Part IA Machine Learning and Real-world Data Notes

1 Statistical classification

- Sentiment Detection
  - the task of automatically deciding whether a review is good or bad, based on the text of the review
  - Common tasking in natural language processing
- Tokenisation
  - The process of breaking text into words, phrases, symbols, or other meaningful elements called tokens.
  - More complex than just splitting on white space, e.g. words at the start of a sentence will be capitalised
- Accuracy
  - Success can be measured in the number of correct decisions c over all decisions
  - For c correct decisions and i incorrect decisions

$$A = \frac{c}{c+i}$$

- Data should be split into training and testing sets
  - This is because if you try and evaluate your system on data you have trained it on, you are doing something unrealistic
- Machine learning typically involves four phases
  1. Training
     - The process of making observations about some known data set
     - The function can be altered in this phase
  2. Development (also known as Validation or Tuning)
     - Used to tune the model
  3. Testing (aka Evaluation)
     - Should only be used once otherwise
     - If used multiple times can tune to test data set
     - Used as a one off test for the system
  4. Use
     - Use in the real-world
- Machine Learning - a program that learns from data, i.e., adapts its behaviour after having been exposed to new data.
- Machine Learning performs two tasks
  - Classification - Which class should the data have?
  - Prediction - Which data is likely to occur in the given situation?
- Two types of machine learning
  1. Supervised ML: you use the classes that come with the data in the training and the testing phase

2. Unsupervised ML: you use the classes only in the testing phase

- There are two types of classifier
  1. Generative - Build a model for each class
     - Given an observation they return the class mist likely to have generated the observation
  2. Discriminatory - Learn what features in an input are useful to discriminate classes
     - More accurate
- A feature or observation is a piece of easily observable data
- A classification function maps a set of features or observations to a class
- Often must use statistical models which are wrong because
  - Computational tractability
  - Acquisition of training data
- In a document
  - A type is a unique word - multiple of the same word count for one type in a text
  - A token is a word - multiple of the same word count for multiple tokens in a text

1.1 Naive Bayesian Classifier

- Bayes' rule

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)}$$

  where $P(c)$ is the prior probability of a class, and $P(d|c)$ is the likelihood of the document

- A statistical classifier maximises the probability that a class c is associated with the observations o, and returns the maximising class $\hat{c}$

$$\hat{c} = \arg\max_{c \in C} P(c|o)$$

- Notice that $P(d)$ does not effect $\hat{c}$
- We assume that the appearance of words is independent - **Naive Bayes Assumption**

$$P(w_1, w_2, \ldots, w_n|c) = P(w_1|c) \times P(w_2|c) \times \cdots \times P(w_n|c)$$

- For a document d, consisting of words w_i

$$C_{NB} = \arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} P(c) \prod_{i \in \text{Positions}} P(w_i|c)$$

- Maximum Likelihood Estimate, is uses the frequency of data to calculate the probabilities

$$\hat{P}(w_i|c) = \frac{\text{count}(w_i, c)}{\sum_{w \in V} \text{count}(w, c)}$$

  where $\text{count}(w, c)$ is the number of times the word w appears in a class c and V is the vocabulary

$$\hat{P}(c) = \frac{N_c}{N_{doc}}$$

  where N_c is the number of documents of a class c in the training data, and N_{doc} is the total number of documents in the training data.

- This is done in log space to avoid underflow and increase speed (addition not multiplication)
- Treat document as a bag of words
  - An unordered set
  - Positions ignored, only frequencies kept
- Zero occurrences of a word result in a probability of zero, and because these a multiplied together result in a zero probability for the class
  - Solved by add one smoothing
- Unknown words are ignored
- Stop words are often removed, these are the most common words in the corpus and offer no information for classifying
- In sentiment detection tasks
  - Can add the prefix `NOT_` to words between a negating word (e.g. not, didn't) and the next piece of punctuation

1.2 Statistical Laws of Language

1.2.1 Zipf's Law

- There is an inverse relationship between a word's frequency rank and the absolute value of the frequency

$$f_w \approx \frac{k}{r_w^\alpha}$$

  where f_w is the frequency of the word w, r_w is the frequency rank of w, α and k are language specific constants
- There are a few very frequent terms and very many rare terms

