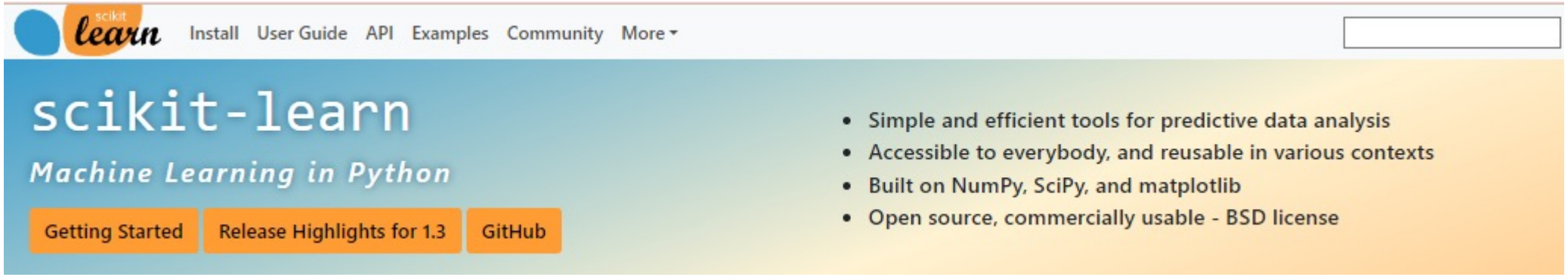
Machine learning algorithms



Instructor: Hsiang-Han Chen (陳翔瀚)

Supervised learning:

- Classification
- Regression

Unsupervised learning:

- Clustering
- Dimensionality reduction

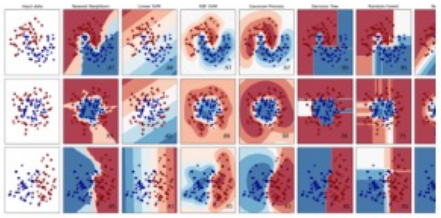
Model selection

Preprocessing

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

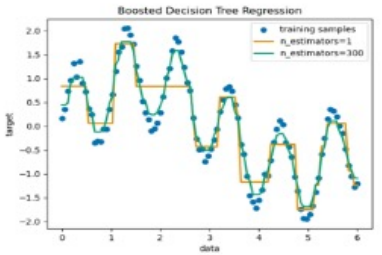


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

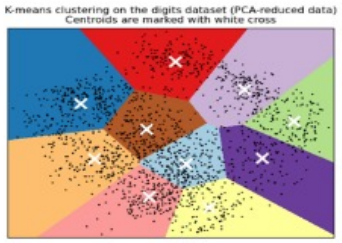


Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...

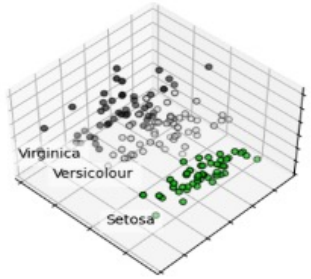


Examples

Dimensionality reduction

Reducing the number of random variables to consider.

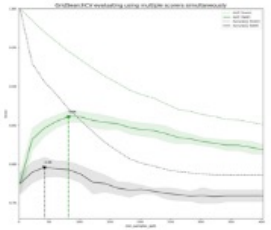
Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization, and more...



Model selection

Comparing, validating and choosing parameters and models.

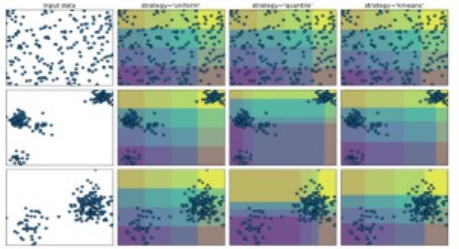
Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...



Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...



Supervised learning

- **Classification:**

Identifying which category an object belongs to.

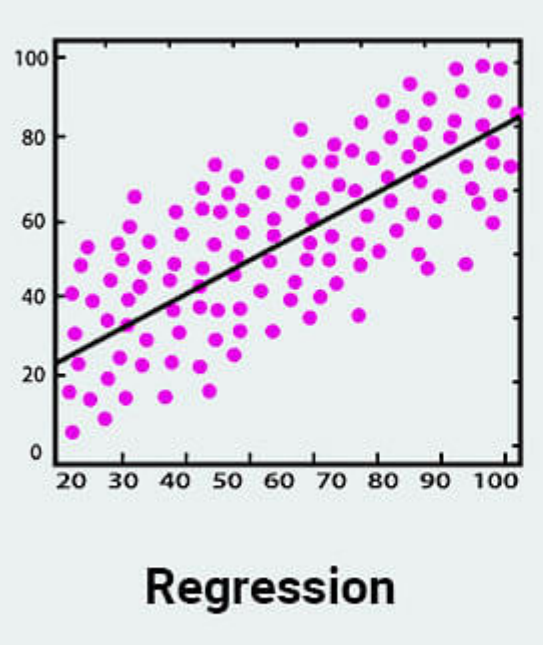
Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

- **Regression**

Predicting a continuous-valued attribute associated with an object.

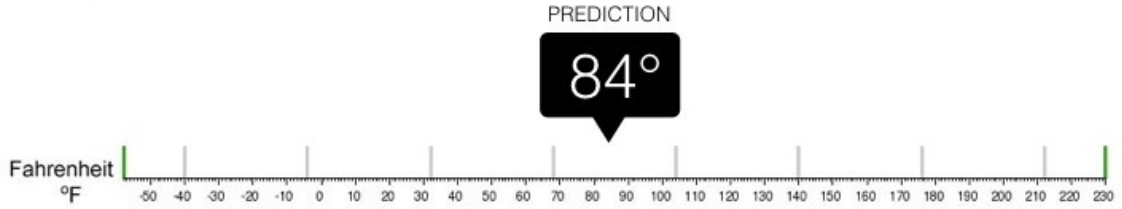
Applications: Drug response, Stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

Classification vs. Regression



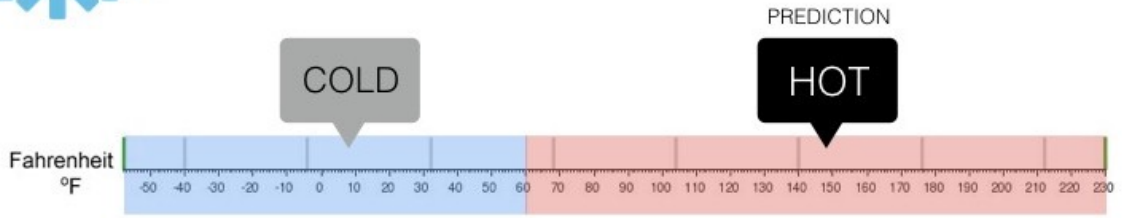
Regression

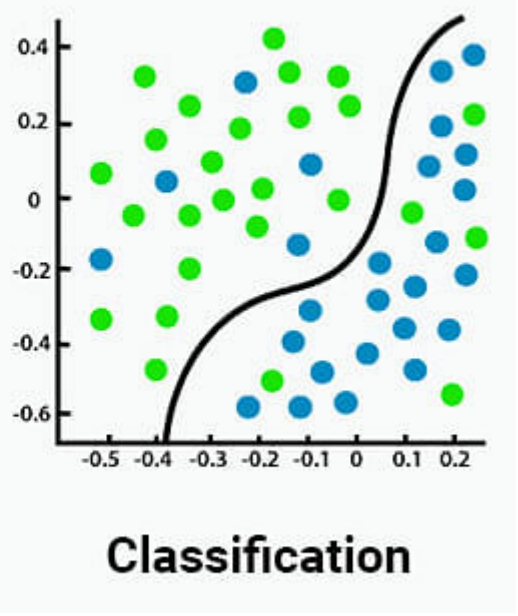
What is the temperature going to be tomorrow?



Classification

Will it be Cold or Hot tomorrow?



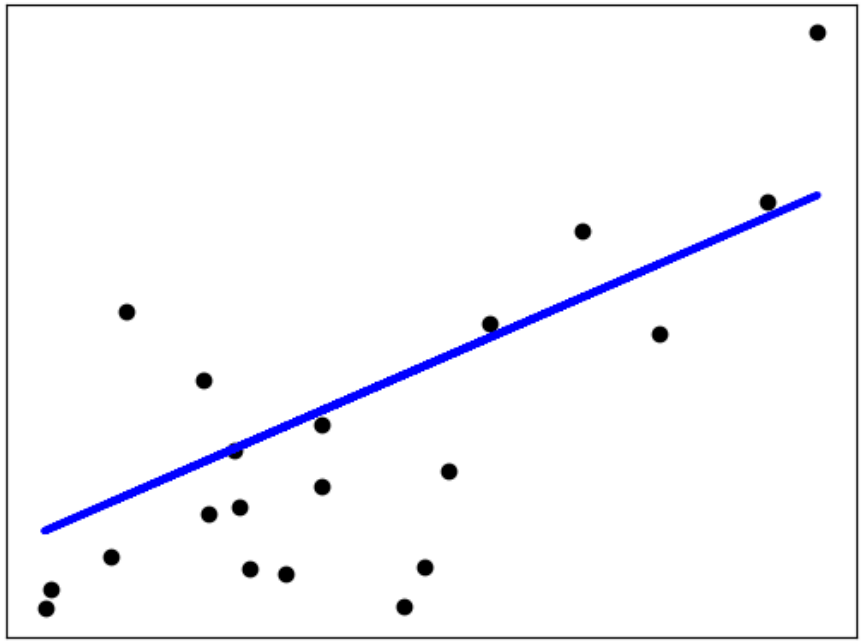


https://www.simplilearn.com/regression-vs-classification-in-machine-learning-article

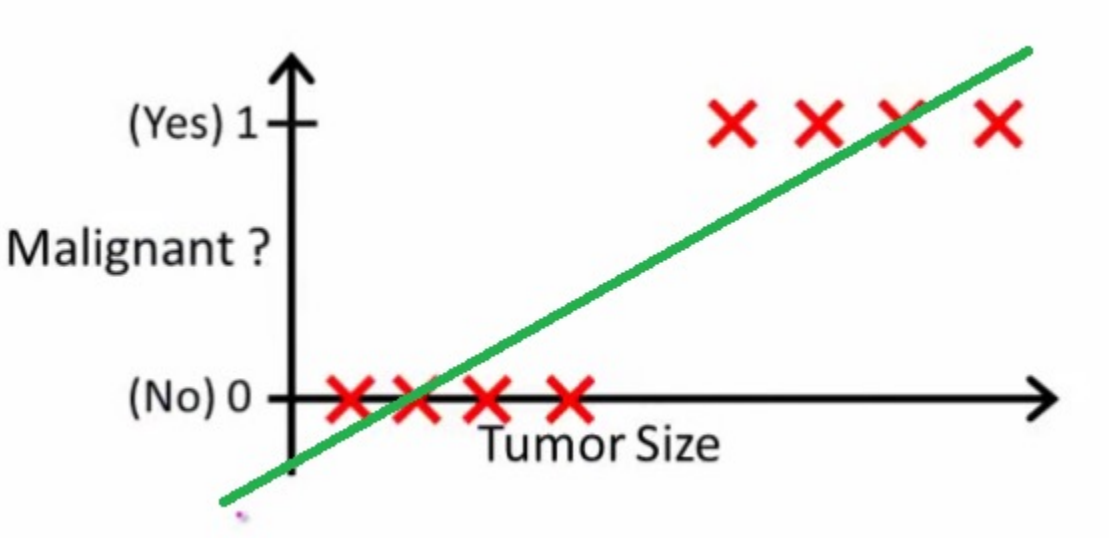
https://www.springboard.com/blog/data-science/regression-vs-classification/

Linear Models

$$\min_{w} ||Xw - y||_2^2$$



For classification?



https://stats.stackexchange.com/questions/22381/

Linear Models – penalization

- Ridge model: $\min_w ||Xw - y||_2^2 + \alpha||w||_2^2$

- Lasso model: $\min_w \frac{1}{2n_{\text{samples}}}||Xw - y||_2^2 + \alpha||w||_1$

- Elastic-Net $\min_w \frac{1}{2n_{\text{samples}}}||Xw - y||_2^2 + \alpha\rho||w||_1 + \frac{\alpha(1-\rho)}{2}||w||_2^2$

Linear Models – Generalized Linear Models

Generalized Linear Models (GLM) extend linear models in two ways.

First, the predicted values $\hat{y}$ are linked to a linear combination of the input variables *X* via an inverse link function *h* as

$$\hat{y}(w, X) = h(Xw)$$

For example: with link='log', the inverse link function becomes $\exp(X\omega)$

Secondly, the squared loss function is replaced by the unit deviance d of a distribution in the exponential family (or more precisely, a reproductive exponential dispersion model (EDM)

$$\min_w \frac{1}{2n_{\text{samples}}} \sum_i d(y_i, \hat{y}_i) + \frac{\alpha}{2}||w||_2^2$$

Linear Models – Generalized Linear Models

- Some specific EDMs and their unit deviance

Distribution	Target Domain	Unit Deviance $d(y, \hat{y})$
Normal	$y \in (-\infty, \infty)$	$(y - \hat{y})^2$
Bernoulli	$y \in \{0, 1\}$	$2(y \log \frac{y}{\hat{y}} + (1-y) \log \frac{1-y}{1-\hat{y}})$
Categorical	$y \in \{0, 1, \ldots, k\}$	$2 \sum_{i \in \{0,1,\ldots,k\}} I(y = i) y_i \log \frac{I(y=i)}{I(\hat{y}=i)}$
Poisson	$y \in [0, \infty)$	$2(y \log \frac{y}{\hat{y}} - y + \hat{y})$
Gamma	$y \in (0, \infty)$	$2(\log \frac{y}{\hat{y}} + \frac{y}{\hat{y}} - 1)$
Inverse Gaussian	$y \in (0, \infty)$	$\frac{(y-\hat{y})^2}{y\hat{y}^2}$

Linear Models

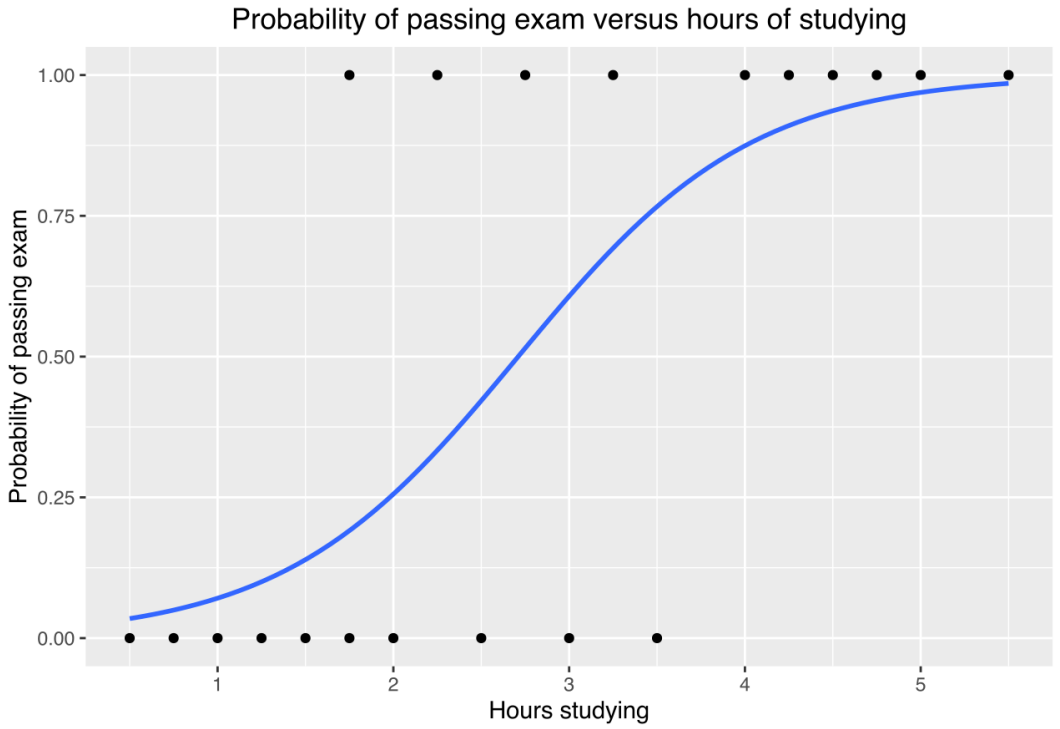
$$\hat{p}(X_i) = \text{expit}(X_i w + w_0) = \frac{1}{1 + \exp(-X_i w - w_0)}$$

$$\min_w C \sum_{i=1}^{n} \left(-y_i \log(\hat{p}(X_i)) - (1 - y_i) \log(1 - \hat{p}(X_i))\right) + r(w).$$

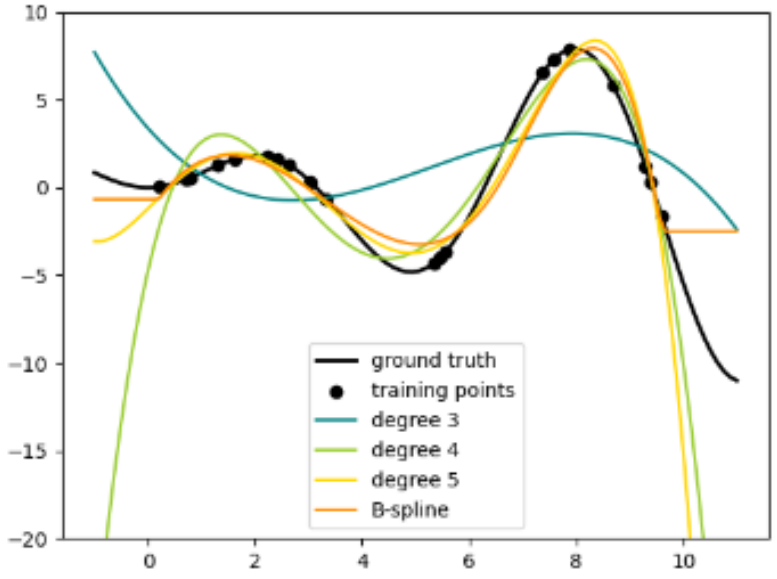
- Logistic regression

Despite its name, it is implemented as a linear model for classification rather than regression in terms of the scikit-learn/ML nomenclature.

