Introduction to Applied Machine Learning I

David Pinelle, Ph.D., Computational Research Manager
Social Sciences Research Laboratories
University of Saskatchewan
Saskatoon, Saskatchewan, Canada

Outline

- Introduction to machine learning
- Understanding data in scikit-learn
- K-nearest neighbor

Code samples and assignment

- Available on Github:
- https://github.com/pinelle

ssrl.usask.ca

What is machine learning?

- Finding patterns and relationships in data
- Using these patterns to make useful *predictions* or to *summarize* the data automatically.
- Some reason machine learning is used:
  - Human expertise does not exist (navigating on Mars)
  - Humans are unable to explain their expertise (speech recognition)
  - Solution changes over time (routing on a computer network)
  - Solution needs to be adapted to particular cases (user biometrics)

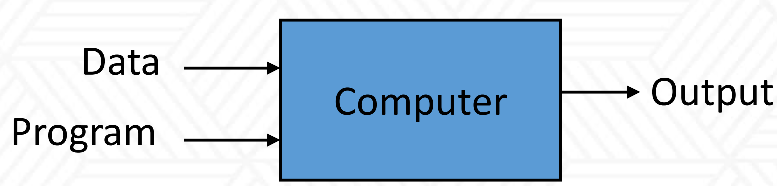
ssrl.usask.ca

What is machine learning?

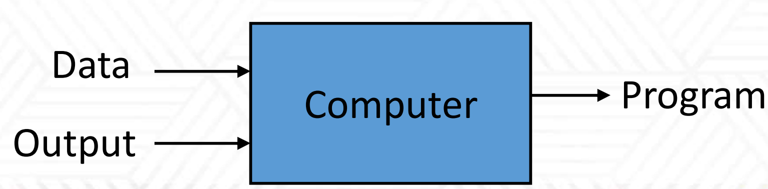
- Machine Learning Definition : ... In 1959, Arthur Samuel defined machine learning as a "Field of study that gives computers the ability to learn without being explicitly programmed".

- Traditional programs use if / then logic, often with human interaction
- This works well in many cases, but fails on many problems – image recognition, face recognition, etc.

Traditional Programming



Machine Learning



ssrl.usask.ca University of Saskatchewan | SSRL

Related disciplines

- Computer science
- Artificial intelligence
- Probability and Statistics
- Data Mining

Progress in machine learning

- Improved machine learning algorithms, toolkits
- Improved networking, faster computers
- New sensors / IO devices
- Access to large datasets
- Improved capacity to store data

Sample applications

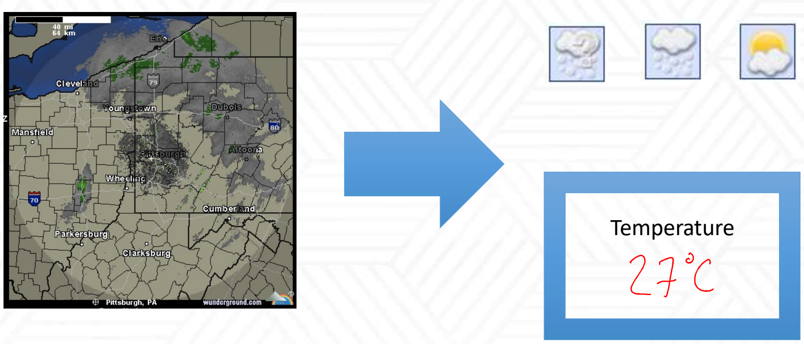
- Web search
- Computational biology
- Finance
- E-commerce
- Robotics
- Social networks
- Computer vision
- Speech recognition

