Contents:

1. COMP30027 2018 Exam
2. COMP30027 2017 Exam

[Feel free to start typing answers! If an answer is already typed, and you'd like to respond to it / add something else, you can type with a different colour]

Please try to avoid extremely bright colors like #FF0000, #00FF00, #0000FF, #FFFF00, #FF00FF, #00FFFF (on the second row) to write or highlight long text. They look really horrible if you stare at them long enough. Colors on the ***first row*** *and* ***third row onwards*** *are recommended.*

COMP30027 2018 Exam

Good luck for the ML exam tomorrow :)

- **Don't forget what an Oblivious Tree[1] is (it's a joke)**
- **Don't write paragraphs, remember your marker is a human and is looking for keywords (or the lack of keywords) in an answer**
- **Try to be as brief as possible, but cover the important stuff within a sentence**
- **You can answer in dot point**
- **Tim has also confirmed that you can show calculations for a few steps and explain the rest (as he did with k-means), most likely if you're doing a page worth of calculations its wrong**
- **Contingency tables may get you most marks if you screw up calculations for NB problems (and it's helpful to show in case you have arithmetic error)**
- **Arithmetic errors won't be penalized heavily as long as you show working out!**
- **As for section D, aim to walk the marker through your steps/procedures/assumptions without side-tracking. Be creative with your answers and it has been confirmed that you may use classifiers outside the scope of this subject so long as you justify and explain it**

https://app.lms.unimelb.edu.au/bbcswebdav/pid-6923739-dt-content-rid-60490928_2/courses/COMP30027_2019_SM1/lectures/exam2018.pdf

- Blue = Isabel's responses (most likely terrible)
- Red = Akira's responses (answers should be similar to the marking criteria style since Tim wrote the exam this year)
- Jin's alternative responses
- Yong See's addenda
- Anonymous alternative responses
- 🅃 Tim's answers (or equivalents in concept) in revision lecture Thu 20 June

[1] "I think Akira is joking about needing to remember Oblivious Tree, Random Tree was mentioned though (base of a Random Forest). If you're interested, an Oblivious Tree is like where at each level the attribute to split on for that level has to be the same" — Hannah Jean Perry (yeh it was a joke but inb4 hahaha)

Section A:

1:

Supervised learning trains on a set of labelled instances (eg. naïve Bayes), unsupervised is given no labelled data (eg. *k*-means)

- Supervised: Uses data with labelled instances to train model (e.g. Decision Tree)
- Unsupervised: Model finds its own internal groupings of instances without labels (e.g. *k*-means)
- Defining an unsupervised model as a model that “finds its own internal groupings” might be problematic - it’s safer to have a broader definition (e.g. that also encompasses association learning tasks). I’d answer something like learning patterns from unlabelled data e.g. k-means clustering.

2:

An instance is each individual example, attribute are the values within the instance representing some information about the instance

- Instances contain features
- T Instances are an example of a concept, and attributes measure aspects of an instance

3:

Assign random value to each category

- One-Hot Encoding to binarize / vectorize nominal data
- T Multiple linear regression is an alternative (TIM)

4:

In soft k-means

- In a multi-class logistic regression model
- T Final layer in a neural network (TIM)
- Fully connected feedforward neural network with multiple class prediction
    - More specifically, when we want the output for a NN to be between 0 and 1 (eg. class probabilities)

5:

Evaluation bias is when the evaluation metric under or overestimates the effectiveness of the classifier, while model bias is the tenancy for a model to make systematically wrong predictions. This is undesirable as we cant accurately know how our learner is doing/hard to improve it

- The model will not make the same errors, but the biased evaluation strategy will over/under estimate the true performance of the model
- Undesirable since we want to know the true performance (or at least have a good estimate) of the model so we can improve it
- T Model like Zero-R, undesirable since it is wrong a lot of the time (TIM)

6:

When activation function is sigmoid function ($\frac{1}{1+e^{-x}}$)

- When a perceptron (single-neuron NN) uses a sigmoid activation function
- Specifically, an NN with no hidden layers, one output layer and uses sigmoid as activation function

7:

Both semi-supervised, but one tries to label the unlabelled instances "itself", the other asks a human/oracle

- Similarity: Both are semi-supervised and are forms of incremental learning
- DIfference: Active will query unlabelled instances for labelling to an oracle whereas a human gives train instances in self learning (T relying on model's evaluation - TIM)
- Difference: active learning augments its labelling data by querying an oracle on unlabelled data; whereas self training augments its labelled data by taking the most confident predictions on unlabelled data