A special case of Generalized Linear Models with a Binomial / Bernoulli conditional distribution and a Logit link.



Linear Models



- Polynomial regression

Extending linear models with basis functions.

EX: $\hat{y}(w, x) = w_0 + w_1 x_1 + w_2 x_2$

EX: $\hat{y}(w, x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2$

This is still a linear model: to see this, imagine creating a new set of features

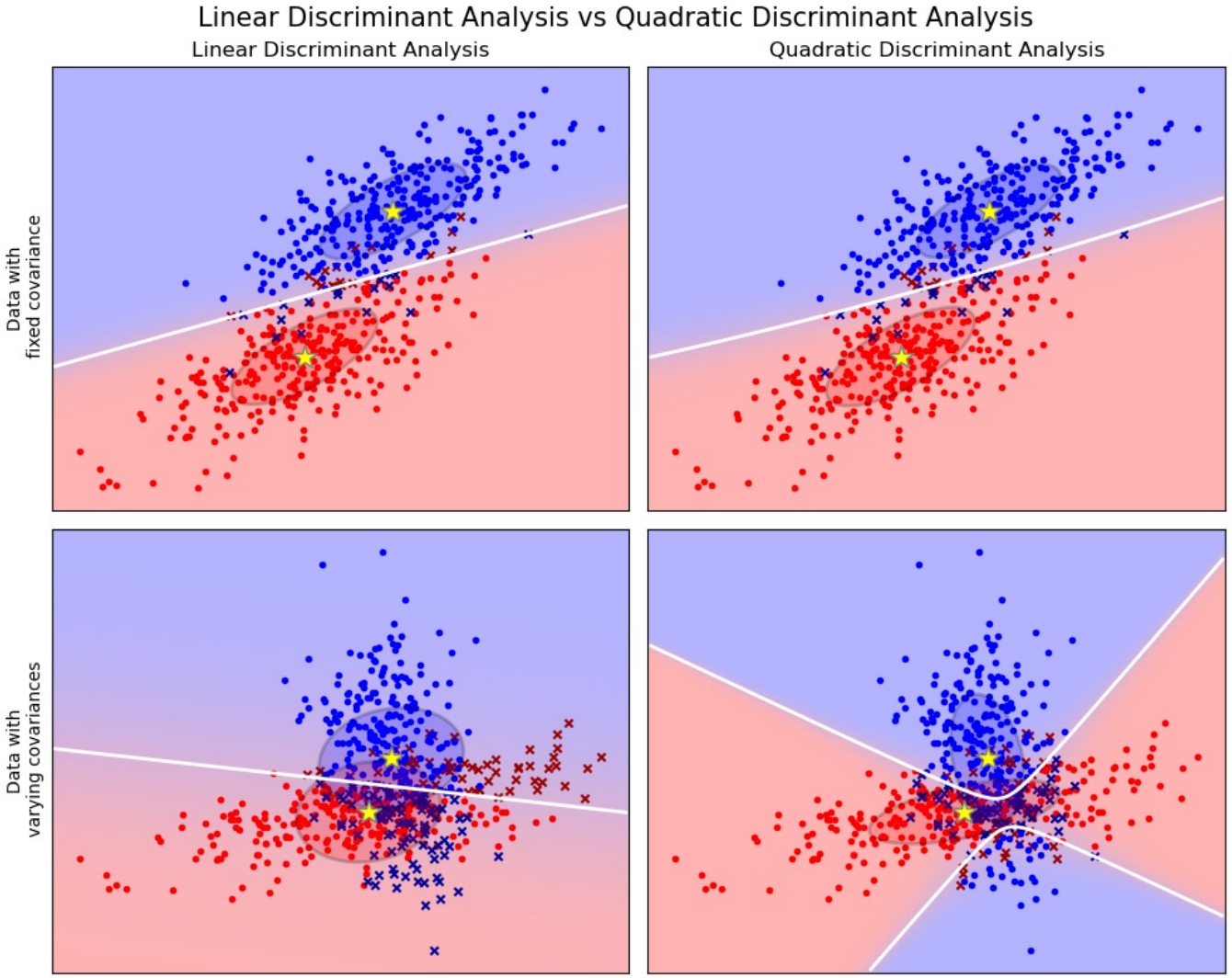
$$z = [x_1, x_2, x_1 x_2, x_1^2, x_2^2]$$

Linear and Quadratic Discriminant Analysis

- Both LDA and QDA can be derived from simple probabilistic models which model the class conditional distribution of the data.
- Predictions can then be obtained by using Bayes' rule, for each training sample:

$$P(y=k|x) = \frac{P(x|y=k)P(y=k)}{P(x)} :$$

$$= \frac{P(x|y=k)P(y=k)}{\sum_l P(x|y=l) \cdot P(y=l)}$$



Linear and Quadratic Discriminant Analysis

Mathematical formulation of the LDA and QDA classifiers

For linear and quadratic discriminant analysis, $P(x|y)$ is modeled as a multivariate Gaussian distribution with density:

$$P(x|y=k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^t \Sigma_k^{-1}(x-\mu_k)\right)$$

where d is the number of features.

QDA: According to the model above, the log of the posterior is:

$$\begin{aligned}\log P(y=k|x) &= \log P(x|y=k) + \log P(y=k) + Cst \\ &= -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^t \Sigma_k^{-1}(x-\mu_k) + \log P(y=k) + Cst\end{aligned}$$

where the constant term *Cst* corresponds to the denominator $P(x)$ – see previous slide, and other constant terms.

The predicted class is the one that maximizes this log-posterior.

Linear and Quadratic Discriminant Analysis

- LDA is a special case of QDA, where the Gaussians for each class are assumed to share the same covariance matrix: $\sum_k = \sum$ for all *k*.
- This reduces the log posterior to:

$$\log P(y=k|x) = -\frac{1}{2}(x-\mu_k)^t \Sigma^{-1}(x-\mu_k) + \log P(y=k) + Cst$$

- The log-posterior of LDA can also be written as:

$$\log P(y=k|x) = \omega_k^t x + \omega_{k0} + Cst$$

- From the above formula, it is clear that LDA has a linear decision surface.

Support Vector Machines

SVM methods are used for classification, regression and outliers detection.

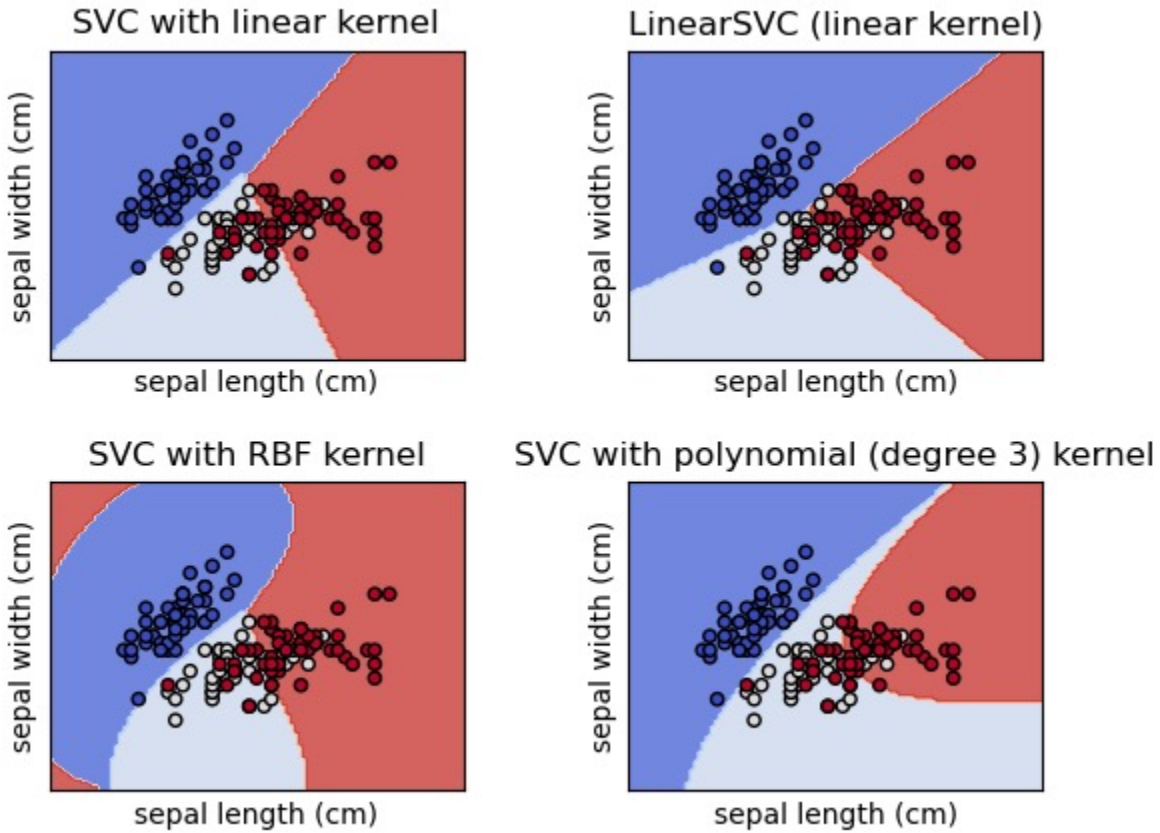
The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where # of dimensions > # of samples.
- Uses a subset of training points (support vectors) in the decision function, memory efficient!
- Various Kernel functions can be specified for the decision function.

The disadvantages of support vector machines include:

- If the # of features >> # of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

Support Vector Machines



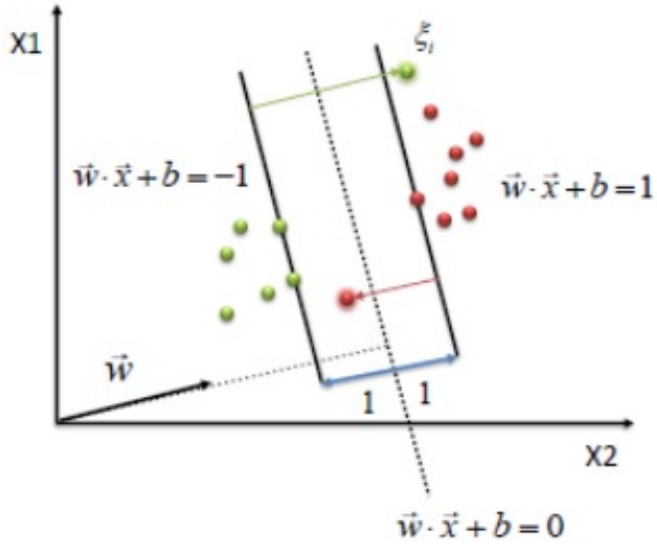
Support Vector Machines

Classification

- SVC solves the following primal problem:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$

$$\text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$$
$$\zeta_i \geq 0, i = 1, \ldots, n$$



slack variable:

ξ_i

Allow some instances to fall off the margin, but penalize them

- Intuitively, we're trying to maximize the margin (by minimizing $w^T w$), while incurring a penalty when a sample is misclassified or within the margin boundary.
- The penalty term *C* controls the strength of this penalty.

$$\text{width} = \frac{2}{\|w\|}$$

https://www.saedsayad.com/support_vector_machine.htm

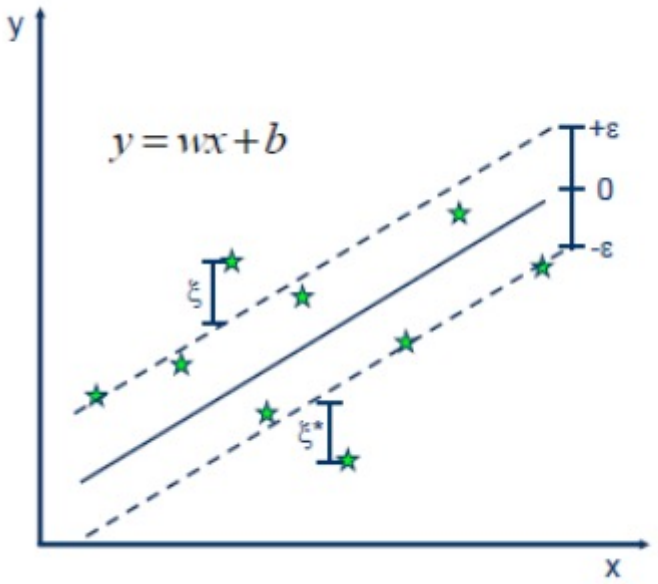
Support Vector Machines

Regression

- SVR solves the following primal problem:

$$\min_{w,b,\zeta,\zeta^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\zeta_i + \zeta_i^*)$$

$$\text{subject to } y_i - w^T \phi(x_i) - b \leq \varepsilon + \zeta_i,$$
$$w^T \phi(x_i) + b - y_i \leq \varepsilon + \zeta_i^*,$$
$$\zeta_i, \zeta_i^* \geq 0, i = 1, \ldots, n$$



- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

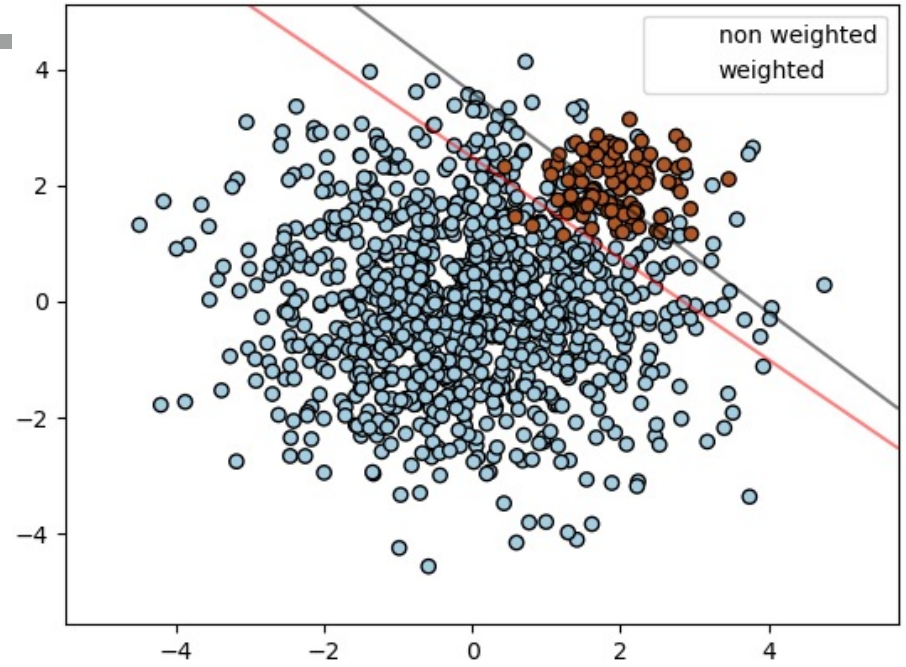
- Constraints:

$$y_i - wx_i - b \leq \varepsilon + \xi_i$$
$$wx_i + b - y_i \leq \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0$$

- Here, we are penalizing samples whose prediction is at least ε away from their true target.
- The penalty term *C* controls the strength of this penalty.

