**Chapter 1 – Introduction**

*Definition of Learning*3 things required in the definition of learning:

1. Experience: Dataset used
2. Task: well defined task
3. Performance metric: test itself to see if the learner is getting better

*Designing a Learning System*Need to create a well-defined task, choose a dataset (experience) and performance metric.

*Learning as search*ML be search because the algorithms search a large space of possible hypotheses to find the one that best fits the observed data. The learner searches through the space to locate the hypothesis that is most consistent with the available training examples.

**Chapter 2 – Concept Learning**

*Supervised vs Unsupervised Learning*3 types of learning:

1. Supervised Learning: model predicts/classifies a dependent variable. Uses labelled data.
2. Unsupervised Learning: model is run on a set of data that does not have an explicit dependent variable. Aim of unsupervised learning is to find a structure and pattern(s) (e.g. grouping to clustering) in the data.
3. Semi-supervised learning: model is fit on dataset where only a portion of the dataset has been labelled. A task of semi-supervised learning is to label unlabelled data.

*Types of supervised learning*3 types of supervised learning:

1. Classification (discrete): task is to identify which of a set of categories a new observation belongs to. This is the category membership problem.
2. Regression (real/continuous): task is to predict a dependent numerical variable. It is used to estimate relationships among variables.
3. Ordinal Regression: task is to predict a dependent variable that is ordered but not continuous.

Concept Learning: subclass of classification. In concept learning the target label is binary (yes/no).

*Inductive Hypothesis*The inductive hypothesis is a hypothesis that approximates the target function well over a sufficiently large enough set of training data such that it will also approximate the target function well over test examples (unobserved examples).

*Version Spaces*  
A version space is set of hypotheses consistent with the training examples.

*Inductive Bias*Allows a learner to generalize and thus predict unseen examples. Without inductive bias the learner is a database (wrote-learning).   
Unbiased learners are not able to generalize beyond the observed examples.

Inductive Bias of the Candidate Elimination algorithm: The target concept (c) can be represented in its hypothesis space (H).

Inductive Bias of Find S:The target concept can be described in its hypothesis space. All instances are negative instances unless demonstrated otherwise.

Every learner has inductive bias and every learner has a different one. That’s why learners perform differently on datasets.

Note: more strongly biased methods make more inductive leaps? In theory strong bias is good because you get to a good hypothesis faster, but you could be taking steps in the wrong direction which is an issue.

**Chapter 3 – Decision Tree Learning**

Decision trees are commonly used in classification problems as the target function is discrete valued. They classify instances down the tree.

The Find-S, Candidate Elimination algorithms cannot handle any errors in the data. Decision tree’s are able to handle errors in the data or missing data.

*Information Theory and ML*Machine learning is the same as compression. For example, in lossy compression we generalize details of image to save space and therefore lost some detail. In Decision Tree’s we generalize information from dataset and predict using it. The base of both compression and machine learning is information theory. Thus, this is how information theory relates to machine learning.

*Searching in Decision Trees*The hypothesis space of ID3 (decision tree algorithm) is the space of all possible decision trees. ID3 searches the space of all possible decision trees from simplest to complex trees using the information gain heuristic.   
ID3 only maintains a single hypothesis at a given time. We cannot tell how many alternative decision trees are also consistent with the available training data.

*Inductive Bias in Decision Trees*- Shorter trees are preferred over longer trees.   
- Trees that place high information gain attributes close to the root are preferred over those that do not.

*Restriction and Preference Biases  
Preference (search bias):* incompletely search a complete hypothesis space. ID3 has a preference of search bias. The bias is on how the space is searched not the space itself.  
*Restriction (language bias)*: completely search an incomplete hypothesis space. Candidate Elimination has restriction bias as it searches a constrained hypothesis space.

Note that preference bias is more desirable than restriction bias. This is because if we search the space enough, we will find true hypothesis but if we have a restriction bias (and the true hypothesis is not within the constrained space function is not within the space).

*Occam’s Razor*Prefer the simplest hypothesis that fits the data. Prefer the shortest tree that fits the data. The idea behind this is shorter tree’s will not overfit.

*Overfitting*The hypothesis that best fit the training data is not the hypothesis that best fit the test data.  
Overfitting you will see the training accuracy is way higher than the test accuracy.

