SCHOOL OF COMPUTER TECHNOLOGY

AASD 4001
Mathematical Concepts for Machine Learning

Lecture 3

Reza Moslemi, Ph.D., P.Eng.

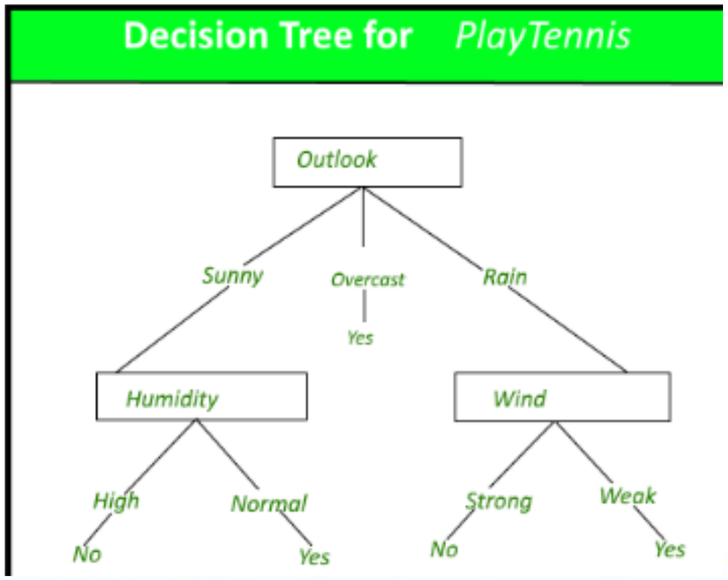
Session 3

- Decision Tree algorithms
- Gradient Descent algorithm
- Support Vector Machines
- Clustering algorithms

Decision Tree algorithms

What is a decision tree?

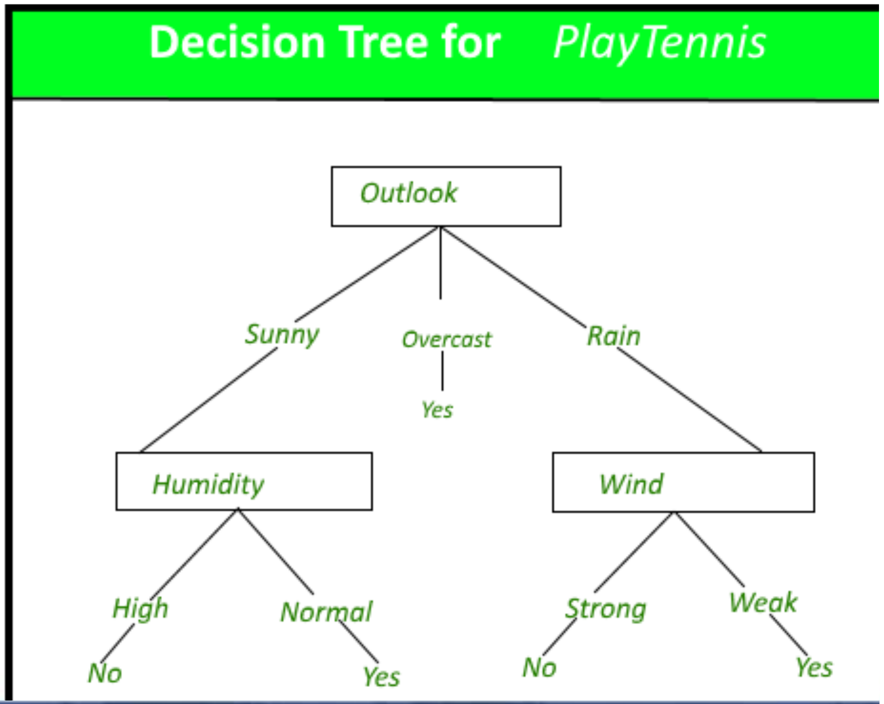
Outlook	Humidity	Wind	Played
Rain	High	Strong	Yes
Overcast	High	Weak	Yes
Rain	High	Strong	No
Overcast	Normal	Weak	Yes
Rain	Normal	Weak	Yes
Sunny	High	Strong	No
Overcast	Normal	Weak	Yes
Overcast	High	Strong	Yes
Rain	Normal	Strong	Yes
Sunny	High	Weak	Yes
Rain	Normal	Strong	No
Overcast	High	Strong	Yes



Decision Tree algorithms (Cont'd)

Elements of a tree

- Node: split for the value of a certain attribute
  - Root: the node where the first split happens
  - Leaves: terminal nodes predicting the outcome
- Edges: Outcome of a split to next node



Decision Tree algorithms (Cont'd)

How does a decision tree work?

