1. Machine Learning Algorithms Pseudocode

Algorithm 1 k-Nearest Neighbor

Input: X: training data, Y:Class labels of X, x: unknown sample
Output: Class with the highest number of occurrence

function $\textsc{Classify}(X, Y, x)$
  for $i = 1$ to m **do**
    Compute distance $d(X_i, x)$
  end for
  Compute set I containing indices for the k smallest distances $d(X_i, x)$
  Return majority label $\{Y_i$ where $i \in I\}$
end function

1. Ensemble Algorithm

Algorithm 2 Adaboost

Input:
  Training data $\{(x_i, y_i)_{i=1}^N$ where $x_i \in \mathbb{R}^k$ and $y_i \in \{-1, 1\}\}$
  Large number of classifiers denoted by $f_m(x) \in \{-1, 1\}$
  0-1 loss function I defined as

$$I(f_m(x, y)) = \begin{cases} 0, & \text{if } f_m(x_i) = y_i \quad (1) \\ 1, & \text{if } f_m(x_i) \neq y_i \quad (2) \end{cases}$$

Output: The final classifier

for $i = 1$ to N **do**
  for $i = 1$ to M **do**
    Fit weak classifier m to minimize the objective function:
    $\epsilon_m = \frac{\sum_{i=1}^N w_i^m I(f_m(x_i)) \neq y_i}{x^2+2x+1}$
    where $I(f_m(x_i) \neq y_i) = 1$ if $f_m(x_i) \neq y_i$ and 0 otherwise
    $\alpha_m = \ln \frac{1-\epsilon_m}{\epsilon_m}$
  end for
  for all i **do**
    $w_i^{m+1} = w_i^{(m)} e^{\alpha_m I(f_m(x_i) \neq y_i)}$
  end for
end for

1. Another Pseudocode for Adaboost

1. Machine Learning Algorithms Pseudocode

1. Machine Learning Algorithms Pseudocode

1. Machine Learning Algorithms Pseudocode

Algorithm 3 Adaboost

Input:
Training data $\{(x_i, y_i)_{i=1}^{N}$ where $x_i \in \mathbb{R}^k$ and $y_i \in \{-1, 1\}\}$
Output: The final classifier

Given Training data $\{(x_i, y_i)$ where $y_i \in \{-1, 1\}\}$
initialize D_1 = uniform distribution on training examples
for $t = 1$ to T **do**
  Train weak classifier h_t on D_t
  choose $\alpha_t > 0$
  compute new distribution D_{t+1}:
  for all i **do**
    multiply $D_t(x)$ by

$$\begin{cases} e^{-\alpha_t}, & (< 1) \text{ if } y_i = h_t(x_i) \quad (3) \\ e^{\alpha_t}, & (> 1) \text{ if } y_i \neq h_t(x_i) \quad (4) \end{cases}$$

    renormalize
  end for
  output final classifier $H_final(x) = sign(\sum \alpha_t h_t(x))$
end for

Algorithm 4 Random forest

Input: S: training set, F:Features and number of trees in forest B
Output: Constructed tree

function RANDOMFOREST(S, F)
  $H \leftarrow \emptyset$
  for $i \in 1, ...B$ **do**
    $S^{(i)} \leftarrow$ A bootstrap sample from S
    $h_i \leftarrow RANDOMIZEDTREELEARN(S^i, F)$
    $H \leftarrow H \bigcup \{h_i\}$
  end for
  return H
end function
function RANDOMIZEDTREELEARN(S, F)
  At each node:
  $f \leftarrow$ a very small subset of F
  Split on best feature in f
  return The learned tree
end function

Algorithm 5 Iterative Dichotomiser 3

Input: D : Training Data, X : Set of Input Attributes
Output: A decision tree

function ID3(D, X)
    Let T be a new tree
    if all instances in D have the same class c **then**
        Label (T) = c; Return T
    end if
    if $X = \emptyset$ or no attribute has positive information gain **then**
        Label (T) = most common class in D; Return T
    end if
    $X \leftarrow$ attribute with highest information gain
    Label(T) = X
    for each value x of X **do**
        $D_x \leftarrow$ instances in D with $X = x$
        if D_x is empty **then**
            Let T_x be a new tree
            Label(T_x) = most common class in D
        else
            $T_x = \text{ID3}(D_x, X - \{x\})$
    end if
    Add a branch from T to T_x labeled by x
end for
return T
end function

Algorithm 6 Perceptron

Input: $ProblemSize, InputPatterns, iterations_max, learn_rate$
Output: $Weights$

for $i = 1$ to $iterations_{max}$ **do**
    $Pattern_i \leftarrow SelectInputPattern(InputPatterns)$
    $Activation_i \leftarrow ActivateNetwork(Pattern_i, Weights)$
    $Output_i \leftarrow TransferActivation(Activation_i)$
    $UpdateWeights(Pattern_i, Output_i, learn_{rate})$
end for
Return $Weights$