Concept Learning

* Learn an unknown Boolean function *f*
* If a dataset has *n* discrete features and each attribute has *k* discrete possibilities, then the full hypothesis space contains

hypotheses

* Measure hypothesis by its consistency
* Prior knowledge:
  + Each attribute can be:
    - A specified value
    - ? – Any value is acceptable
    - ∅ – No value is acceptable
  + A hypothesis is represented as
    - *<* val1, val2, val3, … *>*
    - For example, the hypothesis would classify all instances where *val1* is *sunny* and *val2* is *warm* as Yes. All other instances would be No.
  + The hypothesis means each instance is a Yes instance.
  + All hypothesis containing even one ∅ classifies all instances as No.
  + Full hypothesis space contains

hypotheses

* + (more specific) is a subset of the hypothesis (more general).
  + The goal is to find one or more hypotheses and choose the most general one.
* Assumptions
  + The hypothesis space contains *the* hypothesis that describes the true concept
  + The training data has no error
* Inductive Bias
  + Restriction bias: assume the true target is a conjunction of constraints
  + No preference bias: output the whole version space

|  |  |
| --- | --- |
| Find-S | Candidate Elimination |
| Find a hypothesis consistent with training data. | Find a compact representation of all hypothesis consistent with training data. |
| Only consider the most specific one. | Consider all possible consistent hypothesis. |
| Ignore No instances. | Consider both Yes and No instances |

Decision Tree

* Learn an unknown Boolean function *f*
* Measure: information gain
* ID3 Algorithm
  + Select the attribute that can maximally reduce the *impurity* of the instances.
  + Entropy is highest when instances are 50/50

Where *A* is the attribute to split the data by, and *i* represents each value of the attribute.

* + Continue selecting the best remaining attribute until every leaf is *pure* (all remaining instances are the same classification)
* Hypothesis space
  + Every finite discrete function can be represented by a decision tree.
  + Each *n­*-feature Boolean function can be represented by a binary decision tree with *n* depth
* Inductive bias
  + ID3 algorithm produces only one tree, though there can be many trees consistent with the training data. ID3 prefers shorter trees (preference bias).
* Overfitting
  + Noise (errors in data), imbalanced distribution
  + Control this by pruning the tree.
    - Stop growing the tree when further classification is not statistically significant
    - Prune branches and test accuracy on test-data
* Numerical attributes
  + Check the midpoint of each instance (sorted) and check the impurity of using the midpoint as a threshold
  + Can have multiple thresholds per attribute

Bayesian Learning

* Maximize , Maximum a posteriori (MAP)
* If we do not have any prior knowledge and every *h* in H is equally likely, we can drop . This is the Maximum likelihood estimate (MLE)
* Point estimation
  + Take the *ln* of the argmax function

,

Solve for *θ*

Naïve Bayes Classifier

* Same as Bayesian learning, except additional assumption that all features are independent.
* Don’t forget to multiply by as well when deciding how to classify an instance.
* Smoothing
  + When an instance has never been observed, this method breaks. Smooth the formula:

Where and

* + For every , add the *mp / m* in the calculation so that no values are 0 anywhere.
* This assumption requires less data
* If the values get too small, use lns

Sigmoid

* This is a node within a neural network
* The node performs the following calculation
* Input: a real vector
* Output: a real value (the float produced by the above function)
* Define the error:

Linear Regression

* LSM: Select *h* such that *Error* is minimized

Assume

Because math, the *a* and *b* that minimize this error is

* But there are multiple features (multiple x values), so it’s not that simple.

