\log(1)\boxed{-0\log(0)} = 0 \text{ bits}$$

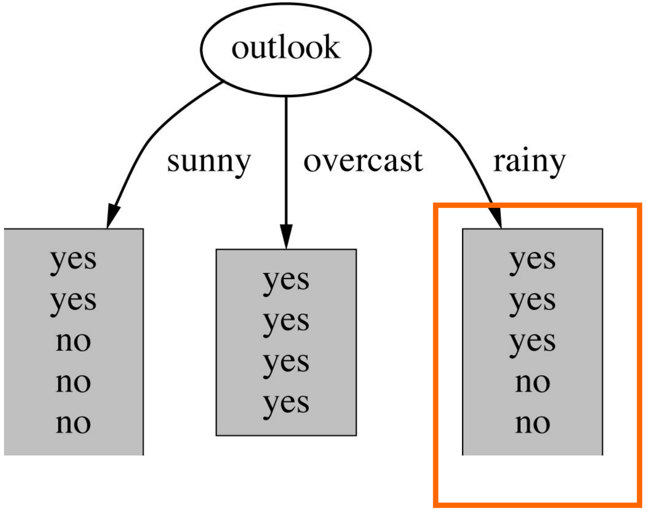
Note: log(0) is not defined, but we evaluate 0*log(0) as zero.



Attribute: "Outlook" = "Rainy"

- "Outlook" = "Rainy":

$$\text{info}([3,2]) = \text{entropy}(3/5,2/5) = -\frac{3}{5}\log(\frac{3}{5}) - \frac{2}{5}\log(\frac{2}{5}) = 0.971\ \text{bits}$$



Expected Information of Attribute “Outlook”

Expected information for attribute:

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$

$$= 0.693 \text{ bits}$$

Compute Information Gain

Information gain:

(information before split) – (information after split)

$$\text{gain}(\text{"Outlook"}) = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) = 0.940 - 0.693$$
$$= 0.247 \text{ bits}$$

Information gain for attributes from all weather data:

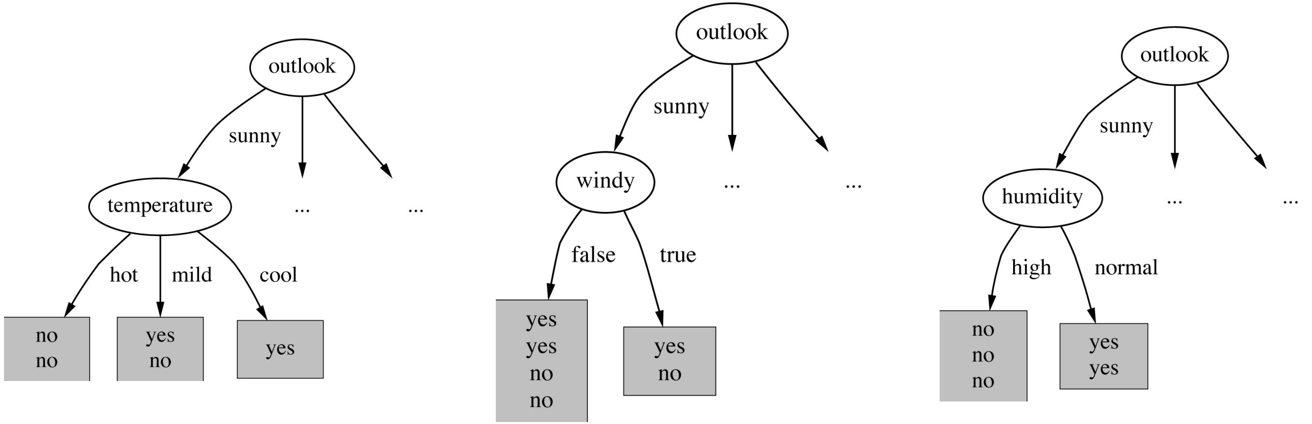
$$\text{gain}(\text{"Outlook"}) = 0.247 \text{ bits}$$