8:

Logistic regression - one class is chosen as "1" or positive class, the other chosen as "0" or negative class. If more than 2 classes, can do one-vs all and repeat over each category,

- A linear model can be used in classification contexts by using Logistic Regression.
- We assume the pivot initialization is irrelevant (comparisons with an irrelevant pivot does not affect performance)
- T Multi-response Linear Regression using a one vs rest ensemble that have normalized outputs so they are comparable (TIM)
- Assuming that predictors are continuous and that this is binary classification. Map positive class to +1, negative class to -1. Perform linear regression, and map positive predictions to the positive class, negative predictions to the negative class. If there are more than two classes, perform one vs rest classification and select the class that had the highest (regression) prediction value as the class prediction.
- The specific transformation we want is a sigmoid function such that the transformed regression output tends towards 0 or 1

9:

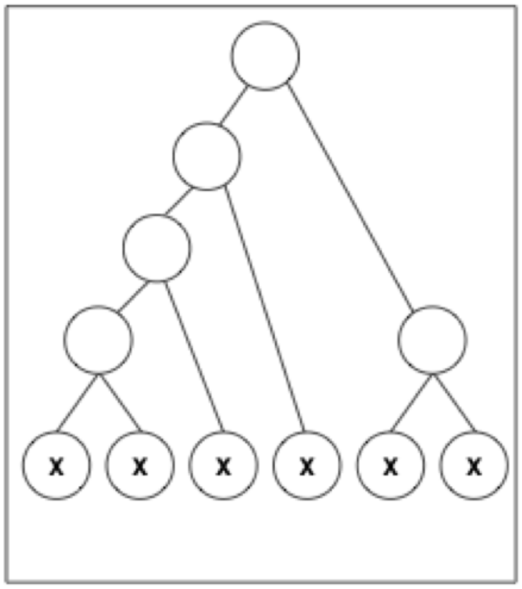
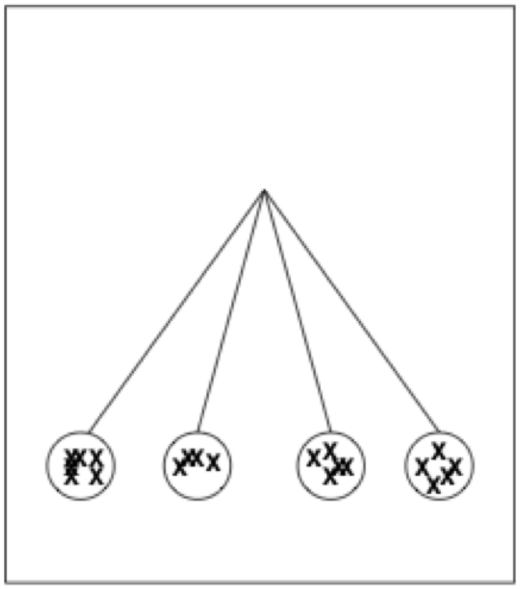
		True	
		Y	N
Predicted	Y	5	20
	N	15	60

- I had (with respect to Y as positive class): TP=5, FP=20, FN=15
- Prec = 5/(5+20) = 1/5, Rec = 5/(5+15) = 1/4
- Hence F1-score = $\frac{2\times\frac{1}{5}\times\frac{1}{4}}{\frac{1}{5}+\frac{1}{4}} = \frac{2}{9}$ (T verified by Tim, only need F1-Score on the exam)

10:

- HMM is good for text classification problems, but can be extended to speech recognition / image recognition
- T Weather and NLP problems are examples of where HMMs are used (Tim)
- Requires a sequence
- Assumes that there are hidden states, each corresponding to a probability distribution of possible observations
- Assumes current (hidden) state only depends on its immediate previous state
- Also assumes that the emission/output probability only depends on its current state
- By Tim's answer I think he actually mean the assumption here is each instance is not independent with each other(for example weather, if today's weather is rainy, then it is very likely that tomorrow's weather is also rainy. So the assumption here, is, there is some relationship between instances)
- Akira's (red) answer captures exactly that - "only depends on its immediate previous state".

11:

-diagram from lectures
-??

