*By Hawk update 2018-1-13*

## What is AI

- Thinking humanly
- Thinking Rationally
- Acting Humanly:  Turing test.  A computer passes the test of intelligence. if it can fool a human interrogator
- Acting Rationally

## Basic Search

- Problem Formulation
  - States: all reachable states from the initial state by any sequence of actions
  - Initial state: the state where the problem starts
  - Actions: legal actions on state
  - Next state: a description of what each actions does, Result(s, a)
  - Goal test: determine whether a given state is a goal state.
  - Path cost: function that assigns a numeric cost to each other.
- Search Methods: Performance Metrics
  - Completeness: Does it always find a solution if it exists
  - Optimality: Does it always find the least-cost solution
  - Time complexity:  nodes generated  expanded
  - maximum nodes in memory
- Uniformed Search Methods
  - BFS: Breadth-first search
    - Costs in the search tree is same
    - Expand the shallowest unexpanded node
  - UCS: Uniform-cost search
    - Expand the cheapest unexpected node
    - the costs in the search tree may be different
  - DFS: Depth-first search
    - expand the deepest unexpanded node
    - not complete, not optimal
  - DLS: Depth-limited search
    - nodes at depth $l$ have no successors
    - Limit $l$ is defined based on domain knowledge
  - IDS: Iterative deepening search(IDS)
    - Apply DLS with increasing limits
    - IDS is the preferred uninformed search method when the search space is large and the depth of the solution is unknown

    - Bidirectional search
        - search from forward and backward directions simultaneously

## Heuristic Search

> Basic search: use no domain knowledge
>
> Heuristic Search: use domain knowledge

Heuristic function $h(n)$ estimates the cheapest cost from $n$ to Goal

- (1) $h(n) = 0$ if $n$ is the goal node
- (2) nonnegative
- (3) problem-specific

A 'good' heuristic can be powerful only it is a 'good' quality. It should be **admissible**

***admissible:*** never overestimates the cheapest cost from $n$ to goal

## Heuristic Search Methods

- Greedy Best-first Search:
    - expand the node that seems closest to the Goal
    - expand node $n$ that has the minimal $f(n) = h(n)$
    - It is not complete, can stuck in loops.  And it is not optimal.
- A* Search:
    - expand node $n$ has the minimal $f(n) = h(n) + g(n)$
    - It is complete. And it is optimal.
    - **the prove of optimality if A***
- Generate admissible heuristics
    - from relaxed problems
    - from sub-problem
    - from experience

## Optimization

- Unconstrained Optimization
    - Steepest descent
        - Iterative algorithm: $x^{k+1} = x^k - \alpha^k \bigtriangledown f(x^k)$
        - Step size $\alpha^k$ is important:  high -> jumping ; low-> little progress
        - **step size selection ???**
    - Zig-zagging
        - why?
    - Newton's method
        - Newton's method to solve the formula
        - $x^{k+1} = x^k - (\bigtriangledown^2 f(x^k))^{-1} \bigtriangledown f(x^k)$

    > Steepest descent: the most basic method, used very often, needs $\bigtriangledown f(x) \in \mathbb{R}^n$

> Newton Method: Extremely fast convergence ($x^*$ is the optimal), needs second-order derivate $\nabla^2 f(x) \in \mathbb{R}^{n*m}$

- Constrained Optimization
  - consider them directly (projected gradients)
    - first steepest descent
    - second keeps feasibility
  - Lagrangian duality
  - KKT conditions

$$\begin{aligned} & minimize \;\; f(x) \\ subject \;\; to \;\; & \begin{cases} g_i(x) \leq 0, i = 1, \ldots, I \\ h_j(x) = 0, j = 1, \ldots J \end{cases} \end{aligned}$$

    - Define the Lagrangian function $L(x; \lambda, \mu) = f(x) + \lambda^T g(x) + \mu^T h(x)$
    - conditions

$$L(x; \lambda, \mu) = f(x) + \lambda^T g(x) + \mu^T h(x)$$

$$\begin{cases} \lambda_j g_j(x) = 0 \\ h_i(x) = 0 \\ g_j(x) \leq 0 \\ \lambda_j \geq 0 \end{cases}$$

    - KKT example (homework) 🐻
    - KKT reference 🐻
    - SQP method (sequential quadratic programming)

## Optimization without derivatives

- Bisection method: $if \; F(a)F(b) < 0, then \; a \; solution \; exists$
- Golden -section search