$$\text{gain}(\text{"Temperature"}) = 0.029 \text{ bits}$$

$$\text{gain}(\text{"Humidity"}) = 0.152 \text{ bits}$$

$$\text{gain}(\text{"Windy"}) = 0.048 \text{ bits}$$

Continue to Split



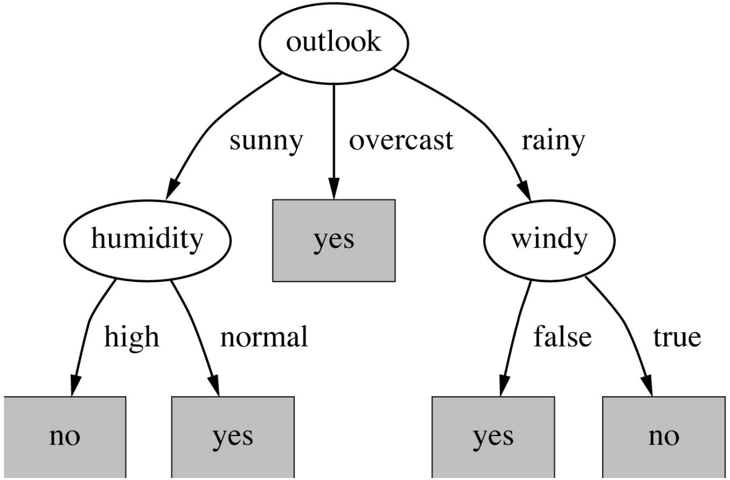
gain("Temperature") = 0.571 bits

gain("Windy") = 0.020 bits

gain("Humidity") = 0.971 bits

Decision Tree: Example for Practice

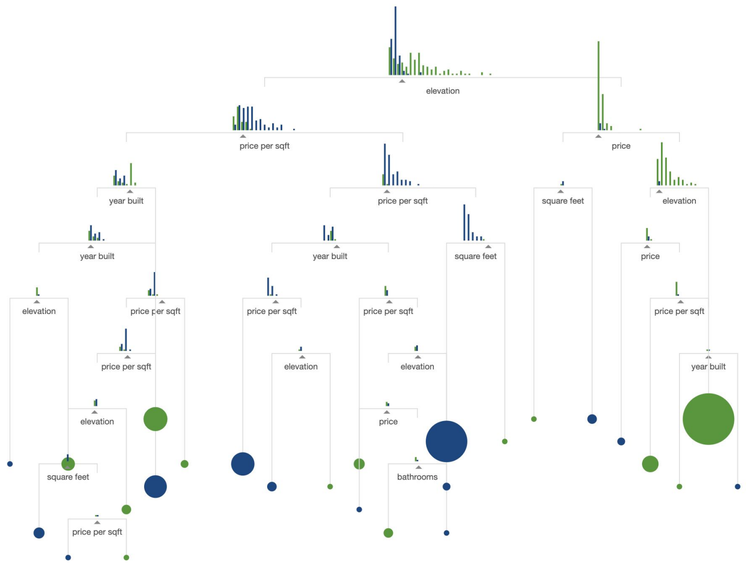
Final Tree



- Note: Not all leaves need to be pure. Sometimes identical instances have different classes.
- Splitting can stop when data can't be split any further.

Decision Tree: Visual Tutorials

- Demo links
  - http://www.r2d3.us/visual-intro-to-machine-learning-part-1/
  - http://explained.ai/decision-tree-viz/

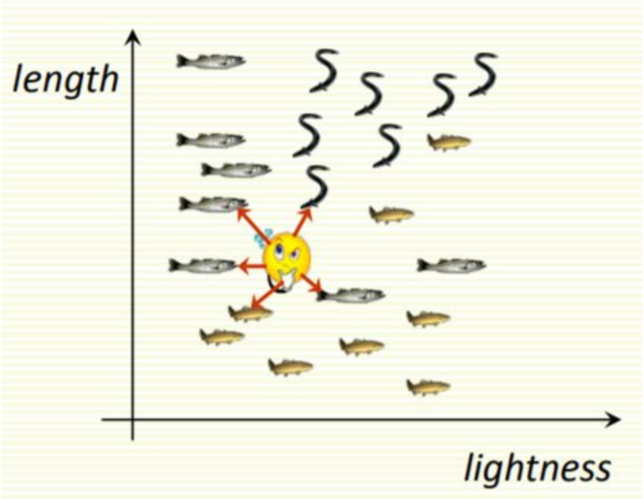


KNN

- Classify an unknown example with the most common class among K nearest examples
  - “Tell me who your neighbors are, and I’ll tell you who you are”
- Example
  - K = 3
  - 2 sea bass, 1 salmon
  - Classify as sea bass

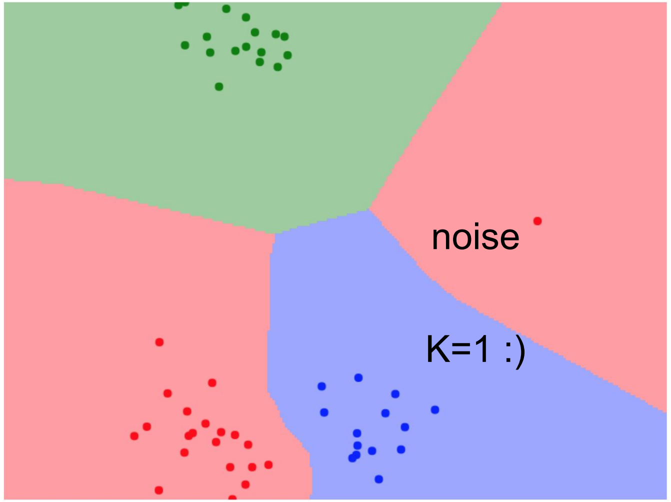
KNN: Multiple Classes

- Easy to implement for multiple classes
- Example for K = 5
  - 3 fish species: salmon, sea bass, eel
  - 3 sea bass, 1 eel, 1 salmon → classify as sea bass

