Machine Learning - 2

M. Fuat Kına

Outline

- Naive Bayes Classification
- Support Vector Machines
- Decision Trees and Random Forests
- Principal Component Analysis
- Neural networks

Credits for:

https://jakevdp.github.io/PythonDataScienceHandbook/

https://app.datacamp.com/learn/courses/introduction-to-deep-learning-in-python

Naive Bayes Classification

https://scikit-learn.org/stable/modules/naive_bayes.html

$$P(\text{parameters}|\text{data}) = \frac{P(\text{data}|\text{parameters}) * P(\text{parameters})}{P(\text{data})}$$

- Supervised learning: Classifier
- Depends on Bayes' theorem (with naive assumption).
- Gaussian, Multinomial, Complement, Bernulli…

P(parameters|data) -> posterior distribution what we know about parameters after we see the data

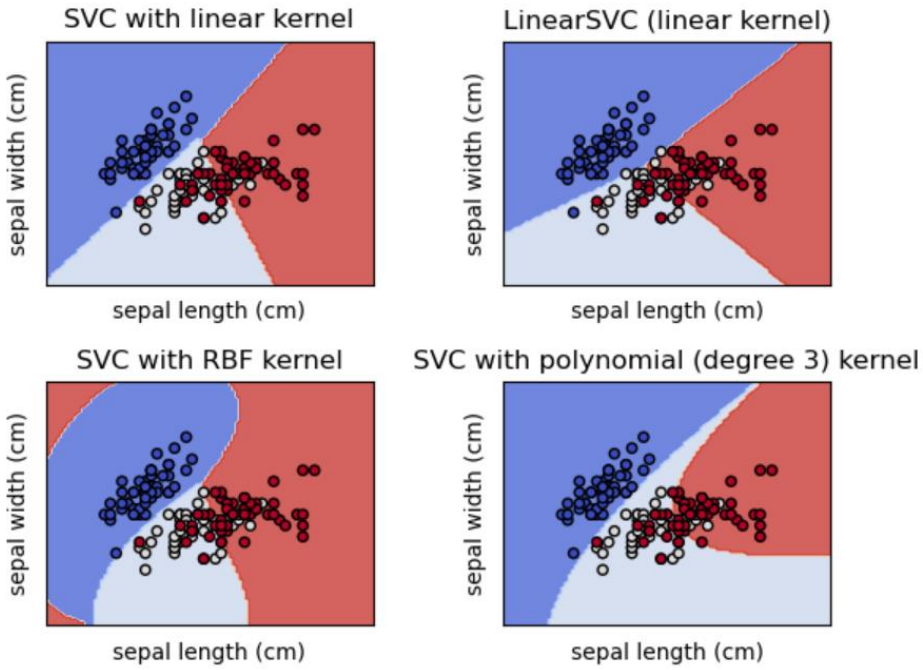
P(parameters) -> prior distribution what we know about parameters

P(data|parameters) -> likelihood the data according to statistical model

Support Vector Machines

https://scikit-learn.org/stable/modules/svm.html

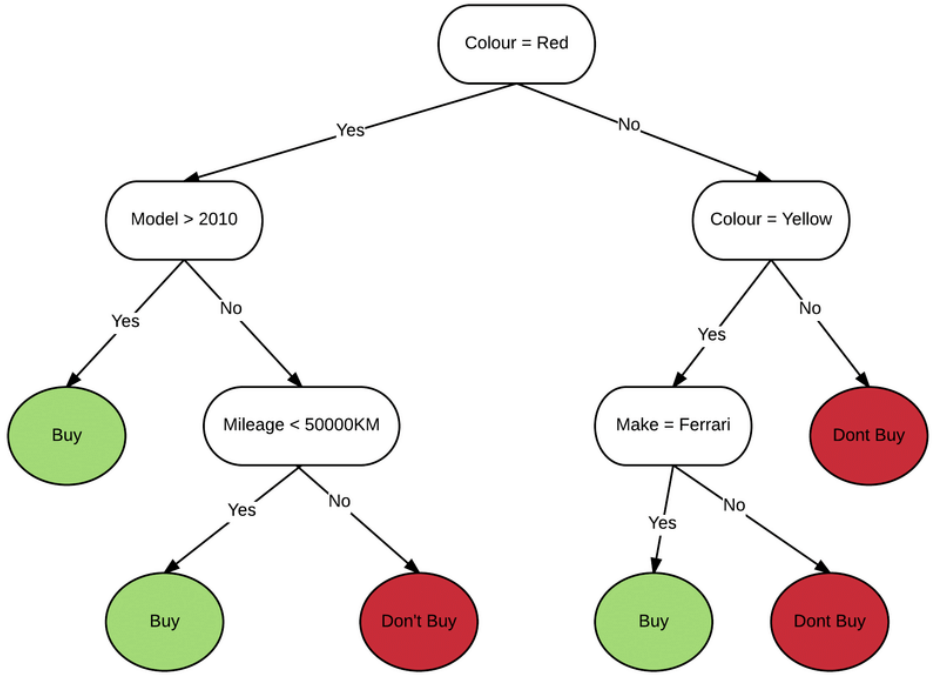
- Supervised Learning: Classifier
- SVM maps training examples to points in space to maximize the width of the gap between the two categories.
- N-dimensional space
- Considers the distances from the closest point in each of the two groups will be the farthest away.
- Specify kernel: linear, polynomial, RBF…