ssrl.usask.ca

UNIVERSITY OF SASKATCHEWAN | SSRL

Example machine learning tasks

- Weather prediction



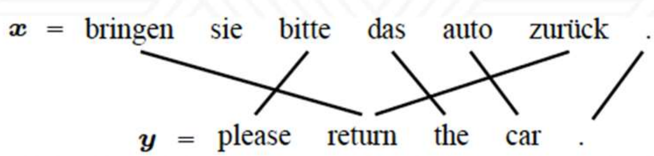
ssrl.usask.ca

UNIVERSITY OF SASKATCHEWAN | SSRL

Slide credit: Carlos Guestrin

Example machine learning tasks

- Machine translation



ssrl.usask.ca

UNIVERSITY OF SASKATCHEWAN | SSRL

Slide credit: Dhruv Batra, figure credit: Kevin Gimpel

Example machine learning tasks

- Speech recognition

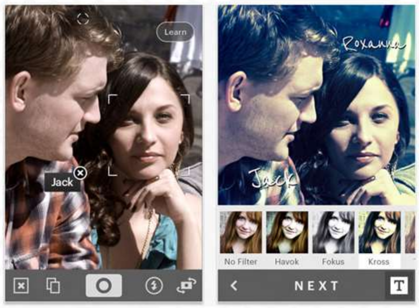
ssrl.usask.ca

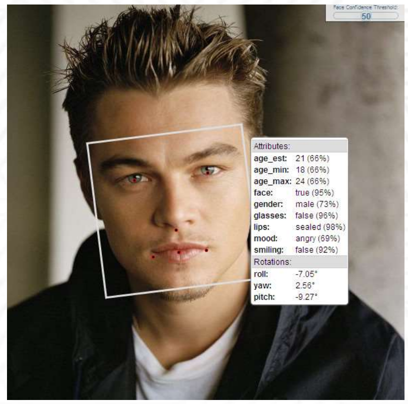
UNIVERSITY OF SASKATCHEWAN | SSRL

Slide credit: Carlos Guestrin

Example machine learning tasks

- Face recognition





ssrl.usask.ca

UNIVERSITY OF SASKATCHEWAN | SSRL SOCIAL SCIENCES RESEARCH LABORATORIES

Slide credit: Noah Snavely

Example machine learning tasks

- Image categorization

→ Pizza
Wine
Stove

ssrl.usask.ca

UNIVERSITY OF SASKATCHEWAN | SSRL SOCIAL SCIENCES RESEARCH LABORATORIES

Slide credit: Dhruv Batra

Types of learning

- Supervised learning
  - Training data includes desired outputs
- Unsupervised learning
  - Training data does not include desired outputs
- Reinforcement learning

Types of learning

- Supervised learning
  - Classification (Discrete data)
  - Regression (Continuous data)
- Unsupervised learning
  - Clustering
  - Dimensionality reduction
- Reinforcement learning

Types of supervised learning

- Classification
  - Takes some sort of input and assigning a label to it.
  - Usually used when predictions are of a discrete, or “yes or no” nature.
  - Example: Mapping a picture of someone to a male or female classification.
- Regression
  - Takes some sort of input and assigns it to a continuous, usually numeric, variable
  - Regression systems could be used, for example, to answer questions of “How much?” or “How many?”

Why use python for machine learning?

- So many tools
  - Preprocessing, analysis, statistics, machine learning, natural language processing, network analysis, visualization, deep learning
- Community support
- “Easy” language to learn
- Both a scripting and production-ready language

ssrl.usask.ca

UNIVERSITY OF SASKATCHEWAN | SSRL SOCIAL SCIENCES RESEARCH LABORATORIES

Anaconda

- Includes more than 1400 popular data-science packages + applications
  - Spyder
  - sci-kit learn
  - Numpy, Pandas
  - TensorFlow, Theano
  - Matplotlib

Technology for this workshop

- Python 3 version of Anaconda
- Spyder as the IDE

- Spyder is bundled with Anaconda
- Anaconda is available at https://www.anaconda.com/distribution/

