Machine Learning

Supervised Learning

Dr. Sherif Saad

Learning Objectives

Introduce the students to machine learning concepts

Explain the main three types of learning and ML terminology

Understand the building blocks for successfully designing machine learning systems.

How to apply machine learning algorithms (not really how to create them)

Outlines

- Introduction To Supervised Learning
- Eager and Lazy Learning
- K-Nearest Neighbor Classifier

Supervised Learning: Review

- Using labeled data we can apply supervised learning techniques to build ML system that could solve many complex problems.
- Problems in supervised learning are divided into classification problems and regression problems.
- The goal of learning in supervised learning techniques is to learn a mapping function that map set of input variables to a set output variables. In other words it infer a mapping function from a set of labeled training data.

Supervised Learning: Classification

- A classification machine learning algorithm maps samples to one class from a set of predefined classes.
- A class could be something like SPAM or NOT SPAM and a sample could be an email message.
- To build a good classifier we we must have enough labeled data (e.g. large collection of email messages marked as SPAM or NOT SPAM).
- The labeled data will be used to train the classifier and eventually the classifier will be able to generalize what are the rules, features, or characteristics, that distinguish between classes|labels (e.g. SPAM and NOT SPAM)

Supervised Learning: Classification

Examples of Classification Problems:

- Facial Recognition
- Cancer Diagnosis
- Character Recognition
- Malware Detection
- Loan Approval

Why classification is supervised learning?

Supervised Learning: Classification

Types of Classifier (Classification Problems):

- Binary Classifier: Classify the sample into one of two available classes. The training dataset contains two labels on ly
- One Class Classifier: also known as novelty or outlier detection. The training dataset contains one label only.
- Multiclass Classifier: Classify the sample into one of more than two available classes. The training dataset contains more than two labels

Most of the classifier algorithms are binary classifiers.

Which classifier type is the most difficult to implement?

Supervised Learning: Classification

Binary Classifier to Multiclass Classifier

One vs All: aka One vs Rest, we can train one classifier per class, where the particular class is treated as the positive class and the samples from all other classes are considered as the negative class.

One vs One: in this approach given K number of labels we create K(K-1)/2 binary classifiers. Where each classifier learn to distinguish between two classes. At prediction time a voting scheme is applied to new sample and the results from the classifiers.

Supervised Learning: Regression

- Regression supervised learning algorithms predict a continuous-valued associated with the sample.
- Given a number of predictor (explanatory) variables and a continuous response variable (outcome), and we try to find a relationship between those variables that allows us to predict an outcome.
- Example given the number of hours the student spend on preparing for the test we can predict the score of the student in this test.
- A small change in the predictor result in a small change in the outcome

Supervised Learning: Regression

Examples of Regression Problems:

- Predict the size of the house given its price and location
- Predict the student GPA score given the student
- Predict the stock price of a company, given tweets by key people in the company.
- Predict the number of hours it will take to prepare for moving given the size of the house.
- Predict the emergency waiting time , given ??

Eager Learning Vs Lazy Learning

What is Lazy Learning?

- Training data are stored as it, or with simple processing (e.g. sorting samples) and waits until it is given a test sample.
- Training time and cost is almost zero (lazy), but prediction time is more expensive than eager learning.
- Lazy Learning algorithms effectively uses a richer hypothesis space. It uses many local linear functions to estimate the target function.

Eager Learning Vs Lazy Learning

Common Lazy Learning algorithms are usually instance-based learning, that store training examples and delay the processing|learning (lazy evaluation) until a new instance must be classified or estimated.

Examples Of Lazy Learning Algorithms:

- K-Nearest Neighbor Algorithm
- Locally Weighted Regression
- Case-Based Reasoning

Eager Learning Vs Lazy Learning

What is Eager Learning?

- Given a training dataset (set of labeled samples) eager learning algorithms construct a classification or a regression model before any attempt to classify or estimate any new (unlabeled) sample.
- Training (Learning) is very expensive in term of time and storage, while the prediction is usually very cheap.
- Eager learning is committed to a single hypothesis that model the entire dataset (instance space)

Eager Learning Vs Lazy Learning

Most of the supervised learning algorithms use eager learning methods. In general we can not claim that lazy learning models are better than eager learning models. The two model have their pros and cons. The performance of the model is mainly affected by the problem we are trying to solve.

