Introduction to Machine Learning

Tan Bui-Thanh *

22nd August 2023

*Department of Aerospace Engineering and Engineering Mechanics, The Oden Institute for Computational Engineering and Sciences, UT Austin, Austin, Texas (tanbui@oden.utexas.edu, https://users.oden.utexas.edu/~tanbui/).

CONTENTS

CHAPTER 1

LECTURE 1: INTRODUCTION

1.1 Welcome to Applied Machine Learning!

Machine learning is one of today's most exciting emerging technologies.

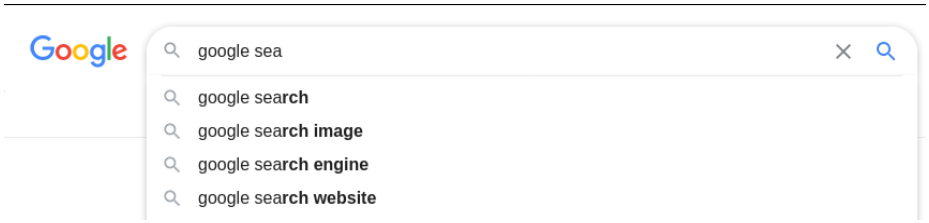
In this course, you will learn what machine learning is, what are the most important techniques in machine learning, and how to apply them to solve problems in the real world.

1.2 What is Machine Learning?

We hear a lot about machine learning (or ML for short) in the news. But what is it, really?

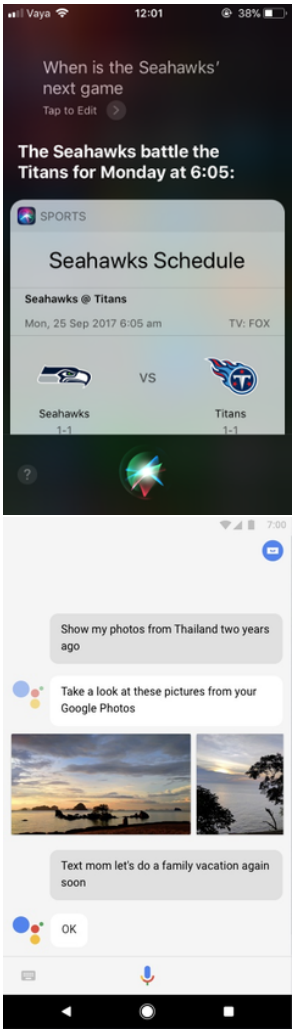
1.2.1 ML in Everyday Life: Search Engines

You use machine learninng every day when you run a search engine query.



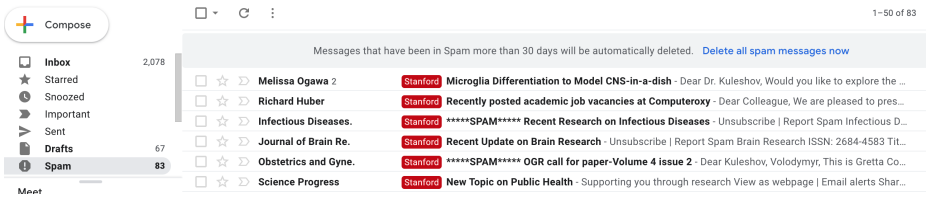
1.2.2 ML in Everyday Life: Personal Assistants

Machine learning also powers the speech recognition, question answering and other intelligent capabilities of smartphone assistants like Apple Siri.



1.2.3 ML in Everyday Life: Spam/Fraud Detection

Machine learning is used in every spam filter, such as in Gmail.



ML systems are also used by credit card companies and banks to automatically detect fraudulent behavior.