- At each step, DT model tries to find the attribute that minimizes the entropy of the data in the next step
    - In simple terms, making the remaining data as consistent (belonging to one class) as possible
- In mathematics, it is known as a *greedy algorithm*
    - It makes the *optimal* choice *at each step* as it attempts to find the overall optimal way to solve the entire problem.
- Although the concept seems simple, it works well in practice!
    - May not be the optimal solution
    - But works often well
    - There are ways to improve it (e.g. random forest)

Decision Tree algorithms (Cont'd)

Entropy – mathematical and intuitive approach

- Degree of disorder or randomness
- Mathematically, entropy is given by $H(S) = -\sum_i p_i(S) log_2 p_i(S)$
  - where $p_i(S)$ is probability of an element/class 'i' in the data.

- Intuitively, if you were to choose an attribute for the root, what would you choose to minimize the entropy of the data in the second level?

Attribute A	Attribute B	Attribute C	Class
0	0	1	Y
1	0	1	Y
0	0	1	Y
1	0	1	Y
1	0	0	Y
1	1	1	Z
1	1	0	Z
0	1	0	Z
0	1	0	Z

Decision Tree algorithms (Cont'd)

Which attribute at the root level would minimize the entropy of the data in the second level (remaining data belonging to one class as much as possible)?

- In simple terms, which attribute makes the remaining data belong to one class as much as possible?
  - Splitting over A: YY ZZ and YYY ZZ
  - Splitting over B: YYYYY and ZZZZ
  - Splitting over C: Y ZZZ and YYYY Z

Split on A		Split on B		Split on C	
A = 0	A = 1	B = 0	B = 1	C = 0	C = 1
YY ZZ	YYY ZZ	YYYYY	ZZZZ	Y ZZZ	YYYY Z

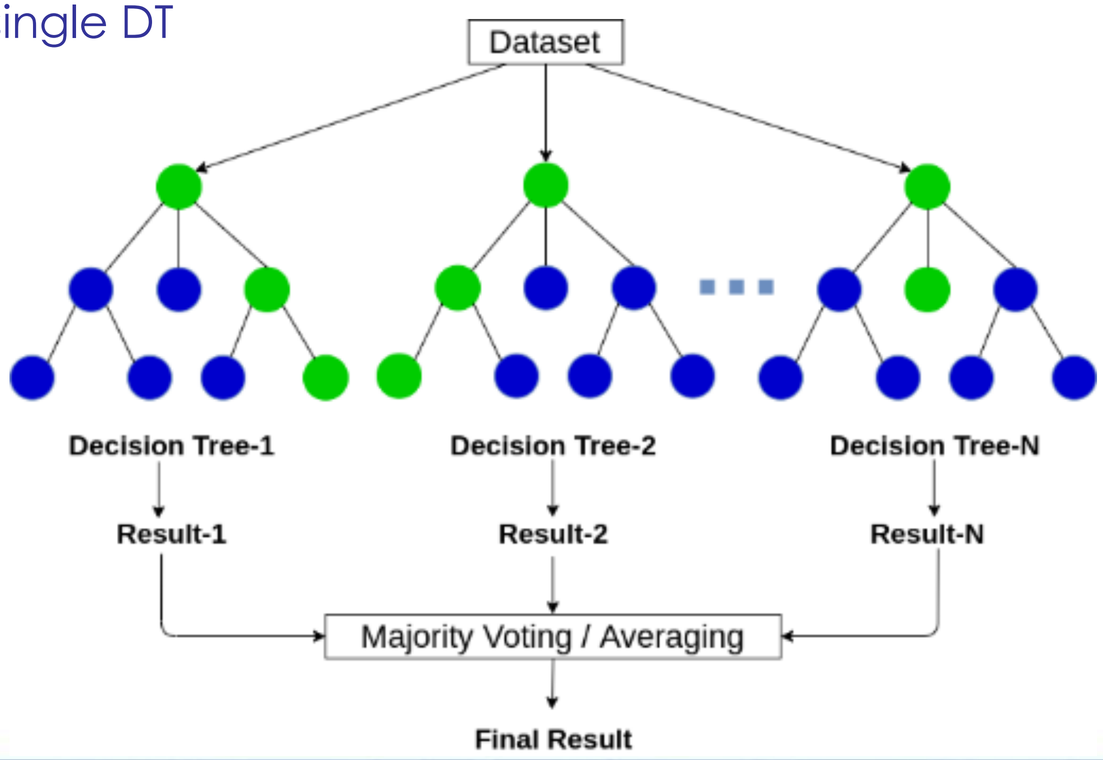
Decision Tree algorithms (Cont'd)

