**Unsupervised Learning**

UL consists of algorithms that are meant to "explore" on their own and provide the user with valuable information concerning their dataset/problem

* Randomized Optimization
* Clustering
  + Single Linkage
    - Consider each of n points a cluster
    - Find the distance between the closest two points in every cluster
    - Merge the closest two clusters
    - Repeat n - k times to get k clusters
    - Method:
      1. Consider each point a cluster
      2. Merge two closest clusters
      3. Unless we have K clusters GOTO 2
    - Links points closest to each other.
    - Can result in "stringy" non-compact clusters
  + K-Means
    - Each iteration is polynomial
    - Finite (exponential) iterations in theory, but usually much less in practice
    - Always converges, but can get stuck with "weird" clusters depending on random starting state
      1. Place k centers
      2. Claim closest points
      3. find the centers of the points
      4. Move the centers to the clusters of points
      5. Unless converged GOTO 2
  + Expectation Maximization
    - Gaussian Means
    - Uses expectation and maximization steps
    - Monotonically non-decreasing likelihood
    - Does not converge (practically does)
    - Can get stuck
    - Works with any distribution (not just Gaussian)
  + Properties of Clustering Algorithms (Pick 2)
    - Richness
    - Scale Invariance
    - Consistency
  + Richness
    - For any assignment of objects to clusters, there is some distance matrix, D, such that P\_D returns that clustering
  + Scale-Invariance
    - Scaling distances by a positive value does not change the clustering
  + Consistency
    - Shrinking intra-cluster distances and expanding intercluster distances does not change the clustering.
  + **No clustering scheme can acheive all of Richness, Scale-Invariance, Consistency**
* Feature Selection
  + Filtering
    - Choose features independent of learner. i.e. "filter" the data before it is passed to the learner
    - Faster than wrapping (don't have to pay the cost of the learner)
    - Tends to ignore relationships between features
    - Decision Trees do this naturally (Filter on information gain)
  + Wrapping
    - "Wrap" the learner into the feature selection. Choose features based on how the learner performs.
    - Takes into account learner bias
    - Good at determining feature relationships (as they pertain to the success of the learner)
    - Very slow (have to run the learner for each feature search)
    - Speed Ups
      1. Randomized optimization
      2. Forward/Backward sequential selection: [good description and implementation](http://sebastianraschka.com/Articles/2014_sequential_sel_algos.html)
  + Relevance
    - [x_i](https://camo.githubusercontent.com/0f6ca4794fffc6c7791fa1c95c12ad9431645f7c/687474703a2f2f6d61746875726c2e636f6d2f32617a3263376d2e706e67) is strongly relevant if removing it degrades the Bayes' Optimal Classifier
    - [x_i](https://camo.githubusercontent.com/0f6ca4794fffc6c7791fa1c95c12ad9431645f7c/687474703a2f2f6d61746875726c2e636f6d2f32617a3263376d2e706e67) is weakly relevant if
      1. it is not strongly relevant
      2. [There exists](https://camo.githubusercontent.com/f2aaa39ea637e908e9ea07ac470b037f16bb3c68/687474703a2f2f6d61746875726c2e636f6d2f79687936676c612e706e67) a subset of features **S** such that adding [x_i](https://camo.githubusercontent.com/0f6ca4794fffc6c7791fa1c95c12ad9431645f7c/687474703a2f2f6d61746875726c2e636f6d2f32617a3263376d2e706e67) to **S** improves Bayes' Optimal Classifier
      3. [x_i](https://camo.githubusercontent.com/0f6ca4794fffc6c7791fa1c95c12ad9431645f7c/687474703a2f2f6d61746875726c2e636f6d2f32617a3263376d2e706e67) is otherwise irrelevant