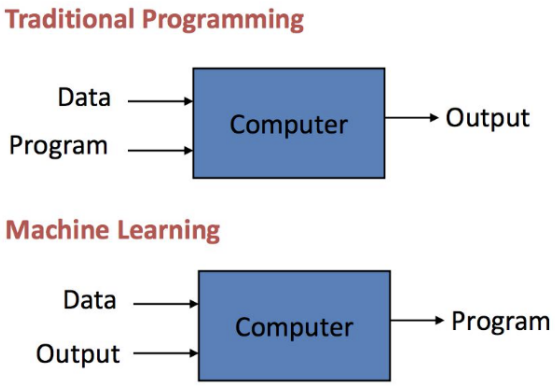
1.2.4 ML in Everyday Life: Self-Driving Cars

One of the most exciting and cutting-edge uses of machine learning algorithms are in autonomous vehicles.

1.3 A Definition of Machine Learning

In 1959, Arthur Samuel defined machine learning as follows.

> *Machine learning is a field of study that gives computers the ability to learn without being explicitly programmed.*



For example, in traditional programming we would like to calculate the addition rule

$$y = \textbf{sum}\ (x_1, x_2)$$

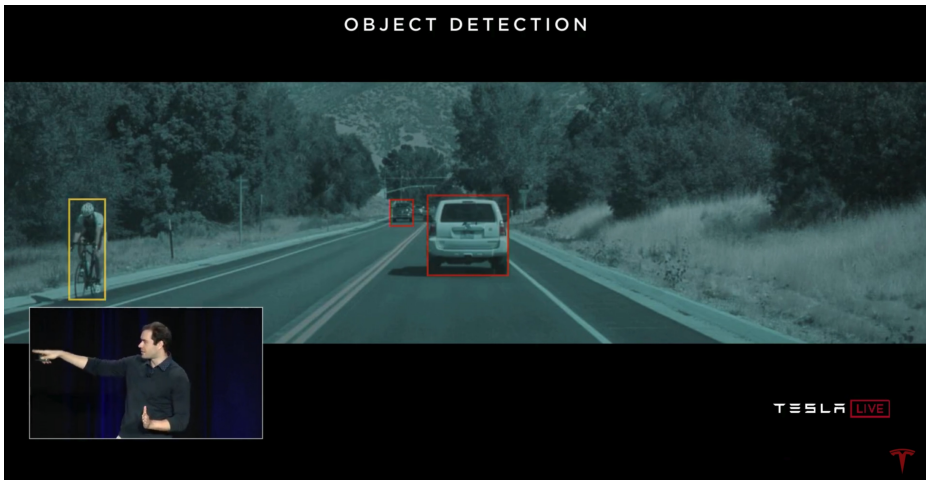
we simply use a calculator or call the functions to implement

$$y = x_1 + x_2$$
$$\textbf{return}\ y$$

On the other hand, assume that we do not know the addition rule, however, we have some pairs of $\left\{x_1^i, x_2^i, y^i\right\}_{i=1}^N$, we can use machine learning to learn/approximate the underlying rule (addition rule).

What does “learn” and “explicitly programmed” mean here? Let’s look at an example of Self Driving Cars

A self-driving car system uses dozens of components that include detection of cars, pedestrians, and other objects.



1.3.1 Self Driving Cars: A Rule-Based Algorithm

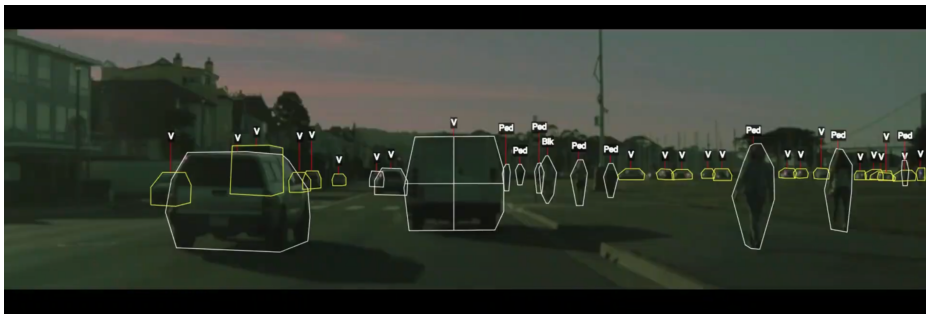
One way to build a detection system is to write down rules.