- Diagram from lectures (hierarchical looks like a dendrogram)
- Partitional clustering since it computes likelihoods (probabilistic distribution)
- T EM is a soft partitioning cluster (Tim)
- Fairly sure EM is an example of partitional clustering - cf. [EM wikipedia article]
- Hierarchical - nested clusters represented as a tree; partitional - clusters have no hierarchical difference (no sub-structures), but treated at the same level
- EM produces partitional clustering (since there is no hierarchy between the produced clusters)

Section B:

2:

1.
   Measures unevenness of a probability distribution - low entropy means highly uneven, high means close to uniform. Decision tree we want entropy to decrease down the tree - therefore use highest info gain to choose new attribute for next node
    - Entropy measures the amount of bias in the probability distribution where low entropy means there is more information to be gained
    - Used to build a DT by acting as a split criterion to partition train instances
    - T A good attribute is one which reduces entropy and increases Info Gain (Tim)

- The entropy of a node measures the impurity of classes of that node. The lower the entropy, the more homogeneous the classes of that node.
- The splitting attribute at a node such that the entropy of its child nodes are minimised.
- A common way of doing this is by using maximising information gain, which measures the expected reduction in entropy.

2.

...look along tree, until reaching a leaf that'll tell you the class..?

- Traverse down the nodes of the DT until we reach the bottom of the tree (usually a pure leaf)
- If leaf is not pure (has more than 1 possible label), take the majority vote

3.

- Refer to Yong See's answer
- T NB assumes independence, DT assumes dependence of attributes
- Naïve Bayes / nearest prototypes classify instances based on a series of numeric values
- Decision trees classify instances by using a set of rules instead
- NP doesn't quite use the numeric values of each instance though (it aggregates instances of the same class), but perhaps that answer is okay anyway.

- Here's an alternative: T NB/NP consider all relevant features simultaneously during prediction, to predict the most probable class; DT considers features sequentially (one by one via traversal) before reaching a final prediction

4.

a.

Random forest is an ensemble of random trees - a decision tree but at each node only some possible attributes contribute - each tree built using a different bagged dataset

- RF: An ensemble of Random Trees utilising bagging

b.

Less prone to overfitting??

- Robust to overfitting since it reduces model variance without introducing combined model bias

3:

1.

Linear regression

- Used in Linear Regression
- And also Logistic Regression (LR also uses Gradient Ascent I believe)(could be Descent, if put a - sign in front of the “cost function”, inverse the log curve)

2.

Want to minimise the sum of squared errors, gradient descent searches for minimum

- Used to minimise a loss or error function until it converges to a minimum (global if it is convex)
- In linear regression, we wish to minimise the linear estimator b for β since $SSRes = (y - Xb)^T(y - Xb)$ is in terms of b
- *I didn't mean it in the context of s^2 is dependent on b*
- No more LSM[2] pls i’m so done
- 🆃 When the cost function is differentiable
- Residual Sum of Squares (RSS) = $\sum_i (y_i - \hat{y}_i)^2$

3.

Too small - get ‘stuck’ on loc min, too large - might overshoot it/diverge

- Small: May never converge due to time complexity
- Large: Overshoot the global minimum

4:

1.

Support vectors are the vectors of the train instances closest to the decision boundary (), SVM finds a hyperplane separating the classes such that the distance from the hyperplane to the support vectors (i.e. margin) is maximised

- Support vectors are the points close to the margin
- They define the constraints of the hyperplane
- The hyperplane splits the data (assume binary class) into pos / neg, which is then used to classify test instances by seeing which side of the hyperplane they lie in

- Support vectors act as data points on the boundary of the classes
- The margin (between two classes) is defined as the scalar projection of the difference between support vectors (of different classes), where the projection is in the direction of the normal of the separating hyperplane; SVM maximises such a margin

[2] Linear Statistical Models, in case any of you don’t know.

- Predictions are made depending on which side of the hyperplane a test instance lies (same label as that of the corresponding support vector

2.

- Same predictions when (SVM) point lies on the hyperplane and (NP) the point lie in the middle of two prototypes
- (Assuming binary classification) Same predictions when the bisecting hyperplane of the two prototypes is sufficiently close to the separating hyperplane from SVM (think of ‘bisecting hyperplane’ as the n-dimensional analogue of a ‘perpendicular bisector’ in 2-D - it is the decision boundary of NP).
- This is because all points on one side of such a hyperplane will then be closer to the corresponding prototype than the other prototype.
- T SVM is superior since it can work with non-linear (i.e. not linearly separable) data by transforming it through a kernel function

3.