> $x_1, x_2, x_3$ with $f(x_2) < min(f(x_1), f(x_3))$, test $x_4 \in [x_2, x_3]$
>
> if $f(x_4) < f(x_2)$, pass to interval $[x_2, x_3]$ otherwise pass to $[x_1, x_4]$

- Nelder-Mead method (details in Slides)
  - reflected, expanded, contracted
  - The worse point is replaced by the best point of above three types points.
  - Otherwise, the simplex shrink.
- Simulated annealing

> if $f(y^k) < f(x^k)$ set $x^{k+1} = y^k$
>
> if $f(y^k) > f(x^k)$ set $x^{k+1} = y^k$ with probability $P(x^k, y^k, T^k)$, and $x^{k+1} = x^k$ with probablity $1 - P(x^k, y^k, T^k)$

- Evolutionary algorithm

```
set initial population P(0)
for k=0,1...K do
  evaluate the fitness of all individuals in P(k)
  based on fitness select the parrents
  generate offsprings using crossover, mutation,...
  form new population P(k+1)
  if terminatin satisfied then
    break
  end if
end for
```

> Exploration : move to unknown territories, discover better
>
> Exploitation: stay close and explore the neighborhood

> Individual representation:
>
> Crossover: between two or more parents
>
> Mutation: single parent variation , small random changes to individual genes
>
> Selection:
>
> - Fitness proportionate selection:  select individual with probability $p_i = \frac{F(x_i)}{\sum F(x_j)}$ (Roulette wheel selection )
> - ranking selection: sort population, for each individual assign probability $p_i = G(i)$
> - tournament selection: randomly pick k individuals from the population select the highest fitness

- Niching

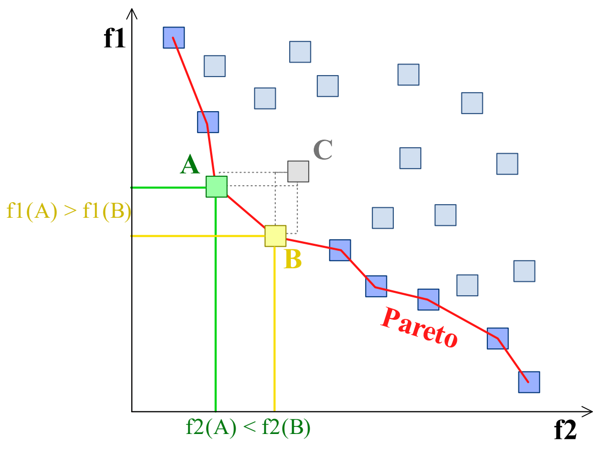
> motivation: close points does not contribute to the algorithm much
>
> Goal: spread points across the whole search space
>
> basic techniques:
>
> - sharing: modified the raw fitness by considering near points
> - crowding: removes nearby individuals (or prevents their creation)

- Constraint handling approaches:

> Penalty function: penalize the constraint violation
>
> Repair approach: similar to the projection approach
>
> Purist approach: Reject all infeasible points
>
> Separatist approach: Consider the objective and constraints separately
>
> Hybrid approach: Combination of the methods above

- Multi-objective programming

> Dominance: $\forall i \in \{1, \dots n\}, a \leq b$ **and** $\exists \in \{1, \dots, n\}, a < b$
>
> non-dominated: there is no point $x_2$ such that $x_2$ $dominates$ $x_1$
>
> Pareto front: set of all non-dominated points



Method:

- reformulate into one objective

  > scalarization : $minimize \sum w_m f_m(x)$
  >
  > - simple
  > - difficult to choose parameters
  > - only finds one point at a time
  >
  > $\varepsilon$ - constrained method: $minimize\ f_{m_0}(x)\ subject\ to\ f_m(x) \leq \varepsilon_m$ for $m \neq m_0$
  >
  > Goal programming

- Multi-object genetic algorithms
- NSGA-II (Non-dominated Sort Genetic Algorithm II)

```
Initialize population of size n
Non-dominated sort:
1. find pareto front in current population
2. From the rest find all non-dominated individuals and form second front
3. repeat till points remain
Fitess evaluation and crowding distance
1. assign fitness as the rank of the front
2. crowding distance measures the distance from "neighboring" points
Selection
1. tournament selection with k=2
2. first criterion is the fitness, thus lower front rank
3. when equal, individual with lower crowding distance is selected
Genetic operators
1. apply crossover and mutation based on the data representation and possibily on
the knowledge
Merge parents and offspring into one population with 2n individuals
Reduce the population size by selecting n individuals by criteria specified above
```