[ ]:
```
# pseudocode example for a rule-based classification system
object = camera.get_object()
if object.has_wheels(): # does the object have wheels?
    if len(object.wheels) == 4: return "Car" # four wheels => car
    elif len(object.wheels) == 2:,
        if object.seen_from_back():
            return "Car" # viewed from back, car has 2 wheels
        else:
            return "Bicycle" # normally, 2 wheels => bicycle
return "Unknown" # no wheels? we don't know what it is
```

In practice it takes too much time to account for all rules. Even when it is possible, the program is too complicated. Furthermore, it’s almost impossible for a human to specify all the edge cases. So what are we going to do?

1.3.2 Self Driving Cars: An ML Approach

The machine learning approach is to teach a computer how to do detection by showing it many examples of different objects, just like how we learn.



No manual programming is needed: the computer learns to identify a pedestrian or a car on its own! Just like a human does. This principle can be applied to countless domains: medical diagnosis, factory automation, machine translation, and many more!

Note that the output of a machine learning algorithm is a program that try to learns the rule/law in the data.

1.3.3 Why Machine Learning?

Why is this approach to building software interesting?

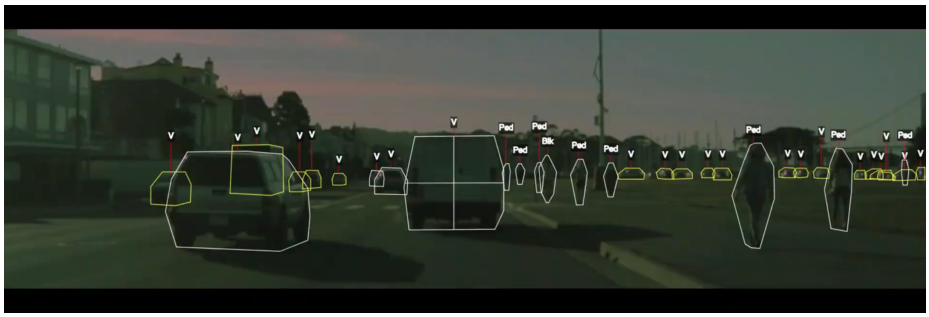
- It allows building practical systems for real-world applications that couldn't be solved otherwise. For example, for application in which we do not know its governing physics/equations: all we have is data!
- Learning is wildly regarded as a key approach towards building general-purpose artificial intelligence systems.
- As we have discussed, machine learning mimics human learning. Thus, the science and engineering of machine learning offers insights into human intelligence.

1.4 Three Main Approaches to Machine Learning

We have distinguished traditional programming and machine learning. W would like to emphasize that the output of a machine learning algorithm is a program that tries to learning the underlying rule in the data. We now discuss three main approaches for machine learning. This provides us the first peek into how the learning process is carried out within a particular machine learning approach.

1.4.1 Supervised Learning

The most common approach to machine learning is supervised learning. The self-driving car example, shown again here, is an example of supervised learning.



A supervised learning approach consists of three steps (will be made precise later when we get to supervised learning):

1. First, we collect a dataset of labeled training examples $\{x_i, y_i\}_{i=1}^N$ with N being the total number of data, where x_i is the image and y_i is the label (e.g. vehicle, pedestrian, etc).
2. We train a model to output accurate predictions on this dataset.
3. When the model sees new, but similar data, it will also be accurate.

A Supervised Learning Dataset

Consider a simple dataset for supervised learning: house prices in Boston.