- SVM can be equivalent to LR if the cost function C (hyperparameter) is adjusted to become a “hard” margin (i.e. made larger)
- “Hard” margins are margins that avoid having points on the wrong side of the line, which is what a LR classifier does
- A mathematical justification of the above (trivial proof): I’m going to write w.x+b from SVM as just w.x, and the b.x from LR as w.x (for the sake of consistency, they’re the same idea). The slack variable in SVM can be written as max(0,1-y*w.x). Then, squeeze enough of your brain juices until you see that the summands in the log-likelihood of LR can be each rewritten as -log(1+exp(-y*w.x)), where the possible values of y are now -1 and 1, instead of 0 and 1 (again, to be consistent with SVM). From now on, let’s minimise the sum of log(1+exp(-y*w.x)) for LR. In SVM, C is multiplied to the sum of slack variables. If we set C->infinity, then we can ignore the |w|^2/2 term (c.f. SVM lecture slide 30). Both models now minimise a sum, so let’s look at the summands, i.e. the functions max(0,1-z) and log(1+exp(-z)). As z -> +infinity, both of these functions are ~0; as z -> -infinity, both of these are ~-z (You can see what I mean here). This means that the summands of either model are similar if y*w.x is large. As C->infinity, |w| would tend to be larger (again, SVM lecture slide 30), the preceding asymptotic equivalences occur, which means that the summands of either model are similar, which means they are approximately equivalent.
- P.S. I do not think this is close to exam expectations at all, but I couldn’t think of any intuitive reason to why Akira’s answer works…
- They are both linear classifiers trying to find the correct $\boldsymbol{w}$ and $\boldsymbol{b}$ (separating line), and also the regularisation hyperparameter in logistic regression is related to the margin. SVM also should be soft (the C hyperparameter).
    i. The kernel in SVM should be linear to optimise w and b similar to logistic regression.

    ii. The logistic regression model should use L2 regularisation equivalent to maximising margin in SVM.
    iii. The SVM model should be soft to allow for points to be in the wrong side similar to logistic regression.
- [Ref: Ash Rahimi]

Section C:

5.

1.

- Use one hot encoding to get the following new dataset (confirmed that this method is valid with Afshin). Also note that we cannot assume the data is ordinal (i.e L < M < H).

ID	**L**	**M**	**H**	**ibu**	**Label**
A	1	0	0	1	ale
B	1	0	0	.1	ale
C	0	1	0	.15	4ale
D	0	1	0	.45	stout
E	0	0	1	.3	stout
F	0	0	1	.45	stout
G	0	1	0	.11	stout

- Hence our test instance is T = (M,0.8) -> (0,1,0,0.8)
- Using Manhattan distance we find:
  d(T,A) = 1 + 1 + 0 + 0.2 = 2.2
  (the rest is trivial and you can just calculate it out)
- Using 1-NN we therefore classify as "stout"
- SIDE NOTE: ~~Same distance metric is to be used throughout. Mix of Hamming distance and Manhattan/Euclidean distance is *not* allowed. [Ref: Afshin]~~[3]
- T Tim said different distance metrics can be mixed.

2.

Stout

[3] Let Tim override Afshin for now XD

- ILD is given as $\frac{d_k - d_j}{d_k - d_1}$ where $k = 5$, $d_1 = 0.35$, $d_5 = 2.35$
- Calculate it for the 5 closest neighbours *(trivial calculations so you can calculate it)*
- Weighting for “`ale`” = 0.925, “`stout`” = 1.83
- Hence we predict using ILD 5-NN we classify as “stout”

6.

1.

   1 - High
   2 - Low
   3 - Low
   4 - med
   5 - Low
   6 - Med
   7 - Low

   - T Range is 1 - 0.1 = 0.9 (by 3 bins) so each bin is of size 0.3
   - Bin1 = [0.10, 0.11, 0.15, 0.30]
   - Bin2 = [0.45, 0.45]
   - Bin3 = [1.00]

2.