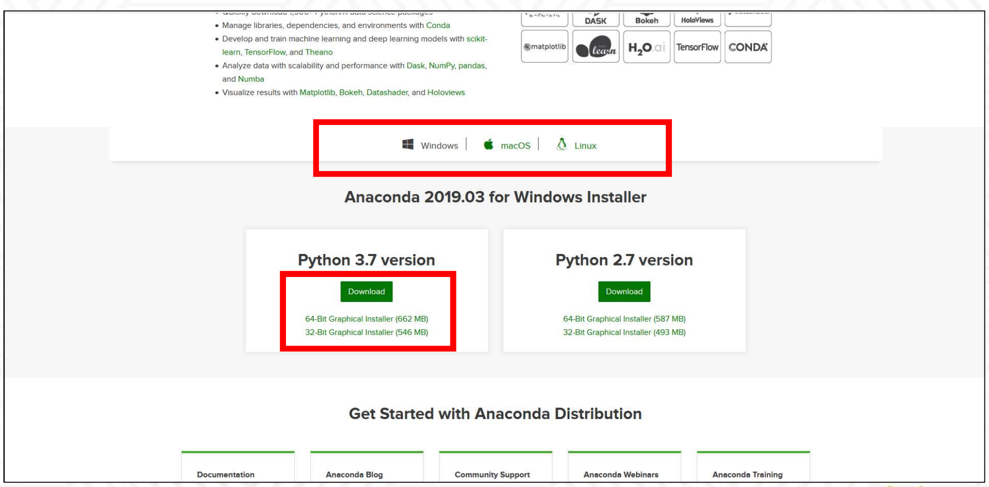
ssrl.usask.ca

Installation instructions

- Install the proper version of Anaconda 3
  - Windows
  - MacOS
  - Linux
- Start Spyder
  - Accept defaults
  - …But you may want to specify a different directory for storing your work