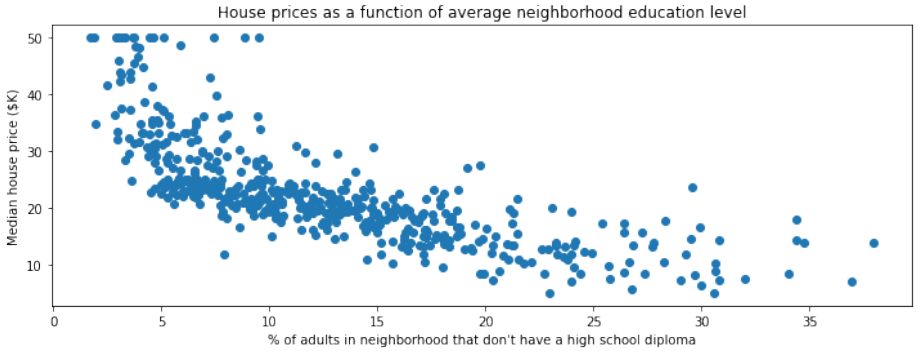
- Each datapoint is a house.
- We know its price, neighborhood, size, etc.

[ ]:
```
# We will load the dataset from the sklearn ML library
from sklearn import datasets
boston = datasets.load_boston()
```

We will visualize two variables in this dataset: house price and the education level in the neighborhood.

[ ]:
```
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = [12, 4]
plt.scatter(boston.data[:,12], boston.target)
plt.ylabel("Median house price ($K)")
plt.xlabel("% of adults in neighborhood that don't have a high school diploma")
plt.title("House prices as a function of average neighborhood education level")
```

[ ]:
```
Text(0.5, 1.0, 'House prices as a function of average neighborhood education
level')
```



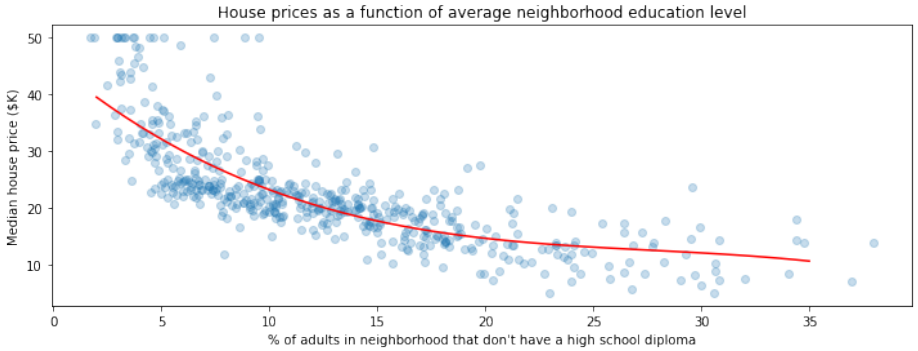
A Supervised Learning Algorithm

We can use this dataset of examples to fit a supervised learning model.

- The model maps input x (the education level) to output a y (the house price).
- It learns the mapping from our dataset of examples (x, y).

[ ]:
```
import numpy as np
from sklearn.kernel_ridge import KernelRidge

# Apply a supervised learning algorithm
model = KernelRidge(alpha=1, kernel='poly')
model.fit(boston.data[:,[12]], boston.target.flatten())
predictions = model.predict(np.linspace(2, 35)[:, np.newaxis])

# Visualize the results
plt.scatter(boston.data[:,[12]], boston.target, alpha=0.25)
plt.plot(np.linspace(2, 35), predictions, c='red')
plt.ylabel("Median house price ($K)")
plt.xlabel("% of adults in neighborhood that don't have a high school diploma")
plt.title("House prices as a function of average neighborhood education level")
```

[ ]:
```
Text(0.5, 1.0, 'House prices as a function of average neighborhood education
level')
```



Applications of Supervised Learning

Many of the most important applications of machine learning are supervised:

- Classifying medical images.
- Translating between pairs of languages.
- Detecting objects in a self-driving car.