1.2.2 Heap's Law

- There is an exponential relationship between the size of a vocabulary and the size of the text that gave rise to it

$$u_n = kn^\beta$$

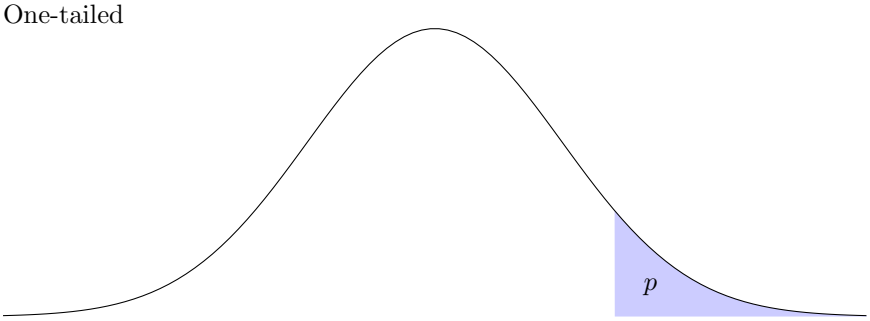
  where u_n is the number of unique words, n is the number of tokens, and k, β are language specific constants
- Vocabulary size continues to grow with a larger corpus, rather than trailing off

1.2.3 Smoothing

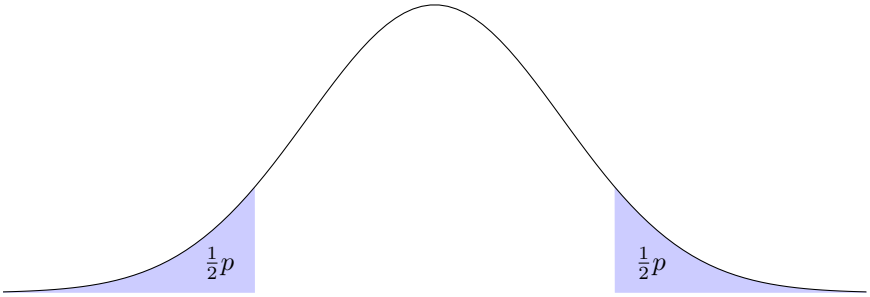
- Maximum likelihood estimation overestimates the likelihood of seen words
- In reality the majority of the probability mass is in the tail, with words with a lower frequency
- Smoothing redistributes this probability mass away from the seen words, making it more realistic

1.3 Significance Testing

- H_0 - Null hypothesis - No effect, formulated to be rejected, 2 results come from the same distribution
- H_1 - Alternative hypothesis - What you want to prove
- Choose a level of significance, p
- If the probability of observing events under the null hypothesis is very small $< p$, we can reject the null hypothesis
- 2 types of error
  1. Type I - Rejecting the null hypothesis when it is in fact true
  2. Type II - Accepting the null hypothesise when it is in fact false
- The region of rejection is a subset of all possible values such that the probability of lying within the region is p
  - If alternative hypothesis is directional then one-tailed, else two-tailed
  - One-tailed



  - Two-tailed



    * Always use two tailed because otherwise assuming our chosen system is better
- Error bars are used to show the range of data

1.3.1 Sign Test

- Used to compare two systems
- Null hypothesis

$$\text{The number of pairs where } X_i < Y_i \text{ is equal to } X_i > Y_i$$

- Use a binomial distribution with $p = q = \frac{1}{2}$
- In the case of ties
  1. Ignore
  2. Add half to `PLUS`, half to `MINUS` and round up

1.3.2 Agreement and Uncertainty

- Human agreement is the only empirical available source of truth in decisions which are influenced by subjective judgement
  - Something is true if several humans independently agree on it
  - The more they agree the more true it is