## Machine Learning

- Types:

  > Supervised learning
  >
  > unsupervised learning

> semi-supervised learning

- > Under-fitting: high training error and high test error
  >
  > - deal:  set a more complex model
  >
  > Over-fitting: low training error and high test error
  >
  > - set a less complex model
  > - regularization: penalize certain parts of the parameter space
  > - get more data

- Linear Model (detail) 🙀
  - Univariate Linear Regression (ULR)

    > Model Formulation: Linear Model $h_w(x) = w_0 + w_1 x$
    >
    > Optimization: $min_w L(w) = \frac{1}{2}\sum_{n=1}^{N}[y^n - (w_1 x^n + w_0)]^2$
    >
    > - closed form solution: First order equations equal to zero
    > - Iterative solution:
    >
    >   > Batch GD: update w once with all training data
    >   >
    >   > Stochastic GD: update w N times with one training data at a time
    >   >
    >   > Mini-batch GD: update w several times with a subset of D for one update

  - Multivariate Linear Regression (MLR)

    > Model Formulation: Linear model $h_w(x) = w_0 + w_1 x + \ldots + w_m x_m = \mathbf{w}^T \mathbf{x}$
    >
    > Optimization: $min_{\mathbf{w}} L(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}[y^n - \mathbf{w}^T \mathbf{x}^n]^2$
    >
    > - closed-form solution:  $\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$
    > - (how about the solution when adding the regularization) ❓❓❓
    >   $\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}$
    >   $$
    >   \begin{align}
    >   &\mathbf {Y =  Xw} \
    >   &\mathbf {X^TY  = X^T X w} \
    >   &\mathbf{w = (X$^{TX)}${-1}X^TY}
    >   \end{align}

    > $$
    >
    > - Iterative Solution
    >
    > Over-fitting for MLR: Regularization: $min_w L_{tr}(w) + \lambda\Omega(w)$ and $\Omega(w) = \sum_i |w_i|^p$

  - Multivariate Linear Classification (MLC)

    > Aim: find the optimal W fitting the observations in D
    >
    > Optimization: $min_w L(w) = \frac{1}{2N}\sum_{n=1}^{N}[y^n - h_w(x^n)]^2$
    >
    > MLR Model: $h_w(x) = w^T x$
    >
    > Problem: can not constrain 0/1 output

> - Hard-threshold Linear Classifier
> - Logistic Regression (Soft-threshold)

- Tree Model (detail) 🙆
  - Decision Tree
  - Information Gain: Good Feature Heuristics

    > Entropy: $H(Y) = -\sum_k p(y_k) log_2 p(y_k)$
    >
    > Conditional entropy: $H(Y|X) == \sum_j p(X = x_j) H(Y|X = x_j)$
    >
    > Information Gain: $IG(X) = H(Y) - H(Y|X)$

  - Decision Tree construction

    > When to stop:
    >
    > - all records in current subset have the same label
    > - all records have the same set input features
    > - all features have small information gain
    >
    > Tree Over-fitting: Decision Pruning

  - Decision tree for Regression
- Neural Networks
  - Loss function
  - Partial derivate for any w: "chain rule"
  - Back propagation to train ANN
- K nearest neighbour
  - For classification: find k nearest neighbours of the testing point and take a vote
  - For regression: take mean or median of the k nearest neighbours, or do a local regression on them.
  - Distance metric: Manhattan Distance; Euclidean distance
  - > Advantage: training is very fast, Learn complex target functions; do not lose information
    >
    > Disadvantage: slow at query time; Easily fooled by irrelevant attributes.
- Support Vector Machine