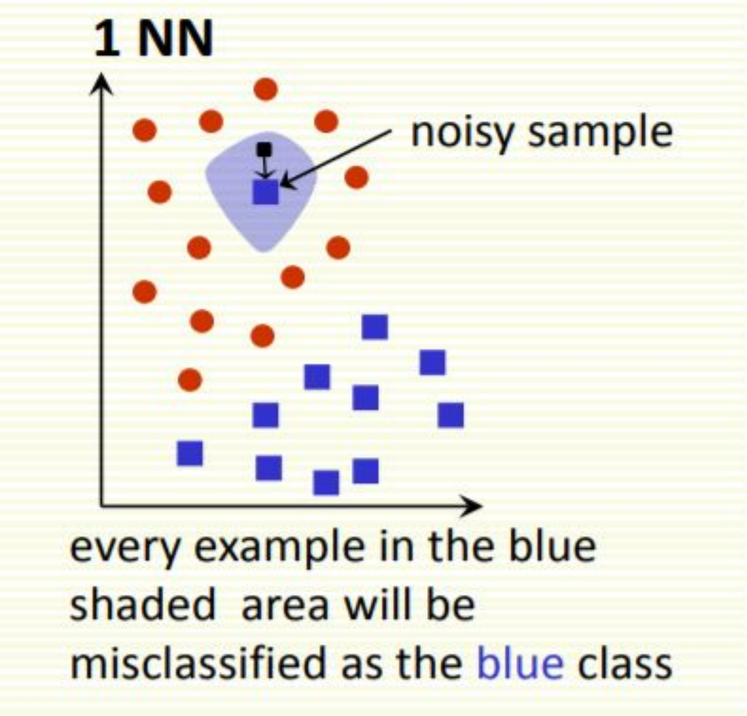


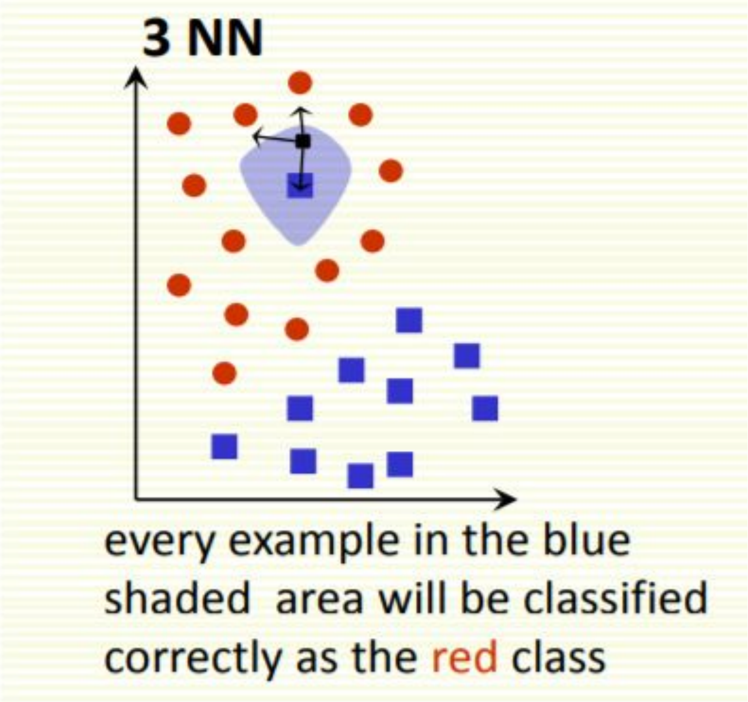
KNN: How to Choose K?

- In theory, if infinite number of samples available, the larger K, the better classification result you'll get.
- Caveat: all K neighbors have to be close
  - Possible when infinite # samples available
  - Impossible in practice since # samples if finite
- Should we "tune" K on training data?
  - Underfitting → Overfitting
- K = 1 → sensitive to "noise" (e.g. see right)



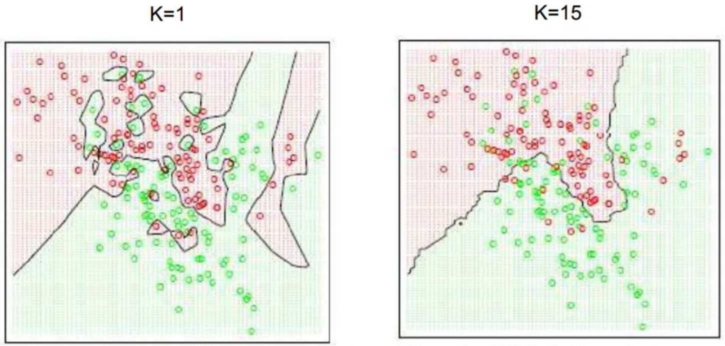
KNN: How to Choose K?