1.4.2 Unsupervised Learning

Here, we have a dataset *without* labels, that is, we only have $\{x_i\}_{i=1}^N$. Our goal is to learn/identify/detect something interesting about the structure of the data:

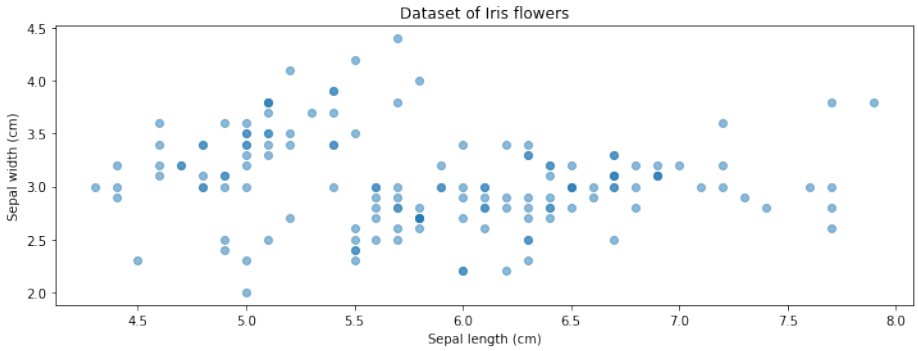
- Clusters hidden in the dataset.
- Outliers: particularly unusual and/or interesting datapoints.
- Useful signal hidden in noise, e.g. human speech over a noisy phone.

An Unsupervised Learning Dataset

Here is a simple example of an unsupervised learning dataset: Iris flowers.

[ ]:
```
# Load and visualize the Iris flower dataset
iris = datasets.load_iris()
plt.scatter(iris.data[:,0], iris.data[:,1], alpha=0.5)
plt.ylabel("Sepal width (cm)")
plt.xlabel("Sepal length (cm)")
plt.title("Dataset of Iris flowers")
```

[ ]:
```
Text(0.5, 1.0, 'Dataset of Iris flowers')
```



An Unsupervised Learning Algorithm

We can use this dataset of examples to fit an unsupervised learning model.

- The model defines a probability distribution over the inputs.
- The probability distribution identifies multiple components (multiple peaks).

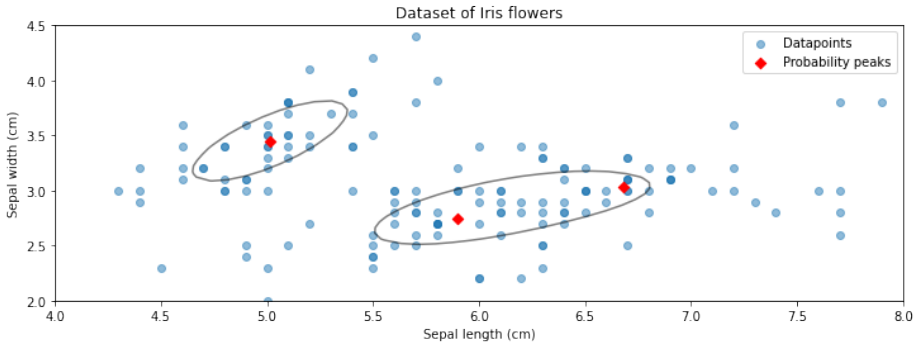
- The components indicate structure in the data.

[ ]:
```
# fit a Gaussian Mixture Model with three components
from sklearn import mixture
model = mixture.GaussianMixture(n_components=3, covariance_type='full')
model.fit(iris.data[:,[0,1]])
```