Decision Trees and Random Forests

https://scikit-learn.org/stable/modules/tree.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

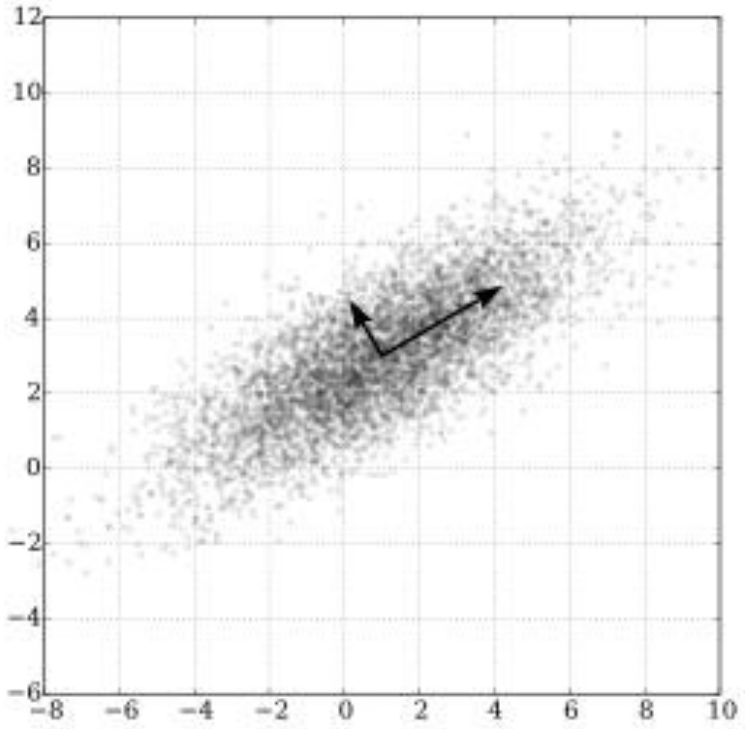
- Supervised learning
- The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.
- Works with both categorical and continuous target variables
- For random forests, specify number of trees.
- Criteria for divisions: Gini, entropy, chi-square
- To avoid from over-fitting, specify a maximum depth of tree (vertical depth). It is used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample, and should be tuned using CV.



Principal Component Analysis

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

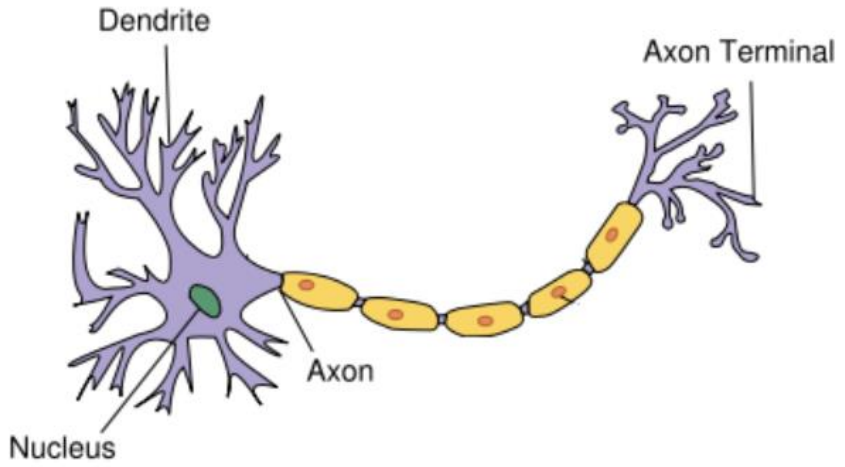
- Unsupervised learning
- The process of computing the principal components and using them to perform a change of basis on the data.
  - Maximizes variance
  - Extract orthogonal dimensions
- Decorrelating dimensions of data: PCA features are not correlated