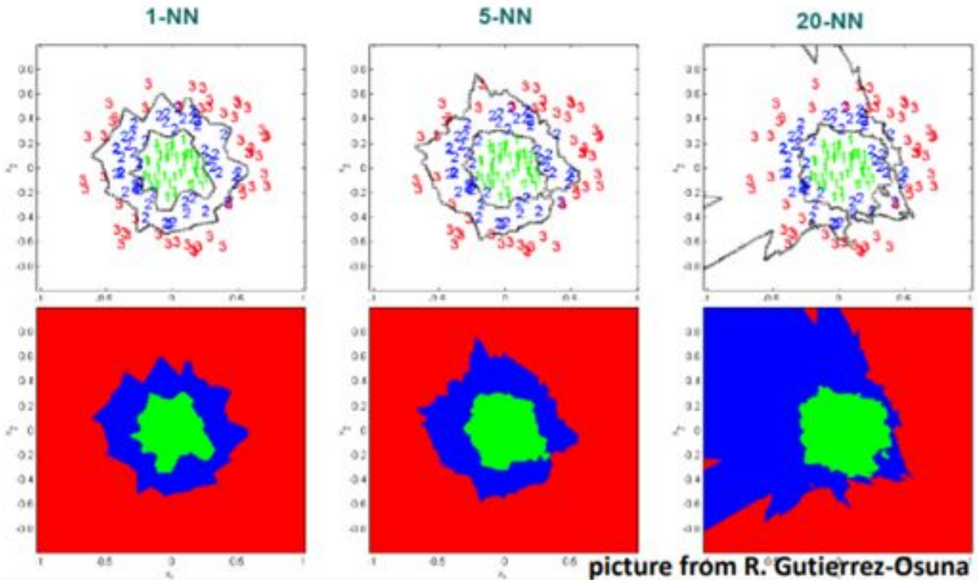


KNN: How to Choose K?

- Larger K gives smoother boundaries, better for generalization
    - Only if locality is preserved
    - K too large → looking at samples too far away that are not from the same class
- Can choose K through cross-validation



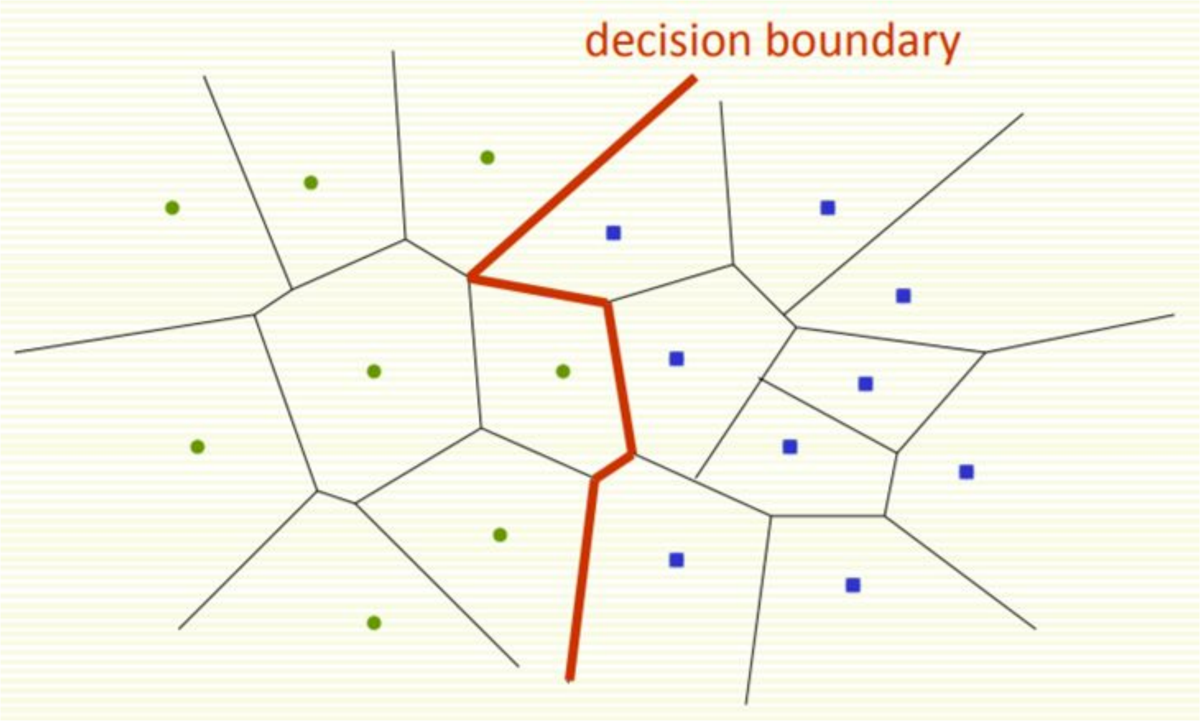
Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)