   1 - High
   2 - Low
   3 - Low
   4 - med
   5 - med
   6 - Med
   7 - Low

   - Use Jin’s answer below for full working out
   - T d(1, 1) > d(1, 0.3) > d(1, 0.45) hence assign to cluster 1
   - Iteration 0: 0.10; 0.30; 0.45
   - Iteration 1: 0.12; 0.30; 0.633...
   - Iteration 2: 0.12; 0.40; 1.00
   - Iteration 3: [CONVERGED]
   - Bucket 0 [0.12]: {0.10, 0.11, 0.15}
   - Bucket 1 [0.40]: {0.30, 0.45, 0.45}
   - Bucket 2 [1.00]: {1.00}

7.

1.

   Ale: prob = 3/7 *⅓ * 0 = 0
   Stout: prob = 4/7 *2/4 * 0 = 0

8.

Equal, could be either

- Build a contingency table

ebc	L	M	H	
ale	2	1	0	3
stout	0	2	2	4
	2	3	2	7

ibu	0.10	0.11	0.15	0.30	0.45	1.00	
ale	1	0	1	0	0	1	3
stout	0	1	0	1	2	0	4
	1	1	1	1	2	1	7

- Ale class: $3/7 \times 1/3 \times 0 = 0$
- Stout Class: $4/7 \times 2/4 \times 0 = 0$
- Assume we use the smallest value as tie-breaker (aka left tie-breaker) to classify as "ale"
- Alt.: Using most common class for tie breaking, class predicted: `stout`

2.

(used discretization from 6.2)x

Ale: prob = 3/7 *2/6 * 2/6 = 0.05

Stout: prob = 4/7 *3/7 * 1/7 = 0.035

Therefore ale

- We build another contingency table (it's optional to do so but it allows for consequentials if you screw the arithmetic up) using equal-width discretisation

ibu	bin1	bin2	bin3	
ale	2	0	1	3
stout	2	2	0	4
	4	2	1	7

- Ale Class: $(3/7) \times (1 + 1 \,/\, 3 + 3) \times (1 + 1 \,/\, 3 + 3) = 0.047619$
- Stout Class: $(4/7) \times (1 + 2 \,/\, 3 + 4) \times (1 + 0 \,/\, 3 + 4\,) = 0.0349854$
- Hence we predict as "ale"

8.

1.

- Refer to Q7.1 for contingency table

2.

- Simple calculation of entropy gives H(ebc) = 1.078922, H(ibu) = 1.747861
- T We prefer “ebc” since it offers lower entropy
- This can be extended to IGR, chi-square, and AIC -> they usually prefer features with distinct grouping and by inspection. We can see that in “ebc” having “L” means “ale”, and having “H” means “stout”, whereas this is not so obvious in “ibu” due to continuous values.
- *Using k-means discretisation from 6.2. (in black)*
- T Mutual information: $\sum_{i \in \{v_1, v_2, v_3\}} \sum_{c \in \{ale,\ stout\}} P(i,\ c) \log_2 \frac{P(i.\ c)}{P(i)P(c)}$
    - **ebc**: ~~1.5295~~ 0.59167 [Workings]
    - ibu: 0.5190
- T χ^2 : $\sum_{i \in \{v_1, v_2, v_3\}} \sum_{c \in \{ale,\ stout\}} \frac{(O_{ic} - E_{ic})^2}{E_{ic}}$
    - $E_{ic} = \frac{7}{6}$
    - ebc: 4.14
    - **ibu**: 5.86

Using k-means discretisation, I got equal MI of **0.59** for both features; equal Chi2 of 4.28 for both features. Using equal-width discretisation, ibu gets an MI of 0.4138 and Chi2 of 2.92.

Section D:

- Contact me if you wish to discuss this (just message me on fb - Akira Wang)

-What do people think for dot point 1? HMM?
-Dot point 2? Confusion matrix? Precision? Recall? Accuracy?
-dot point 3 - active learning?

Discussion on this page would be appreciated

Some thoughts:

1. Not sure from the question, but if the embeddings are simple and don’t have any feature extraction then we could use a 1-D CNN? If the embeddings are already feature rich and properly preprocessed then perhaps we could use a standard feed-forward NN. Output would be three nodes, each with a softmax function indicating probabilities for one of the three classes. Exact architecture would have to be determined from experimentation, and I guess it also relies on more specific information that isn’t available in the question.
2. Since we have relatively few labelled samples, and thus a relatively small set of instances to evaluate our model with, it might make sense to run cross validation with a partition size of 1. This would maximise the amount of data that we can use for training, and provide the most accurate evaluation of how the model would perform in a practical

setting. Other evaluation methods like holdout/repeated subsampling might not provide enough data for both training and testing, and thus have really high evaluation variance.