The primary weakness of a DT model is its tendency to overfit.

- Not very good predictive capability

How to improve?

- Limit the depth of the tree
- Random Forest is a well-known approach to improve the performance of a single DT



Decision Tree algorithms (Cont'd)

Random Forest:

- New random sample of features for each tree (*bootstrap or bagging*)

  bootstrapping is random sampling with replacement from the available training data. Bagging (= bootstrap aggregation) is performing it many times and training an estimator for each bootstrapped dataset
- Random subset of attributes for each tree at each level (uncorrelated sub-trees)
  - For classification, it is common to allow an m number of p features at each split where $m = \sqrt{p}$
  - For regression a good default is $m = p/3$
- More robust than a single decision tree
- Reduces overfitting and error due to bias

Let's switch to jupyter notebook to make classification/prediction using DT and RF model for data provided in "kyphosis.csv".

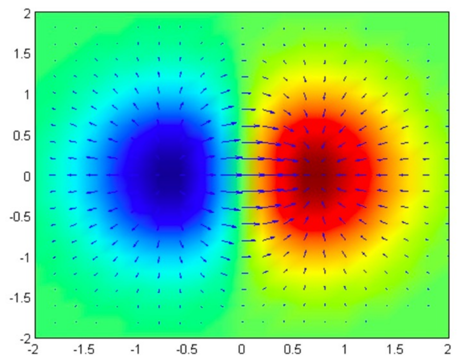
Gradient Descent algorithm

Gradient Descent algorithm

- What is gradient?
- What is a gradient descent algorithm?
- What about stochastic gradient descent algorithm?

Gradient:

- The gradient vector can be interpreted as the "direction and rate of fastest increase"
  - Consider a room where the temperature is given by T(x, y, z). At each point in the room, the gradient of T at that point will show the direction in which the temperature rises most quickly moving away from (x, y, z). The magnitude of the gradient will determine how fast the temperature rises in that direction.

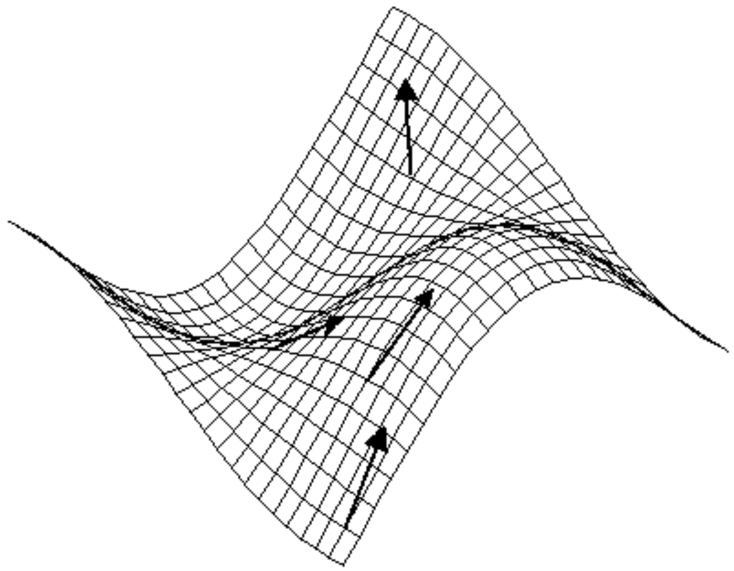


  - Consider a surface whose height above sea level at point (x, y) is H(x, y). The gradient of H at a point is a vector pointing in the direction of the steepest slope at that point.