picture from R. Gutierrez-Osuna

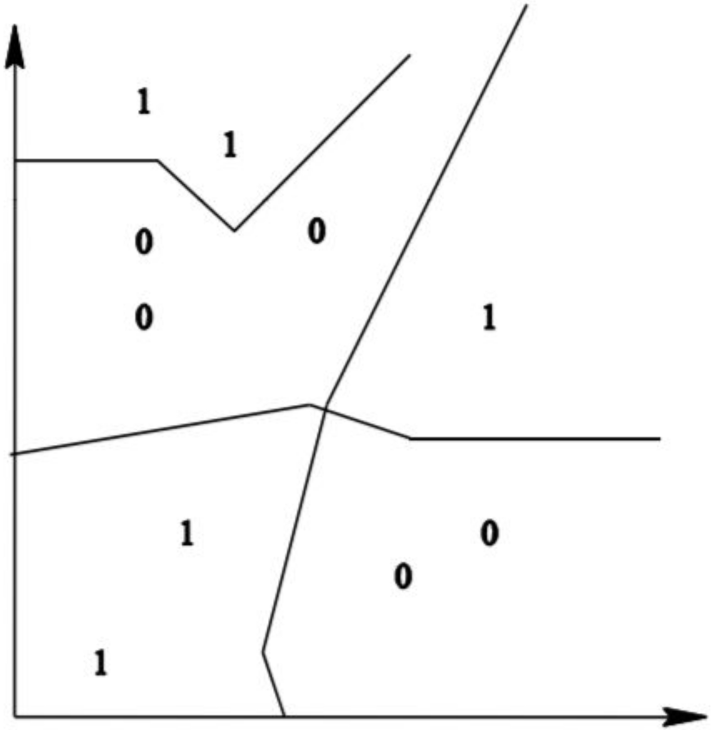
KNN: Decision Boundary

- Voronoi diagram



KNN: Decision Boundary

- Decision boundaries are formed by a subset of the Voronoi Diagram of the training data
- Each line segment is equidistant between two points of opposite class
- The more examples that are stored, the more fragmented and complex the decision boundaries can be.



KNN: Distance

- If we use Euclidean Distance to find the nearest neighbor:

$$D(a,b) = \sqrt{\sum_k (a_k - b_k)^2}$$

- Euclidean distance treats each feature as equally important
- Sometimes, some features (or dimensions) may be much more discriminative than other features

KNN: Distance

- Feature 1 gives the correct class: 1 or 2
- Feature 2 gives irrelevant number from 100 to 200
- Dataset: [1, 150], [2, 110]
- Classify [1, 100]

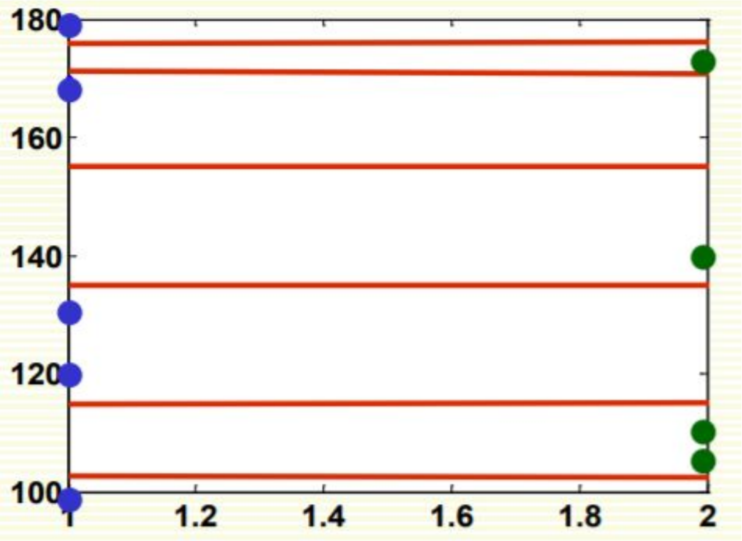
$$D\left(\begin{bmatrix}1\\100\end{bmatrix},\begin{bmatrix}1\\150\end{bmatrix}\right)=\sqrt{(1-1)^2+(100-150)^2}=50$$

$$D\left(\begin{bmatrix}1\\100\end{bmatrix},\begin{bmatrix}2\\110\end{bmatrix}\right)=\sqrt{(1-2)^2+(100-110)^2}=10.5$$

- Use Euclidean distance can result in wrong classification
- Dense Example can help solve this problem

KNN: Distance

- Decision boundary is in red, and is really wrong because:
    - Feature 1 is discriminative, but its scale is small
    - Feature gives no class information but its scale is large, which dominates distance calculation



KNN: Feature Normalization

- Normalize features that makes them be in the same scale
- Different normalization approaches may reflect the result
- Linear scale the feature in range [0,1]:

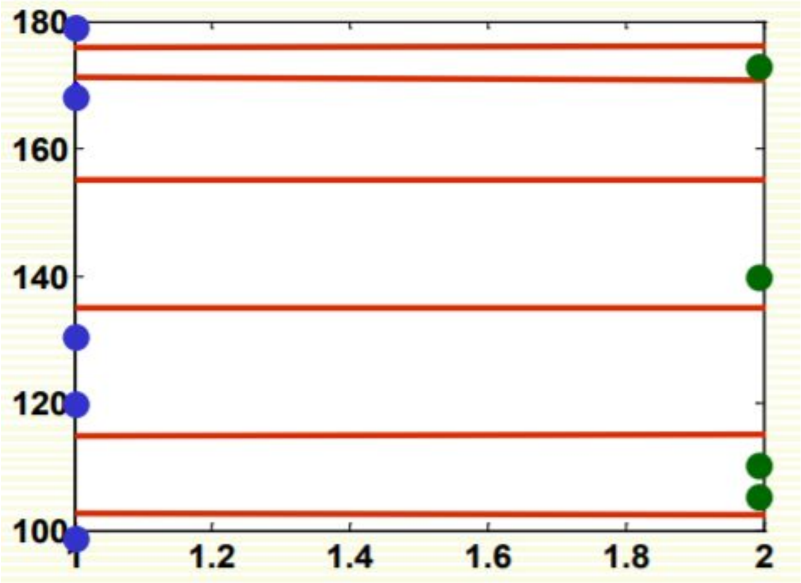
$$f_{new} = \frac{f_{\text{old}} - f_{\text{old}}^{\text{min}}}{f_{\text{old}}^{max} - f_{\text{old}}^{\text{min}}}$$

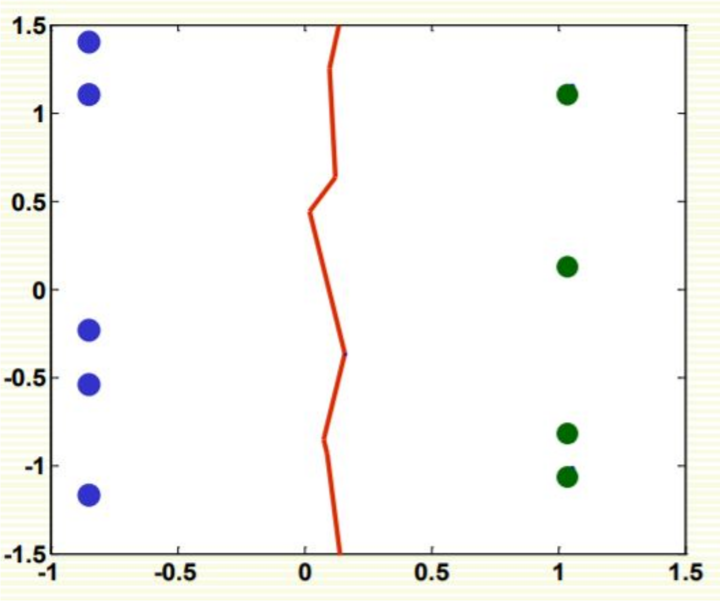
- Linear scale to 0 mean standard deviation 1(Z-score):

$$f_{new} = \frac{f_{\text{old}} - \mu}{\sigma}$$

KNN: Feature Normalization

- Result comparison non-normalized vs normalized





KNN: Feature Weighting

- Scale each feature by its importance for classification

$$D(a,b) = \sqrt{\sum_k w_k (a_k - b_k)^2}$$

- Use prior/domain knowledge to set the weight w
- Use cross-validation to learn the weight w

KNN: Computational Complexity

- Suppose ***n*** examples with dimension ***d***
- Complexity for kNN training?
- Complexity for kNN training?
  - For each point to be classified:
  - Complexity for computing distance to one example
  - Complexity for computing distances to all examples
  - Find ***k*** closest examples
- Is it expensive for a large number of queries?

KNN: Summary

- Advantages:
  - Can be applied to the data from any distribution
  - The decision boundary is not necessarily to be linear
  - Simple and Intuitive
  - Good Classification with large number of samples
- Disadvantages:
  - Choosing k may be tricky
  - Test stage is computationally expensive
    - No training stage, time-consuming test stage
    - Usually we can afford long training step but fast testing speed
  - Need large number of examples for accuracy