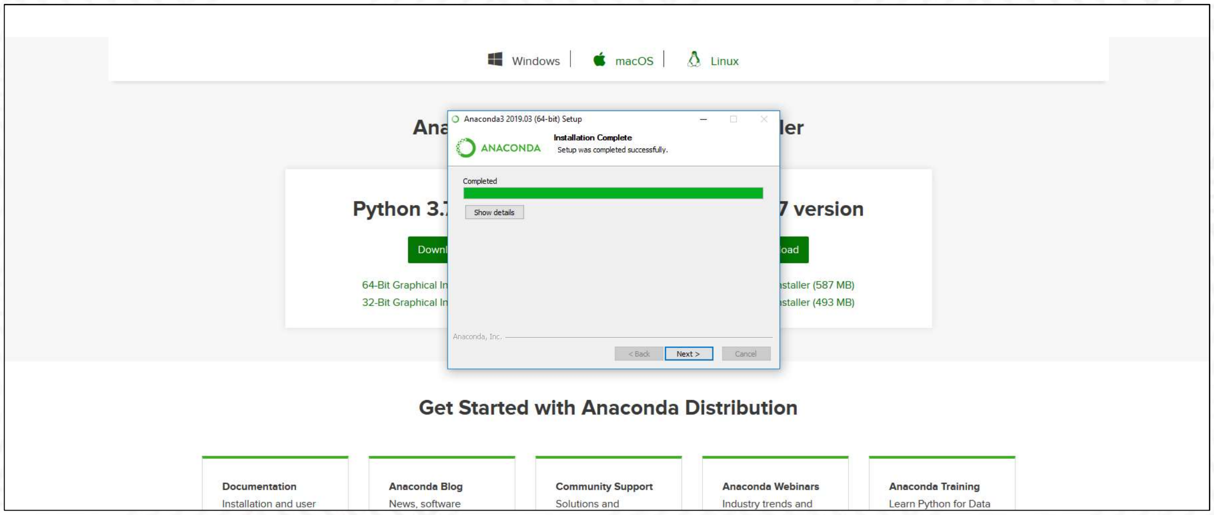
ssrl.usask.ca

Select the proper version of Anaconda 3

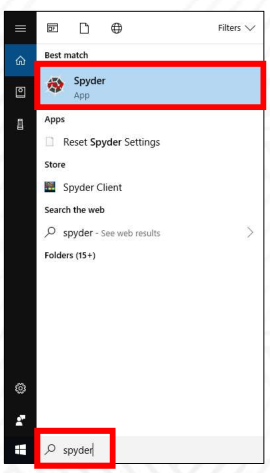


ssrl.usask.ca

Download Anaconda 3 and accept defaults



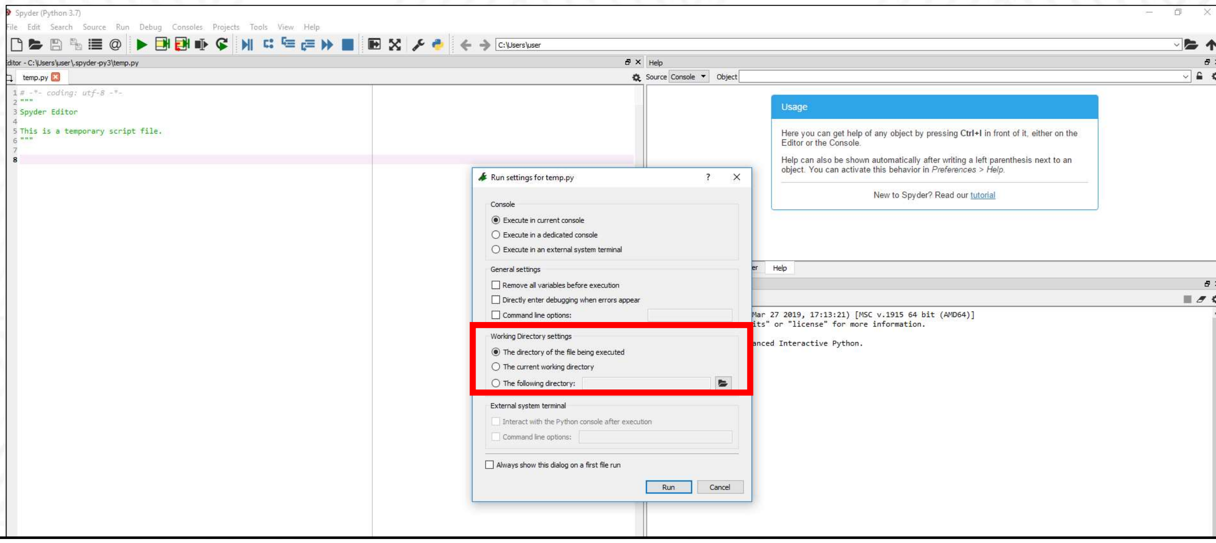
Find and start Spyder



ssrl.usask.ca

University of Saskatchewan | SSRL

Accept the Spyder defaults



Scikit-learn

- Python machine learning library
- Open-source
- Built-in datasets
  - Iris, digits datasets for classification
  - Diabetes, Boston house prices datasets for regression
- Many good online resources and books

ssrl.usask.ca

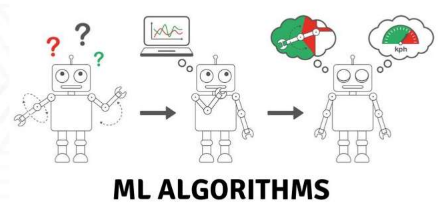
Scikit-learn

- Other things
  - Preprocessing tools
  - Very comprehensive set of machine learning algorithms
  - Methods for testing the accuracy of your model

  - Scikit-learn uses Numpy NDArrays for data storage and manipulation

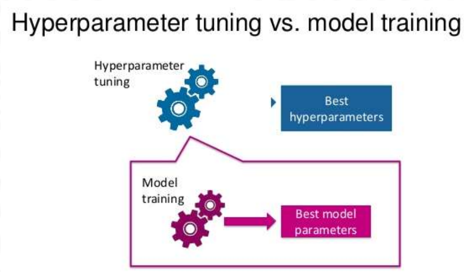
Machine learning terms

- Algorithm
  - Machine learning algorithms defines rules and calculations that are used to make decisions and predictions from training data.
- Model
  - A machine learning model is the internal representation that is created after the algorithm is trained with training data.



