# Supervised VS Unsupervised

* *Feature*: measurable property/characteristic of data instance
* *Concept*: output label/variable; what we are predicting
* *Supervised*: labelled
  + Classification, Regression
  + Learn relation between features and target
* *Unsupervised*: unlabelled data
  + Clustering, Dimensionality Reduction, Anomaly Detection
  + *Strongly*: no idea how many/what groups are
  + *Weakly*: know how many/what groups are
* *Semisupervised*: some labelled, majority unlabelled
* *Reinforcement Learning*: interaction and feedback
* *Hyperparameters*: parameters which define and constrain the learning process
* *Parameters*: values learnt and applied to particular training dataset

# Feature Types

| Numerical | | Categorical | |
| --- | --- | --- | --- |
| *Discrete* | *Continuous* | *Nominal* | *Ordinal* |
| - whole number  distinct categories/lvls | - range/interval  - decimal | - no inherent order/ranking | - has natural order/ranking |

## Categorical → Numeric

*Label/Ordinal Encoding*

* assign unique number label to each category
* for ordinal: based on order/ranking
* eg. red, blue, green → 1, 2, 3
* CON: artificial order

*One-Hot Encoding*

* binary columns for each feature
* eg. “red” → [0, 0, 1]
* PRO: equal distance, CON: increases dimensionality

*Target Encoding*

* replace category with statistical aggregation (count, mean, median, etc.)
* eg. red, blue, green → 4, 10, 4
* CON: prone to overfitting

*Hashing*

* hash feature to fixed, unique numeric value
* PRO: reduces dimensionality of large-scale datasets

## Categorical → Numeric

*Equal-Width Discretisation*

* partition into bins of width
* CON: sensitive to outliers; choosing requires domain knowledge

*Equal Frequency*

* sort values → find breakpoints to produce bins with equal frequency

*K-Means Clustering*

Text

Bag of Words

* Vector of frequencies of each word
* CONS: curse of dimensionality, missing context, order, synonyms

Images

Bags of Pixels

* Can work for constrained tasks
* CONS: curse of dimensionality, no shapes/objects, just colours

## Distance

| *Euclidean Distance* | *Manhattan Distance* | *Hamming Distance* |
| --- | --- | --- |
|  | |  |

# 0-R/1-Rule Classifier

*0-R*: Majority class classifier – predict majority class

*1-R*: Predict concept based on feature with least error

1. For each attribute/feature, create a frequency table and **calculate error**
   * assume we always pick majority class for that feature
   * Error = total all instances in minority classes
2. Pick feature with **least error**
3. Use feature to determine concept of test instances

# Naive Bayes

* *Marginal Probability*: probability of event occurring for single random variable, without considering values of other variables
* : probability of outcome X=A, considering all possible values of Y
* *Joint Probability*: probability of two or more events occurring together
* P(X=A, Y=B)
* Bayes’ Rule:
* assumes that all features are independent given the class
* : Predicted Class
* : Prior Probability of class
* : Likelihood – probability of feature given class
* : product – multiplying lots of numbers between (0, 1] can lead to underflow

## Underflow

* Log Transformation:
* take log of each probability and sum instead of product

## Zero Values

* If any ⇒ Final value = 0
* *Simple Probabilistic Smoothing:* replace 0 with positive constant
* *Laplace Smoothing*: increase counts by (usually 1)

| Unsmoothed | Smoothed |
| --- | --- |
|  |  |

* CONS: drastic change for small dataset; overestimates rare events
* *Other Options*: Add-k Smoothing, Regression

# Decision Trees

*Entropy*: amount of uncertainty in dataset

ID3

Information Gain

Gain Ratio

# Support Vector Machines (SVM)

|  | * GOAL: find optimal hyperplane to separate classes and maximises margin * PRO: good for high-dimensional spaces (captures feature relationships implicitly); strong generalisation; robust against overfitting (due to ) * CONS: sensitive to hyperparameter tuning (kernel, kernel params, ); computationally expensive; best for supervised binary classification (multi-class and regression is more expensive) |
| --- | --- |

* *Confidence Measure*: distance between new instance and hyperplane
* *Soft Margins*: permits few points to be on the wrong side
* *Kernel Function*: transforms input data to higher-dimensional feature space