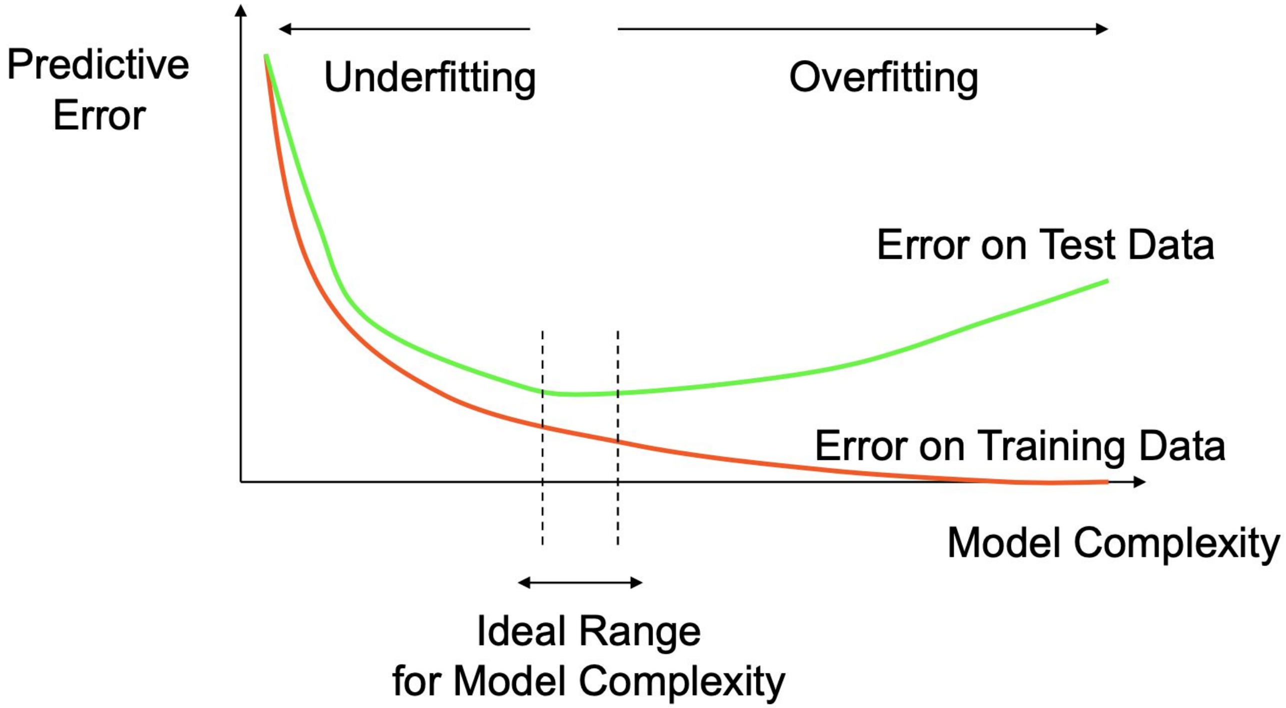
ML Pipeline: Evaluation Your Models

- Content

Cross Validation

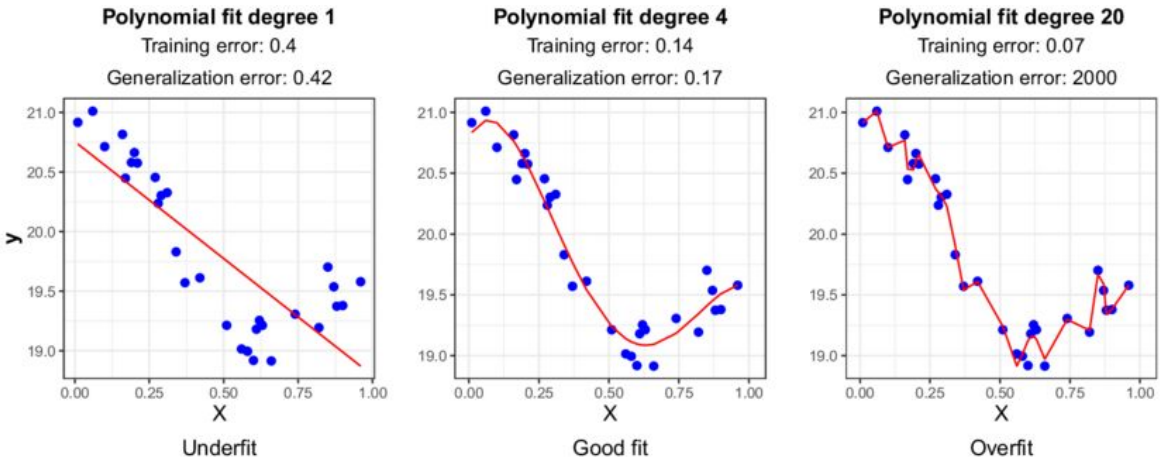
- Content

Analyze Your Model: Underfit or Overfit?