[ ]:
```
GaussianMixture(n_components=3)
```

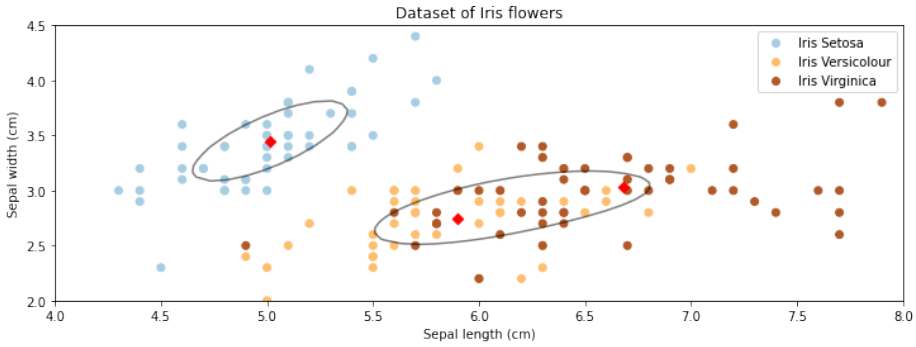
[ ]:
```
# display learned probabilities as a contour plot
x, y = np.linspace(4.0, 8.0), np.linspace(2.0, 4.5)
X, Y = np.meshgrid(x, y)
Z = -model.score_samples(np.array([X.ravel(), Y.ravel()]).T).reshape(X.shape)
plt.contour(X, Y, Z, levels=np.logspace(0, 10, 1), cmap="gray", alpha=0.5)
plt.scatter(iris.data[:,0], iris.data[:,1], alpha=0.5)
plt.scatter(model.means_[:,0], model.means_[:,1], marker='D', c='r')
plt.ylabel("Sepal width (cm)")
plt.xlabel("Sepal length (cm)")
plt.title("Dataset of Iris flowers")
plt.legend(['Datapoints', 'Probability peaks'])
```

[ ]:
```
<matplotlib.legend.Legend at 0x12293ad68>
```



[ ]:
```
CS = plt.contour(X, Y, Z, levels=np.logspace(0, 30, 1), cmap='gray', alpha=0.5)
p1 = plt.scatter(iris.data[:,0], iris.data[:,1], alpha=1, c=iris.target,␣
↪cmap='Paired')
plt.scatter(model.means_[:,0], model.means_[:,1], marker='D', c='r')
plt.ylabel("Sepal width (cm)")
plt.xlabel("Sepal length (cm)")
plt.title("Dataset of Iris flowers")
plt.legend(handles=p1.legend_elements()[0], labels=['Iris Setosa', 'Iris␣
↪Versicolour', 'Iris Virginica'])
```

[ ]:
```
<matplotlib.legend.Legend at 0x1229d3668>
```



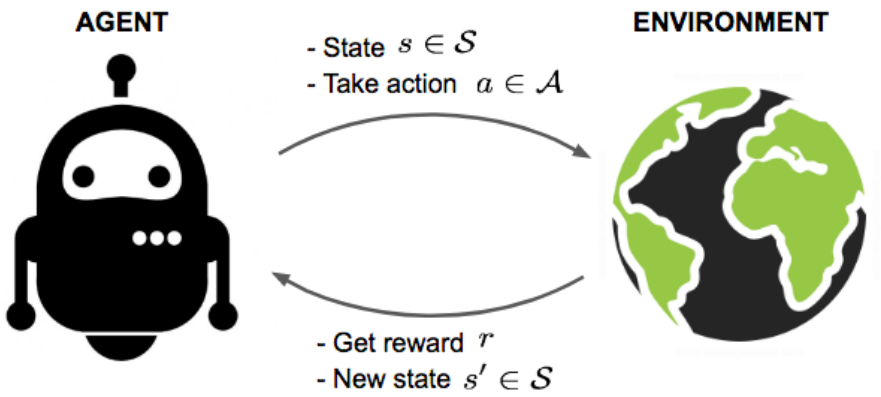
Applications of Unsupervised Learning

Unsupervised learning also has numerous applications:

- Recommendation systems: suggesting movies on Netflix.
- Anomaly detection: identifying factory components that are likely to break soon.
- Signal denoising: extracting human speech from a noisy recording.

1.4.3 Reinforcement Learning

In reinforcement learning, an agent is interacting with the world over time. We teach it good behavior by providing it with rewards.



We will not cover Reinforcement learning in this class.