Machine learning terms

- Hyperparameters
  - Hyperpameters must be set and tuned to improve model performance, and it is based on experience, and at times, based on trial-and-error.



Machine learning terms

	features				target
samples (train)	type (category)	# rooms (int)	surface (float m2)	public trans (boolean)	sold (float k€)
	Apartment	3	50	TRUE	450
	House	5	254	FALSE	430
	Duplex	4	68	TRUE	712
	Apartment	2	32	TRUE	234

- Feature
  - An individual independent variables that acts as the **input** in your system.
  - You can consider one column of your data set to be one feature.
  - The number of features are called dimensions.
- Target
  - A dependent variable (**Y**) that is the **output** of the input variables (i.e. the set of features **X**).
  - The target is the variable that is being predicted in a machine learning system.

Some common algorithms

- **Supervised learning**
  - K-Nearest Neighbor
  - Linear regression
  - Logistic regression
  - Decision trees
  - Random forest
  - Support vector machines
  - Naive Bayes
- **Unsupervised learning**
  - Clustering algorithms
  - Dimensionality reduction
- **Deep neural networks**

ssrl.usask.ca

UNIVERSITY OF SASKATCHEWAN | SSRL SOCIAL SCIENCES RESEARCH LABORATORIES

Some common algorithms

- **Supervised learning**
  - ***K-Nearest Neighbor***
  - ***Linear regression***
  - ***Logistic regression***
  - ***Decision trees***
  - ***Random forest***
  - Support vector machines
  - Naive Bayes
- **Unsupervised learning**
  - Clustering algorithms
  - Dimensionality reduction
- **Deep neural networks**

ssrl.usask.ca

UNIVERSITY OF SASKATCHEWAN | SSRL SOCIAL SCIENCES RESEARCH LABORATORIES

Machine learning function

$$f(\mathbf{x}) = y$$

prediction function — features — output

- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, create prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $f(\mathbf{x}) = y$

Machine learning function

- Apply a prediction function to a feature representation of the image to get the desired output:

f() = "apple"

f() = "tomato"

f() = "cow"

Supervised learning

- One of the most commonly used types of machine learning
- Used when we want to predict an output from a given input
- Need a reasonable set of high-quality known cases
- Cases must contain an input, usually a vector of values (**x,** called features), and an output (y, called a target)
- The training data will be used to build models using different supervised learning algorithms
- The test data will be used to generate metrics that can evaluate the accuracy of the model

ssrl.usask.ca

UNIVERSITY OF SASKATCHEWAN | SSRL SOCIAL SCIENCES RESEARCH LABORATORIES

Training vs Testing

- The goal
  - High accuracy on *unseen/new/test data*
- Training data
  - Features (x) and target (y) used to learn mapping f : f(**x**) = y
- Test data
  - Features (x) used to make a prediction
  - Targets (y) only used to test the performance of the model f: f(**x**) = y

ssrl.usask.ca

UNIVERSITY OF SASKATCHEWAN | SSRL SOCIAL SCIENCES RESEARCH LABORATORIES

Iris dataset

Iris dataset

- Contains measures for previously identified irises
- It records the length and width for the petals and sepals
- Each iris in the dataset belongs to one of three species:
  - Setosa
  - Versicolor
  - Virginica

Iris dataset

- Goal: Build a model that learns from known cases so that it can predict the species of unknown irises
- Classification: Each iris is assigned to one of three classes

Load the data

- *data* contains the features (input) and *target* contains the labels (output)

```
from sklearn.datasets import load_iris
iris_dataset = load_iris()

print("Keys of iris_dataset:\n", iris_dataset.keys())
```

```
Keys of iris_dataset:
dict_keys(['data', 'target', 'target_names', 'DESCR',
'feature_names', 'filename'])
```

ssrl.usask.ca

Description: DESCR

```
print(iris_dataset['DESCR'][:193] + "\n...")

.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, pre
...
```


