Programming Principles in Python (CSCI 503)

Machine Learning

Dr. David Koop
Grammar of Graphics & Altair

- "Grammar of Graphics", L. Wilkinson
- "A Layered Grammar of Graphics" + ggplot, H. Wickham
- Vega: "Declarative language for creating, saving, and sharing interactive visualization designs"
- Vega-Lite: higher-level language than Vega, carefully crafted rules for defaults
- Altair: Python interface to Vega-Lite (J. VanderPlas)
  - "spend more time understanding your data and its meaning"
  - Specify the what, minimize the amount of code directing the how
  - Python can write JSON specification just as well as any other language
  - Bindings make it more Python-friendly, integrate with pandas, add support for Jupyter, etc.
Basic Example

• import altair as alt
  import pandas as pd
  data = pd.DataFrame({'x': [1,3,4,6,10],'y': [1,5,2,7,3]})
  alt.Chart(data).mark_line().encode(x='x', y='y')

• Easiest to use data from a pandas data frame
  - Another option is a csv or json file
  - Can support geo_interface, too

• Chart is the basic unit

• Mark: .mark_*() indicates the geometry created for each data item

• Encode: .encode() allows visual properties to be set to data attributes
Visual Marks

- **Marks** are the basic graphical elements in a visualization
- Marks classified by dimensionality:
  - Points
  - Lines
  - Areas

- Also can have surfaces, volumes
- Think of marks as a mathematical definition, or if familiar with tools like Adobe Illustrator or Inkscape, the path & point definitions
- Altair: area, bar, circle, geoshape, image, line, point, rect, rule, square, text, tick
  - Also compound marks: boxplot, errorband, errorbar
Encode via Visual Channels

- **Position**
  - Horizontal
  - Vertical
  - Both

- **Color**

- **Shape**

- **Tilt**

- **Size**
  - Length
  - Area
  - Volume

[Munzner (ill. Maguire), 2014]
Data Attributes and Altair Types

- Categorical
  - Symbols
  - +, ●, ■, △

- Ordered
  - Ordinal
  - Symbols
  - ™, ™, ™

- Quantitative
  - Symbols
  - —, —, —, —

[Munzner (ill. Maguire), 2014]
Data Attributes and Altair Types

- Categorical data = Nominal (N)
- Ordinal data = Ordinal (O)
- Quantitative data = Quantitative (Q)
- Temporal data = Temporal (T)

[Munzner (ill. Maguire), 2014]
Different Channels for Different Attribute Types

**Magnitude Channels: Ordered Attributes**
- Position on common scale
- Position on unaligned scale
- Length (1D size)
- Tilt/angle
- Area (2D size)
- Depth (3D position)
- Color luminance
- Color saturation
- Curvature
- Volume (3D size)

**Identity Channels: Categorical Attributes**
- Spatial region
- Color hue
- Motion
- Shape

Altair will use its rules to pick whether to use color hue or saturation based on the type.

[Altair (ill. Maguire), 2014]
Altair Supports Concatenation, Layering, & Repetition

- **Layering:**
  - + Operator

- **Concatenation:**
  - Horizontal: | operator
  - Vertical: & operator

- **Repetition**
  - Use of .repeat for layout
  - Reference repeated variables in the encoding
Layering
Concatenation

[Improvise, Weaver, 2004]
Repetition
Seattle Weather: 2012-2015

Date
Jan 01 Feb 01 Mar 01 Apr 01 May 01 Jun 01 Jul 01 Aug 01 Sep 01 Oct 01 Nov 01 Dec 01

Maximum Daily Temperature (°C)
-5 0 5 10 15 20 25 30 35 40

Count of Records
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470 480 490 500 510 520 530 540 550 560 570 580 590 600 610 620 630 640 650 660 670 680 690 700 710 720 730 740 750 760 770 780 790 800

Weather
- sun
- fog
- drizzle
- rain
- snow

Precipitation
- 0
- 10
- 20
- 30
- 40
- 50
Weather Selection: Rain vs. Sun

Seattle Weather: 2012-2015

D. Koop, CSCI 503, Spring 2021
Date Selection: July-September Sun
• Back to Pokémon Data
• Calculate MaxCP in pandas and find highest per generation
• Analyze attack, defense, and speed by primary type and generation using visualizations created with matplotlib and altair
Final Exam