3. The obvious answer would be to make the interns go crazy labelling as much data as possible, but perhaps we could save them some stress by using an active learning strategy. This could potentially also make the model require less data for training. I'm sure they would make good oracles.

Although I'm not too sure about the first point - not much justification for specifically using NNs over other models like logistic regression other than the fact that they seem to be widely favoured in both industry and academia rn. I suppose they can model 'deeper' patterns?

Some ideas, feel free to give feedback:

- Note on point 1 from above: Each message gets an embedding already, so we probably are not going to do any processing on the word/token level (e.g. NLP often uses CNN on a word/token level, but that's not applicable here, since we did not get word embeddings).
    - True, the point I was making was that the question doesn't really specify what the 'embedding' actually is - could be a simple count vectorizer, or could be something more complex in the style of word2vec.
    - My thought: like he said, since the question did not say word vector, but some real valued densen vector. I think we could just treat it as some real valued vector and use methods like logistic regression. I think using complex algorithm like neural network or HMM is not appreciated here(since it is in the exam condition, but of course if you are really good at those algorithm, then you should use them).

- A simple idea would be to use a cosine similarity based k-NN. Cosine similarity because the vector magnitude may depend on the length of the message, so metrics similar Euclidean-based distances may not work well, especially on high-dimensional data. The use of efficient spatial data structures (k-D tree etc.) can help us incrementally add labelled instances efficiently. This method works on the assumption that messages of different sentiments have their corresponding embedding vectors pointing in different directions.
- Typical evaluation metrics such as precision/recall/F1 can be used, however some specific cases may be more important to look at, e.g. the recall of negative messages, or perhaps calculate weighted precision, where more recent messages are weighted more heavily. A key challenge to note is the relatively small amount of labelled data. We can use cross validation here, initially with more partitions since there is little training data (such that a larger proportion of data is used for training each time), but later with less partitions when the training data is large enough that the computational time becomes intractable with too many partitions.
- Active learning can be done here, where a batch of queries can be made to the interns (oracles) each day, for them to label the sentiment of unlabelled instances. We could choose those instances that have the highest entropy of the classes of their k nearest

neighbours to be queried, but this could result in certain regions of the data not being labelled (skewed distribution of labelled instances). We need to find unlabelled instances that are difficult to classify (far away from labelled instances), yet are close to other unlabelled instances (so that it is useful for labelling more data). For each instance i, define D1(i) as the distance from instance i to the closest labelled instance, and define D2(i) as the distance from instance i to the closest unlabelled instance. We want to select instances with large D1(i) and small D2(i) to be queried, so D1(i)/D2(i) is a good heuristic for query selection: unlabelled instances with the highest D1(i)/D2(i) can be selected to be queried. One weakness is that the same batch may then contain many similar unlabelled instances, which intuitively seems wasteful. Here's a fix: Suppose the next batch B is partially constructed, and let instance i' be the next instance with the highest D1(i')/D2(i') heuristic (that has neither been accepted to nor rejected from B). We accept i' to batch B only if for every instance i in B, the distance between i and i' is larger than D1(i). This ensures that queries of the same batch will have sufficient distance away from each other.

- I'm probably just having trouble modelling this in my head, but would this fix fully solve the problem? Seems like it might just prevent some of the most immediately obvious redundancies, where we query both instead of one of the instances that are closest to one another (smallest D2(i)). This doesn't necessarily mean that our batch won't contain instances that are still generally close together. Here's a vague idea: how about we cluster the instances first, and then only pick batch instances from separate clusters?
- I made a typo, changed it to D1(i) instead (so that the 'sphere' is bigger - should make more sense since we don't know where the next nearest labelled instance is outside the 'sphere'). I think clustering could be a better idea, you can even make number of clusters equal to the batch size. Thanks!

COMP30027 2017 Exam

https://app.lms.unimelb.edu.au/bbcswebdav/pid-6923739-dt-content-rid-61552665_2/courses/COMP30027_2019_SM1/lectures/exam2017.pdf

Solutions already available at
https://app.lms.unimelb.edu.au/bbcswebdav/pid-6923739-dt-content-rid-61552666_2/courses/COMP30027_2019_SM1/lectures/solutions2017.pdf

Questions not covered by our cohort

A: 1. 3 [PARTIAL ANSWER], 7(e)

B: 2. [PARTIAL ANSWER]

3. 3 (?)