Feature names and target names

- Print the features (input) and *target* contains the labels (output)

```
print("Target names:", iris_dataset['target_names'])
Target names: ['setosa' 'versicolor' 'virginica']

print("Feature names:\n", iris_dataset['feature_names'])
Feature names:
['sepal length (cm)', 'sepal width (cm)', 'petal length
(cm)', 'petal width (cm)']
```

Data

- There are 150 entries in the dataset, each representing a flower
- Each entry contains four measurements

```
print("Shape of data:", iris_dataset['data'].shape)
Shape of data: (150, 4)

print("First five rows of data:\n", iris_dataset['data'][:5])
First five rows of data:
 [[5.1 3.5 1.4 0.2]
  [4.9 3.  1.4 0.2]
  [4.7 3.2 1.3 0.2]
  [4.6 3.1 1.5 0.2]
  [5.  3.6 1.4 0.2]]
```


Target

- Each iris type is coded numerically: 0: Setosa, 1: Versicolor, 2: Virginica

```
print("Shape of target:", iris_dataset['target'].shape)
Shape of target: (150,)

print("Target:\n", iris_dataset['target'])

Target:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

ssrl.usask.ca
UNIVERSITY OF SASKATCHEWAN | SSRL SOCIAL SCIENCES RESEARCH LABORATORIES

Training and testing data

- By default, 75% of the data / target is used for training, 25% is used for testing

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
iris_dataset['data'],  iris_dataset['target'],  random_state=0)

print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
```

```
X_train shape: (112, 4)
y_train shape: (112,)
```

ssrl.usask.ca

University of Saskatchewan | SSRL Social Sciences Research Laboratories

Training and testing data

- Testing data will be used to evaluate the model after it is trained with the training set

```
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
```

```
X_test shape: (38, 4)
y_test shape: (38,)
```

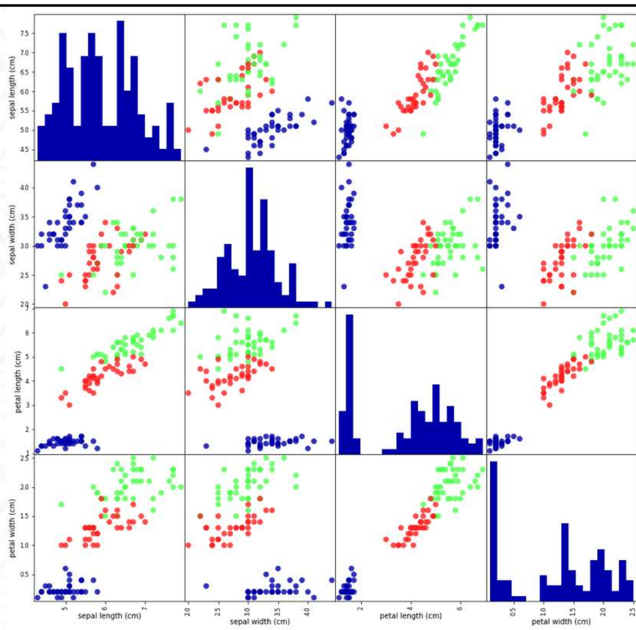
ssrl.usask.ca

University of Saskatchewan | SSRL Social Sciences Research Laboratories

Look at the data

- Create a Pandas DataFrame from data in X-train
- Use labels from iris_dataset.feature_names
- Will do pair-wise comparisions using scatter_matrix from Pandas

```
import pandas as pd
iris_dataframe = pd.DataFrame(X_train,
columns=iris_dataset.feature_names)
# create a scatter matrix from the dataframe, color by y_train
pd.plotting.scatter_matrix(iris_dataframe, c=y_train, figsize=(15, 15),
                           marker='o', hist_kwds={'bins': 20}, s=60,
                           alpha=.8, cmap=mglearn.cm3)
```