Gradient Descent algorithm (cont'd)

Gradient:

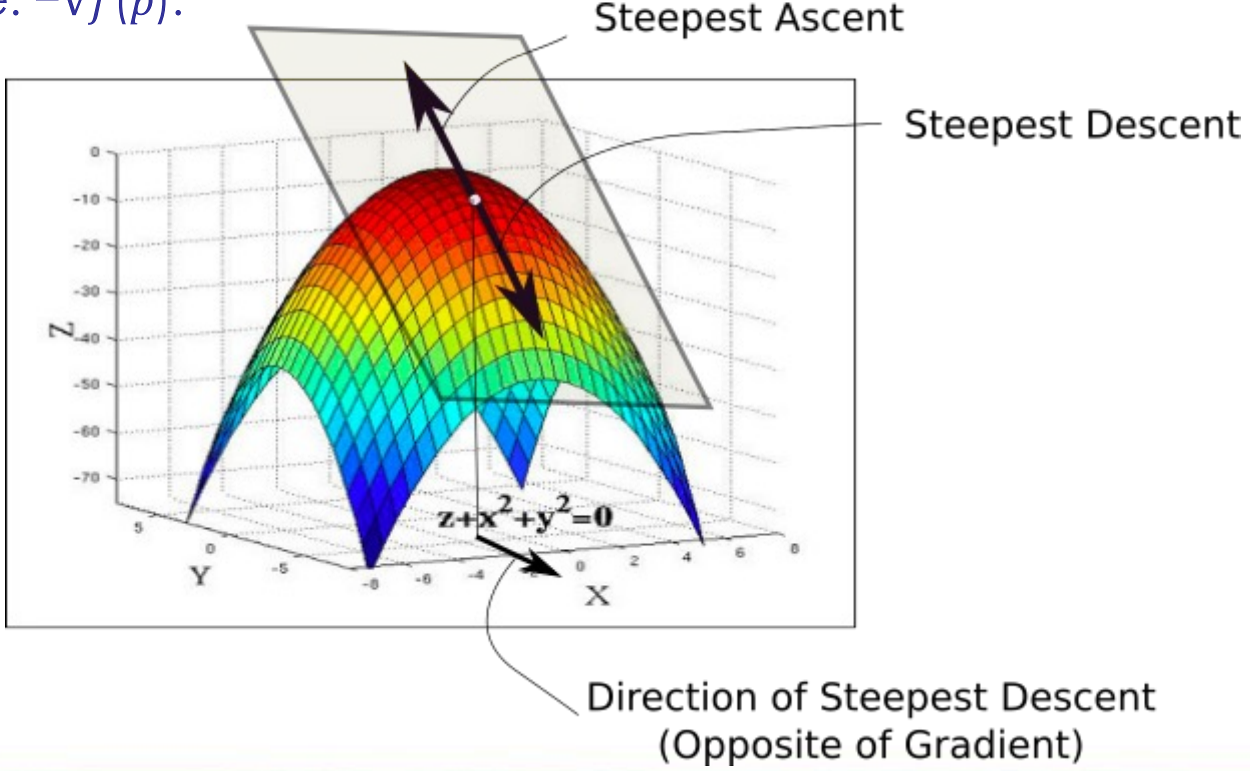
- Mathematically :
  - $\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$ where $p = (x_1, \dots, x_n)$ is an n-dimensional space.
  - In our case, n is the number of features in each sample.