4. 3 [PARTIAL ANSWER]

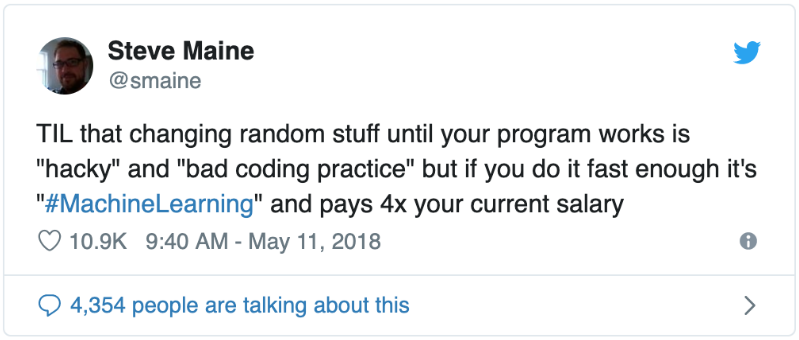
Erratum

~~Minus sign in the paper is missing.~~
[Update: seems like the latest version has this issue fixed now.]

~~C.6.3: should be~~ $GINI = 1 - \sum_{j \in C} [p(j)]^2$

~~C.7: All numbers in `opacity` column is negative; Similarly,~~
~~C.7.2: The test instance should be {`abv` = 5.1, `opacity` = −0.23}~~

Bonus section


Steve Maine
@smaine

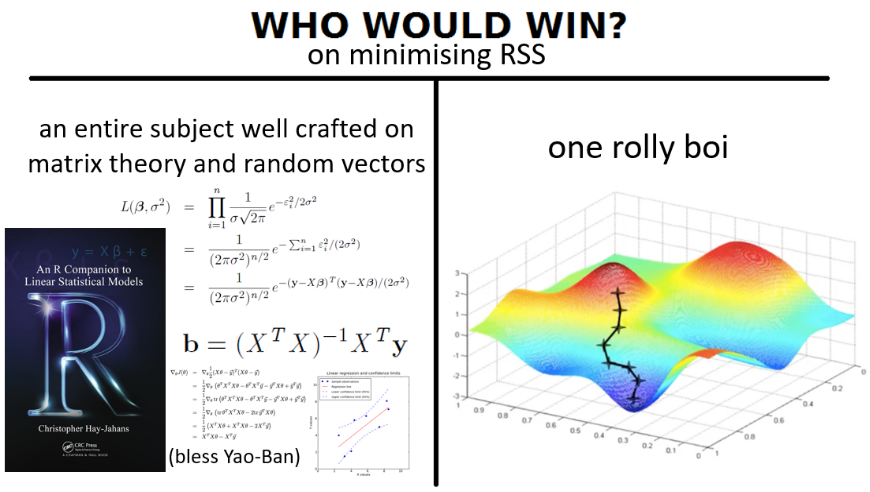
TIL that changing random stuff until your program works is "hacky" and "bad coding practice" but if you do it fast enough it's "#MachineLearning" and pays 4x your current salary

10.9K 9:40 AM - May 11, 2018

4,354 people are talking about this

https://twitter.com/smaine/status/994723834434502658?lang=en

For the LSM veterans



WHO WOULD WIN?
on minimising RSS
an entire subject well crafted on matrix theory and random vectors
$b = (X^T X)^{-1} X^T y$
(bless Yao-Ban)
An R Companion to Linear Statistical Models
Christopher Hay-Jahans
one rolly boi

- Don't write paragraphs because its a ~~cbs~~ reading it if you're a marker
- Building a contingency table is the safest bet for NB if you have the time too (might even get full marks for method even if you get an arithmetic error)
- Perceptron activation function is 1 if $w \cdot x_i + b \geq 0$ else 0
- Negative sampling (`word2vec`) selects words from the corpus on the assumption that they are not going to "fit" the original context of occurrence
- Cluster evaluation
    - **Unsupervised**: High cohesion and high separation.
    - **Supervised**: Purity and Entropy of clusters.
- The soft k-means β parameter determines the stiffness
- Reading a whole chunk of text in #FF0000 for a long time is giving me eye fatigue. Try something more mild next time, like #CC0000.