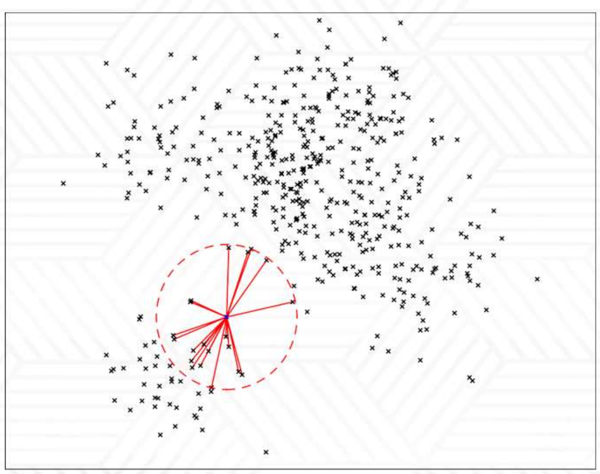
Scikit-learn algorithms

- All algorithms are implemented as their own importable class
- The class is imported and used to create a model
- Parameters are passed to the model to configure it
- The model contains
  - The algorithm that is used to build the model from the training data
  - The algorithm that will make the predictions about new data

ssrl.usask.ca

University of Saskatchewan | SSRL

K-Nearest Neighbors



ssrl.usask.ca

University of Saskatchewan | SSRL

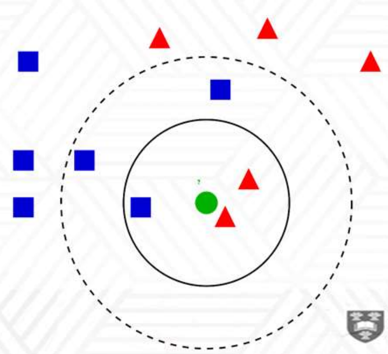
K-Nearest Neighbors

- One of the simplest classification algorithms
- …but it is also flexible and allows multiple features to be used
- It stores training data and uses it to classify new data points (lazy learner)
- Makes no assumptions about the distribution of features, targets

- Performs best when:
  - Features are numeric and have a similar scale
  - Works well with a small number of features, but struggles when the number of features is very large

K-Nearest Neighbors

- KNN classifies unknown cases by finding the points that are most similar to it (the 'nearest neighbors')
- The unknown is assigned the label (i.e. the target value) of the nearest neighbor.

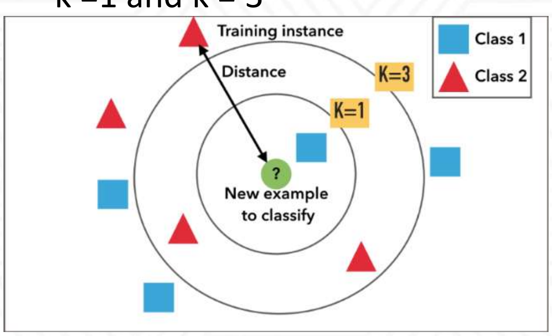


K-Nearest Neighbors

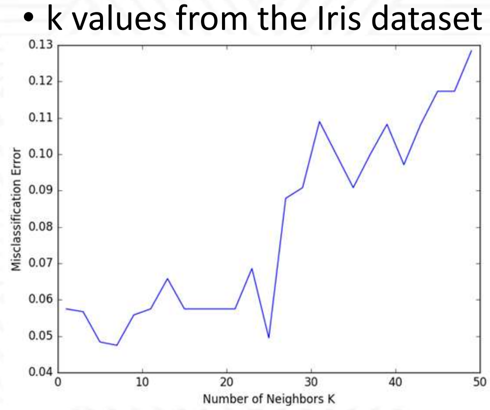
- What is ***k***?
  - It is a hyperparameter that is set before training the algorithm
  - It sets the number of nearest neighbors that should be used when classifying an unknown case
- Unknown cases are classified based on a majority vote of the **k** points closest to it
- **k** nearest neighbors are identified using a distance metric
- A variety of distance metrics are available as hyperparameters, including Euclidean, Manhattan, Chebyshev and Hamming distance.
- The optimal value of **k** is strictly a function of the problem / dataset
- **k** values are usually odd to prevent tie situations.