Examples Of Eager Learning Algorithms:

- Neural Network
- Decision Tree
- Support Vector Machines

Eager Learning Vs Lazy Learning

General Guidelines:

- If your training dataset is highly dynamic either in size or classes. Consider first using lazy learning method
- If short prediction time is desirable (e.g real-time application). Consider using an eager learning method.
- If the scalability of the system is a hard nonfunctional requirement , then consider lazy learning.
- Consider building hybrid ML system that combines lazy and eager learning methods.

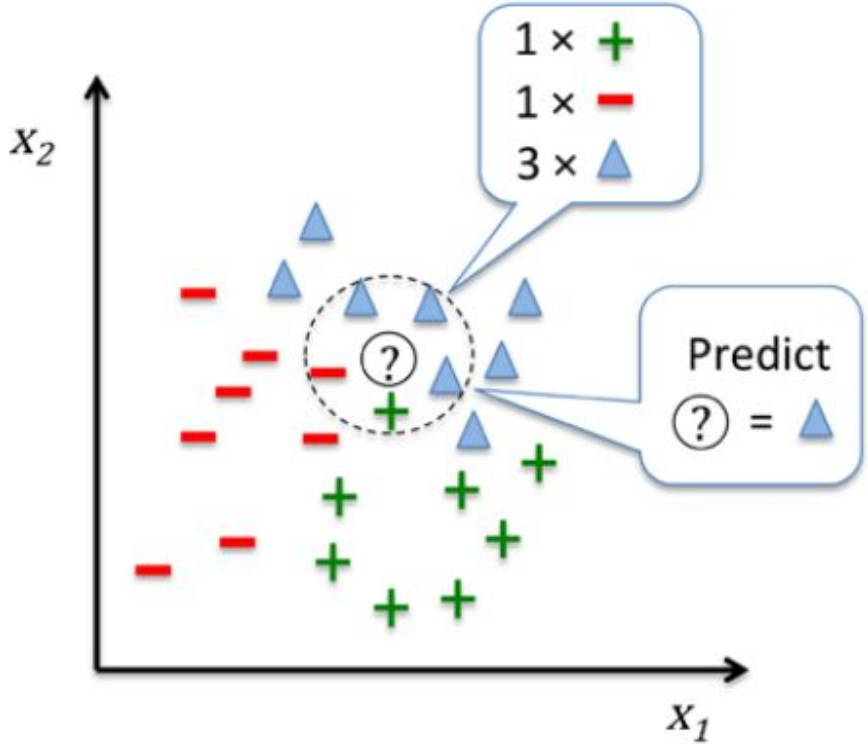
K-Nearest Neighbor Algorithm

- K-NN is a supervised learning algorithm that can be applied for classification tasks and regression tasks.
- K-NN is an instance-based learning approach, and uses a lazy learning technique.
- It is a nonparametric model. Where the ML model can not be characterized by a fixed set of parameters, and the number of parameters grows with the training data. (e.g. K-NN, Decision Tree, Random Forest)
- In K-NN all the samples (instances) correspond to points in n-d space.

K-Nearest Neighbor Algorithm

The KNN Algorithm is very simple:

1. Choose the number of k and a distance metric
2. Find the k nearest neighbors of the new sample that we want to classify or estimate.
3. Assign the class label by majority voting



K-Nearest Neighbor Algorithm

The prediction complexity is ***O(n.m),*** where n is the number of labeled samples in the dataset and m is the number of features that describe a sample.

There are two common approaches to reduce the complexity of the KNN algorithm:

- Using KD tree: where the samples are stored and sorted in k-dimensions space. (think of multidimensional binary search tree)
- What else?

Selecting the value of K for KNN Algorithm

In general, K yields smoother predictions, because we average the prediction over more samples.

What if K = 1 → The prediction function become a piecewise constant function

What if K = N → The label that dominates the dataset will dominate class for every new sample.