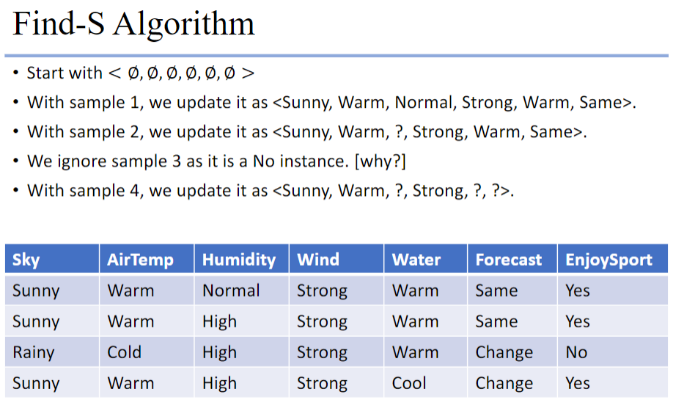
Here,

and and

The that minimizes this error is

* Regularization
  + Regularization reduces overfitting
  + Add to the Error term
  + Small is better for smoother curves

|  |  |
| --- | --- |
| **Naïve Bayes** | **Logistic Regression** |
| Generative classifier | Discriminative classifier |
| Can generate new data; we know and | Cannot generate new data; we only know |
| Generally not a linear classifier | Always a linear classifier |
| Gaussian Naïve Bayes with class independent variance is representationally equivalent to logistic regression | |



Perceptron

* This is a node within a neural network
* This is basically just a weighted sum of all the inputs

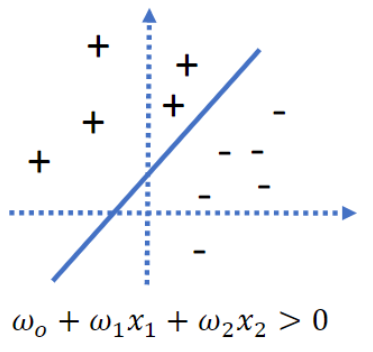
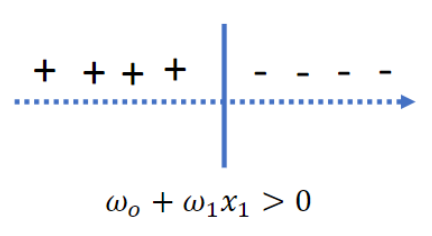
If

A value of 1 is returned and passed onto the next node

If

A value of -1 is returned and passed onto the next node

* Decision surface of Perceptron
  + Linearly separable



* Training a perceptron
  + Method 1: Perceptron training rule
    - *t*: target value
    - *o*: current prediction
    - : learning rate, small value
    - Continue updating until it converges
    - It will converge if the data is linearly separable and the learning rate is small enough. This is important because the training data may not be linearly separable, even if the underlying truth is linear
  + Method 2: LSM + Gradient Descent
    - Linear unit:
    - Error
    - Least squares method
    - Batch Mode: define the error for the whole training data, and update the parameters by minimizing the total error.
    - Incremental Mode: consider the error for each instance and update the parameters by minimizing the error for a single attribute. Then, move on to the next attribute and repeat.

Logistic Regression

* Classify an instance based on the higher probability
* This is still a linear classifier, as the classification is still ultimately determined by the same rule as a linear regression
* Learn using Bayesian learning

Take

For all

where is the learning rate

is the difference between the observed value and the predicted probability

* Regularization
  + Method 1: Add term and maximize
  + Method 2: Assume follows a Gaussian distribution with

\*\*\*cry\*\*\*

Neural Networks

* A neural network is a series of interconnected units that map input values to an output.
* Input units: usually a real vector
* Processing units
  + Node containing some function (perceptron, sigmoid, etc.)
  + receive input from other units
  + output result to other units
* Edges
  + These are the branches that connect the nodes
  + They are weights (the parameters)
* Train a neural network
  + Define the error E
  + Calculate
  + Batch and incremental mode apply

|  |  |  |
| --- | --- | --- |
| **Perceptron** | **Sigmoid** | **Linear** |
| Not differentiable | Differentiable | Differentiable |
| Output a classification (-1 or 1) | Output a bounded float/value (0,1) | Output a float/value (unbounded) |
|  |  |  |

Notes

* If the data is not linearly separable, the gradient descent algorithm *can* still converge because the objective function is concave.
* If a decision tree classifier has 100% accuracy on the training data, this does *not* mean logistic regression will also have 100% accuracy (think linearly separable).
* A decision tree is *not* the smallest possible representation of the target function. Some Boolean functions may generate very large trees.
* If values are missing for regression, you can:
  + Ignore the instances with missing values
  + Take the most likely value for the missing values.