*What increases* overfitting

* Number of training instances decreases
* Number of attributes (features) increases
* Noise (errors) in the data increases
* Signal is more complex

*Approaches to overfitting*

* Stop growing the tree early
* Post-prune the tree
  + Prune the tree using a validation set
  + Reduced Error Prune
  + Rule Post Prune
* Create a validation set
* Statistical test

**Chapter 4 – Statistics**

*Sample Error vs True Error*Sample Error – fraction of instances in sample S that algorithm misclassifies  
True Error – probability the learner will misclassify a single randomly drawn instance from the population

*Bias vs Variance*Bias is how far your hypothesis is from the true value  
Variance: how much your hypotheses differ from one another

*Underfitting and Overfitting*Underfitting – High bias low variance  
Overfitting – Low bias and high variance

Overfitting – performance on the test set is much worse than the training set  
Underfitting – performance on the training set and test set are the same (both are bad!)

*Types of bias*

1. *Machine Learning Bias*

Every inductive learning algorithm must adopt a bias inorder to generalize beyond the training data.

1. *Systematic Error Bias*

If there is systematic error in the training set (systematic error = equipment error etc) the algorithm cannot tell the difference between systematic error and real signal. Cannot tell error from signal.

1. Statistical Bias

Statistical bias is the systematic error for a given sample size. As the sample size gets smaller the statically bias goes up (errors goes up)

*4 sources of error*

1. Random variation in the selection of the test data
   1. Poor train test split
2. Random variation in the selection of the training data
   1. Poor train test split
3. Randomness in the learning algorithm
   1. Seeing initial weights to random seeds
4. Random classification error
   1. Human error

To account for train variation and randomness perform cross-validation.  
To account for test data variation and random classification error use statistical test that considers the size of the test set and the consequences of changing the test set.

*Overfitting*

Multiple comparisons problem: more dof’s, more variables more complex the problem becomes and the more likely to overfit to the dataset.

*What causes overfitting?*Multiple comparisons causes overfitting, over searching and feature selection problem.   
*Solutions to multiple comparisions?*1. Everytime you change the model get a new training/test set

2. Bonferroni adjustment

3. Cross validation

4. Randomization testing

*Randomization Testing*Tells you the probability of your result being purely by chance. Thus, it tells you the probability of overfittings for this particular dataset.   
So the randomization test is testing whether the accuracy of a model is just from random luck or whether it's actually finding patterns and correlations in the data.

If the accuracy of your ML algorithm is not out in the tails then you have overfit to the data.

*Types of Error (Type 1 and Type 2)*

Type 1: False Positive (no effect)

* The error of rejecting a null hypothesis when its actually true

Type 2: False Negative

* Failing to reject a null hypothesis when it is in fact not true

We care more about Type 1 error than Type 2 error. This is because the conclusion is drawn that the null hypothesis is false when, in fact, it is true. Type 1 is a traditionally thought error we have.

**Chapter 5: Ensembles**

*Key to ensembles*1. Individual classifiers must be better than random  
2. Individual classifiers must be different enough from each other

*Methods for constructing ensembles*1. Manipulating the training set – bagging, boosting  
2. Manipulating the input features – attribute bagging  
3. Manipulating the output features  
4. Injecting randomness

Manipulating the training set  
- Bagging: Sample with replacement to create multiple training samples  
- Cross validation  
- Boosting: adjust probability distribution over training instances (duplicate datapoints deliberately changing the distribution of training examples)

Manipulate the input features (feature selection)  
- Re sample the input features  
- This does not work in cases where all the features are needed to predict the class. Features must be highly redundant.

Manipulate the output target  
- Error correcting output code (EROC)

Inject Randomness  
- Initialize weights of ANN with random numbers  
- To inject randomness into decision trees: Select attribute at top of tree using information gain. To inject randomness, select top 3 attributes according to information gain and then randomly pick way attribute that goes onto the top.

*Ways to combine classifiers*1. Unweighted vote – allow each classifier to vote and calculate majority vote  
2. Weighted vote – weight votes of classifiers according to accuracy of individual classifier  
3. Learn good weights – use a gating network  
4. Stacking

*Why do ensembles work? (3 reasons)*1. Statistical

The training data may not be enough to narrow the hypothesis space down to a single good hypothesis – there are many equally good hypotheses. So, if we take a bunch of learners and have them vote, so hopefully we will get closer to the true hypothesis.