  + Relevance vs. Usefulness
    - **Relevance** measures the effect the variable has on the Bayes' Optimal Classifier
    - **Usefulness** measures the effect the variable has on the *error* of a *particular predictor* (ANN, DT, etc.)
* Feature Transformation
  + Polsemy: Same word different meaning - False Positives
  + Synonomy: Different word same meaning - False Negatives
  + PCA: [Good Slides](http://www.cc.gatech.edu/~agray/4245fall10/lecture18.pdf)
    - Example of an eigenproblem
    - Finds direction (eigenvectors) of **maximum variance**
    - All principal components (eigenvectors) are mutually orthogonal
    - Reconstructing data from the principal components is proven to have the least possible L2 (squared) error compared to any other reduction
    - Eigenvalues are monotonically non-increasing and are proportional to variance along each principal component (eigenvector). **Eigenvalue of 0 implies zero variance which means the corresponding principal component is irrelevant**
    - Finds **"globally"** varying features (image brightness, saturation, etc.)
    - Fast algorithms available
  + ICA
    - Finds new features that are completely **independent** (from each other). i.e. they share no mutual information
    - Attempts to maximize the mutual information between the **original** and **transformed** data. This allows original data to be reconstructed fairly easily from the transformed data.
    - Blind Source Separation (Cocktail Party Problem)
    - Finds **"locally"** varying features (image edges, facial features)
  + RCA
    - Generates random directions
    - It works! If you want to use it to preprocess classification data...
      1. Is able to capture correlations between data, but in order for this to be true, you must often reduce to a larger number of components than with PCA or ICA.
    - Can't really reconstruct the original data well.
    - Biggest advantage is speed.
  + LDA
    - Requires data labels
    - Finds projections that discriminate based on the labels. i.e. separates data based on class.
* Information Theory
  + Entropy: [A characterization of uncertainty about a source of information](http://en.wikipedia.org/wiki/Entropy_(information_theory))
    - [Entropy Formula](https://camo.githubusercontent.com/5f8440a5c9922ddb21110dfb0e600c5646410ac8/687474703a2f2f6d61746875726c2e636f6d2f70646d7a36366b2e706e67)
  + Joint Entropy: [The entropy contained by the combination of two variables](http://en.wikipedia.org/wiki/Joint_entropy)
    - [Joint Entropy Formula](https://camo.githubusercontent.com/57b6695ec3ee3aea2e2459a2e009a6dfc258f200/687474703a2f2f6d61746875726c2e636f6d2f6c337432656b6c2e706e67)
  + Conditional Entropy: [The entropy of one variable, given another](http://en.wikipedia.org/wiki/Conditional_entropy)
    - [Conditional Entropy Formula](https://camo.githubusercontent.com/26731d6001bcb57ef4a5074dab28a7095c5f07ac/687474703a2f2f6d61746875726c2e636f6d2f707671376e71342e706e67)
  + Mutual Information: [The reduction of entropy of a variable, given knowledge of another variable](http://en.wikipedia.org/wiki/Mutual_information)
    - [Mutual Info Formula](https://camo.githubusercontent.com/e3cb387204b83988a1972354a066978a390ed5a3/687474703a2f2f6d61746875726c2e636f6d2f6f3765733467682e706e67)
  + KL Divergence: [A non-symmetric measure of the difference between two probability distributions P and Q](http://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence)
    - [KL Divergence Equation](https://camo.githubusercontent.com/b88743fe880bd5ebf053e9f18121073705bb9ade/687474703a2f2f6d61746875726c2e636f6d2f6b6c6435756d762e706e67)
    - Can be used in supervised learning as an alternative to squared error