1.3.3 Kappa

- Kappa is used to measure agreement
  - Varies between -1 and 1
  - 0 random
  - 1 complete agreement
  - < 0 less than random agreement
  - 0.8 good agreement
  - 0.69 marginal agreement
- Calculated by

$$k = \frac{P(A) - P(E)}{1 - P(E)}$$

where

  - $P(A)$ - Proportion of times people agree

$$P(A) = \frac{\text{number of observed pairwise agreements}}{\text{number of possible pairwise agreements}}$$

    for N judges there $\frac{N(N-1)}{2}$ possible pairwise agreements
  - $P(E)$ - Proportion of times that we would expect people to agree
    - Probability of both choosing the same category
    - Sum of squares of probabilities of each category

1.4 Overtraining

- Want classifier to generalise well
  - Recognise general characteristics that apply to unseen data
  - Ignore overly specific characteristics
- Overtraining can happen even if you use a separate test data set
- Overtraining is when you think you are making improvements but are in fact make it worse as it generalises less well to data other than your test data
  - Pick up accidental properties of the test data
  - May go unnoticed until deployed on real data
  - Safe if have large test data and use new test set each time
  - Can look for overly specific rules in classifier
- **Wayne Rooney Effect** - With time opinions change

1.5 Cross-Validation

- Can't afford to get new test data each time
- Use as much test data as possible
- N-Fold cross-validation
  - Split data randomly into N-Folds
  - For each fold X, all other folds as training data, test on fold X
  - Final performance is average of all folds
- Stratified cross-validation
  - Each fold mirrors the distribution of classes observed in the overall data
- Variance

$$\text{var} = \frac{1}{n}\sum_{i}^{n}(x_i - \mu)^2$$

  where x_i is the score of the i^{th} fold, and μ is the average score
- Leave one out cross-validation
  - Train the model on all but one piece of data, use the remaining piece to test
  - Repeat for all pieces of data, taking an average

2 Hidden Markov Models

- Model sequential problems
- Uses
  - Speech Recognition
    - Goal: Determining the words spoken
    - States: Words
    - Observations: The acoustics signals
  - Parts of speech tagging
    - Goal: Determine the parts of speech for text
    - States: Parts of speech
    - Observations: Words
  - Protien Analysis
    - Goal: Find which sections of proteins are in cell membranes
    - States: Zones relating to cells
    - Observations: Amino acids
- States are hidden and transitions occur probabilistically
- States produce an emission probabilistically, these are observations
- Can only see observations
- Must infer sequence of states that correspond to a sequence of observations
- Has a set of N emitting states $S_e = \{s_1, \ldots, s_n\}$, a start state s_0, and a finish state s_f

- Has a set of the output alphabet $k = \{k_1, \ldots, k_m\}$
- A sequence of observations is written as $O = o_1, \ldots, o_T$
- A sequence of states is written as $X = X_1, \ldots, X_T$
- A hidden Markov model is defined by its matrix parameters A and B, $\mu = (A, B)$
  - A is the state transition probability matrix
    - a_{ij} is the probability of moving from state s_i to s_j

      $$a_{ij} = P(X_t = s_j | X_{t-1} = s_i)$$

    - In each time t must transition for a state (can be the same state) and therefore

      $$\forall i \left(\sum_{j=1}^{N} a_{ij} = 1 \right)$$

    - Transitions into s_0 and out of s_f are undefined
  - B is the emission probability matrix
    - $b_i(k_j)$ is the probability of emitting k_i from state s_i

      $$b_i(k_j) = P(O_t = k_j | X_t = s_i)$$

    - At a time t must produce an emission

      $$\forall i \left(\sum_{j=1}^{N} b_i(k_j) = 1 \right)$$

    - s_0 and s_f are not associated with an observation
- The Markov Assumptions
  1. **Output Independence** - Each observation only depends on the current state, not the history

     $$P(O_t | X_1, \ldots, X_t, \ldots, X_T, O_1, \ldots, O_t, \ldots, OT) = P(O_t | X_t)$$