K-Nearest Neighbors

- k =1 and k = 3



- k values from the Iris dataset



ssrl.usask.ca
UNIVERSITY OF SASKATCHEWAN | SSRL

K-Nearest Neighbors

- The KNN algorithm follows these steps
  - A positive integer ***k*** and a distance metric are set as hyperparameters.
  - It calculates the distance between the unknown case and all points using the chosen distance metric.
  - It finds the ***k*** points that are closest based on the previously calculated distances.
  - Finally, the label is chosen based on the majority of the surrounding points.

K-Nearest Neighbors

- This version of the algorithm has a single ***k*** variable

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=1, p=2,
           weights='uniform')
```

ssrl.usask.ca

UNIVERSITY OF SASKATCHEWAN | SSRL SOCIAL SCIENCES RESEARCH LABORATORIES

Evaluation

- Evaluate the model with test data

```
print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
Test set score: 0.97
```


Prediction

- Evaluate the model with a sample iris

```
import numpy as np
X_new = np.array([[5, 2.9, 1, 0.2]])

prediction = knn.predict(X_new)
print("Prediction:", prediction)
print("Predicted target name:",
       iris_dataset['target_names'][prediction])
Prediction: [0]
Predicted target name: ['setosa']
```

Summary

```
X_train, X_test, y_train, y_test = train_test_split(
    iris_dataset['data'], iris_dataset['target'], random_state=0)

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)

print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
Test set score: 0.97
```


K-Nearest Neighbors

- Advantages
  - Simple to understand and easy to implement
  - Stores training data in memory, so it immediately adapts as new training data is added
  - Can be used both for classification and regression
- Disadvantages
  - Slows down significantly as the dataset grows and uses significant memory
  - Works well with small number of features, but as the numbers of features grow it struggles to predict the output of new data points
  - Needs homogenous features with the same scale since distance (e.g. Euclidean) is used to classify unknowns.
  - Very sensitive to outliers since it simply choses the neighbors based on distance criteria.

Assignment

- Set the **k** value in the KNN algorithm and identify the **k** values where the precision drops below:
  - 90%
  - 80%
  - 70%
- Start with a k value of 15, and then increment by 10 for each successive run.
- When you get close to a target score, start using smaller increments (stick with odd k values) until you find the proper **k** values

Assignment

- Copy and paste the assignment into a new Anaconda window
- The code is available at: **https://github.com/pinelle**

```
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

iris_dataset = load_iris()
X_train, X_test, y_train, y_test = train_test_split( iris_dataset['data'],
iris_dataset['target'],  random_state=0)
knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(X_train, y_train)
print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
```

Resources: Datasets

- UCI Repository: http://www.ics.uci.edu/~mlearn/MLRepository.html
- UCI KDD Archive: http://kdd.ics.uci.edu/summary.data.application.html
- Statlib: http://lib.stat.cmu.edu/
- Delve: http://www.cs.utoronto.ca/~delve/

Resources

- NumPy
  - https://www.numpy.org/
- Pandas
  - http://pandas.pydata.org/
- scikit-learn
  - http://scikit-learn.org/
- matplotlib
  - http://matplotlib.org/

Resources

- https://www.geeksforgeeks.org/ml-machine-learning/
- https://elitedatascience.com/start-here
- https://www.tutorialspoint.com/machine_learning/index.htm
- https://www.datacamp.com/community/tags/machine-learning
- https://machinelearningmastery.com/start-here/

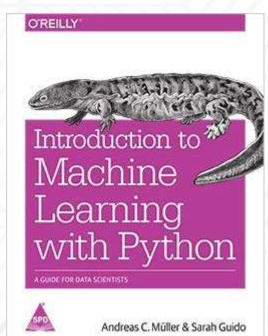
ssrl.usask.ca

University of Saskatchewan | SSRL

Resources

Introduction to Machine Learning with Python: A Guide for Data Scientists

Andreas C. Müller and Sarah Guido



ssrl.usask.ca

University of Saskatchewan | SSRL