2. Computational

Difficult search problem – finding an optimal single hypothesis that fits the data may take a long time. Instead of just keeping searching, run a number of individual classifiers (an ensemble) for a shorter period of time. Let each ensemble find suboptimal parameters but when you combine them you should get a more optimal overall set of parameters.

3. Representational

Hypothesis space may not contain the true hypothesis because of our restrictive representation. However, if we let ensemble run, we may be able to combine the individual classifier results in the ensemble to find an approximate of the target function that is outside of the hypothesis space.

*Cross validated communities vs Bagging*Bagging always does better than cross validation (CV). Bagging always does better than CV because some of the datapoints are duplicated causing a warping of the distribution of datapoints. This means the trees are more different from each other than if you use CV only.

*Random Forests*Random forest perform so well because they perform bagging (bagging on the data) and feature/attribute bagging. The tree’s produced in the ensemble are very diverse thus covering a larger variety of cases.

*Hypothesis space of ensembles?*

**Chapter 6 – ANN**

*Hypothesis space of ANN?*Every passible assignment of network weights represents a different hypothesis. Thus, the hypothesis space of ANN is a n by n dimensional vector containing the network weights.

SGD and GD are used to find the optimal set of network weights.

*Difference between GD and SGD?*In GD the error is summed over all examples before updating the weights. Summing over all examples requires more computational power but since you are using the true gradient you can be sure you’re taking a step in the right direction and thus can use larger learning rate.   
In SGD weights are updated upon examining each training example. This is computationally less expensive but requires you to use a smaller learning rate typically to be sure you don’t step too far in the wrong direction.

If there are multiple local minima (in the error surface) SGD can avoid falling into these minima compared to GD. This is because for SGD to get stuck it must get stuck on each datapoint individually over the entire dataset.

SGD is used generally more than GD because of its ability to less likely get stuck in a local minimum.

SGD and GD are the methods by which the hypothesis space is searched.

**Chapter 7 – ANN 2**

*Backpropagation Algorithm*Learn weights for a multilayer network*.* Uses GD to minimize squashed error between network outputs and target values.  
If you do too few iterations of backprop then you fail to reduce error sufficiently.  
If you do too many iterations of backprop then you overfit to the data.

*Momentum*Momentum makes the weight update in the nth iteration dependent on the weight update in the n-1th iteration.  
The idea behind this is that if you take a step in the same direction over two iterations then the second time to take that step it will be slightly bigger (than normal). Then, If you take a step in the same direction for a 3rd iteration then the third step will be larger what would be normally. The hope is to converge faster because you are going in the right direction, so you take bigger steps.

Momentum allows you to roll through local minima and as it increases the step size in regions where the gradient is unchanging convergence should be sped up.

*Why ANN’s (back-propagation) does not get stuck in local minima?*In practice networks are so large (have a large number of weights) thus correspond to error surfaces with a very high dimensional space. When GD falls into a local minimum with respect to one weight it won’t fall in the local minima on all the other weights. Hence the more weights (higher dimensions) you can get out of local minima as it only needs one point to jump out of the local minima.

*To overcome local minima (4 methods)*- Add momentum  
- SGD because you evaluate weights at every training instance means you are less likely to get stuck in local minima  
- New seed (random initial weights)  
- Add more dimensions

*Inductive Bias of ANN/Backprop*The assumption made by ANN’s to generalize is a smooth interpolation between data points. For example, if there are two positive training instances (with no negatives between them) backpropagation will label the points in between also as positive (smooth interpolation).

**Chapter 8: ANN 3**

*Hidden Layer Representations (compact representations of data)*Backpropagation can discover useful intermediate representations within the hidden unit layers. They make implicit concepts explicit.

The more layers of hidden units, the more complex features can be generated. The problem with more layers is overfitting.

*ANN Overfitting*Note that ANN’s are hard to underfit but very easy to overfit.

To stop overfitting add stopping criteria:

* Train until the error on the training examples falls below some level
* Stop when you reach the lowest level on the validation set

*Hidden Layers/Units*In general there is a minimum number of hidden units needed. Above this amount any extra will not increase dramatically and will instead:

* Take longer to train the network
* Increases the tendency to overfit

**Chapter 9: Genetic algorithms (GA)**

Genetic algorithms are search algorithm inspired by evolution.

The best hypothesis is defined as the one that optimizes the fitness function.   
The GA iteratively updates a pool of hypotheses (each hypothesis is an individual within the population).