  2. **Limited Horizon** - Transitions depend only on the current state

     $$P(X_t | X_1, \ldots, X_{t-1}) = P(X_t | X_{t-1})$$

     - This is a first order hidden Markov model
     - Generally, transitions in an n^{th} order HMM depend on the last n states
- Hidden Markov Model tasks
  1. Labelled Learning
     - Given parallel observation and state sequences, O and X learn the HMM parameters A and B

     $$a_{ij} = P(X_{t+1} = s_j | X_t = s_i) \sim \frac{\text{count}(X_t = s_i, X_{t-1} = s_j)}{\text{count}(X_t = s_i)}$$

     $$b_i(k_j) = P(O_t = k_j | X_t = s_i) \sim \frac{\text{count}(O_t = k_j, X_t = s_i)}{\text{count}(X_t = s_i)}$$

  2. Unlabelled Learning
     - Given an observation sequence O and the set S_e learn the HMM parameters A and B

3. Likelihood
   - Given an HMM $\mu = (A, B)$ and an observation sequence O determine the likelihood $P(O|\mu)$
4. Decoding
   - Given an observation sequence O and HMM μ discover the best hidden states sequence X
   - Use the Viterbi algorithm
   - The optimal state sequence, $\hat{X}$ is found by

$$\hat{X} = \underset{X_0,\ldots,X_{T+1}}{\arg\max}\ P(X|O,\mu) = \underset{X_0,\ldots,X_{T+1}}{\arg\max} \prod_{t=0}^{T+1} P(O_t|X_t)P(X_t|X_{t-1})$$

2.0.1 Viterbi Algorithm

- Use a trellis, a look up table for memoization, $\delta_i(t)$
- $\delta_i(t)$ is the probability of the most probable state sequence of length t ending at the state s_j

$$\begin{aligned}\delta_i(t) &= \max_{X_0,\ldots,X_{t-1}} P(X_0,\ldots,X_{t-1},o_1,\ldots,o_t,X_t = s_j|\mu) \\ &= \max_{1\le i\le N} \delta_i(t-1)\cdot a_{ij}\cdot b_j(o_t)\end{aligned}$$

- Use the following recursive relationship

$$\begin{aligned}\delta_j(1) &= a_{0j}b_i(o_1) \\ \delta_j(t) &= \max_{1\le i\le N} \delta_i(t-1)\cdot a_{ij}\cdot b_j(o_t) \\ \delta_f(T+1) &= \max_{1\le i\le N} \delta_i(T)\cdot a_{ij}\end{aligned}$$

- Use a helper variable $\psi_j(t)$, this holds the $(t-1)^{\text{th}}$ state in the most probable path
- Use the following recursive relationship

$$\begin{aligned}\psi_0(0) &= \text{undefined} \\ \psi_j(t) &= \underset{1\le i\le N}{\arg\max}\, \delta_i(t-1)\cdot a_{ij}\cdot b_j(o_t) \\ \psi_f(T+1) &= \underset{1\le i\le N}{\arg\max}\, \delta_i(T)\cdot a_{ij}\end{aligned}$$

- $\psi_f(T+1)$ records the last state in the optimal state sequence, follow the ψ to find the optimal state sequence

2.1 Recall and Precision

- When you have interesting and non-interesting categories
- Consider the following

	C_1	C_2	Total
C_1	a	b	$a+b$
C_2	c	d	$c+d$
Total	$a+c$	$b+d$	$a+b+c+d$

- Precision of C_2:

$$P_{C_2} = \frac{d}{b+d} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

- Recall of C_2:

$$R_{C_2} = \frac{d}{c+d} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

- F-Measure of C_2:

$$R_{C_2} = \frac{2P_{C_2}R_{C_2}}{P_{C_2} + R_{C_2}}$$

- Accuracy:

$$\text{ACC} = \frac{a+d}{a+b+c+d} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total}}$$

