Manhattan (L1) distance: $d(x,y) = \sum_{i=1}^{d} |x_i - y_i|$

Euclidian (L2) distance: $d(x,\ y) = \|x - y\| = \left(\sum_{i=1}^{d} (x_i - y_i)^2 \right)^{1/2}$

Minkowski (Lp) distance: $d(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p}$

Laplace correction = $\frac{N_i+1}{|S|+k}$ m-estimate = $\frac{N_i+m\pi_i}{|S|+m}$

Algorithm GrowTree(D, F) – grow a feature tree from training data.

Input : data D; set of features F.
Output : feature tree T with labelled leaves.
if Homogeneous(D) **then return** Label(D);
$S \leftarrow$BestSplit(D, F) ; // e.g., BestSplit-Class (Algorithm 5.2)
split D into subsets D_i according to the literals in S;
for each i **do**
  if $D_i \neq \emptyset$ **then** $T_i \leftarrow$GrowTree(D_i, F) ;
  else T_i is a leaf labelled with Label(D);
end
return a tree whose root is labelled with S and whose children are T_i

Impurity measures:

Minority class

Imp($\dot{p}$) = min($\dot{p}$, 1–$\dot{p}$)

Gini index

Imp($\dot{p}$) = 2$\dot{p}$(1–$\dot{p}$)

Entropy

Imp($\dot{p}$) = – $\dot{p}\log_2(\dot{p})$ – (1–$\dot{p}$) $\log_2(1-\dot{p})$

√Gini index

Imp($\dot{p}$) = $\sqrt{2\dot{p}(1-\dot{p})}$

Total impurity:

$\mathrm{Imp}(\{D_1, \dots, D_l\}) = \sum_{i=1}^{l} \frac{|D_i|}{|D|} Imp(D_i)$

Bayes Rule:

$$P(H_i \mid D) = \frac{P(D \mid H_i)\,P(H_i)}{P(D)}$$

False positive rate (FPR) $= \frac{FP}{N} = \alpha$

False negative rate (FNR) $= \frac{FN}{P} = \beta$

True positive rate (TPR) $= \frac{TP}{P}$

True negative rate (TNR) $= \frac{TN}{N}$

Accuracy $= \frac{TP+TN}{P+N} = \left(\frac{P}{P+N}\right) TPR + \left(\frac{N}{P+N}\right) TNR$

Error rate $= \frac{FP+FN}{P+N}$

Precision $= \frac{TP}{\hat{P}}$

Accuracy + error rate = 1

Average recall $= \frac{TPR+TNR}{2}$

Ranking classifier error rate: *rank-err* $= {}^{err}/_{PN}$

Ranking classifier accuracy: *rank-acc* = 1 – *rank-err*

Multivariate least-squares regression (homogeneous representation):

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon}$$

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

$$= \boldsymbol{S}^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Classifier margin for point $\boldsymbol{x}$

$$\boldsymbol{z}(\boldsymbol{x}) = \frac{y(\boldsymbol{w}^T\boldsymbol{x} - t)}{\|\boldsymbol{w}\|} = \frac{m}{\|\boldsymbol{w}\|}$$

Non-homogeneous representation

PAC learning outputs, with probability at least $1-\delta$, a hypothesis h such that $err_D < \varepsilon$

```
Algorithm Perceptron(D, η) – train a perceptron for linear classification.
Input  : labelled training data D in homogeneous coordinates; learning rate η.
Output : weight vector w defining classifier ŷ = sign(w·x).
w ← 0 ;                          // Other initialisations of the weight vector are possible
converged ← false;
while converged = false do
    converged ← true;
    for i = 1 to |D| do
        if y_i w·x_i ≤ 0              // i.e., ŷ_i ≠ y_i
        then
            w ← w + η y_i x_i ;
            converged ← false;        // We changed w so haven't converged yet
        end
    end
end
```

```
Algorithm DualPerceptron(D) – perceptron training in dual form.
Input  : labelled training data D in homogeneous coordinates.
Output : coefficients α_i defining weight vector w = Σ_{i=1}^{|D|} α_i y_i x_i.
α_i ← 0 for 1 ≤ i ≤ |D|;
converged ← false;
while converged = false do
    converged ← true;
    for i = 1 to |D| do
        if y_i Σ_{j=1}^{|D|} α_j y_j x_i · x_j ≤ 0 then
            α_i ← α_i + 1;
            converged ← false;
        end
    end
end
```