## Theory of Learning

- Bias-variance trade-off
  - Difference between Optimization and Learning

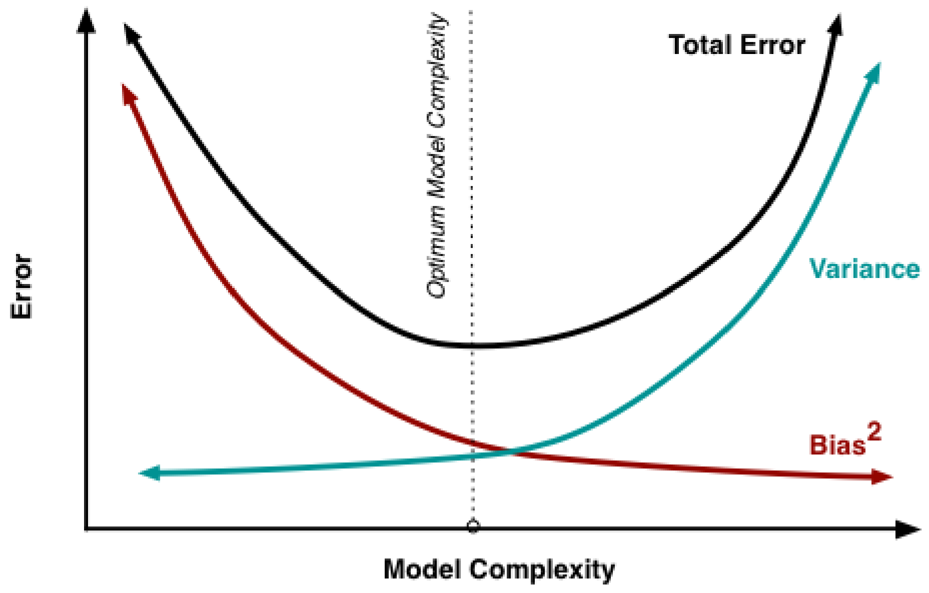
    > Optimization:
    >
    > - assumes all data are deterministic
    > - Computes it with high accuracy
    >
    > Learning :
    >
    > - assumes the data are stochastic (random)
    > - performance on the test set is more important

> - often large datasets than in optimization
> - Low precision usually sufficient

- $L(x) = \sigma^2 + var\ \widehat{f}(x) + bias^2\ \widehat{f}(x)$
  - Loss function consists of three terms

    > Irreducible error: Uncertainty in the data
    >
    > Squared bias: error due to simplification in the model, performance in the model.
    >
    > Variance: how well the method generalizes on different testing data

- Complexity choice



- Model selection
  - Learning process
  - What to do when something is wrong

    > Bad performance on the training set? → more complex, different model,...
    >
    > Good performance on the training set, bad in validation set? → over-fitting
    >
    > Good on training set, good in dev set ? → done.

  - Orthogonalization
    - design hyper parameters which have only one function, make tuning simpler
  - Input normalization
    - instead $x_1, x_2, \ldots, x_n$ consider $z_1, \ldots z_n$
    - $z_i = \frac{x_i - \mu}{\sigma}$ where $\mu = \frac{1}{n}\sum x_i$ and $\sigma^2 = \frac{1}{n-1}\sum(x_i - \mu)^2$
  - Increase training set size: individuals observations have smaller impact
    - how to add observations?
      - Gather them
      - add artificial observations: for example flip or rotate image.
  - Regularization
    - possible regularization: zero-norm, one-norm (LASSO), two-norm

    - Feature selection
        - Goal: reduce the number of features, set some $w_j$ to zero
        - Regularization:

            > $l_2$ regularization shrinks $w_j$ towards to zero
            >
            > $l_1$ regularization set $w_j$ to zero if below a certain threshold
            >
            > $l_0$ regularization counts non-zeros in $w_j$ and tries to set as many of them to zero as possible

        - Filter methods: select the active features based on some criterion
        - Wrapper methods: run the model repeatedly and subsequently add or remove features
- Implementation issues
    - Early stopping
        - don't run the algorithm for the whole time
    - Hyper parameters choice
        - how: fixed or randomly
    - Convexity
        - convex: if all $x_1, x_2$ and $t \in (0,1) f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$
        - if $f$ is differentiable, then $f$ is convex if and only if $\bigtriangledown^2 f(x)$ is positive semi-definite for all $x$.
        - A set $A \in \mathbb{R}$ is convex if all $x_1, x_2 \in A$ and $t \in (0,1)$ , $tx_1 + (1-t)x_2 \in A$
        - all linear function $f$ are convex since $\bigtriangledown^2 f = 0$
        - why convexity is important
            - the set of global minima is convex set
            - there are no local minima
            - For strictly convex functions (linear regression), the global minimal is unique
- Other topics
    - Boosting
        - combine several weal classifiers to create a stronger one
    - PAC learning
        - learning studies how many examples are needed to show that all consistent hypotheses are "good"