3 Social Networks

- Graphs - Unweighted and undirected
- Have two basic properties
  1. Homophily - principle that we connect to others who are like ourselves
  2. Weak ties - links to acquaintances that connect us to parts of the network that would otherwise be distant
- Degree - Number of neighbours a node has
- Distance - The length of the shortest path between two nodes
- Diameter - Maximum distance between any pair of nodes
- Small World Phenomenon - Natural graphs have low average distances
- Natural networks have closely clustered regions connected only by a few links
- Giant Component - A connected component containing most of the nodes in the graph
- Weak and Strong ties depend on the closeness of a link (e.g. the number of times two researchers have collaborated)
- Bridge - An edge connecting two otherwise unconnected components
- Local Bridge - Edge joining two nodes that have no other shared neighbours
- Triadic Closure - If A knows B and C then it is relatively likely that B and C will know each other, a result of homophily
- Ego Network - A network of social connections focusing on one person

3.1 Generating Random Networks

3.1.1 Erdos-Renyi Model

$$G_{n,p} = (V, E)$$

Where

- n is the number of vertices $|V|$
- p is the probability of forming an edge $(u, v) \in E$

3.1.2 Watts-Strogatz Model

$$G_{n,k,p} = (V,\, E)$$

Where

- n is the number of vertices $|V|$
- k is the initial degree of each node (even integer)
- p is the probability of rewiring
- Start with a ring of nodes, each connected to its k nearest neighbours
- Each edge is visited and probabilistically replaced with a new edge
- No duplicate edges

$$n \gg k \gg ln(n) \gg 1$$

- $k \gg ln(n)$ ensures that graph is connected

3.2 Betweenness Centrality

- $G = (V,\, E)$ is a directed graph
- Betweenness centrality half for an undirected graph
- $\sigma(s,\, t)$ number of shortest paths between s and t

$$s = t \iff \sigma(s,\, t) = 1$$

- Use the following recursive relationship

$$\sigma(s,t) = \sum_{u \in \text{pred}(t)} \sigma(s,u)$$

where

$$\text{pred}(t) = \{u \,|\, (u,t) \in E \wedge d(s,t) = d(s,u) + 1\}$$

- $\sigma(s,\, t|v)$ number of shortest paths between s and t that pass through v

$$v \in \{s,t\} \implies \sigma(s,\, t|v) = 0$$

- $C_B(v)$ the betweenness centrality of v

$$C_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

- Define δ

$$\delta(s,t|v) = \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

- Define

$$\delta(s|v) = \sum_{t \in V} \delta(s,t|v)$$

further

$$\delta(s|v) = \sum_{\substack{(v,w) \in E \\ w\,:\, d(s,w) = d(s,v)+1}} \frac{\sigma(s,v)}{\sigma(s,w)} \cdot \big(1 + \delta(s|w)\big)$$

and

$$C_B(v) = \sum_{s \in V} \delta(s|v)$$

3.2.1 The Algorithm

1. Start at source and calculate $\sigma(s,t)$ for all vertices
2. Back track and calculate $\delta(s|v)$ starting at furthest node, use following formula

$$\delta(s|v) = \sum_{\substack{(v,w)\in E \\ w\,:\,d(s,w)=d(s,v)+1}} \frac{\sigma(s,v)}{\sigma(s,w)} \cdot \big(1+\delta(s|w)\big)$$

3. Repeat 1 and 2 for all $s \in V$, sum each $\sigma(s|v)$ to get betweenness centrality

3.3 Edge Centrality

- $\sigma(s,t|v)$ now $\sigma(s,t|e)$, the number of shortest paths between s and t going through e
- Calculate $\delta(v)$ as before and $C_B(v,w)$, equivalent to $\delta(v)$

3.4 Clustering - Newman-Girvan Method

- Clustering - Automatically group data according to notion of closeness or similarity
- Agglomerative Clustering work bottom-up
- Divisive Clustering work top-down, by splitting
- Newman-Girvan Method is divisive

3.4.1 Find connected components

1. Depth first search start at arbitrary node, mark all visited nodes
2. Repeat with all unvisited nodes

3.4.2 Algorithm

```
while number of connected sub graphs < specified number of clusters:
    Calculate the edge centrality
    Remove edge(s) with the highest betweenness
        In case of tie either choose at random or remove all
    Recalculate number of connected components
```

3.4.3 Evaluating Clustering

- Purity
  - Assign each cluster a label based on the majority class
  - Count correct assignments and divide by total assignments
- Best is extrinsic - Use system for a task and evaluate that task