Sample covariance: $\hat{\Sigma}_{ij} = \frac{1}{k}\sum_k (x_{ik} - \hat{\mu}_i)(x_{jk} - \hat{\mu}_j) = \frac{1}{k} S_{ij}$

If X is a matrix that holds all the zero-centered samples as column vectors, then

$$\hat{\Sigma} = \frac{1}{k}\boldsymbol{X}\boldsymbol{X}^T = \frac{1}{k}\boldsymbol{S}$$

S is the scatter matrix

If X is not zero-centered, then

$$G = \boldsymbol{X}^T\boldsymbol{X}$$

G is the Gram matrix

Soft margin optimization problem:

$$\mathbf{w}^*, t^*, \xi_i^* = \underset{\mathbf{w}, t, \xi_i}{\arg\min} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, 1 \leq i \leq n$$

Chebyshev distance:

$$L_\infty(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_\infty = \max_i |x_i - y_i|$$

Hamming distance:

$$L_0(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_0 = \text{count}(|x_i - y_i| > 0)$$

Mahalanobis distance:

$$D_M(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{y})}$$

Algorithm KMeans(D, K) – K-means clustering using Euclidean distance Dis_2.

Input : data $D \subseteq \mathbb{R}^d$; number of clusters $K \in \mathbb{N}$.
Output : K cluster means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \in \mathbb{R}^d$.
randomly initialise K vectors $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \in \mathbb{R}^d$;
repeat
  assign each $\mathbf{x} \in D$ to $\arg\min_j \mathrm{Dis}_2(\mathbf{x}, \boldsymbol{\mu}_j)$;
  for $j = 1$ to K **do**
    $D_j \leftarrow \{\mathbf{x} \in D | \mathbf{x} \text{ assigned to cluster } j\}$;
    $\boldsymbol{\mu}_j = \frac{1}{|D_j|}\sum_{\mathbf{x}\in D_j} \mathbf{x}$;
  end
until no change in $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$;
return $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$;

Algorithm Bagging($D, T, \mathscr{A}$) – train an ensemble of models from bootstrap samples.

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.
Output : ensemble of models whose predictions are to be combined by voting or averaging.

for $t = 1$ to T **do**
  build a bootstrap sample D_t from D by sampling $|D|$ data points with replacement;
  run $\mathscr{A}$ on D_t to produce a model M_t;
end
return $\{M_t | 1 \leq t \leq T\}$

Algorithm Boosting($D, T, \mathscr{A}$) – train an ensemble of binary classifiers from reweighted training sets.

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.
Output : weighted ensemble of models.

$w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
for $t = 1$ to T **do**
  run $\mathscr{A}$ on D with weights w_{ti} to produce a model M_t;
  calculate weighted error ϵ_t;
  if $\epsilon_t \geq 1/2$ **then**
    set $T \leftarrow t - 1$ and break
  end
  $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$; // confidence for this model
  $w_{(t+1)i} \leftarrow \frac{w_{ti}}{2\epsilon_t}$ for misclassified instances $x_i \in D$; // increase weight
  $w_{(t+1)j} \leftarrow \frac{w_{tj}}{2(1-\epsilon_t)}$ for correctly classified instances $x_j \in D$; // decrease
end
return $M(x) = \sum_{t=1}^{T} \alpha_t M_t(x)$

Sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Backpropagation error for output units:

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

Backpropagation error for hidden units:

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh} \delta_k$$