- Monday, April 26, 2:00-3:50pm, Online (Blackboard)
- **More** comprehensive than Test 2
- Expect questions from topics covered on Test 1 and 2
- Expect questions from the last three weeks of class (data, visualization, machine learning)
- Similar format
Machine Learning Intro
Tasks Machine Learning can Help With

- Identifying the zip code from handwritten digits on an envelope

- Detecting fraudulent activity in credit card transactions
- Identifying topics in a set of blog posts
- Grouping customers with similar preferences

[A. Müller & S. Guido, Introduction to Machine Learning with Python, J. Steppan (MNIST image)]
Questions when building a machine learning solution

• What question(s) am I trying to answer? Do I think the data collected can answer that question?
• What is the best way to phrase my question(s) as a machine learning problem?
• Have I collected enough data to represent the problem I want to solve?
• What features of the data did I extract, and will these enable the right predictions?
• How will I measure success in my application?
Machine Learning

- Traditional Programming

  Data → Computer → Output
  Program → Computer

- Machine Learning

  Data → Computer → Program
  Output → Computer
Machine Learning

- Every machine learning algorithm has three components:
  - Representation
  - Evaluation
  - Optimization
Representation

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.
Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- Etc.
Optimization

• Combinatorial optimization
  - E.g.: Greedy search
• Convex optimization
  - E.g.: Gradient descent
• Constrained optimization
  - E.g.: Linear programming
Types of Learning

• Supervised (inductive) learning
  - Training data includes desired outputs

• Unsupervised learning
  - Training data does not include desired outputs

• Semi-supervised learning
  - Training data includes a few desired outputs

• Reinforcement learning
  - Rewards from sequence of actions
Areas of Machine Learning

• Supervised learning
  - Decision tree induction
  - Rule induction
  - Instance-based learning
  - Bayesian learning
  - Neural networks
  - Support vector machines
  - Model ensembles
  - Learning theory

• Unsupervised learning
  - Clustering
  - Dimensionality reduction
Supervised & Unsupervised Tasks

- Identifying the zip code from handwritten digits on an envelope (supervised)
  
  ![MNIST Image]

- Detecting fraudulent activity in credit card transactions (supervised)
- Identifying topics in a set of blog posts (unsupervised)
- Grouping customers with similar preferences (unsupervised)

[A. Müller & S. Guido, Introduction to Machine Learning with Python, J. Steppan (MNIST image)]
Supervised Learning
Supervised Learning: Learned Algorithm (Fit)
Supervised Learning: Prediction
Supervised Learning: Prediction
Unsupervised Learning: Input
Unsupervised Learning: Output
Scikit-Learn

- Started as a Google Summer of Code project! (D. Cournapeau, 2007)
- Rewritten by scientists at INRIA (France) in 2010
- Written in Python using numpy, some optimizations using C (cython)
- The "gold standard" for machine learning in python
scikit-learn Principles

• Consistency: all objects share consistent, documented interface
• Inspection: parameters and parameter values determined by learning algorithms are stored and exposed as public attributes
• Non-proliferation of classes: only learning algs are classes, not datasets or parameters; easier to combine with other libraries
• Composition: create and reuse building blocks
• Sensible defaults: user-defined parameters should have meaningful defaults
scikit-learn entities

- Data: numpy matrices (also pandas series, data frames), process batches
- Estimators: all supervised & unsupervised algs implement common interface
  - estimator initialization does not do learning, only attaches parameters
  - fit does the learning, learned parameters exposed with trailing underscore
- Predictor: extends estimator with predict method
  - also provides score method to return value indicating prediction quality
- Transformer: help modify or filter data before learning
  - Preprocessing, feature selection, feature extraction, and dimensionality reduction vis transform method
  - Can combine fit and transform via fit_transform

[L. Buitinck et al.]
Penguin Example
Deep Learning

• Deep learning is tied to neural networks, attempting to mimic how human neurons work together
• Hierarchical with multiple layers
• Usually takes advantage of GPUs
• Frameworks:
  - pytorch
  - TensorFlow
  - keras
  - theano