The value of K could be problematic and result in overfitting the KNN to the training dataset.

What is a good value of K?

Selecting the value of K for KNN Algorithm

There are many factors to consider while selecting the value of K:

- Select an odd value (e.g. when working with binary classifier)
- Choose K by applying cross-validation
- Use trial and error method.

The key point, is you need to make sure that the value of K is not overfitted to the training dataset.

KNN Distance Function and Feature Normalization

The most common distance metrics used with KNN are Euclidean Distance, Manhattan Distance, Mahalanobis Distance

$$d\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \sqrt[p]{\sum_k \left|x_k^{(i)} - x_k^{(j)}\right|^p}$$

It is also common to use a weighted version of these distance metrics, to reflect the importance of specific features.

More ***complex and problem specific*** distance metric are also common.

KNN Distance Function and Feature Normalization

Feature are commonly on different scales, in general it is important to normalize features to be on the same scale.

There are many methods to normalize features, for example

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

OR

$$x_{new} = \frac{x_{old} - \mu}{\sigma}$$

K-Nearest Neighbor for Regression

In case of prediction we return the mean value of the k nearest neighbors of the new sample.

Use KNN algorithm to predict the price of a house with the following properties:

- Number of Rooms = 6
- Number of Bathrooms = 3
- Number of Floor = 3
- The size of the house = 3200 sq

K-Nearest Neighbor for Regression

ID	# Rooms	# Bathrooms	# Floors	Size	Price
1	2	1	1	1200	**340,000**
2	2	1	1	1350	**170,000**
3	3	1	2	2500	**470,000**
4	4	2	3	2200	**720,000**
5	3	2	2	3150	**540,000**
6	5	2	3	2500	**450,000**
7	4	3	1	2100	**230,000**
8	2	2	2	1180	**320,000**
9	5	1	3	2700	**680,000**

K-Nearest Neighbor Algorithm in Nutshell

Pros:

- Can be applied to any data from any distribution
- Work with classification and regression problem
- Easy to implement and to scale

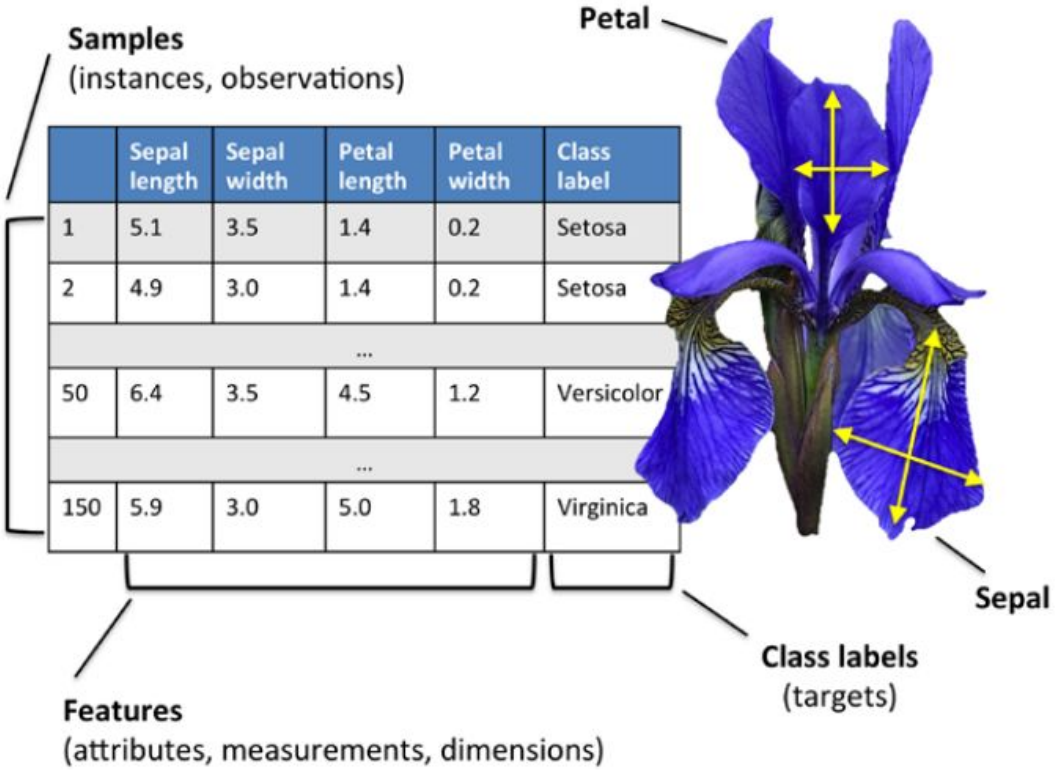
Cons:

- Require a large dataset (labeled samples)
- Prediction stage is very expensive and not helpful for real time.
- Selecting K can be tricky

K-NN Algorithm with Scikit-learn

```
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

import numpy as np
```

K-NN Algorithm: Iris Dataset

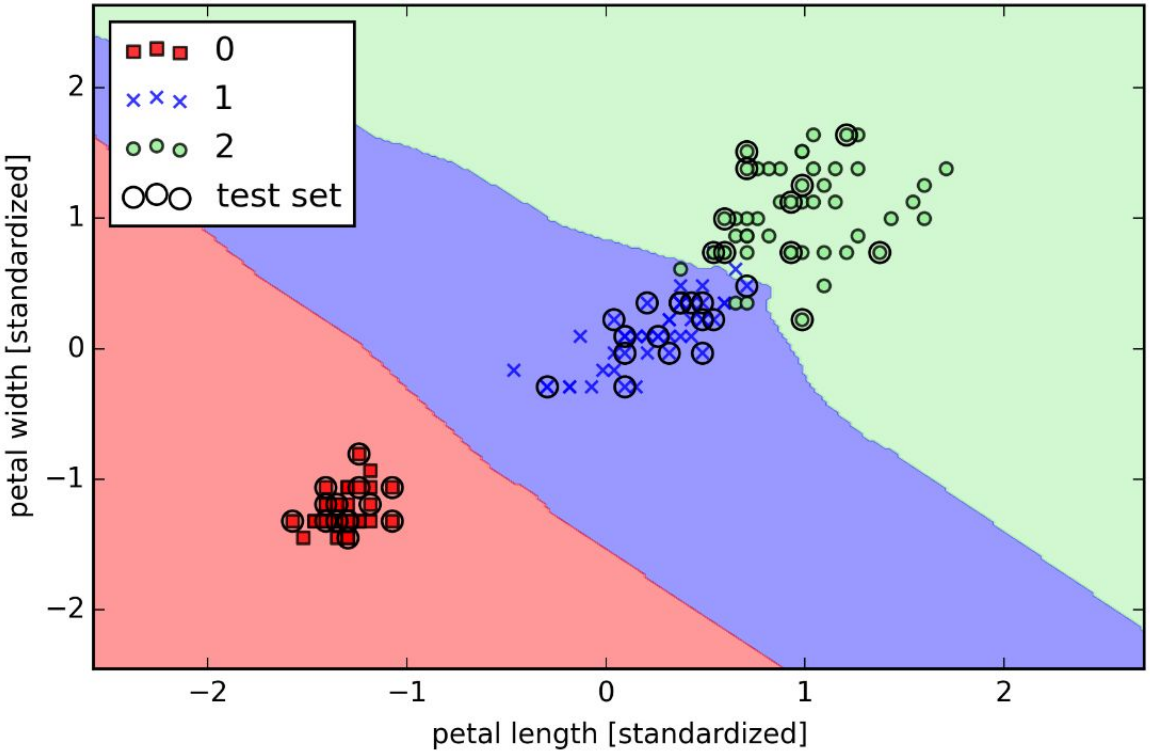


	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

K-NN Algorithm with Scikit-learn

```
iris = datasets.load_iris()

X = iris.data[:, [2, 3]]
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

sc = StandardScaler()
sc.fit(X_train)

X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

knn = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski')
knn.fit(X_train_std, y_train)

X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
```

K-NN Algorithm with Scikit-learn



Questions