Gradient Descent algorithm (cont'd)

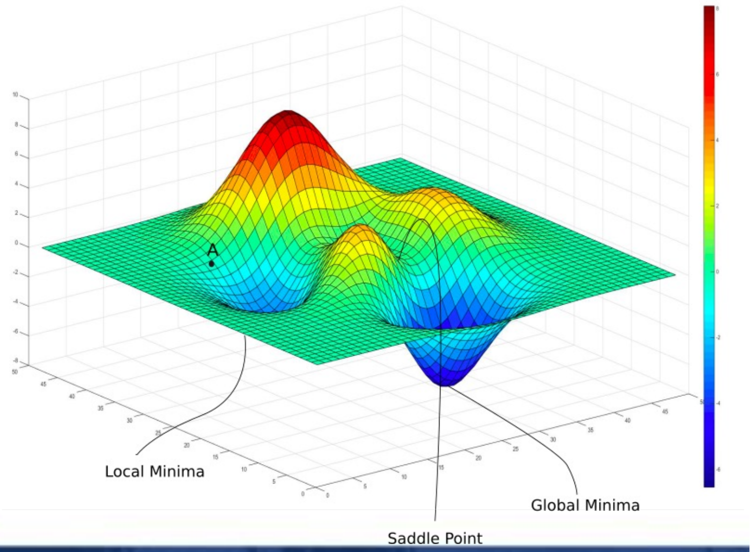
We can use the concept of gradient to minimize any loss function:

- Take the gradient at a point p, it gives the direction at which the loss function grows largest (steepest ascent).
- In order to minimize the function, move exactly in the opposite direction, i.e. $-\nabla f(p)$.



Gradient Descent algorithm (cont'd)

- There is a chance that you miss the minimum if the steps are too large.
  - Move in smaller steps, i.e. $-\nabla f(p)$ multiplied by a small learning rate, usually 0.01 as a starting point
  - A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.
- We can take larger steps when we are far from the minimum and smaller steps when close to the minimum. This process is called learning rate scheduling and is often carried out automatically by the gradient descent libraries.
- Avoid getting stuck at local minima or saddle
  - Randomness helps to arrive at a global minimum and avoid getting stuck at local minima



Gradient Descent algorithm (cont'd)

Gradient descent algorithm does not work well with large datasets. Why?

- Uses the entire training set to compute the gradient of the loss function for each iteration of the gradient descent and then updates the function
- Let's assume we have 10,000 samples with 20 features each. For each point, we need to calculate the partial derivative wrt 20 features.
- Therefore, for each step we have 10,000*20=200,000 sets of calculations.

Disadvantages:

- Redundant computation for the same training example for large datasets
- Can be very slow and intractable

Stochastic gradient descent is a solution that extends the idea of gradient descent to large datasets.

Gradient Descent algorithm (cont'd)

Stochastic Gradient Descent (SGD):

- The same idea to use (-) gradient direction to minimize a loss function
- Instead of using all the data points(the gradient of all-example-loss), only one training example (the gradient of one-example-loss) is used at a time for computation
- While the gradient of the "all-example-loss" may push us down a local minima, or get us stuck at a saddle point, the gradient of "one-example-loss" might point in a different direction, and might help us steer clear of these.
- We can improve SGD performance by using a random mini-batch (subset, certain number of examples) of the data points, rather than just one. This is the method that is often used in practice.

Different types of gradient descents:

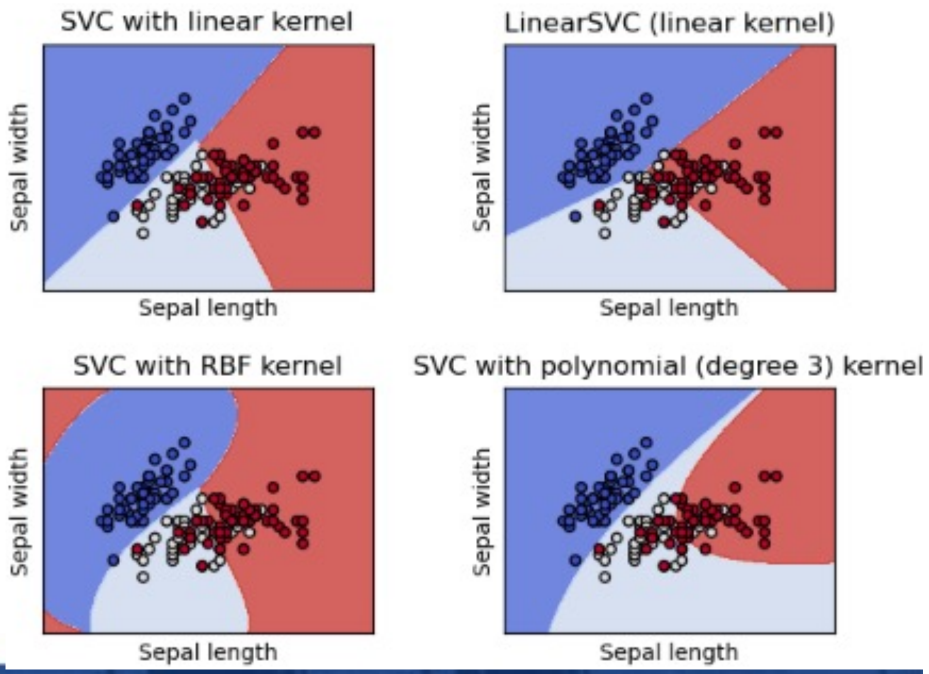
- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini batch Gradient Descent