**Reinforcement Learning**

[Reinforcement Learning: A Survey](http://www.jair.org/media/301/live-301-1562-jair.pdf)

Put an agent into a world (make sure you can describe it with an MDP!), give him some rewards and penalties and hopefully he will learn.

* **Markov Decision Processes**
  + Building a MDP
    - States
      * MDP should contain all states that an agent could be in.
    - Actions
      * All actions an agent can perform. Sometimes this is a function of state, but more often it is a list of actions that could be performed in any state
    - Transitions (model)
      * Probability that the agent will arrive in a new state, given that it takes a certain action in its current state: P(s'|s, a)
    - Rewards
      * Easiest to think about as a function of state (i.e. when the agent is in a state it receives a reward). However, it is often a function of a [s, a] tuple or a [s, a, s'] tuple.
    - Policy
      * A list that contains the action that should be taken by the agent in each state.
      * The **optimal policy** is the policy that maximizes the agent's long term expected reward.
  + Utility
    - The utility of a state is the reward at that state plus all the (discounted) reward that will be received from that state to infinity.
    - Accounts for *delayed* reward
    - Described by the Bellman Equation
      * [Bellman Equation](https://camo.githubusercontent.com/fdd45d7396302a7560a1c545f80908ff365d5acf/687474703a2f2f6d61746875726c2e636f6d2f6f6a37356c6a662e706e67)
  + Value Iteration
    - "Solve" (iteratively until convergence, more like hill climb) Bellman Equation.
    - When we have maximum utility, the policy which yields that utility can be found in a straightforward manner.
  + Policy Iteration
    - Start with random (or not) initial policy.
    - Evaluate the utility of that policy.
    - Update policy (in a hill climbing-ish way) to the neighboring policy that maximizes the expected utility.
  + Discount Factor, [gamma](https://camo.githubusercontent.com/8574d5702702e0b630a697606a9ee03439a92c09/687474703a2f2f6d61746875726c2e636f6d2f7062686d78642e706e67) (typically between 0 and 1), describes the value placed on future reward. The higher [gamma](https://camo.githubusercontent.com/8574d5702702e0b630a697606a9ee03439a92c09/687474703a2f2f6d61746875726c2e636f6d2f7062686d78642e706e67) is, the more emphasis is placed on future reward.
* **Model-Based vs. Model-Free**
  + Model-Based requires knowledge of transition probabilities and rewards
    - Policy Iteration
    - Value Iteration
  + Model-Free gets thrown into the world and learns the model on its own based on "[s, a, s', r]" tuples.
    - Q Learning
* Three types of RL
  + Policy Search - direct use, indirect learning
  + Value function based - ^Argmax
  + Model based - indirect use, direct learning ^Solve Bellman
* **Q Learning**
  + Q Function is a modification of the Bellman Equation
    - [Q Function](https://camo.githubusercontent.com/009464fd2a28d6d8a8301b65d5817d1e5065699b/687474703a2f2f6d61746875726c2e636f6d2f6b6879733538762e706e67)
    - [U(s)](https://camo.githubusercontent.com/30b38d7fffd9c14862b4192b51cfe7f2d10b14ac/687474703a2f2f6d61746875726c2e636f6d2f6f386c6e6e6e6b2e706e67)
    - [Pi(s)](https://camo.githubusercontent.com/ca1fa8f1e1a4644e57aa46bc623b27dc4918be12/687474703a2f2f6d61746875726c2e636f6d2f706e667a357a362e706e67)
  + Learning Rate, [alpha](https://camo.githubusercontent.com/82866c0b4fd1a65228fd1ae56d56c9b3804dc665/687474703a2f2f6d61746875726c2e636f6d2f3832377461672e706e67), is how far we move each iteration.
  + If each action is executed in each state an infinite number of times on an infinite run and [alpha](https://camo.githubusercontent.com/82866c0b4fd1a65228fd1ae56d56c9b3804dc665/687474703a2f2f6d61746875726c2e636f6d2f3832377461672e706e67) is decayed appropriately, the Q values will converge with probability 1 to Q\*
  + Exploration vs Exploitation
    - Epsilon Greedy Exploration
      * Search randomly with some decaying probability like Simulated Annealing
    - Can use starting value of Q function as a sort of exploration
* **Game Theory**
  + [**Zero Sum Games**](http://en.wikipedia.org/wiki/Zero-sum_game)
    - A mathematical representation of a situation in which each participant's gain (or loss) of utility is exactly balanced by the losses (or gains) of the utility of the other participant(s).
  + **Perfect Information Game**
    - All agents know the states of other agents
    - minimax == maximin
  + **Hidden Information Game**
    - Some information regarding the state of a given agent is not know by the other agent(s)
    - minimax != maximin
  + [**Pure Strategies**](http://en.wikipedia.org/wiki/Strategy_%28game_theory%29#Pure_and_mixed_strategies)
  + [**Mixed Strategies**](http://en.wikipedia.org/wiki/Strategy_%28game_theory%29#Pure_and_mixed_strategies)
  + [**Nash Equilibrium**](http://en.wikipedia.org/wiki/Nash_equilibrium)
    - No player has anything to gain by changing only their own strategy.
  + Repeated Game Strategies
    - Finding best response against a repeated game finite-state strategy is the same as solving a MDP
    - Tit-for-tat
      * Start with cooperation for first game, copy opponent's strategy (from the previous game) every game thereafter.
    - Grim Trigger
      * Cooperates until opponent defects, then defects forever
    - Pavlov
      * Cooperate if opponent agreed with your move, defect otherwise
      * **Only strategy shown that is subgame perfect**
  + Folk Theorem: Any feasible payoff profile that strictly dominates the minmax/security level profile can be realized as a Nash equilibrium payoff profile, with sufficiently large discount factor.
    - In repeated games, the possibility of retaliation opens the door for cooperation.
    - Feasible Region
      * The region of possible average payoffs for some joint strategy
    - MinMax Profile
      * A pair of payoffs (one for each player), that represent the payoffs that can be achieved by a player defending itself from a malicious adversary.
    - Subgame Perfect
      * Always best response independent of history
    - Plausible Threats
  + **Zero Sum Stochastic Games**
    - Value Iteration works!
    - Minimax-Q converges
    - Unique solution to Q\*
    - Policies can be computed independently
    - Update efficient
    - Q functions sufficient to specify policy
  + **General Sum Stochastic Games**
    - Value Iteration *doesn't* work
    - Minimax-Q *doesn't* converge
    - *No* unique solution to Q\*
    - Policies *cannot* be computed independently
    - Update *not* efficient
    - Q functions *not* sufficient to specify policy