Applications of Reinforcement Learning

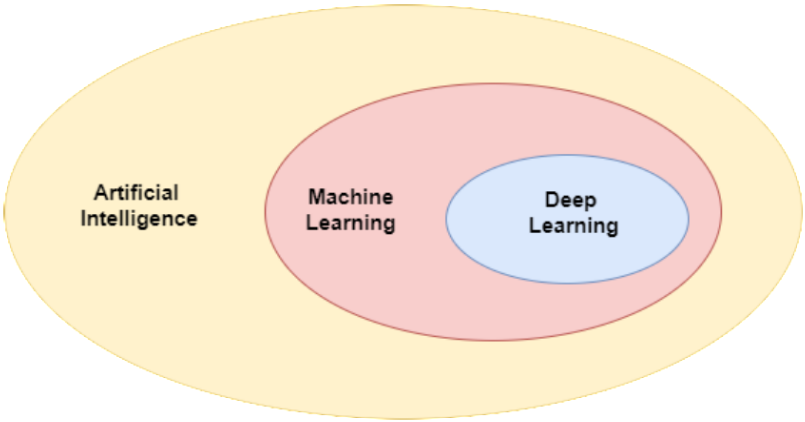
Applications of reinforcement learning include:

- Creating agents that play games such as Chess or Go.
- Controling the cooling systems of datacenters to use energy more efficiently.
- Designing new drug compounds.

1.5 Artificial Intelligence and Deep Learning

Machine learning is often discussed in the context of these two fields.

- AI is about building machines that exhibit intelligence.
- ML enables machines to learn from experience, a useful tool for AI.
- Deep learning focuses on a family of learning algorithms loosely inspired by the brain.



1.6 About this class

Next, let's look at the machine learning topics that we will cover.

1.6.1 Teaching Approach

The focus of this course is on applied machine learning.

- We will cover a broad toolset of core algorithms from many different subfields of ML.
- We will emphasize both theories (the math behind) and applications and show how to implement and apply algorithms via examples and exercises.

Why are we following this approach?

- Applying machine learning is among the most in demand industry skills right now.
- There can be a gap between theory and practice, especially in modern machine learning. We try to bridge that gap in this class. We will make effort to understand how an algorithm works. The implementation aspect of the class is a mix of using already-implemented algorithms and self-implementation.

1.6.2 What will you Learn?

- What are the core algorithms of ML and their mathematics.
- How to implement algorithms from scratch as well as using ML libraries and apply them to problems in computer vision, language processing, medical analysis, and more.

- Why machine learning algorithms work and how to use that knowledge to debug and improve them.

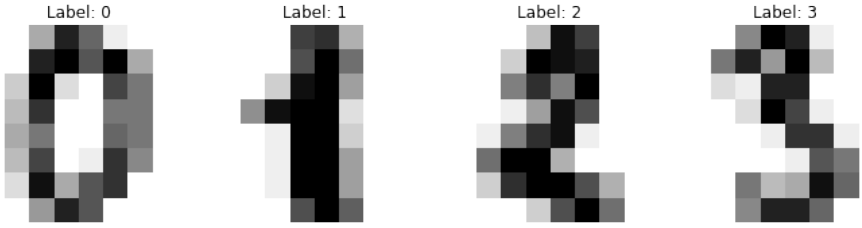
1.6.3 Software You Will Use

You will use `scikit-learn` for our class. It implements most classical machine learning algorithms.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, neural_network
plt.rcParams['figure.figsize'] = [12, 4]
```

We can use these libraries to load a simple datasets of handwritten digits.

[ ]:
```
# https://scikit-learn.org/stable/auto_examples/classification/
 ↪plot_digits_classification.html
# load the digits dataset
digits = datasets.load_digits()