*Crossover*Additional members are generated using crossover. Crossover takes two parents’ hypotheses from the current population and creates two offspring hypotheses by recombining portions of both parents.

*Mutation*A proportion of the population are chosen at random and random mutations are performed (for example randomly flip a bit).

*Hypothesis space of GA*Populations are defined as sets of hypotheses {H1, H2 etc}. Thus the hypothesis space of genetic algorithms is the sets of possible hypotheses.

*How the space is searched?*Randomized beam search method to seek maximally fit hypothesis.

*Why do GA’s work?*1. Children of high fitness parents should have high fitness  
2. Blocks of close genes: the closer to genes are two each other, the more likely the are to be passed onto the same child. Thus closely related genes should be close to one another.

*Early convergence*The early convergence problem occurs when a relatively very individual generates copies of itself and creates similar individuals that slowly take over the population. Early convergence lowers diversity of the population and slows down search of the GA.

*Overcrowding – Early convergence*Whole population becomes copies and mutations of one individual. Early convergence is due to overcrowding. Early convergence is a local minima.

*Niching Solutions – avoid early convergence*1. Tournament/rank selection – delays early convergence

2. No incest - Not allowing people to crossover with themselves or ancestors of themselves. Delays early convergence.

3. Restricting the kinds of individuals allowed to combine

Run GA within little groups every once in a while, allowing crossover between the groups to add diversity. This means if there is one fit individual it takes them longer to take over the whole population. The fit individual will eventually take over, but this delays that problem. The hope is you will be closer to the global minima by the time when you converge. Delays early convergence.

4. Crowing: Create a new offspring and then check population for person closest to the new offspring (Use some sort of similarity metric). Replace that person with the new offspring. This delays early convergence.

5. Clearing Procedure: This method ensures early convergence does not occur. Minimal distance between all members of a population. Every person has a to have an area around them with no people. If I have a new child and put it into the population any current members in the population within their radius area will be removed.

Why it works? Guarantees genetic diversity and that people are different enough from one another.

6. Sharing: Individuals close to one another share fitness values. As you put more and more of this very fit person in their fitness will decrease and they are less likely to be selected. This method preserves genetic diversity.

*Theory of GA – Discovery and Retention*

*Lamarkian Evolution:* Experiences of an individual during their lifetime can be passed to their offspring.

*Baldwin Effect:* Any individual that improves themselves by learning during their lifetime that maximises their fitness can support is going to do better in the gene pool. More diverse gene pool and thus more rapid evolution.

*Baldwin vs Lamarkian (example ANN)*

* **Baldwin**: Train NN using backprop and then evaluate each NN based on their final fitness. But when we cross them over, we use the initial weights they started with.

In essence Baldwin is saying this NN has a good fitness/chance of going into the next generation because when trained it got good weights. However, when going to the next generation use initial weights as that knowledge **(knowledge is not passed on)**

* **Lamarkian:** Pass on knowledge in their lifetime. Train neural nets then cross them over to generate offspring neural nets. The things those neural nets learnt was passed onto their children. **(knowledge is passed on)**

**Chapter 11 – Particle Swarm Optimisation**

*Main Characteristics*Collect behaviours (global behaviour) that result from local interactions of the individuals with each other or with the environment.

*Properties of Swarm Intelligence*1. Many agents follow very simple rules

2. No central control structure

3. Simple local interactions lead to emergence of complex global behaviour

*The Power –* the power of swarm technology is in interactive collaboration.

*ACO Hypothesis Space:* The hypothesis space of ACO is the vector of pheromone values.

*ACO advantage:*

1. Near optimal solutions
2. When graph changes dynamically, ACO will adapt and continue to work

*PSO Hypothesis space:* The hypothesis space is the vector of each particles position

*Convergence:*

PSO will always converge due to one-way communication, just may not converge to the best solution.   
PSO will also do early convergence.  Solve it down by using neighbourhood updates.

*One-way information sharing*

* Only best information gets shared with other particles. This is why PSO converges quicker than GA.

*One way or multi-way information sharing?*

Case by case scenario.

Multiway sharing will not converge as quickly and therefore is more likely to end in the global minima. But it does take longer to converge. Further, low quality characteristics are still able to propagate through multiple iterations in GA’s vs in PSO only the best information is shared.

One-way sharing will converge quicker and can get stuck in local minima (early convergence).