# K Nearest Neighbours (KNN)

# Hill Climbing

* finds best solution for search space (local optima)
* starts with initial solution and iteratively makes incremental improvements
* 1 initialise → 2 evaluate → 3 neighbourhood search (incremental change) → 4 selection (based on performance/evaluation) → repeat 3-4 until termination criterion

# Logistic Regression

* binary classification

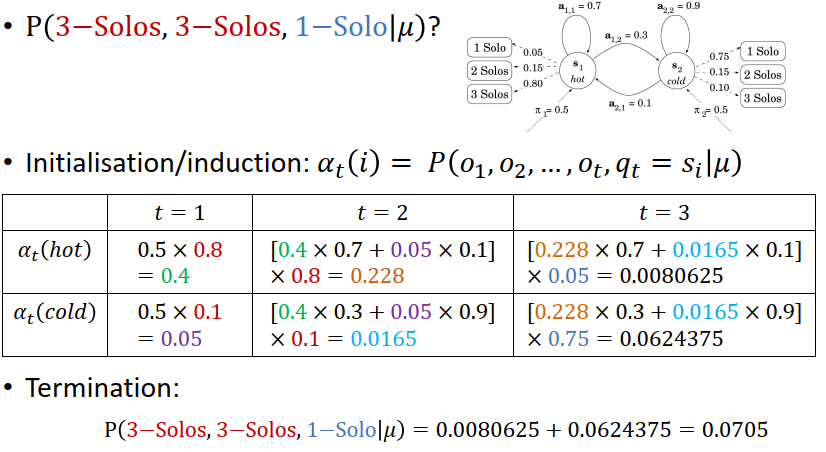
1. linear combination of input features: multiply each by weight and sum them tgt
2. logistic function (sigmoid function): maps value between 0 and 1
3. determine division boundary (threshold) to determine predicted class
4. trained using maximum likelihood estimation

* O(num classes \* num features)

# Hidden Markov Model

* *Markov assumption*: the likelihood of transitioning into a given state depends only on the current state, and not the previous state(s) (or output(s)), i.e. P (qt|q1···qt−1) ≈ P (qt|qt−1)
* *Output independence assumption*: the likelihood of a state producing a certain observation (as output) does not depend on the preceding (or following) state(s) (or output(s)), i.e., P (ot|q1···qt, o1···ot−1) ≈ P (ot|qt)

## Forward Algorithm



* calculate the probability of observing a particular sequence of observations

## Viterbi Algorithm

* find most likely sequence of hidden states given sequence of observations

# K-Means Clustering

1. Select points at random as initial cluster centroids
2. compute distance to each centroid for each instance & assign to nearest centroid
3. compute new centroid for each cluster (centroid = mean of all instances in cluster)
4. Repeat form 2. until no instances are reassigned

# Gaussian Mixture Model (GMM) / Soft K-Means

* assigns probability associated with each cluster; allows instance to belong to multiple clusters simultaneously
* each cluster represented by Gaussian distribution
* algorithm iteratively updates means and covariance matrices of distribution and recalculates probabilities
* **Gaussian Mixture Model (GMM)**: distribution as composed of Gaussian distributions
* *Finite Mixture*: distribution composed of component distributions

# Expectation-Maximisation (EM) Algorithm

* iterative optimization algorithm; generalisation of (soft) k-means; guaranteed positive hill-climbing characteristics
* used to estimate parameters in statistical models with hidden variables

1. *(E)xpectation Step*: calculate posterior probability (expectation) of latent variables given observed data and current estimates of model parameters
2. *(M)aximisation Step*: maximises expected log-likelihood by updating model parameters
3. Repeated until algorithm converges (estimates of model parameters stabilise//reach convergence criterion)