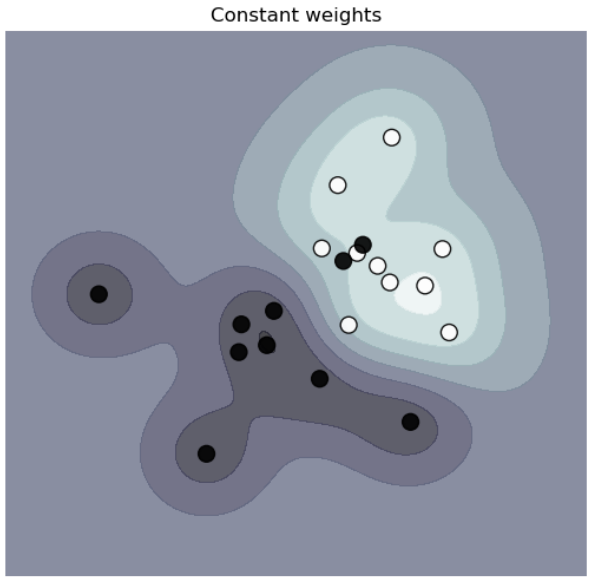
https://www.saedsayad.com/support_vector_machine_reg.htm

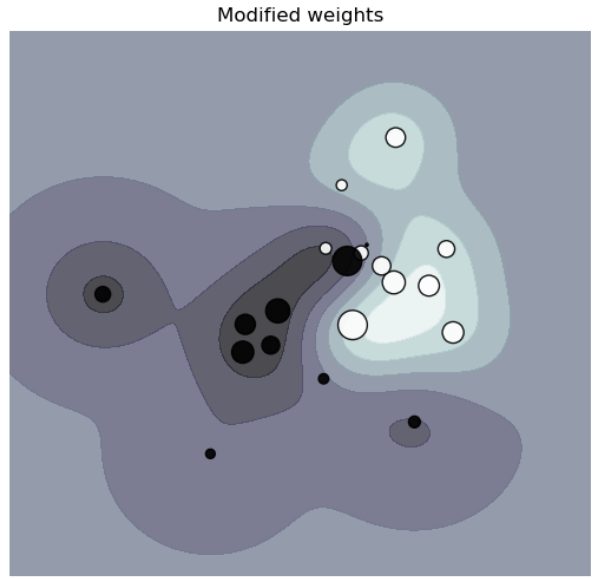
Support Vector Machines



Unbalanced problems:

In problems where it is desired to give more importance to certain classes or certain individual samples, the parameters class_weight and sample_weight can be used.





Support Vector Machines

Kernel functions

- linear: $\langle x, x' \rangle$.
- polynomial: $(\gamma \langle x, x' \rangle + r)^d$, where d is specified by parameter `degree`, r by `coef0`.
- rbf: $\exp(-\gamma \|x - x'\|^2)$, where γ is specified by parameter `gamma`, must be greater than 0.
- sigmoid $\tanh(\gamma \langle x, x' \rangle + r)$, where r is specified by `coef0`.

Nearest Neighbors

- The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these.
- The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning).
- The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice.
- Neighbors-based methods are known as non-generalizing machine learning methods, since they simply "remember" all of its training data

Nearest Neighbors

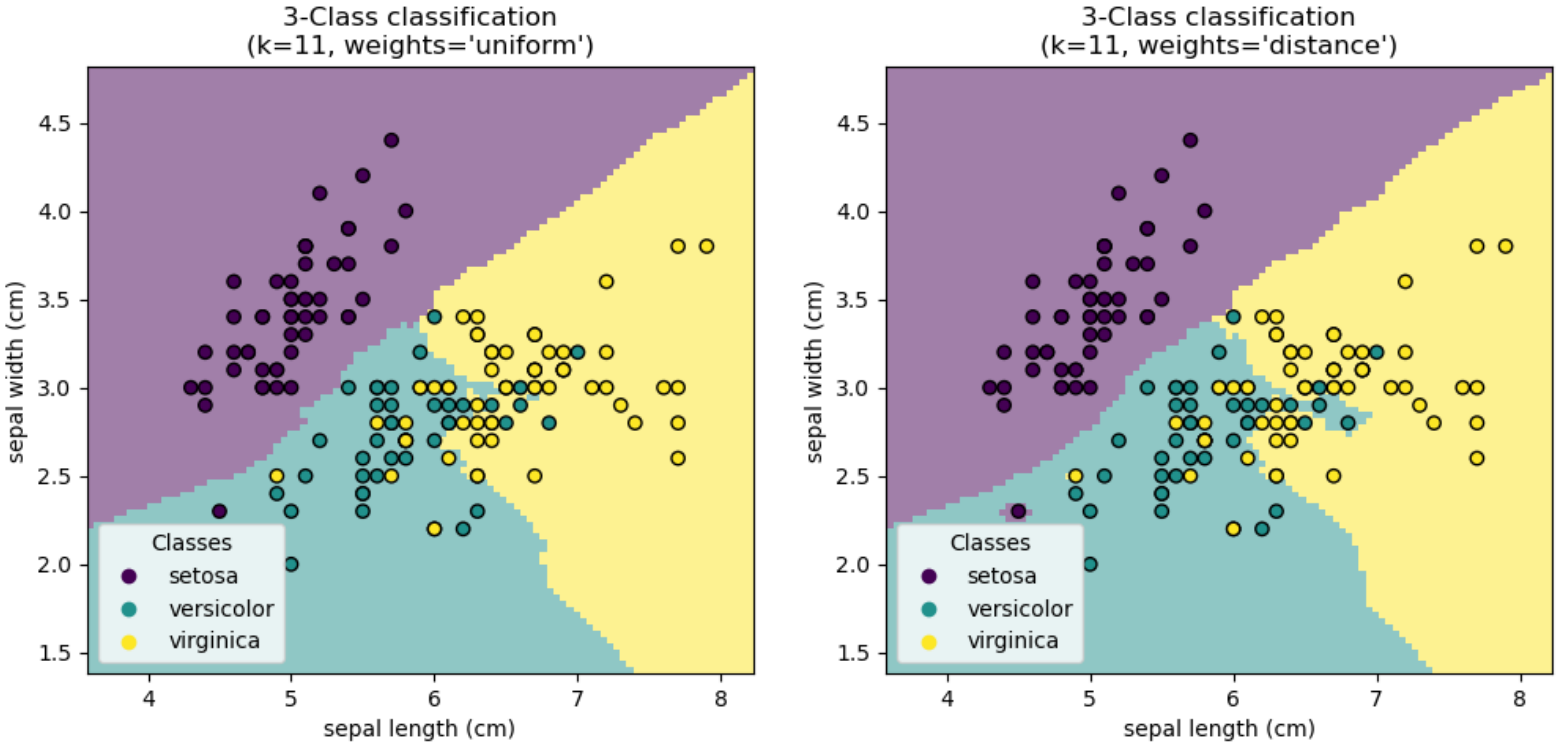
- Despite its simplicity, nearest neighbors has been successful in a large number of classification and regression problems, including handwritten digits and satellite image scenes.
- Being a non-parametric method, it is often successful in classification situations where the decision boundary is very irregular.

Nearest Neighbors

- Classification

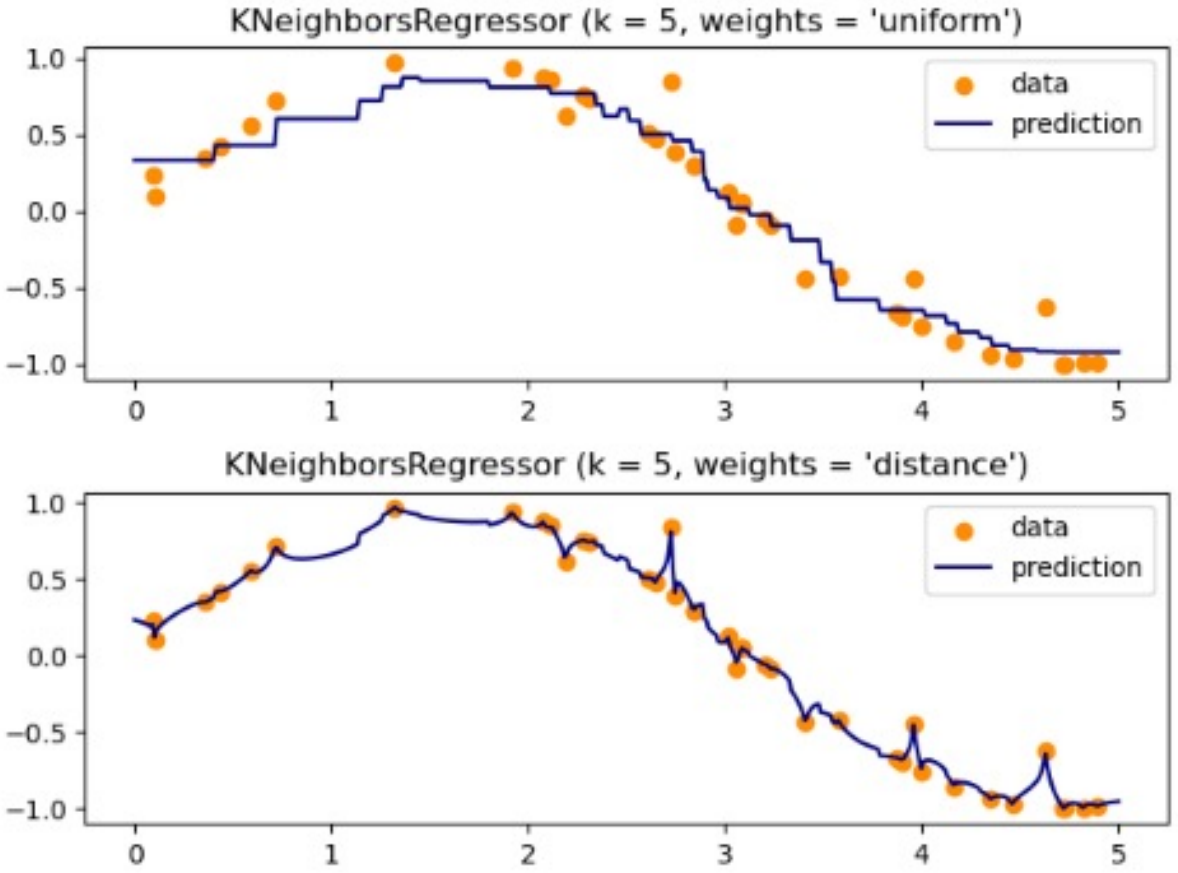
  Uniform weights: the value assigned to a query point is computed from a simple majority vote of the nearest neighbors. The default value, weights = 'uniform'.

Under some circumstances, it is better to weight the neighbors such that nearer neighbors contribute more to the fit.
weights = 'distance' assigns weights proportional to the inverse of the distance from the query point.



Nearest Neighbors

- Regression



Decision Trees

- Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression.
- 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.

Decision Trees

Advantages:

- Simple to understand and to interpret. Trees can be visualized.
- Requires little data preparation (doesn't need normalization).
- Able to handle both numerical and categorical data. However, the scikit-learn implementation does not support categorical variables for now. See algorithms for more information.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

Decision Trees

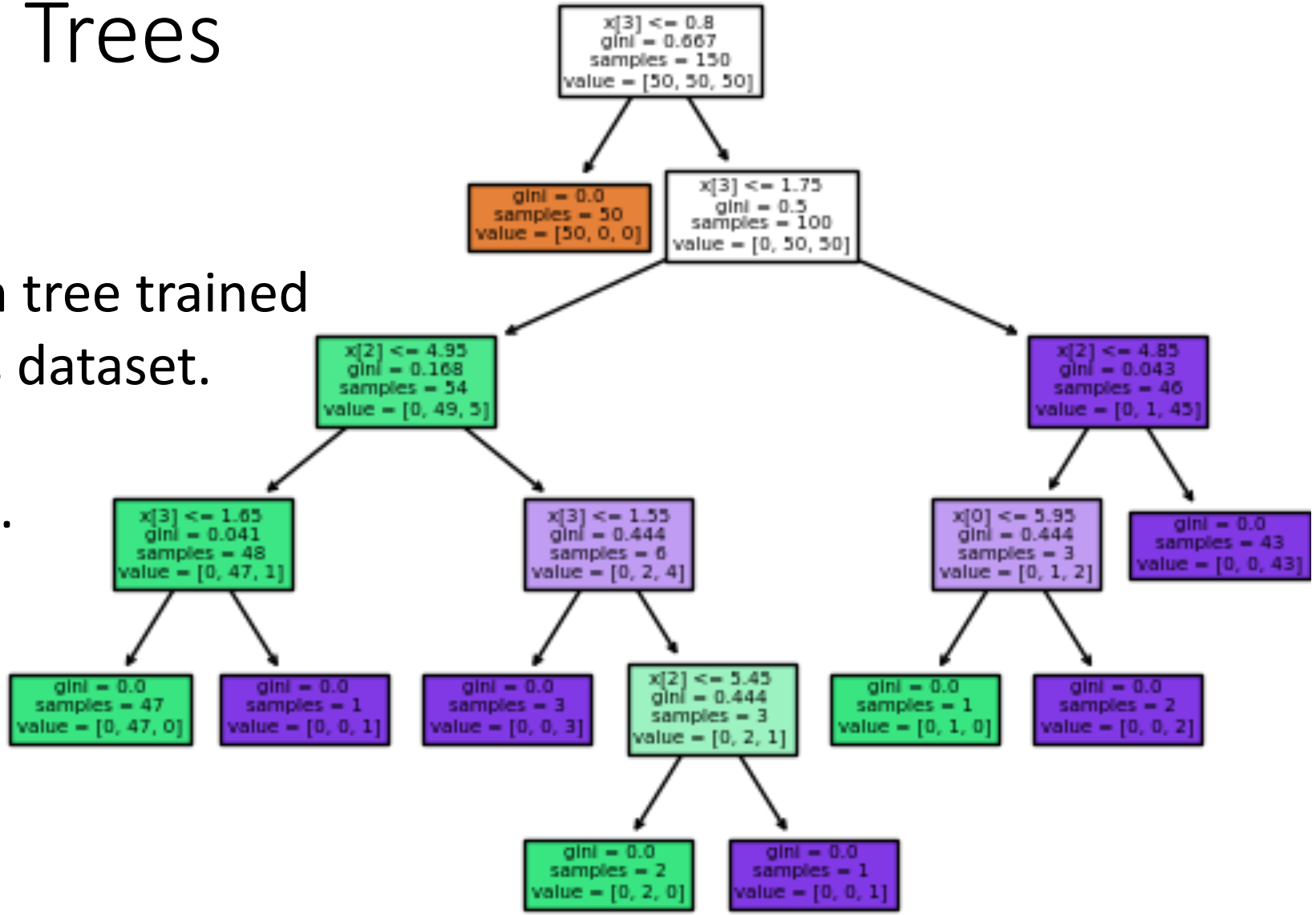
Disadvantages:

- Decision-tree learners can create over-complex trees that do not generalize the data well.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated.
- Predictions of decision trees are neither smooth nor continuous. Therefore, they are not good at extrapolation.
- Learning an optimal decision tree is an NP-complete problem => cannot guarantee to return the globally optimal decision tree.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

Decision Trees

Classification

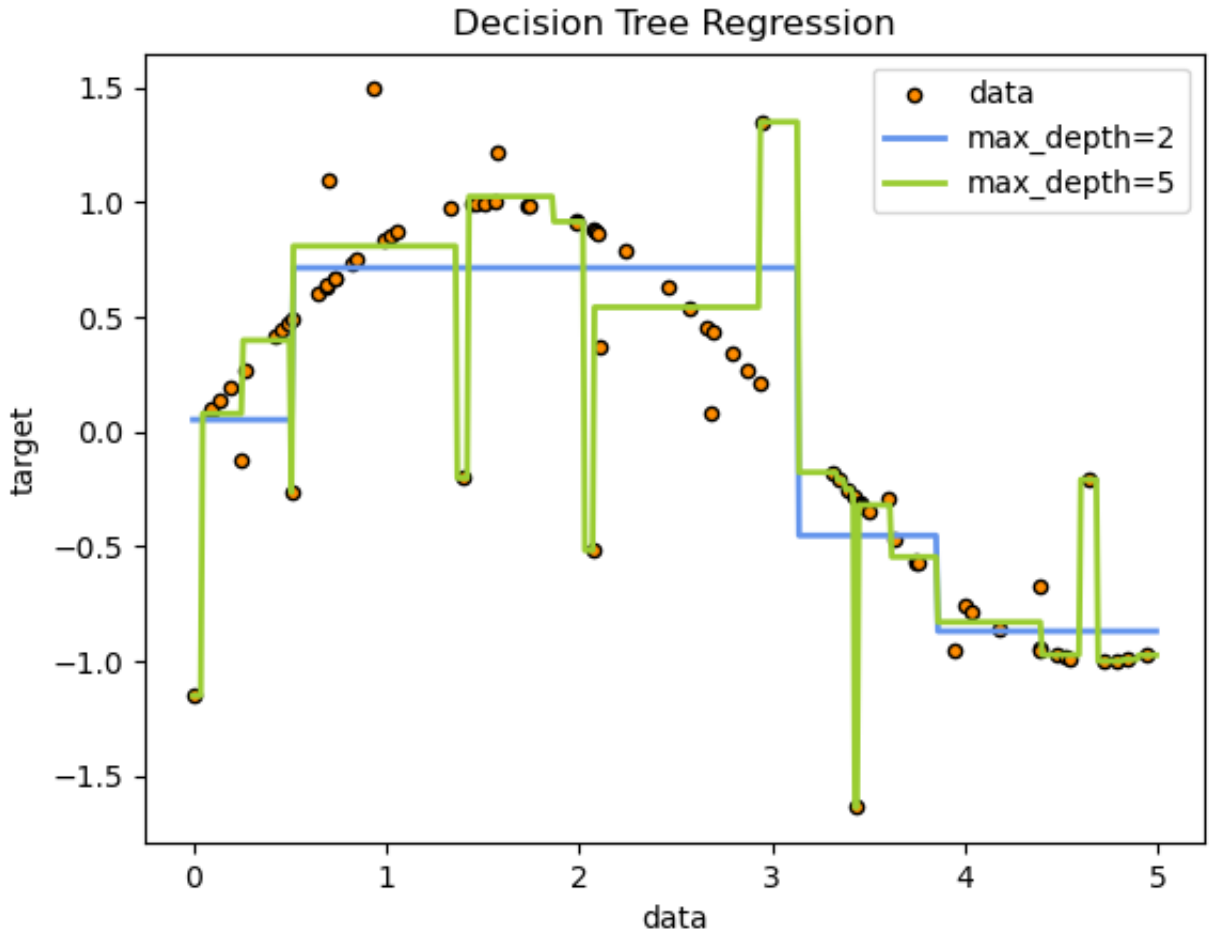
- A decision tree trained on the iris dataset.
- Gini score.