# The data that we are interested in is made of 8x8 images of digits, let's
# have a look at the first 4 images.
_, axes = plt.subplots(1, 4)
images_and_labels = list(zip(digits.images, digits.target))
for ax, (image, label) in zip(axes, images_and_labels[:4]):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title('Label: %i' % label)
```



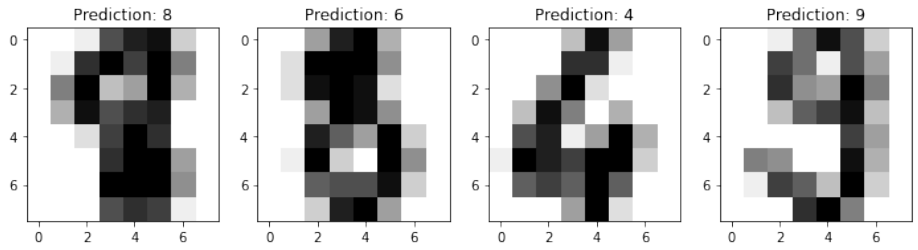
We can now load and train this algorithm inside the slides.

[ ]:
```
np.random.seed(0)
# To apply a classifier on this data, we need to flatten the image, to
# turn the data in a (samples, feature) matrix:
data = digits.images.reshape((len(digits.images), -1))

# create a small neural network classifier
from sklearn.neural_network import MLPClassifier
classifier = MLPClassifier(alpha=1e-3)

# Split data into train and test subsets
X_train, X_test, y_train, y_test = sk.model_selection.train_test_split(data,␣
 ↪digits.target, test_size=0.5, shuffle=False)
```

```
# We learn the digits on the first half of the digits
classifier.fit(X_train, y_train)

# Now predict the value of the digit on the second half:
predicted = classifier.predict(X_test)
```

We can now visualize the results.

[ ]:
```
_, axes = plt.subplots(1, 4)
images_and_predictions = list(zip(digits.images[n_samples // 2:], predicted))
for ax, (image, prediction) in zip(axes, images_and_predictions[:4]):
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title('Prediction: %i' % prediction)
```



1.7 Class Content

The course spans about 26-28 lectures approximately divided up into a set of blocks:

- Supervised and unsupervised algorithms.
- Foundations of machine learning.
- Applying machine learning in practice.
- Advanced topics and **guest lectures (?)**.

1.7.1 Machine Learning Algorithms

- Supervised learning algorithms: linear models and extensions, kernel machines, tree-based algorithms.
- Unsupervised learning algorithms: density estimation, clustering, dimensionality reduction
- Introduction to learning with deep neural networks.

1.7.2 Foundations of Machine Learning

- The basic language of machine learning: datasets, features, models, objective functions.
- Tools for machine learning: optimization, probability, linear algebra.
- Why do algorithms work in practice? Probabilistic foundations.

1.7.3 Applying Machine Learning

- Evaluating machine learning algorithms.
- Diagnosing and debugging performance.

- Analyzing errors and improving models.
- Deploying and debugging pipelines.

1.7.4 Advanced Machine Learning Topics

- Convolutional Neural Networks, Recurrent Neural networks, Graph Neural networks (if time permits)

1.8 Course Assignments

Homework will include both derivations (either expanding some details that lectures do not cover or new related approaches/derivations) and programings in python within the Scikit-learn environment. There will be approximately 7-10 homework assignments in the semester. Students will be given a minimum of one or two weeks to complete each assignment. Assignments must be submitted electronically via canvas. There will be a group project due on the last day of the class when each group will present their project.

1.9 Prerequisites. Is this class For You?

This course is designed to aimed at a very general technical audience. **Boldface** topics will be reviewed. Main requirements are:

- Programming experience (at least 1 year), preferably in Python.
- College-level linear algebra, Matrix operations, and the curiosity and the desire to learn the math behind machine learning algorithms/approaches. This is the key different from this class and the other machine learning classes.
- **College-level probability. Probability distributions, random variables, Bayes' rule, etc.** We will review most of them in class.

Other Logistics

- The majority of course materials will be accessible online.
- Grading will be based on a combination of homework assignments, exames and the final project. See the syllabus for more details.
- There is no required textbook, but we recommend Elements of Statistical Learning by Hastie, Tibshirani, and Friedman.