Neural Networks and Deep Learning

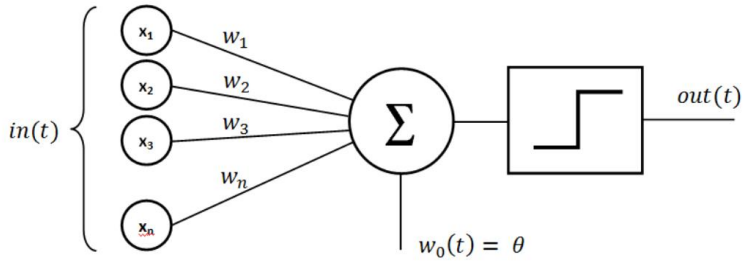
https://stackabuse.com/introduction-to-neural-networks-with-scikit-learn/
https://keras.io/guides/

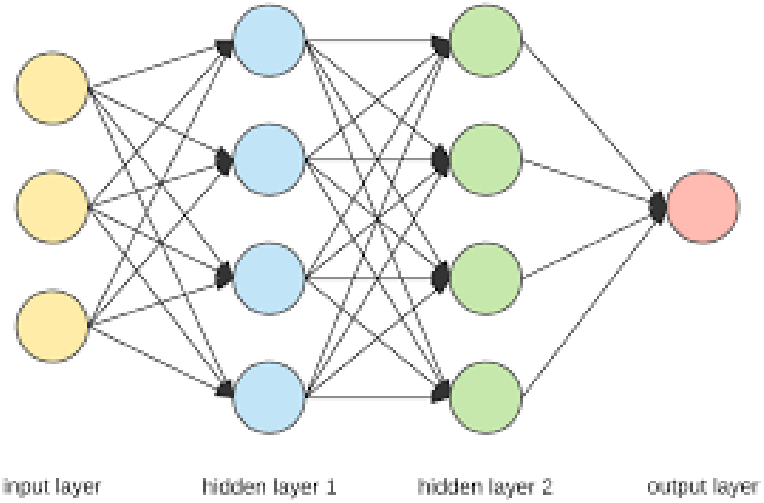
- The main purpose: recognizing things like human brain
- Inspired by the human nervous system
- Supervised learning
- Easily can be run by scikit-learn in Python
- Extremely efficient, especially when one wants to analyze different types of features together



Perceptrons and Artificial Neural Networks

- The simplest neural network consists of only one neuron and is called a perceptron, as shown in the figure above.
- First group of nodes (left-side) has the same number of features in the data.
- Artificial Neural Network (Multilayer Perceptron)





Artificial Neural Networks

- Forward Propagation: Weighting the input and running the activation function until reaching the output layer
  - The values received in the input layer are multiplied with the weights. A bias is added.
  - In the first hidden layer, neurons have an activation function that operates upon the value received from the input layer.
  - The outputs from the first hidden layer neurons are multiplied with the weights of the second hidden layer.
- Back Propagation
  - The error is calculated by quantifying the difference between the predicted output and the desired output.
  - Minimize the error!!!

Applications with Sklearn and Keras

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
https://predictivelearning.github.io/projects/Deep_Learning_1__Sequential_models.html

- Specify architecture
  - How many layers do you want? how many nodes in each layer?
  - What activation function do you want to use in each layer?
  - Identity function, relu, softmax, etc.
- Compile
  - This specifies the loss function, and some details about how optimization works.
  - Loss function: Means squared error, categorical_crossentropy
  - Optimizers: Adam
- Fit the model
- Predict the outcome