Decision Trees

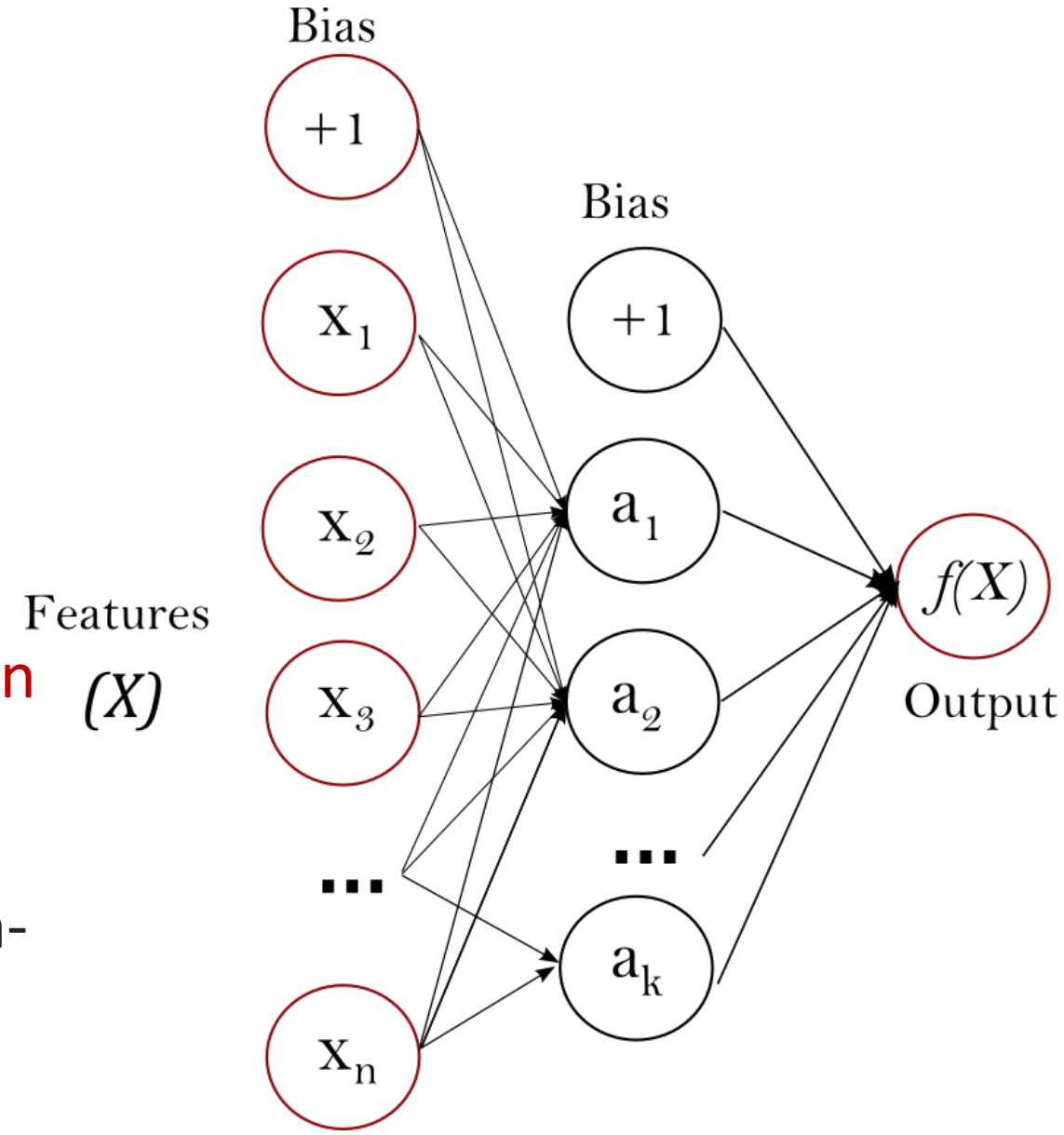
Regression

- Complexity control
- Max depth



Neural network models

- Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function $f(\cdot)$: $R^m \rightarrow R^o$ by training on a dataset.
- It can learn a non-linear function approximator for either classification or regression.
- Between the input and the output layer, there can be one or more non-linear layers, called hidden layers.



Neural network models

Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation

$$w_1 x_1 + w_2 x_2 + \ldots + w_m x_m$$

followed by a non-linear activation function

$$g(\cdot) : R \rightarrow R$$

Neural network models

Advantages:

- Capability to learn non-linear models.
- Capability to learn models in real-time (on-line learning) using partial_fit.

Disadvantages:

- MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore different random weight initializations can lead to different validation accuracy.
- MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations.
- MLP is sensitive to feature scaling.

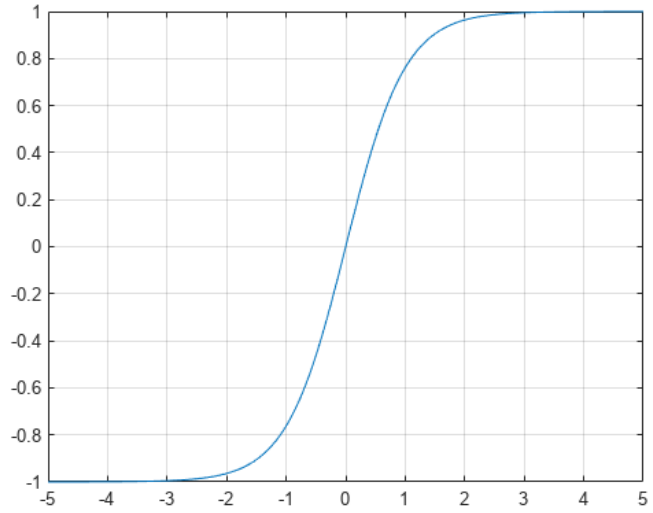
Neural network models

MLP learns the function $f(x) = W_2 g(W_1^T x + b_1) + b_2$

W_1, W_2 represent the weights of the input layer and hidden layer, respectively; and b_1, b_2 represent the bias added to the hidden layer and the output layer, respectively.

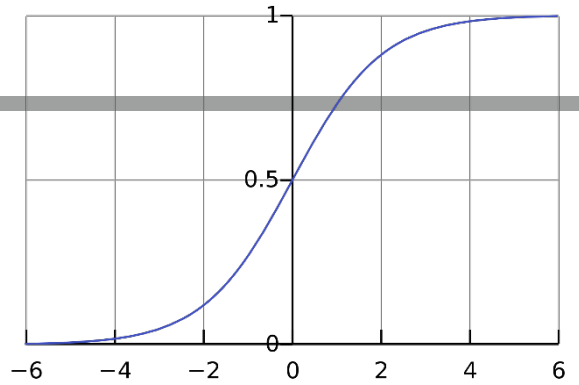
The default activation function is the hyperbolic tan

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$



https://www.mathworks.com/help/matlab/ref/tanh.html

Neural network models

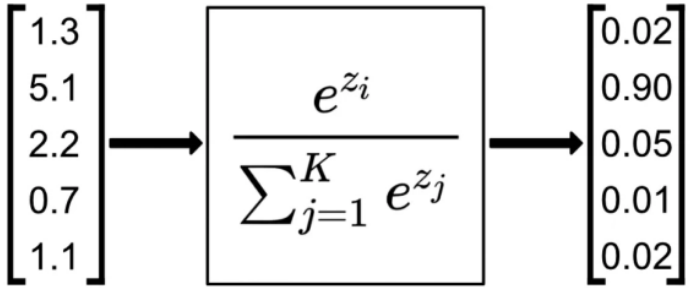


- For binary classification, *f*(x) passes through the logistic function

$$g(z) = 1/(1 + e^{-z})$$

- For more than two classes, *f*(x) passes through the softmax function

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{l=1}^{k} \exp(z_l)}$$



- In regression, the output remains as *f*(x). Therefore, output activation function is just the identity function.

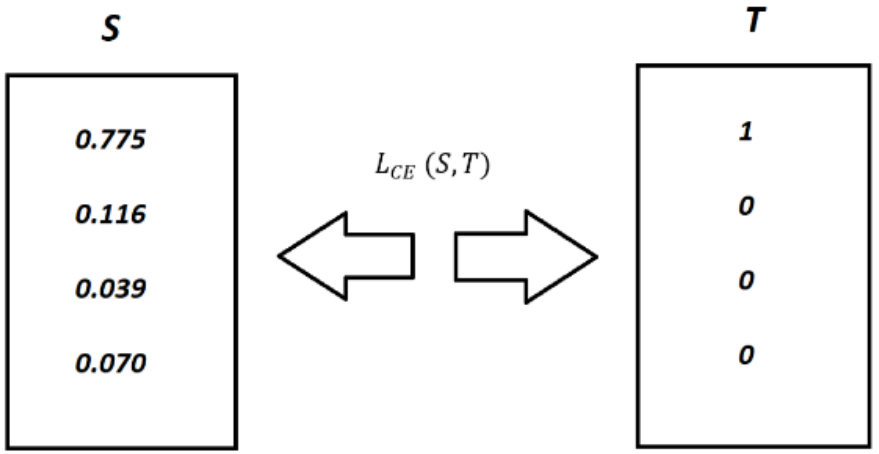
https://en.wikipedia.org/wiki/Logistic_function
https://www.singlestore.com/blog/a-guide-to-softmax-activation-function/

Neural network models

- For classification, MLP uses Average Cross-Entropy, which in binary case is given as,

$$Loss(\hat{y}, y, W) = -\frac{1}{n}\sum_{i=0}^{n}(y_i \ln \hat{y}_i + (1 - y_i) \ln (1 - \hat{y}_i)) + \frac{\alpha}{2n}||W||_2^2$$

- Case 1:



$$L_{CE} = -\left[1log_2(0.775) + 0log_2(0.116) + 0log_2(0.039) + 0log_2(0.070)\right]$$

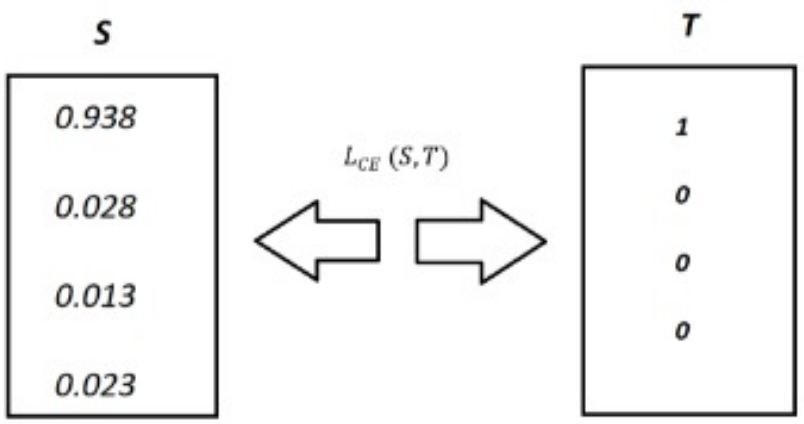
$$L_{CE} = -log_2(0.775)$$

$$L_{CE} = 0.3677$$

https://www.shopdev.co/blog/cross-entropy-in-machine-learning

Neural network models

- Case 2:



$$L_{CE} = -[1log_2(0.938) + 0 + 0 + 0]$$

$$L_{CE} = 0.095$$

- For regression, MLP uses the Mean Square Error loss function; written as,

$$Loss(\hat{y}, y, W) = \frac{1}{2n}\sum_{i=0}^{n}||\hat{y}_i - y_i||_2^2 + \frac{\alpha}{2n}||W||_2^2$$

https://www.shopdev.co/blog/cross-entropy-in-machine-learning

Neural network models

- To find the model parameters, we minimize the regularized training error given by

$$E(w,b) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$

- MLP trains using Stochastic Gradient Descent, Adam.

Stochastic Gradient Descent (SGD) updates parameters using the gradient of the loss function with respect to a parameter that needs adaptation, i.e.

$$w \leftarrow w - \eta(\alpha\frac{\partial R(w)}{\partial w} + \frac{\partial Loss}{\partial w})$$

where η is the learning rate which controls the step-size in the parameter space search. *Loss* is the loss function used for the network. *R*(ω) is the regularization term.

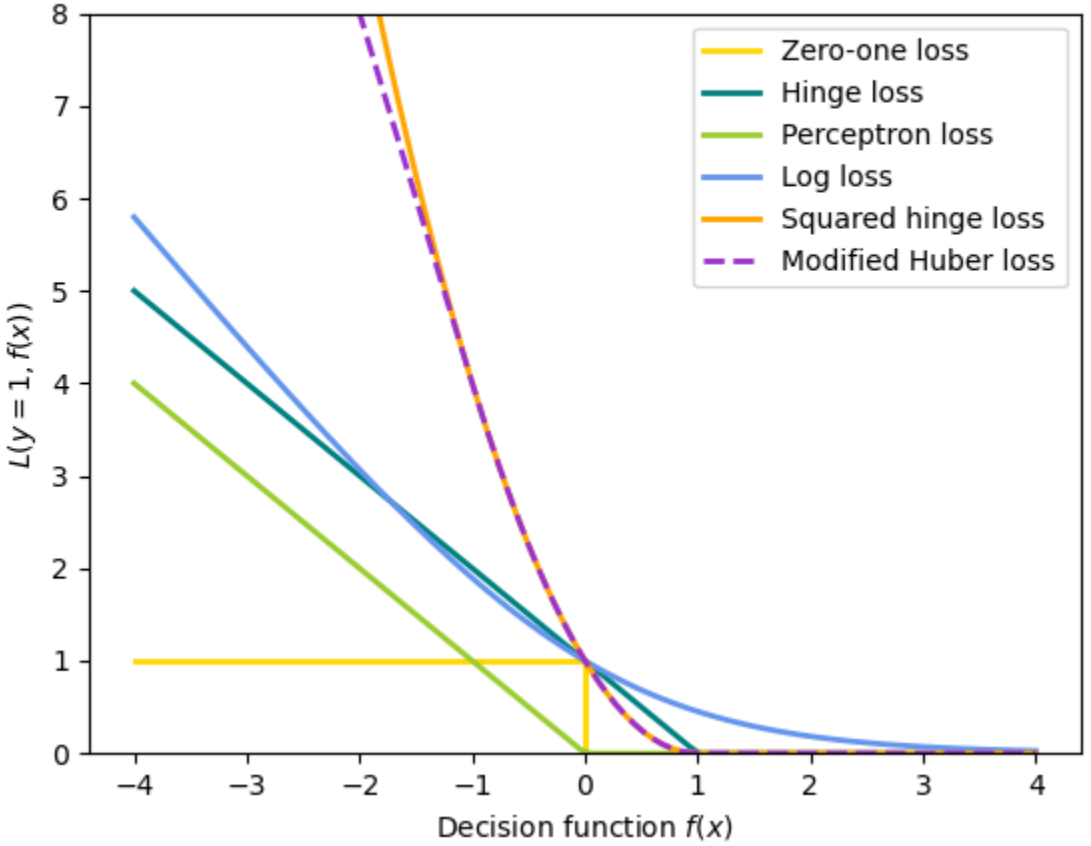
Neural network models

- Adam is similar to SGD in a sense that it is a stochastic optimizer, but it can automatically adjust the amount to update parameters based on adaptive estimates of lower-order moments.
- With SGD or Adam, training supports online and mini-batch learning.