Let's move to jupyter notebook and open "SGD.ipynb" to practice SGD classification.

Support Vector Machines

What are Support Vector Machines?

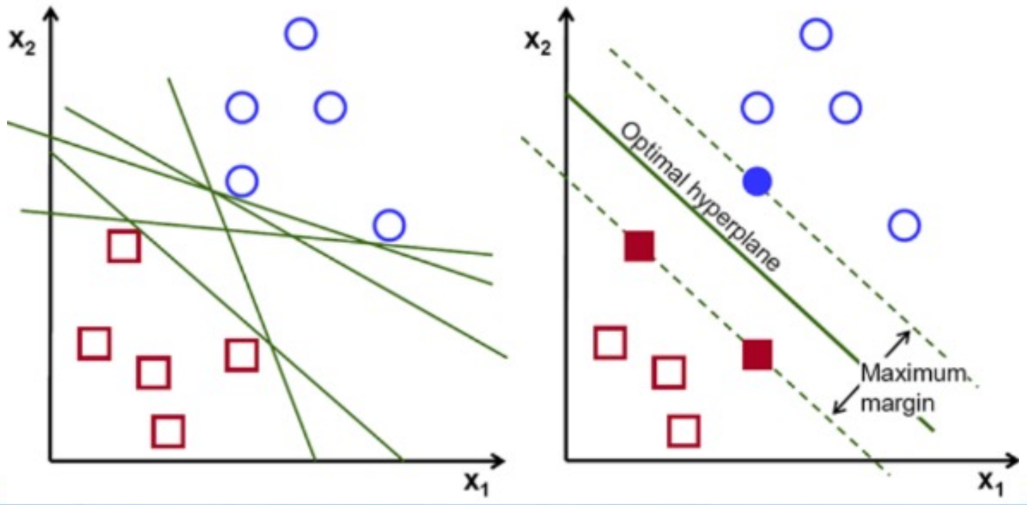
- Supervised machine learning algorithms to analyze data and recognize patterns
- Used for both regression and also classification
- Can perform linear and non-linear analysis (decision boundaries)
- Work well with high-dimensional data (data with more than a few number of features), but can be computationally expensive
- It chooses extreme vectors or support vectors to create the hyperplane
- Support vectors are defined as the data points, which are closest to the hyperplane and have some effect on its position. As these vectors are supporting the hyperplane, therefore named as Support vectors
- For non-linear decision boundaries, data can be converted into a linear one using higher dimensions



Support Vector Machines (cont'd)

Example:

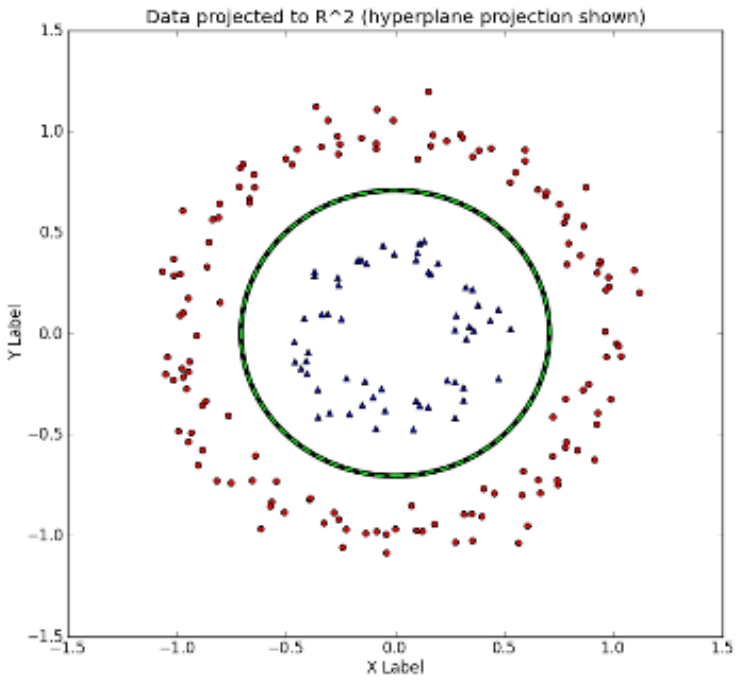
- Let's consider data points with 2 features x_1 and x_2
- It is possible to draw several lines that completely separate the classes
- Support Vector Machines (SVM) provide a mathematical framework to determine the optimal separation between classes with the maximal margin
- It is done so that the model can efficiently predict future data points
- The data points closest to the optimal hyperplane are called support vectors
- SVM maps training examples to points in space so as to maximize the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall



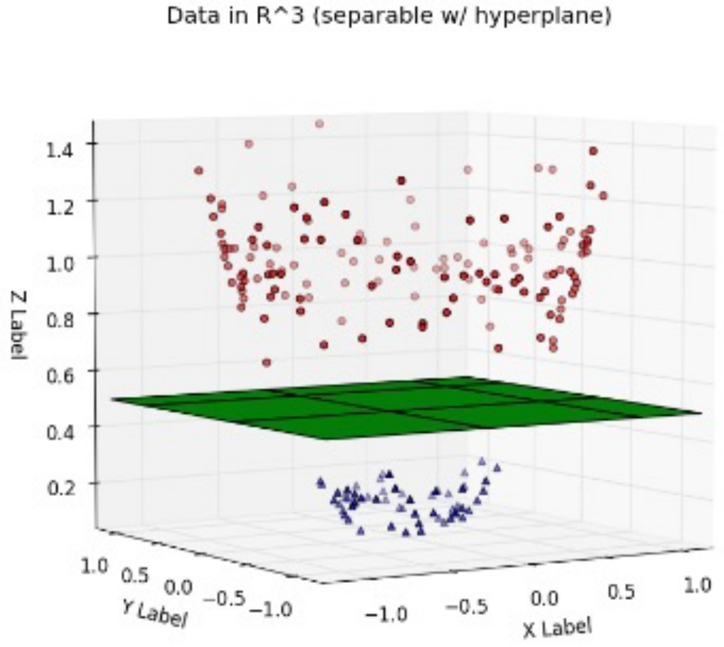
Support Vector Machines (cont'd)

But what about not linearly-separable datasets?

- The approach here is known as kernel trick.
  - A Kernel is a mathematical transformation to project a given datapoint (dataset) onto a higher dimension space, it may make it easier to classify the data where it could be linearly divided by a plane.