# Ensemble Learning

* Aggregate results from base classifiers ⇒ reduces variance
* *Bagging/Bootstrapping*: construct new datasets with random sampling w/ replacement
* Instance Absent Probability:
* PRO: simple (1 algo + simple vote); noisy datasets (outliers vanish); can parallelise; less overfitting
* CON: unstable classifiers ⇒ high variance (and vv)
* *Stacking*: combining output of a number base classifiers as input to further supervised learning (meta) model; to smooth errors over range of algorithms with different biases
* PRO: results usually better than best of base classifiers
* CON: computationally expensive; many algorithms
* *Boosting*: focuses on hard-to-classify instances; iterative: sample → update weights → weighted voting
* PRO: computationally cheap; guaranteed performance (error bounds over training data); minimise instance bias
* CON: more expensive than bagging; overfitting
* *Random Forest*

1. Create new datasets using Bagging
2. Use random subset of features for each tree

|  | * ensemble of Random Trees * *Random Tree*: decision tree with only some possible attributes |
| --- | --- |

# Neural Networks

Learns feature hierarchy

*Embedding*: low dimensionality representation of input; can be used for other tasks (not only original task)

# 

**Neuron**

|  | * Input: vector , * Output: scalar * Hyperparam: activation function * Parameters: weights , |
| --- | --- |

**Perception**

NN with single neuron

Binary linear classifier

Usually step function (if y\_i > 0 then output 1, else 0)

**Training Perceptrons**

Find weights to minimise errors

1. Iterate over training set (1 iteration = 1 epoch); 2. compare prediction and true values; 3. Update weights

Lambda: learning rate

More penalty on larger inputs; less penalty on smaller inputs

# 

# Evaluation

* *Generalisation*
* *Overfitting*
* *Learning Curve*
* *Bias*: tendency to produce same errors; 🠕 Bias ⇒ predicted label distribution ≠ true labels//makes many mistakes; 🠗 Bias ⇒ no mistakes//different kinds of errors on different instances; still similar distribution to true labels
* *Variance*: measure of model inconsistency; produce different classifications on different training sets (randomly sampled on same population)

|  | Precision   * 🠕 precision ⇒ 🠗 FP   Recall   * 🠕 recall ⇒ 🠗 FN   Error Rate   * proportion of incorrect predictions * = (FP + FN)/Total * = 1 – Accuracy |
| --- | --- |

* *Micro-average*: calculate metric for overall; kind of treat all data as from same class
* *Macro-average:* calculate metric for each class then average the resulting metric values (eg. )

MSE, RMSE and RRSE usually used for regression

## Mean Squared Error (MSE)

1. for each prediction, subtract the corresponding true value
2. square the differences
3. calculate average of squared differences

## Root Mean Squared Error (RMSE)

1. take square root of MSE

* same units as original data

## Root Relative Squared Error (RRSE)

1. divide RMSE by range (max-min) of true values
2. multiply result by 100 (%)

* relative measure of error compared to range of true values

## Baseline

## Pearson’s Correlation

# Feature Selection

## Dimensionality Reduction

* lossy: not possible to reproduce original data from reduced version

*PCA*

* Projects data to new coordinate system
* Identified Principal Components (linear combinations of original features that capture max variance in data)
* Arrange in descending order
* CONS: assumes linear relationship between features; hard to interpret; assumes 🠕 variance components = informative features; influenced by outliers

## Filtering

## Pointwise Mutual Information (PMI)

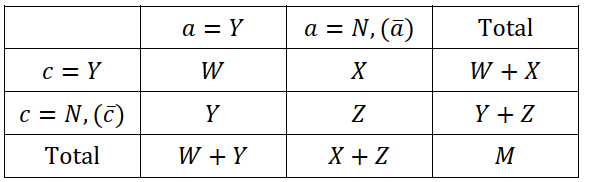
## Mutual Information (MI)

* amount of information shared between two random variables

## Chi-Square,

* association between two categorical variables (usually feature and label)
* eval difference between observed and expected freq. → assess deviation
* 🠕 ⇒ dependency between variables

1. Contingency Table

* 

1. Calculate expected frequency for each cell
2. Calculate Chi-Square Value for each cell and sum all up

set of patterns/principles of **assigning responsibilities** into an object; supports Responsibility-Driven Design (RDD)

| Naive Bayes |
| --- |
| B should create instances of A if:   * B “contains” / compositely aggregates A * B records/closely uses A * B has initializing data for A |
| K Nearest Neighbours (KNN) |
| objects should communicate through **intermediary** rather than directly |
| reduces coupling; more flexible; easier maintain |

## 