Loss functions

- Hinge: $L(y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))$
- Perceptron: $L(y_i, f(x_i)) = \max(0, -y_i f(x_i))$
- Log Loss: $L(y_i, f(x_i)) = \log(1 + \exp(-y_i f(x_i)))$
- Squared Error: $L(y_i, f(x_i)) = \frac{1}{2}(y_i - f(x_i))^2$
- Epsilon-Insensitive:

$$L(y_i, f(x_i)) = \max(0, |y_i - f(x_i)| - \varepsilon)$$



Ensemble methods

- Voting,
- Stacking,
- Boosting,
- Bagging,

Voting Classifier

- The idea is to combine conceptually different machine learning classifiers and use a majority vote or the average predicted probabilities (soft vote) to predict the class labels.
- Such a classifier can be useful for a set of equally well performing models in order to balance out their individual weaknesses.
- EX:

  classifier 1 -> class 1

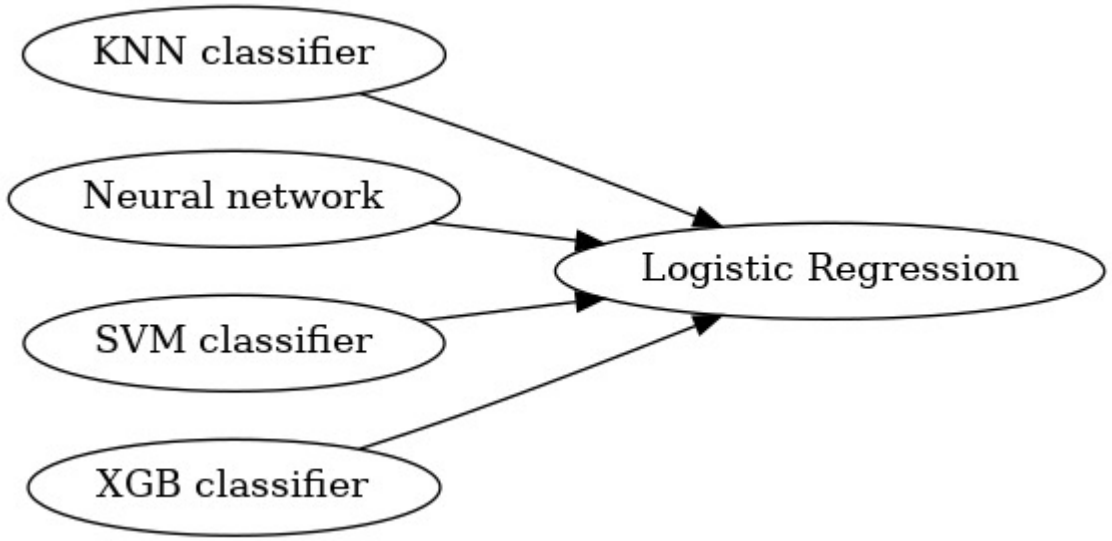
  classifier 2 -> class 1

  classifier 3 -> class 2

  The VotingClassifier (with voting='hard') would classify the sample as “class 1” based on the majority class label.

Stacking

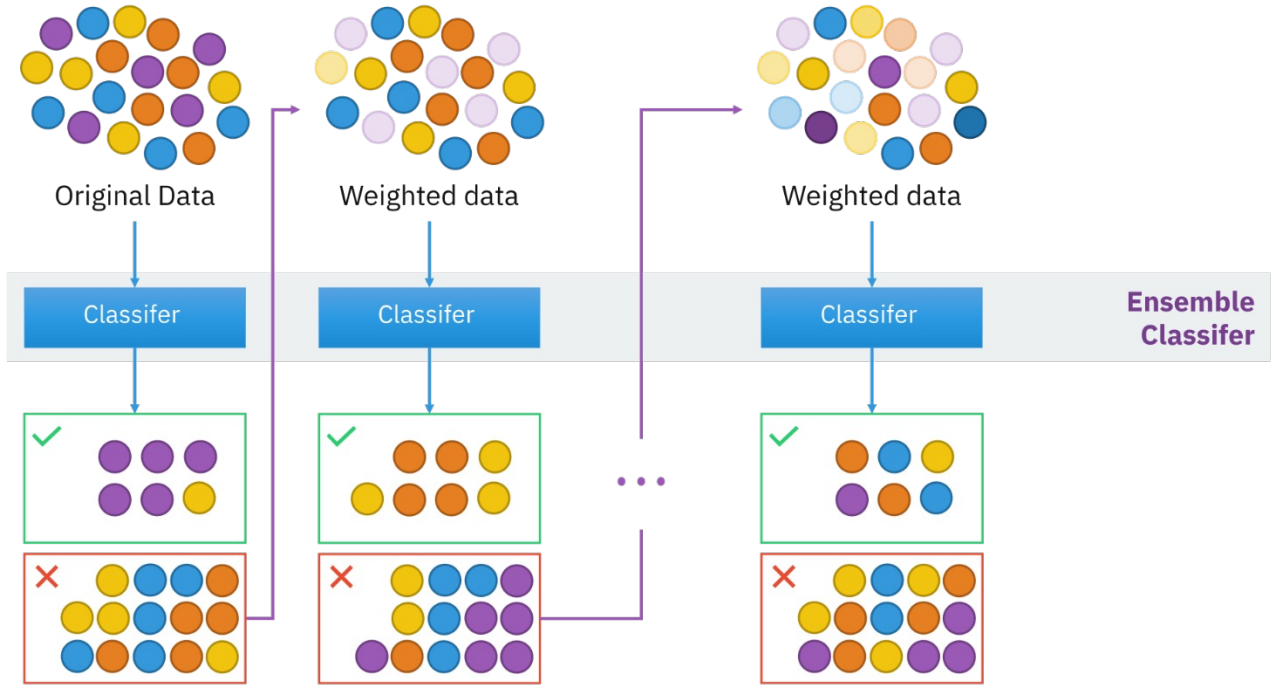
- The predictions of each individual estimator are stacked together and used as input to a final estimator to compute the prediction.
- This final estimator is trained through cross-validation.



https://wolpert.readthedocs.io/en/latest/user_guide/intro.html

Boosting

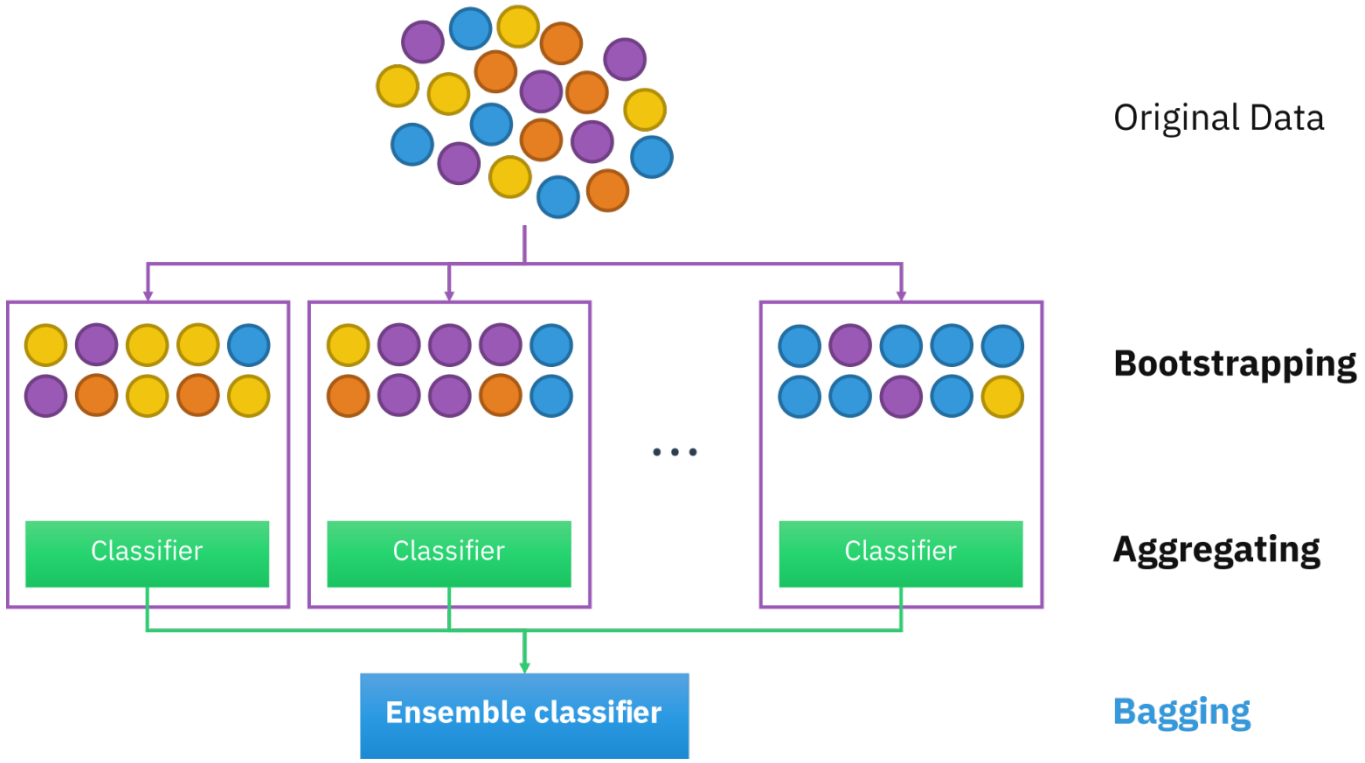
1. Form a large set of simple features
2. Initialize weights for training images
3. For T rounds
    1. Normalize the weights
    2. For available features from the set, train a classifier using a single feature and evaluate the training error
    3. Choose the classifier with the lowest error
    4. Update the weights of the training images: increase if classified wrongly by this classifier, decrease if correctly
4. Form the final strong classifier as the linear combination of the T classifiers (coefficient larger if training error is small)



https://en.wikipedia.org/wiki/Boosting_(machine_learning)

Bagging (Bootstrap aggregating)

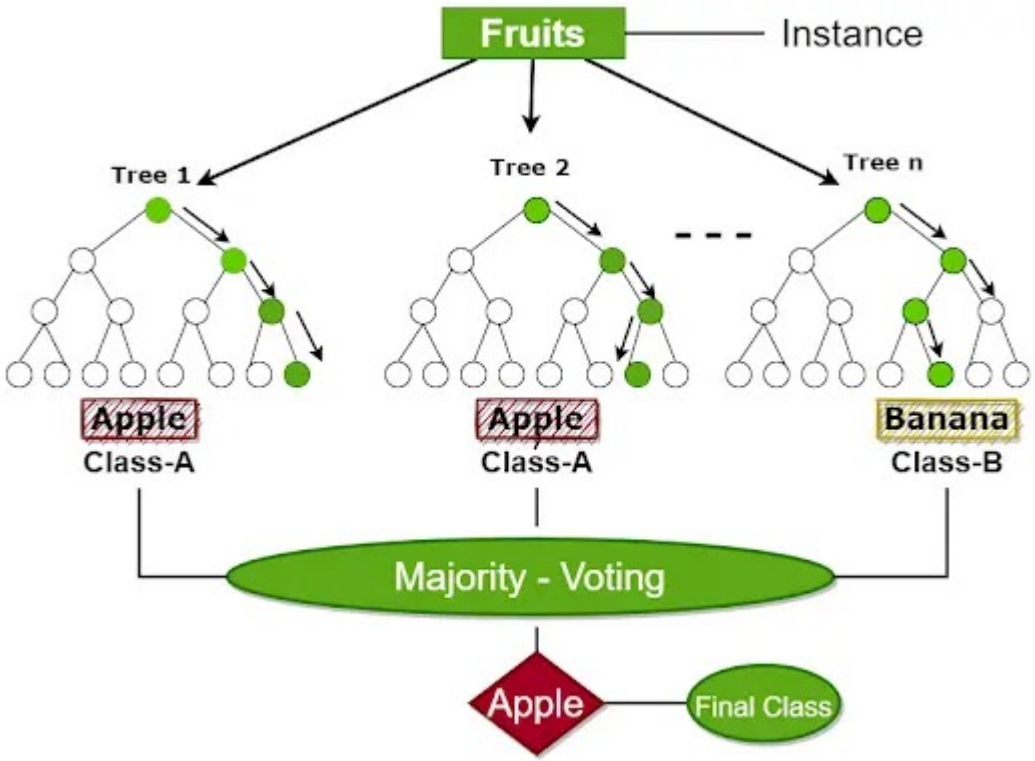
- Reduce variance
- A random sample of data in a training set is selected with replacement.
- After several data samples are generated, these weak models are then trained independently.
- The average or majority of those predictions yield a more accurate estimate.



https://www.ibm.com/

Random Forest

- In random forest, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set.
- Furthermore, when splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size max_features.



https://www.turing.com/kb/random-forest-algorithm

Aspect	Random Forest	Decision Tree
Nature	Ensemble of multiple decision trees	Single decision tree
Bias-Variance Trade-off	Lower variance, reduced overfitting	Higher variance, prone to overfitting
Predictive Accuracy	Generally higher due to ensemble	Prone to overfitting, may vary
Robustness	More robust to outliers and noise	Sensitive to outliers and noise
Training Time	Slower due to multiple tree construction	Faster as it builds a single tree
Interpretability	Less interpretable due to ensemble	More interpretable as a single tree
Feature Importance	Provides feature importance scores	Provides feature importance, but less reliable
Usage	Suitable for complex tasks, high-dimensional data	Simple tasks, easy interpretation

https://www.analyticsvidhya.com/

Unsupervised learning

- Clustering

Clustering is a data mining technique which groups unlabeled data based on their similarities or differences.