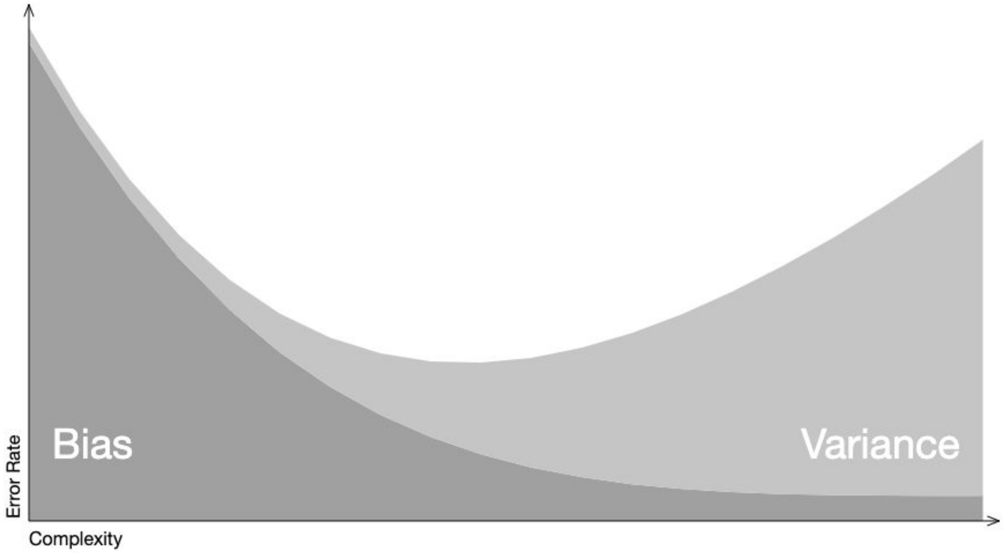
Analyze Your Model: Underfit or Overfit?

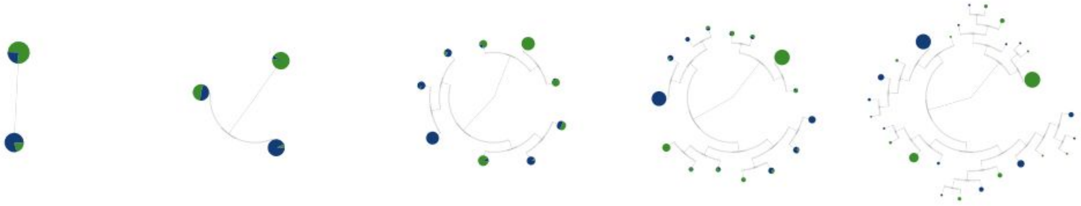
- Another example on regression



Analyze Decision Tree: Too simple or too complex?

- Examples on Decision Tree
- Another two concepts: Model Bias & Variance
- Demo: [Link]





Programming Prep Guide

“Do it local”: Python & Jupyter Notebook

- **Step 1:** Install Anaconda (with Python 3.X and Jupyter Notebooks)
- **Step 2:** Try out Python in command line and open Jupyter Notebooks
- **Step 3:** Familiarize yourself with Python 3
- **Step 4:** Use Jupyter Notebooks for coding and writing together
- **Step 5:** Customize your Python environment and install Python packages
    - Example packages: Numpy, Pandas, Matplotlib

Note: This slide is only intended for students who want to program on local desktop instead of Google Colab.

Where is your Python?

- Install Conda/Anaconda
  - Conda: https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html
  - Anaconda: https://docs.anaconda.com/anaconda/install/mac-os/
- Install Jupyter Notebook from anaconda (this step may be skipped once Anaconda is installed)
  - Link: https://jupyter.org/install
  - Command Line: conda install -c conda-forge notebook
- Check out Python and Jupyter notebook
  - Command Line: python or ipython
  - Version/Source: python --version or which python
  - Open Jupyter Notebook: jupyter notebook (automatically into something URL like: http://localhost:8888/tree)

Note: This slide is only intended for students who want to program on local desktop instead of Google Colab.

Create customized Python environment

- Checklist:
  - Create a customized virtual environment
  - Activate/Deactivate your environment
  - Install packages for your virtual environment
- Helpful links:
  - Managing conda environment: https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

A Program Notebook: Write and Code

- Apply both on Jupyter Notebook and Google Colab!

- Checklist:
  - Identify **Markdown** cell and **Code** cell
  - Learn how to use markdown and latex to input math formula
  - Run Python code

- Markdown tutorial → *It is a notebook interface!*
  - Checklist: paragraph, bold, italic, list, code (courier), math formula (in latex)
  - Link: https://www.markdowntutorial.com/

- Latex → *It is for typing math symbols and equations!*
  - No need to install Tex or Mactex
  - Cheatsheet: http://tug.ctan.org/info/undergradmath/undergradmath.pdf

Checklist: Python, Numpy, Pandas, Matplotlib

- Shown in the demo
- Python
  - Data types and control flow
- Numpy
  - Array and matrix
  - Matrix operation
  - Broadcasting
- Pandas
  - Data load and export
  - Dataframe operations
- Matplotlib
  - Plot types, settings and output figure files
- Scikit-learn
  - ML pipeline (data prep, model selection, train and development, evaluation)

Demo [Link]

- **Google Colab: A starter guide**
  - Create and connect online codebook
  - Run code and commands
  - Save and output results
- **Text cell**
  - Markdown and Latex
- **Code cell**
  - Python
  - Numpy
  - Pandas
  - Matplotlib

Thank you!

Q & A