Kernel trick

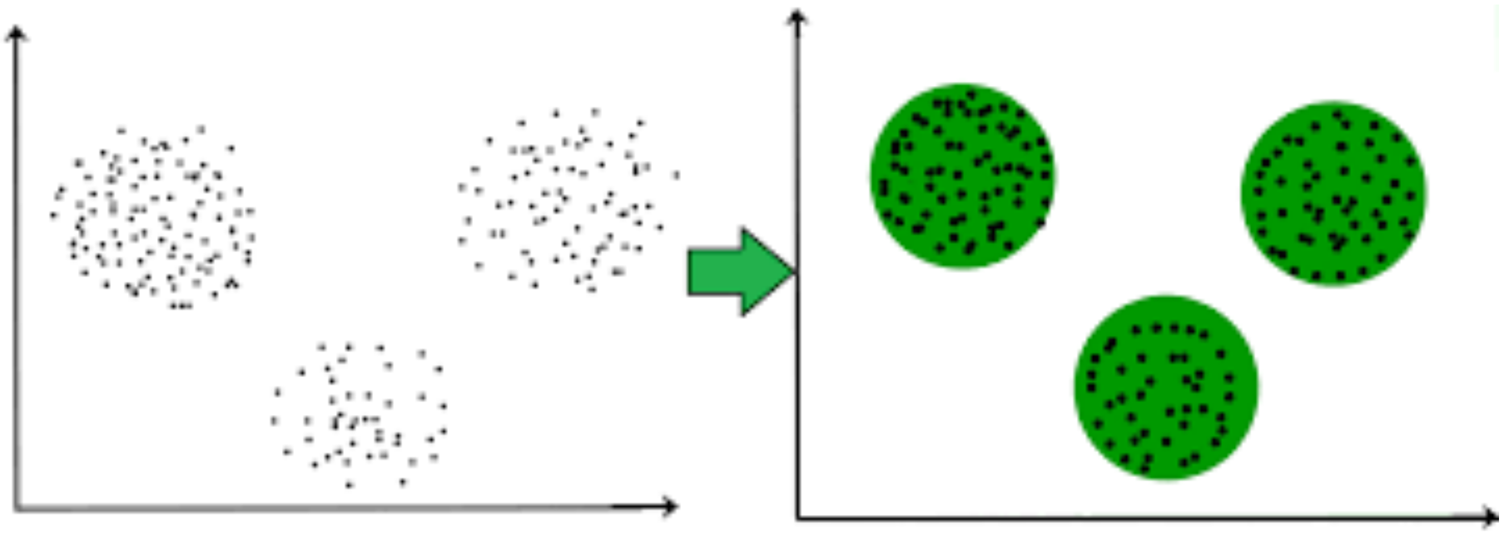


Let's move to jupyter notebook and open "SVM.ipynb" to further explore SVMs with scikit-learn.

Clustering algorithms

Clustering algorithms

- Unsupervised machine learning algorithms to group/classify a set of datapoints in a dataset such that datapoints in one cluster are more similar (in some sense) to each other than to those in other clusters



Clustering algorithms (cont'd)

Different types of clustering methods:

- Connectivity-based clustering
    - Distance based
- Centroids-based clustering
    - Represents each cluster by a single mean vector
- Distribution-based clustering
    - Modeled using statistical distributions
- Density-based Clustering
    - Defines clusters as connected dense regions in the data space

One of the most common and simple clustering algorithm is known as K-Means clustering

Clustering algorithms (cont'd)

K-Means clustering

- Attempts to group similar clusters of data together (based on their
- distance from K centroids)
- Minimizes within cluster variance
- Unsurprised machine learning algorithm – you do not need/have target information
  - In simple terms: only need $x_1, \ldots, x_n$ but not $y_1, \ldots, y_n$
- What are possible applications of K-Means clustering (and clustering, in general)

Clustering algorithms (cont'd)

Mathematically:

- Given a set of datapoints $x_1, \dots, x_n$ data is partitioned into k $(\leq n)$ clusters so as to minimize the within-cluster variance (squared error function):

$$\underset{C}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

where μ_i is the mean of the points in i -th cluster, C_i

In practice, K-Means algorithm is as follows:

- Randomly pick k $(\leq n)$ centroids
- Assign each data point to the closest centroid
- Recompute the new centroids based on the mean of the points in each cluster
- Iterate the above 2 steps until the centroids stop moving (below a given threshold)
- To predict the cluster of a new unseen datapoint, calculate its distance from all centroids and assign the class of its nearest centroid

Clustering algorithms (cont’d)

In general, for clustering algorithms one needs to be mindful of:

- Too many and/or irrelevant features can negatively affect the clustering results
- The number of clusters is not known a priori, you should decide and (probably) fine tune it
- The resultant clusters are not necessarily grouped as you wanted them to
- Last but not least, let’s repeat that K-Means is an unsupervised machine learning algorithm and therefore you cannot verify its performance in real-life scenarios by comparing it to a test set (training/test approach).

Let’s move to jupyter notebook and open “K-Means.ipynb” to explore this in practice.