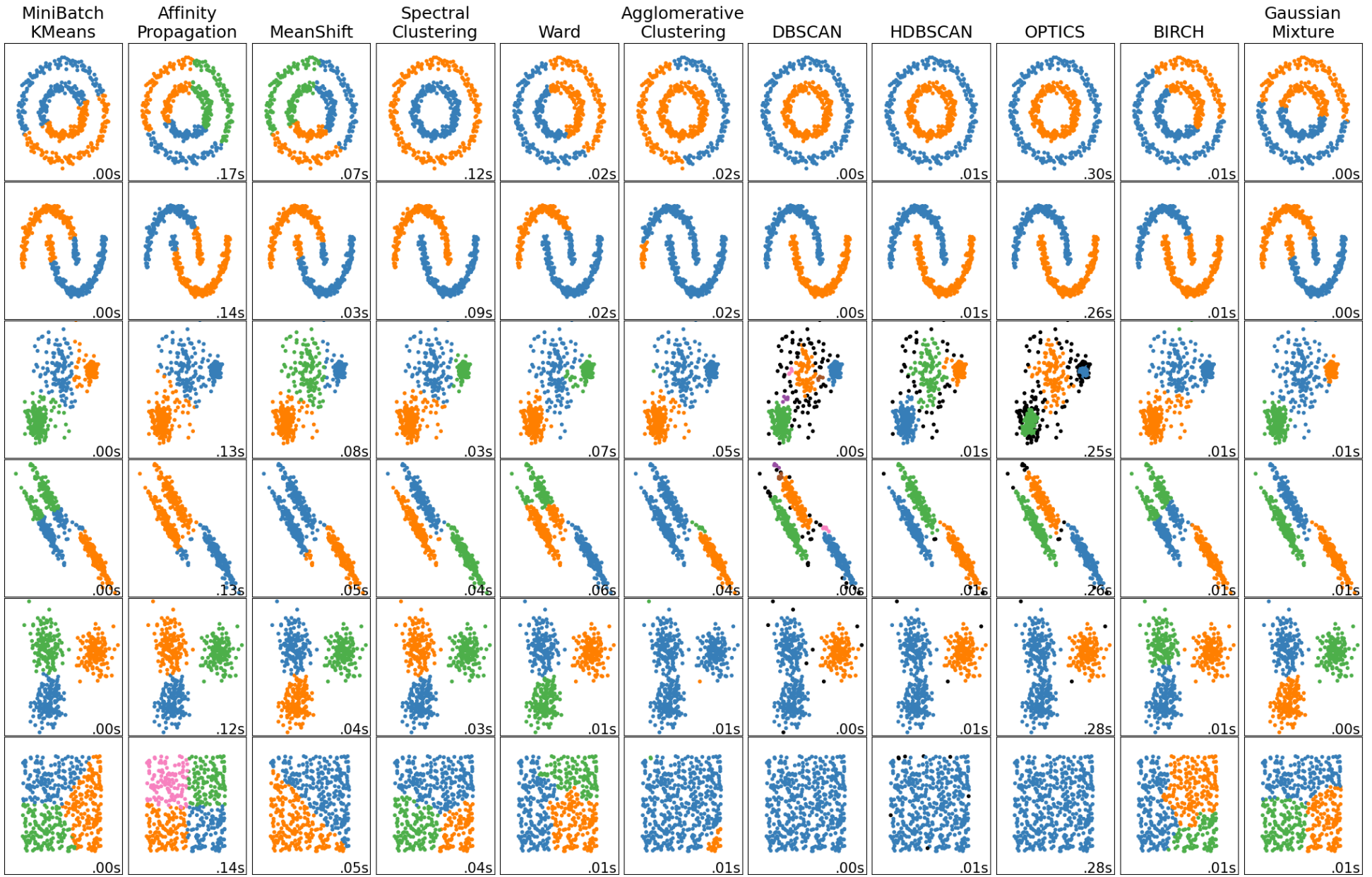
- Dimensionality reduction

Dimensionality reduction reduces the number of data inputs to a manageable size while also preserving the integrity of the dataset as much as possible.

https://www.ibm.com/

Clustering

- A comparison of the clustering algorithms in scikit-learn



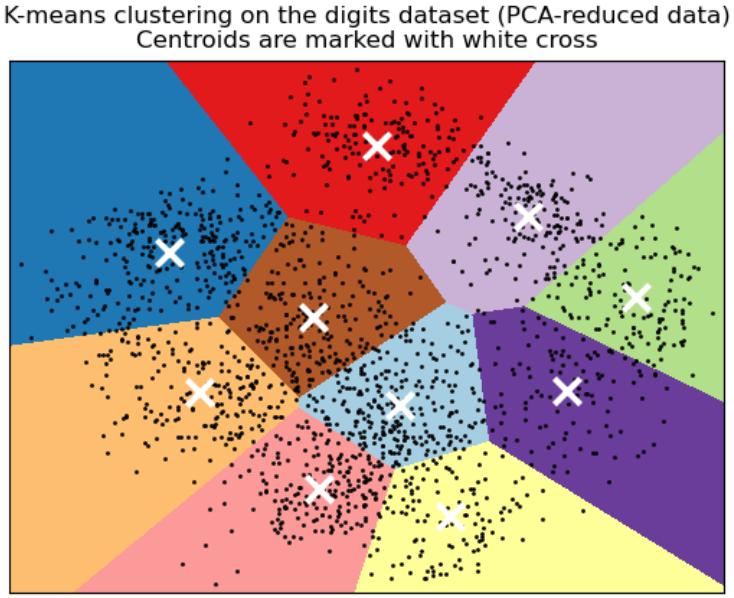
K-means

- The k-means algorithm divides a set of *N* samples *X* into *K* disjoint clusters *C*, each described by the mean μ_i of the samples in the cluster. The means are commonly called the cluster "centroids"; note that they are not, in general, points from *X*, although they live in the same space.
- The K-means algorithm aims to choose centroids that minimize the inertia, or within-cluster sum-of-squares criterion:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

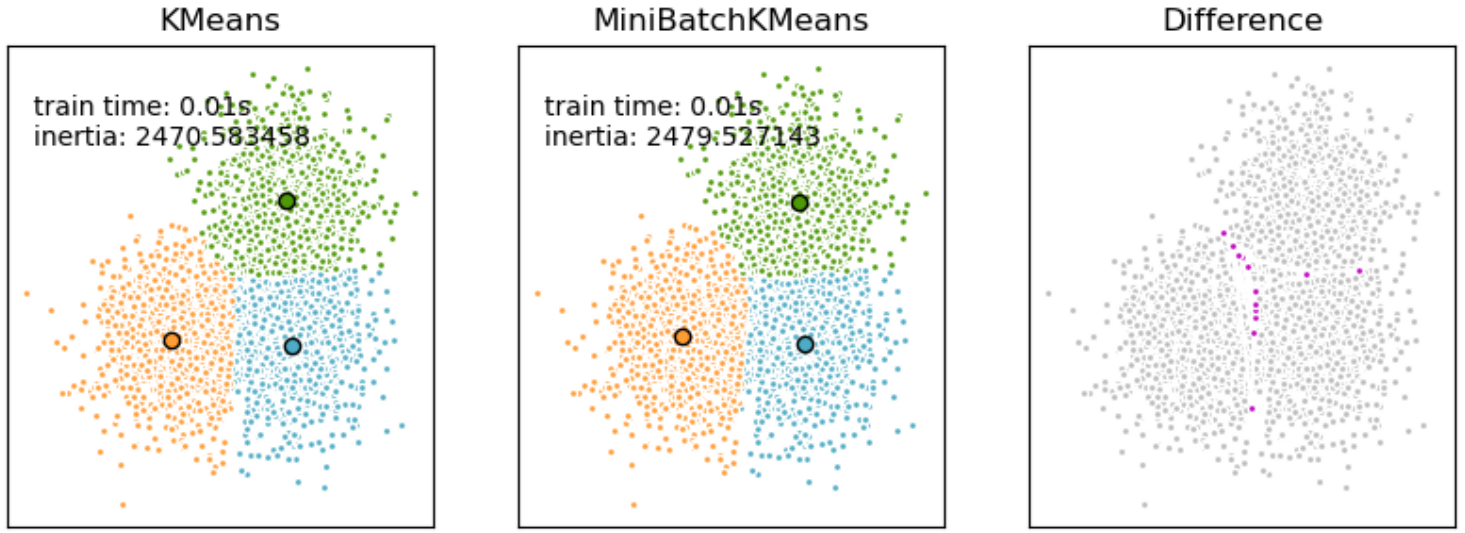
K-means

- A demo of K-Means clustering on the handwritten digits data



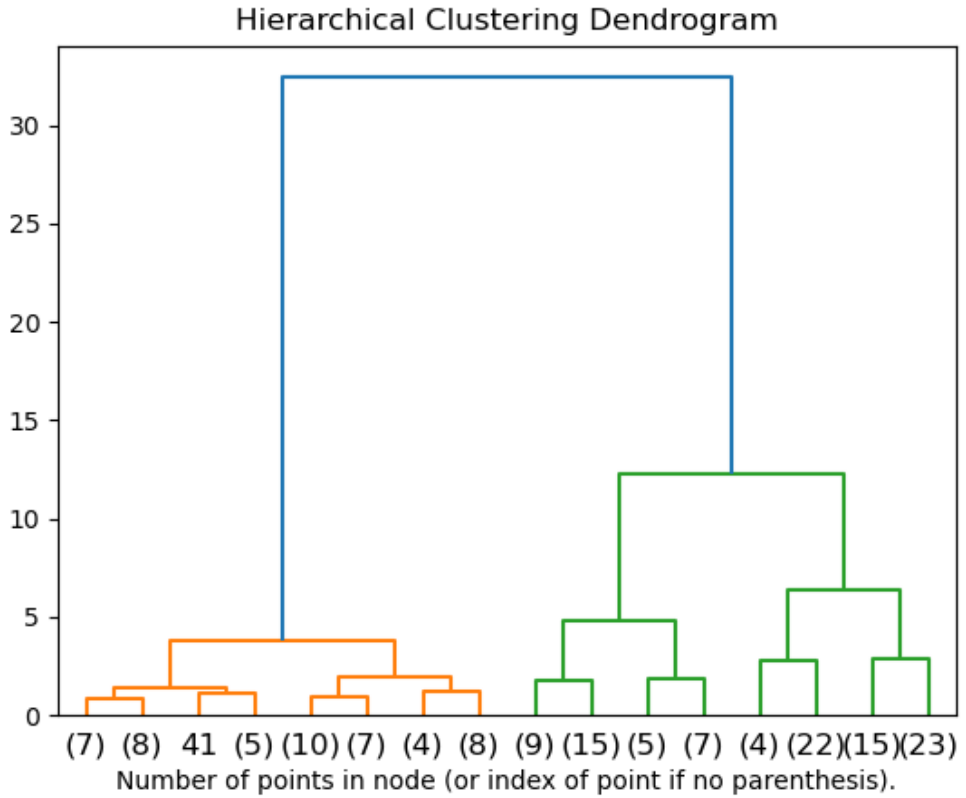
- Mini Batch K-Means:

Mini-batches are subsets of the input data, randomly sampled in each training iteration.



Hierarchical clustering

- Hierarchical clustering is a general family of clustering algorithms that build nested clusters by merging or splitting them successively.
- This hierarchy of clusters is represented as a tree (or dendrogram).
- The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.



Hierarchical clustering

- The Agglomerative Hierarchical Clustering uses a bottom up approach: each observation starts in its own cluster, and clusters are successively merged together.

The linkage criteria determines the metric used for the merge strategy:

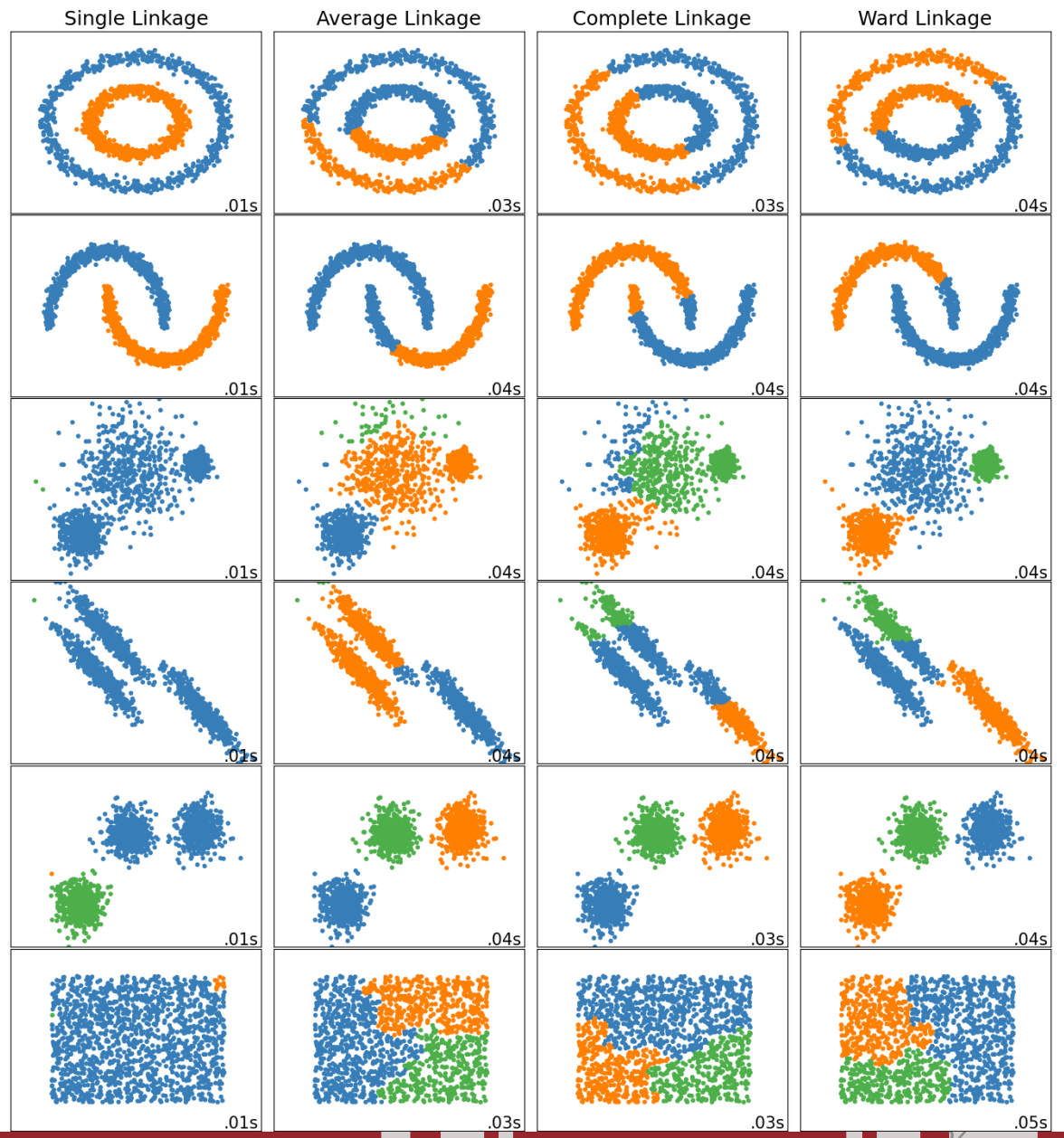
- **Ward** minimizes the sum of squared differences within all clusters. It is a variance-minimizing approach.
- **Maximum** or **complete linkage** minimizes the maximum distance between observations of pairs of clusters.
- **Average linkage** minimizes the average of the distances between all observations of pairs of clusters.
- **Single linkage** minimizes the distance between the closest observations of pairs of clusters.

Hierarchical clustering

- Agglomerative cluster has a “rich get richer” behavior that leads to uneven cluster sizes.

In this example:

- **Single linkage** is the worst strategy.
- **Ward** gives the most regular sizes.
- For non Euclidean metrics, **average linkage** is a good alternative.
- Single linkage can be computed very efficiently and can therefore be useful to provide hierarchical clustering of larger datasets.

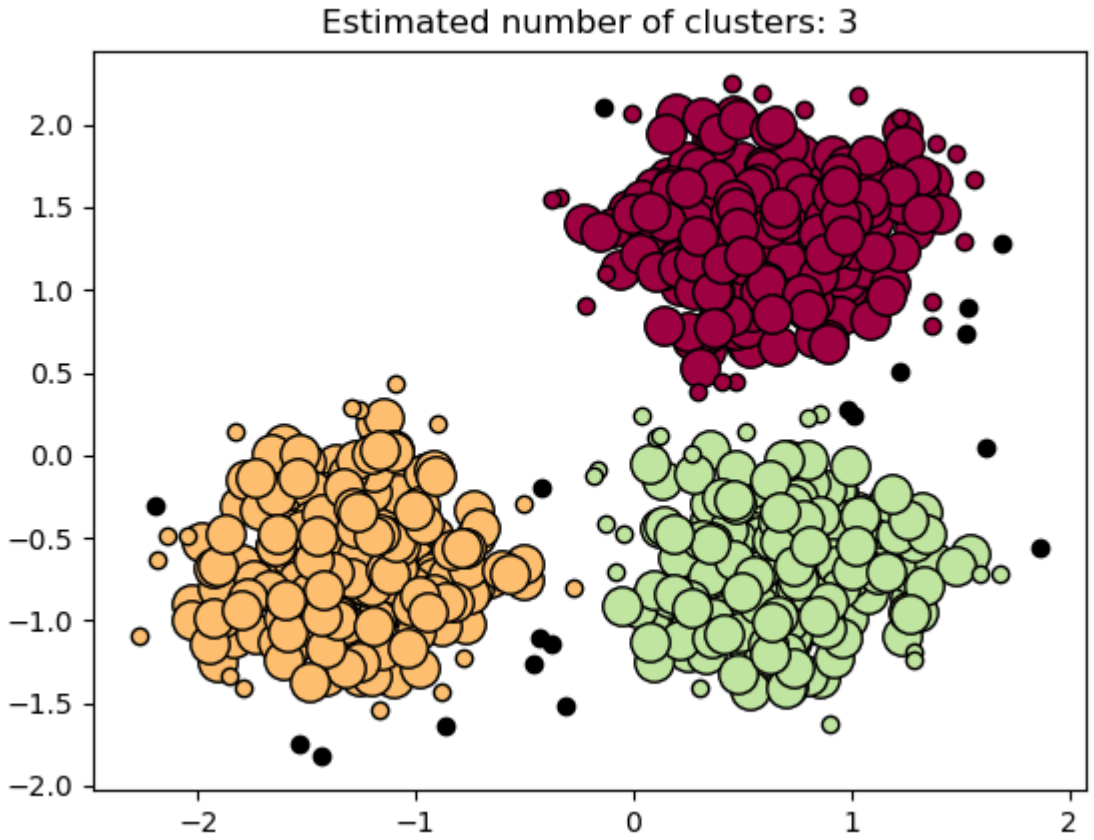


DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- Define a core sample as being a sample in the dataset such that there exist min_samples other samples within a distance of eps, which are defined as neighbors of the core sample.
- This tells us that the core sample is in a dense area of the vector space.
- A cluster is a set of core samples that can be built by recursively taking a core sample,
  - finding all of its neighbors that are core samples,
  - finding all of their neighbors that are core samples, and so on.

DBSCAN

- Large circles indicating core samples found by the algorithm.
- Smaller circles are non-core samples that are still part of a cluster.
- Outliers are indicated by black points below.



Clustering performance evaluation

When the ground truth is known

- Rand index
- Homogeneity, completeness and V-measure
- Fowlkes-Mallows scores

When the ground truth is unknown

- Silhouette Coefficient
- Davies-Bouldin Index

Clustering performance evaluation - Rand index

Rand index

- Given the knowledge of the ground truth class assignments labels_true and our clustering algorithm assignments of the same samples labels_pred.
- Rand index is a function that measures the similarity of the two assignments, ignoring permutations:

$$\mathrm{RI} = \frac{a + b}{C_2^{n_{samples}}}$$

a is number of pairs of elements that are in the same set in G and in the same set in K
b is the number of pairs of elements that are in different sets in G and in different sets in K
c_2^n is the total number of possible pairs in the dataset.
G is a ground truth class assignment and *K* the clustering

```
>>> labels_true = [0, 0, 0, 1, 1, 1]
>>> labels_pred = [0, 0, 1, 1, 2, 2]
>>> metrics.rand_score(labels_true, labels_pred)
0.66...
```

Clustering performance evaluation - Rand index

- Advantages:
  1. Interpretability
  2. Random (uniform) label assignments have an adjusted Rand index score close to 0.
  3. Bounded range in [0, 1] for unadjusted RI & [-1, 1] for adjusted RI.
  4. No assumption is made on the cluster structure
- Disadvantages:
  1. Requires knowledge of the ground truth classes.

Clustering performance evaluation - Homogeneity, completeness and V-measure

- homogeneity: each cluster contains only members of a single class.
- completeness: all members of a given class are assigned to the same cluster.

$$v = \frac{(1+\beta) \times \text{homogeneity} \times \text{completeness}}{(\beta \times \text{homogeneity} + \text{completeness})}$$

beta defaults to a value of 1.0

Clustering performance evaluation - Homogeneity, completeness and V-measure

- Advantages
  1. Bounded scores: 0.0 is as bad as it can be, 1.0 is a perfect score.
  2. qualitatively analyzed in terms of homogeneity and completeness
  3. No assumption is made on the cluster structure
- Disadvantages:
  1. not normalized with regards to random labeling
  2. random labeling won't yield zero scores especially when the number of clusters is large

  This problem can safely be ignored when the number of samples is more than a thousand and the number of clusters is less than 10.

Clustering performance evaluation - Fowlkes-Mallows scores

- The Fowlkes-Mallows index (sklearn.metrics.fowlkes_mallows_score) can be used when the ground truth class assignments of the samples is known. The Fowlkes-Mallows score FMI is defined as the geometric mean of the pairwise precision and recall:

$$\mathrm{FMI} = \frac{\mathrm{TP}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})}}$$

		Predicted condition	
	Total population = P + N	**Predicted positive**	**Predicted negative**
Actual condition	**Positive (P)** [a]	**True positive** (TP), hit[b]	**False negative** (FN), miss, underestimation
	Negative (N)[d]	**False positive** (FP), false alarm, overestimation	**True negative** (TN), correct rejection[e]

Clustering performance evaluation - Fowlkes-Mallows scores

- Advantages:
  1. Random (uniform) label assignments have a FMI score close to 0.
  2. Upper-bounded at 1
  3. No assumption is made on the cluster structure
- Disadvantages:
  1. Requires knowledge of the ground truth classes

Clustering performance evaluation - Silhouette Coefficient

- A higher Silhouette Coefficient score relates to a model with better defined clusters.
- The Silhouette Coefficient is defined for each sample and is composed of two scores:
  - a: The mean distance between a sample and all other points in the same class.
  - b: The mean distance between a sample and all other points in the next nearest cluster.
- The Silhouette Coefficient s for a single sample is then given as

$$s = \frac{b - a}{max(a, b)}$$

Clustering performance evaluation - Silhouette Coefficient

- Advantages:
  1. The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters.
  2. The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster.
- Disadvantages:
  1. The Silhouette Coefficient is generally higher for convex clusters than other concepts of clusters, such as density based clusters like those obtained through DBSCAN

Clustering performance evaluation - Davies-Bouldin index

- A lower Davies-Bouldin index relates to a model with better separation between the clusters.
- This index signifies the average 'similarity' between clusters, where the similarity is a measure that compares the distance between clusters with the size of the clusters themselves.
- Zero is the lowest possible score. Values closer to zero indicate a better partition.

Clustering performance evaluation - Davies-Bouldin index

$$DB = \frac{1}{k}\sum_{i=1}^{k}\max_{i\neq j} R_{ij}$$

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

s_i, the average distance between each point of cluster i and the centroid of that cluster

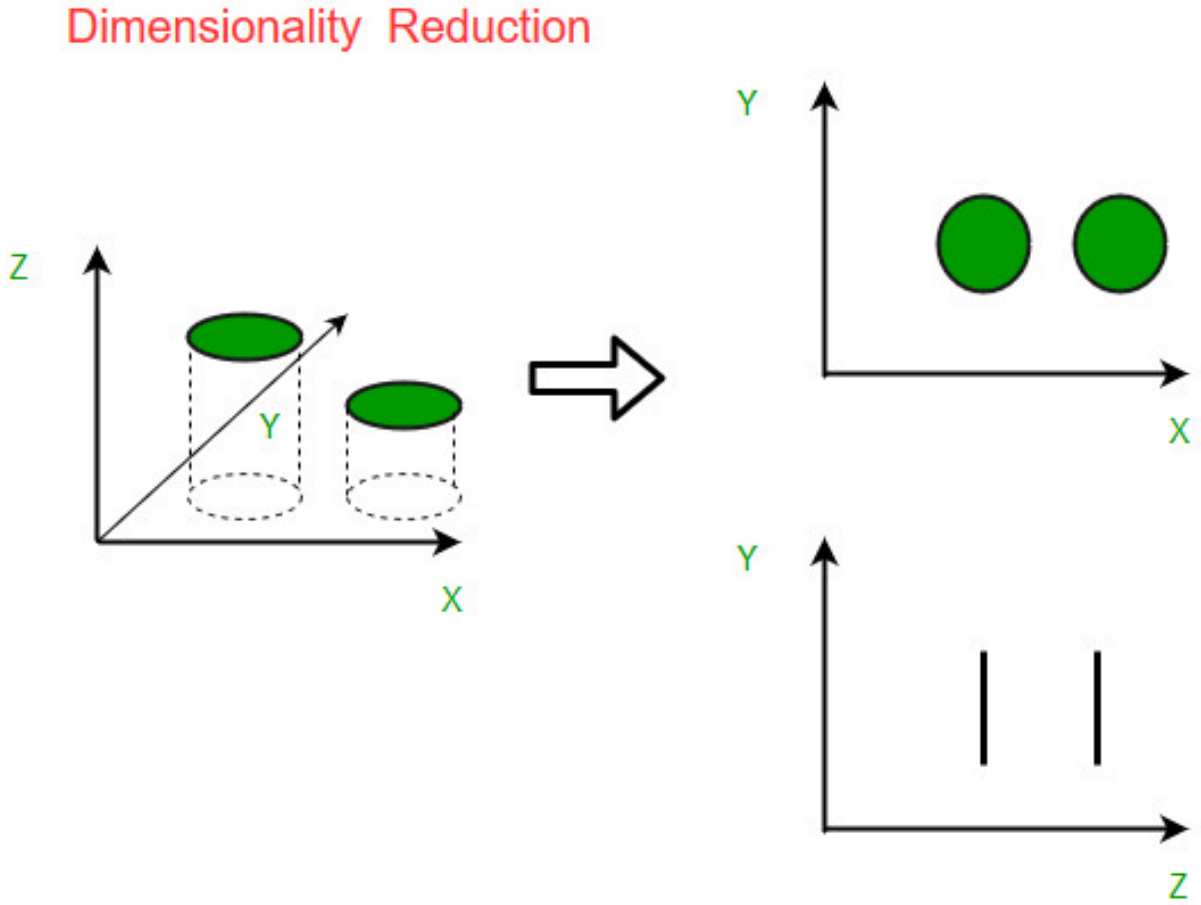
d_{ij}, the distance between cluster centroids i and j.

Clustering performance evaluation - Davies-Bouldin index

- Advantages:
  1. The computation of Davies-Bouldin is simpler than that of Silhouette scores.
  2. The index is solely based on quantities and features inherent to the dataset as its computation only uses point-wise distances
- Disadvantages:
  1. The Davies-Boulding index is generally higher for convex clusters than other concepts of clusters, such as density based clusters like those obtained from DBSCAN.
  2. The usage of centroid distance limits the distance metric to Euclidean space.

Dimensionality reduction

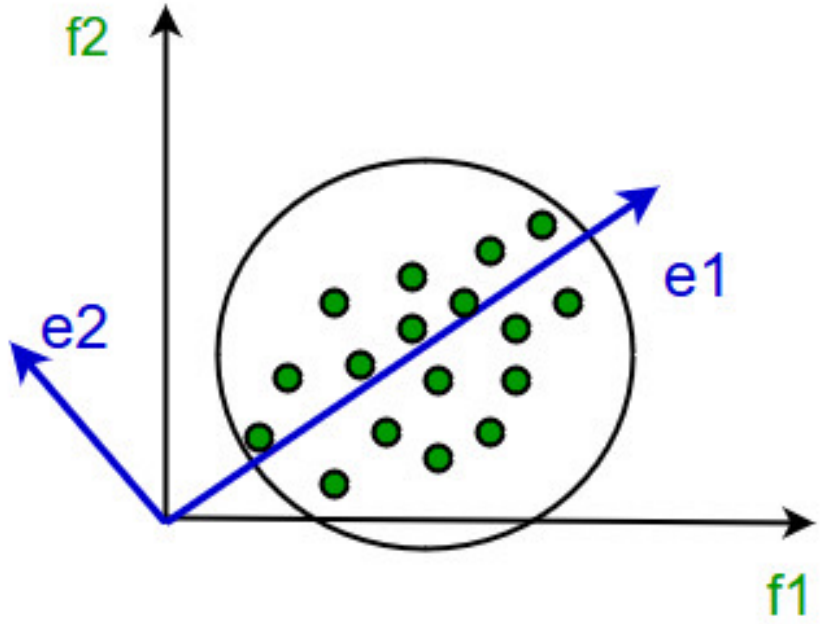
- Dimensionality reduction is a technique used to reduce the number of features in a dataset while retaining as much of the important information as possible.
- It is a process of transforming high-dimensional data into a lower-dimensional space that still preserves the essence of the original data.



https://www.geeksforgeeks.org/dimensionality-reduction/

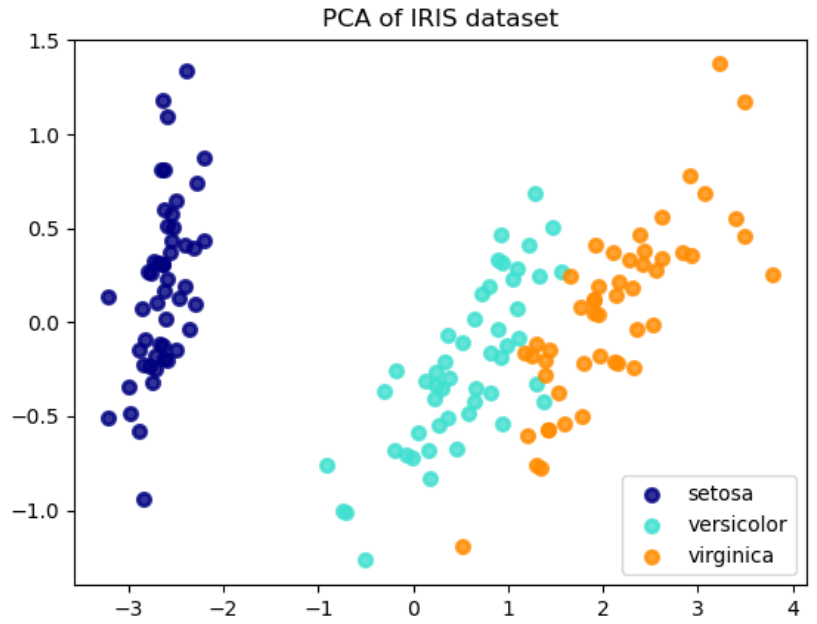
Principal component analysis (PCA)

- PCA is used to decompose a multivariate dataset in a set of successive orthogonal components that explain a maximum amount of the variance.



IRIS dataset

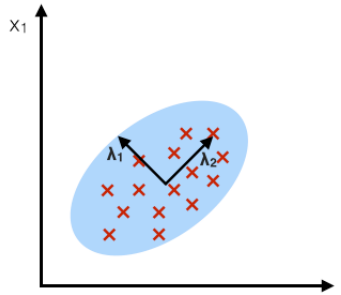
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa



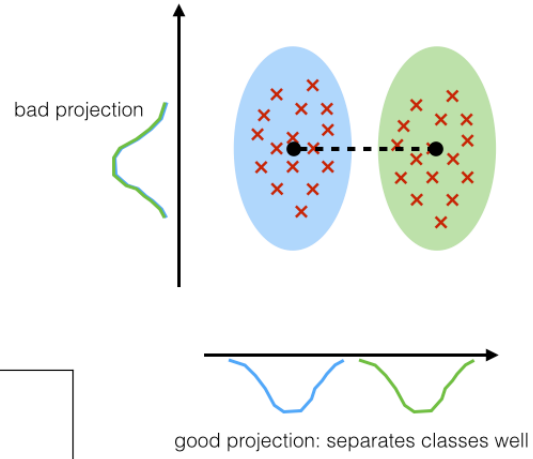
Linear Discriminant Analysis (LDA) for dimensionality reduction

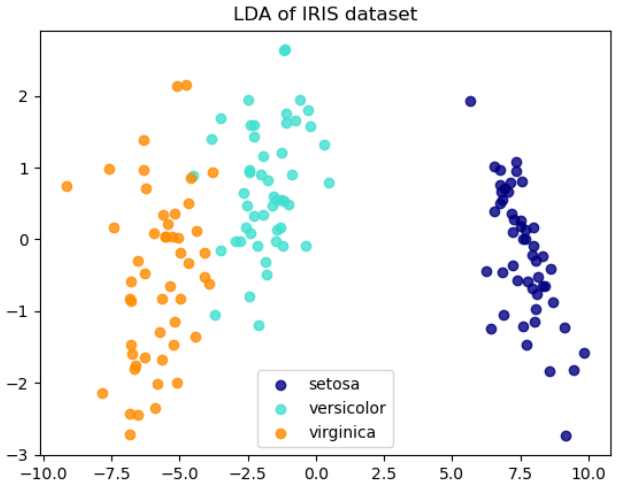
- LDA, in contrast to PCA, is a supervised method, using known class labels.
- Linear Discriminant Analysis (LDA) tries to identify attributes that account for the most variance between classes.

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



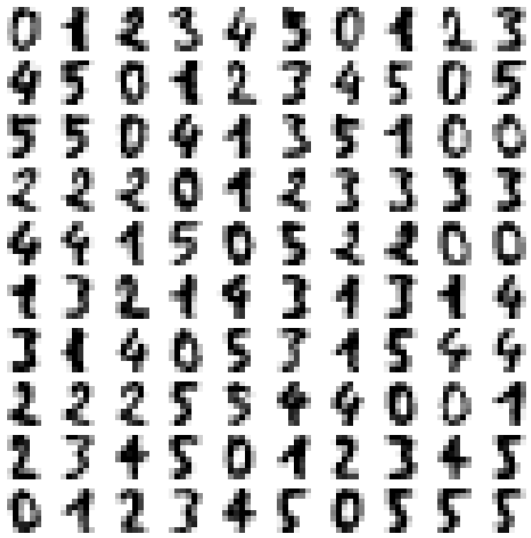


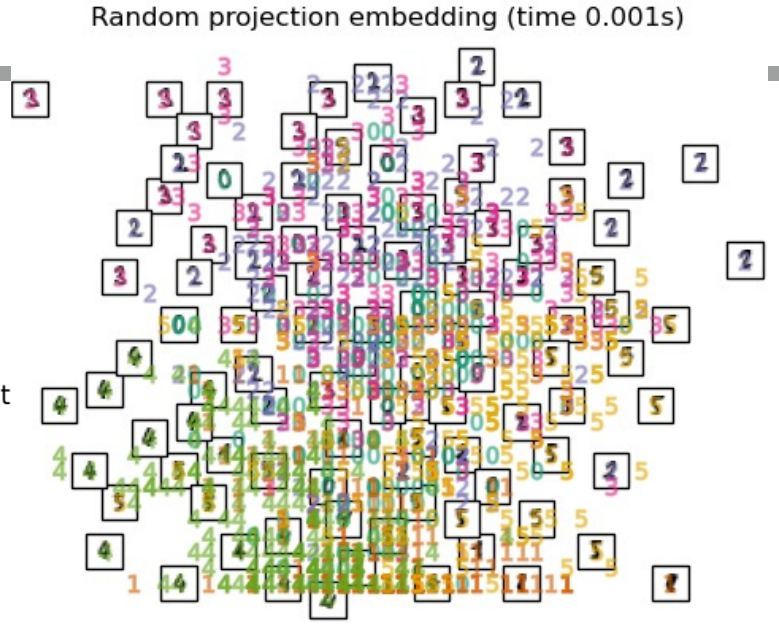
https://sebastianraschka.com/Articles/2014_python_lda.html

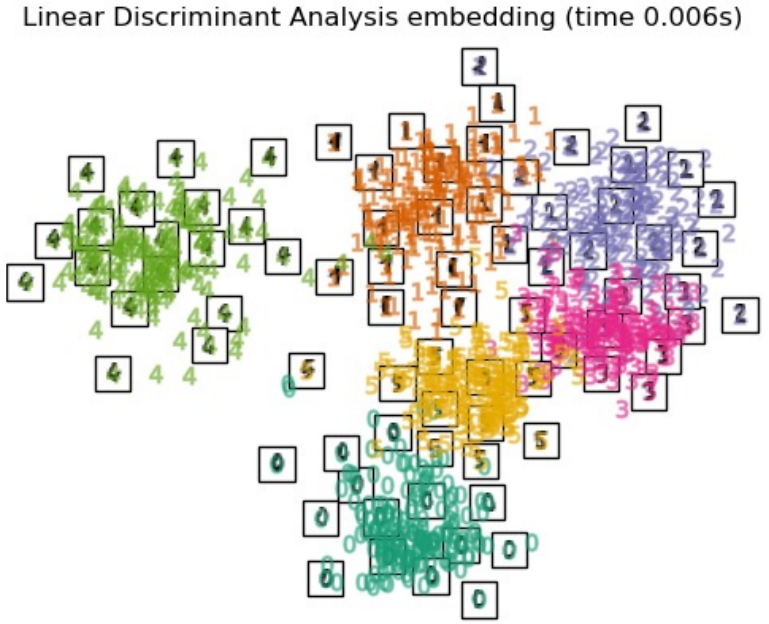
Manifold learning

- Manifold learning is an approach to non-linear dimensionality reduction.
- Manifold Learning can be thought of as an attempt to generalize linear frameworks like PCA to be sensitive to non-linear structure in data.

A selection from the 64-dimensional digits dataset

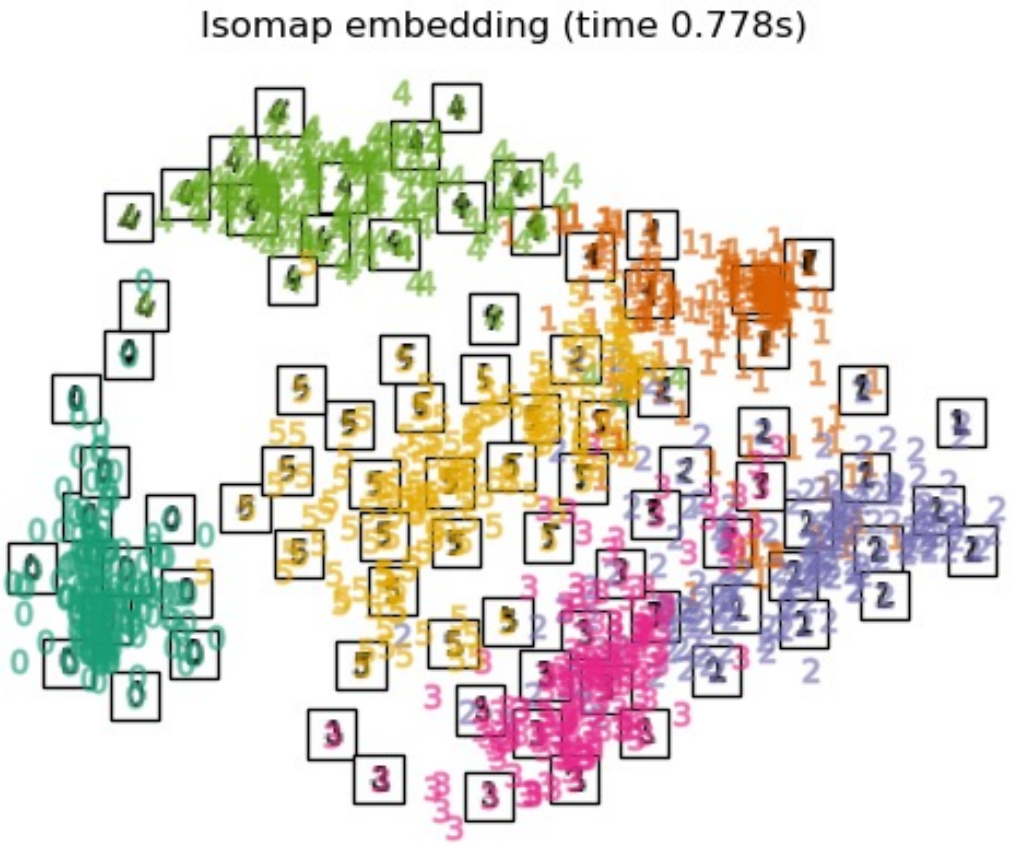






Manifold learning - Isomap

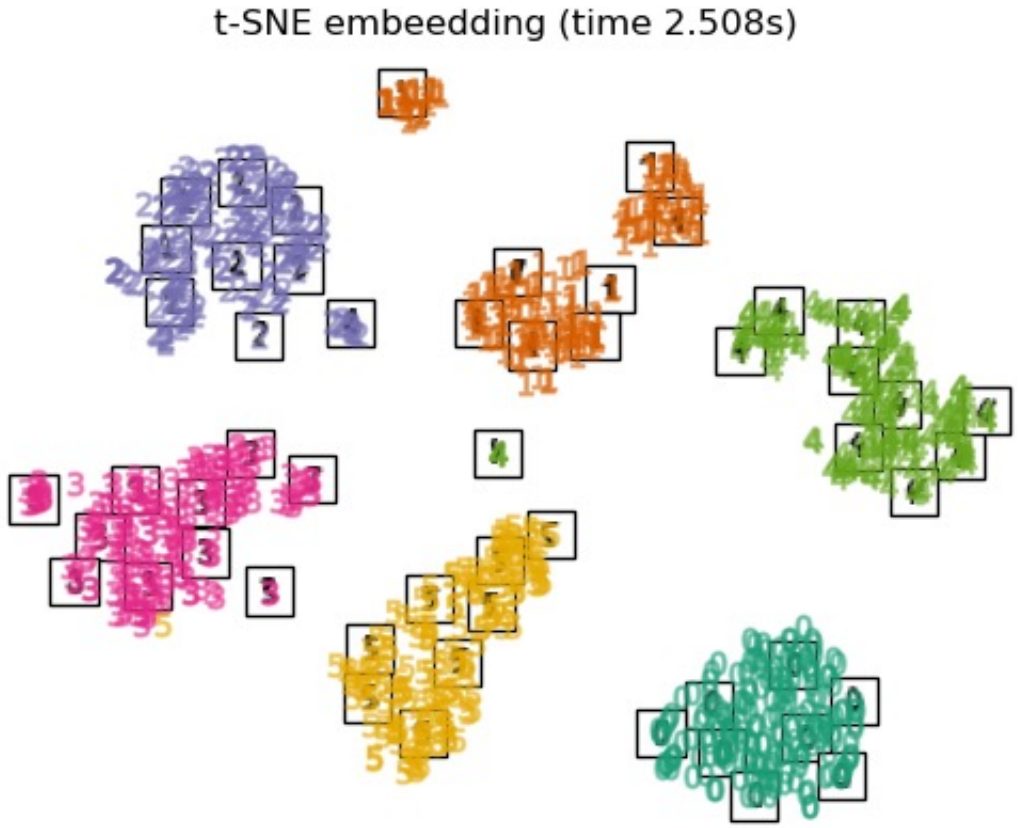
- One of the earliest approaches to manifold learning.
- Isomap can be viewed as an extension of Multi-dimensional Scaling (MDS) or Kernel PCA.
- Isomap seeks a lower-dimensional embedding which maintains geodesic distances between all points.



Geodesic distance between two vertices is the length in terms of the number of edges of the shortest path between the vertices

Manifold learning - t-distributed Stochastic Neighbor Embedding (t-SNE)

- The basic idea is to convert similarities between data points in the high-dimensional space into probabilities, and then map these probabilities to a lower-dimensional space in a way that preserves the relationships between the data points as much as possible.
- This allows t-SNE to be particularly sensitive to local structure.



https://aurigait.com/blog/blog-easy-explanation-of-dimensionality-reduction-and-techniques/

Manifold learning - t-distributed Stochastic Neighbor Embedding (t-SNE)

The procedure for t-SNE:

1. **Compute pairwise similarities**: pairwise similarities between data points in the high-dimensional space are computed (Gaussian distribution).
2. **Construct probability distributions**: From the pairwise similarities, probability distributions are constructed for each data point.
3. **Initialize embedding**: An initial embedding of the data points in the low-dimensional space is randomly generated. Each data point is assigned a position in the low-dimensional space.
4. **Compute similarity in low-dimensional space**: pairwise similarities between data points are computed in the low-dimensional space. (Student's t-distribution).

https://aurigait.com/blog/blog-easy-explanation-of-dimensionality-reduction-and-techniques/

Manifold learning - t-distributed Stochastic Neighbor Embedding (t-SNE)

The procedure for t-SNE(conti.):

5. **Optimize embedding**: The goal of t-SNE is to minimize the divergence between the pairwise similarities in the high-dimensional space and the pairwise similarities in the low-dimensional space (gradient descent).
6. **Convergence:** The optimization process continues until a stopping criterion is met.
7. **Visualization:** the final low-dimensional embedding can be visualized.

https://aurigait.com/blog/blog-easy-explanation-of-dimensionality-reduction-and-techniques/

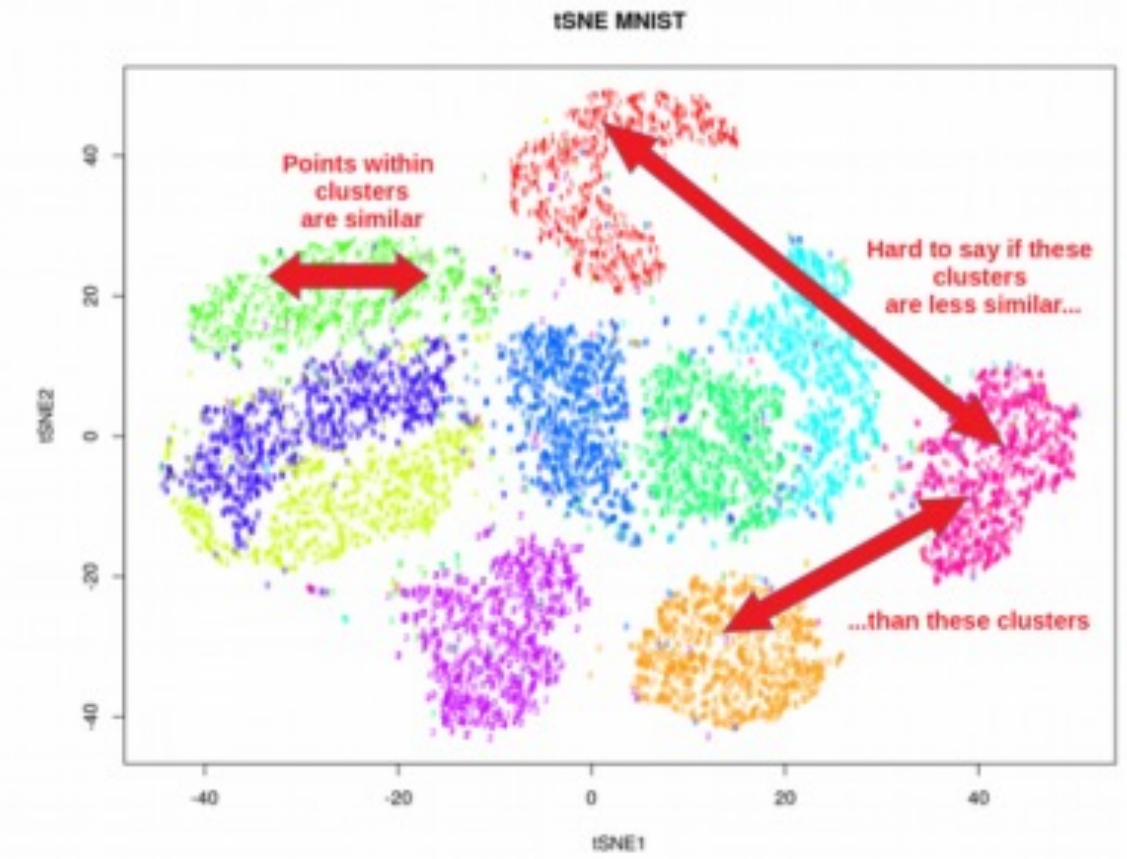
Manifold learning - t-distributed Stochastic Neighbor Embedding (t-SNE)

- Advantages:
  1. Revealing the structure at many scales on a single map
  2. Revealing data that lie in multiple, different, manifolds or clusters
  3. Reducing the tendency to crowd points together at the center
- Disadvantages:
  1. t-SNE is computationally expensive, and can take several hours on million-sample datasets where PCA will finish in seconds or minutes
  2. The Barnes-Hut t-SNE method is limited to two or three dimensional embeddings.
  3. The algorithm is stochastic and multiple restarts with different seeds can yield different embeddings. However, it is perfectly legitimate to pick the embedding with the least error.
  4. Global structure is not explicitly preserved. This problem is mitigated by initializing points with PCA (using init='pca').

https://aurigait.com/blog/blog-easy-explanation-of-dimensionality-reduction-and-techniques/

Manifold learning - Uniform Manifold Approximation and Projection (UMAP)

- It constructs a low-dimensional representation of the data that preserves the global and local structure of the high-dimensional space.
- UMAP uses a graph-based approach to build a topological representation of the data, which is then embedded in a low-dimensional space using stochastic gradient descent.



https://aurigait.com/blog/blog-easy-explanation-of-dimensionality-reduction-and-techniques/

Manifold learning - Uniform Manifold Approximation and Projection (UMAP)

The procedure for UMAP:

1. **Compute the pairwise distances**: calculate the pairwise distances between data points in the high-dimensional space.
2. **Construct a fuzzy simplicial set**: based on the pairwise distances, a fuzzy simplicial set is constructed. A fuzzy simplicial set represents the local neighborhood structure of the data points.
3. **Optimize the low-dimensional embedding**: Initially, a random low-dimensional embedding is generated. Then, UMAP uses stochastic gradient descent to minimize the discrepancy between the pairwise similarities in the high-dimensional space and the low-dimensional space.

https://aurigait.com/blog/blog-easy-explanation-of-dimensionality-reduction-and-techniques/

Manifold learning - Uniform Manifold Approximation and Projection (UMAP)

The procedure for UMAP (conti.):

4. **Construct a graph and perform graph layout**: UMAP constructs a graph representation based on the low-dimensional embedding. This graph captures the global structure of the data.
5. **Refine the embedding**: UMAP performs a refinement step to improve the embedding further. This step involves adjusting the positions of the data points based on the graph structure and the connectivity of the data.
6. **Convergence:** The optimization and refinement steps are iterated until convergence is achieved.
7. **Visualization and analysis**: The low-dimensional representation aims to preserve the neighborhood relationships and the global structure of the high-dimensional data.

https://aurigait.com/blog/blog-easy-explanation-of-dimensionality-reduction-and-techniques/