# CS 345 Fall 2021 Semester Review

The following is not an exhaustive list of what has been covered, and to be very explicit. That said, hopefully the following will be a helpful as you review the broad topics and techniques covered so in CS 345

1. While in principle the Machine Learning (ML) Foundations and Practice course might not need to require skill with the Python language, the reality is success today in ML demands skill with at least one language used to support ML today, and put simply, Python is that language. Consequently, you are comfortable with Python in general and the following specific concepts: dynamic typing (asking the type of something), strings, lists, list comprehensions, boolean expressions, defining functions and of course None.

2. Compiled only languages as opposed to read-eval-print style interpreted languages seldom let you ask an object, say a function, to describe itself. However, Python supports and encourages such behaviors and you fully understand how to take advantage of this ability both in exploring functions written by others and when writing new functions yourself.

3. This class makes extensive use of Jupyter Notebooks as the delivery mechanism for the material covered in the course. One aspect of Jupyter Notebooks is advanced support – through LaTeX syntax – that enables the presentation of well formatted mathematics. Upon finishing CS 345 you will **not** be expected to master details of how to format mathematics in LaTeX. However, you at least know it is available to you, and know enough to match up simple examples of the markup language input with the final displayed output such as already demonstrated in early course notebooks.

4. Quoting from the NumPy homepage: "NumPy is the fundamental package for scientific computing in Python". No one can possibly begin to approach ML in Python without a solid understanding of NumPy and an entire CS345 notebook is devoted just to introducing elements of NumPy which are then relied upon throughout the semester. Just a few key essential you now have mastered include: arrays, array indexing and slicing, boolean indexing and element wise functions.

5. A very powerful aspect of NumPy is the ability to carry out operations on arrays such as adding two arrays together or multiplying an array by a scalar. These concise syntactic constructs are now familiar to you and hence you are comfortable reading them, writing them, and most important understanding what they accomplish.

6. T. S. Eliot in the *Ad-Dressing of Cats* made clear "A CAT IS NOT A DOG", and with apologies to T. S. Eliot, "AN ARRAY IS NOT A LIST". Python lists and NumPy arrays each have their uses and value – they even play well together when initializing and array from a list. However, you've now done enough ML programming to clearly understand the difference between them and avoid the pitfalls of mistaking one for another.

7. About NumPy slices, you now understand the metaphor that a slice becomes another view into the SAME data. You are comfortable with this idea and also generating/understanding simple examples of the behavior.

8. Concise expression of frequent operations is one of the true virtues of the Python/NumPy combination. As an example, the early notebook introducing NumPy illustrates a simple single line of code that computes the accuracy of a process expressed as a pair of arrays, one suggesting a true answer and one a predicted answer. Noteworthy is the absence of explicit iteration – a for loop – in this single line of code. This is precisely the type of Python you can now quickly write or read whenever a similar calculation is needed.

9. Interspersed throughout the CS345 notebooks are little examples of some interesting and basic concepts not necessarily at the heart of ML precisely, but nevertheless worth learning both as good examples of how to write concise python as well as simply because the concept is useful. One such early example is that of the Random Walk, and for the reasons just stated, you will want to understand how one simulates a Random Walk in Python.

10. The first real dataset used in CS345 used to introduce the notion of labeled data is called the Iris Dataset. There are many things to take away from this first introduction, but here are three that you now understand deeply from both that first introduction and subsequent presentations. What is a feature? What is a label (is it an integer or a string)? Can you as a person makes sense of a visual display of the problem posed by the dataset?

11. Consider this superficially simple question: "Is a 1D array the same as a vector?" Or, if you would rather: "Points and vectors are the same?" In the lecture on vectors and dot products CS345 introduces some key mathematical concepts that anyone doing ML should know. Just for example, you should feel comfortable with at least two different answers to the 1D array versus vector question. At one level, sure, they are the same in so much as often a 1D NumPy array is used to represent a

vector. Indeed, the method "dot" is defined for 1D arrays. However, in mathematical terms, a vector is simply a direction and a magnitude – not a data structure as in NumPy. Why does this matter? Because in representing directions vectors play a key role in such things as defining a linear decision boundary in space. What about "Points and vectors are the same thing?". Again, mathematically the correct answer is an emphatic no – points have location vectors direction and magnitude. However, in ML, often the distinction is blurred and a single 1D array may be interpreted as the vector from the origin out to the point or simply the coordinates of the point. Having just walked through this material, being comfortable with the definitions and distinctions raised here is essential to a basic understanding of ML.

12. Early in the semester an image of a cat and an image of a dog were used to drive home the fact that in ML we can change, in this case reshape, data. For example, images have two dimensions, e.g. rows and columns. However, an image can be unrolled (reshaped) to fit in a 1D array. Now it may be thought of as a point in high dimensional space – or perhaps a vector (see previous item about points and vectors). You should feel entirely comfortable with such operations along with how to interpret such actions. For example, images may be thought of as points in k-dimensional space, what is k for a 32 by 32 pixel grayscale image? Along the same lines, how does a nearest neighbor classifier compute the distance between two such images?

13. ML is about understanding data, and in parallel with the application of ML algorithms, is the need to explore the data, and that leads directly to visualization. Throughout CS345 visualizations, often supplied by MatPlotLib, are emphasized as a way of both exploring data. Consequently, you are comfortable writing custom Python code to generate scatter plots, histograms, function plots, etc. You are also facile with the task of selecting colors, such as different colors for different classes, along with axis labels and plot titles. Finally, grids of plots are also easily constructed in Python and you are able to write code to generate grids of plots.

14. Following on the importance of vectors and dot products is the idea of a hyperplane. Few mathematical constructs are more essential to understanding ML than that of the simple line in 2-D and the hyperplane in k-D. The essential underlying idea of using a dot product measuring on what side of a hyperplane an arbitrary point falls is key. Even as important as the "this side" versus "that side" distinction may be, the dot product does much more, revealing how far a point lies away from a hyperplane, and so as a result you would have no difficulty writing down the algebra and writing Python code to compute how far a point lies away from a hyperplane.

15. There is a marvelous little tool we introduced in the module where hyperplanes were discussed, namely the "pairplot" feature provided by "pandas". Put yourself in the position of demonstrating to a team of co-workers that a particular ML problem is trivial in so much as one feature out of many perfectly separates two classes. You should now feel entirely capable doing this with the aid of visualization tools such as "pairplot".

16. Matrices play many roles, but never lose sight of the simpler and more essential conveniences afforded to us by matrices. CS345 is designed to lead you into an early appreciation for dot products. What you also now intuit at a fundamental level is that a matrix vector multiplication can be interpreted as a compact way of writing down a sequence of dot products.

17. Every so often, an out-of-date technique that would no longer be used in its original form still warrants study and understanding as the jumping off point for an entire field. Rosenblatt's Perceptron is exactly such a creation. It is significant first for its historical place in the development of ML. But, perhaps more important, it provides an early (if somewhat flawed) model for how an ML algorithm should behave. Consequently, you are comfortable now using the Perceptron as a means of explaining the following concepts to another person, namely supervised learning, update rule, linear classifier, guaranteed convergence, learning rate, and finally, the role XOR plays in suggesting how to think about "hard" machine learning problems.

18. A simple intuition, namely checking if something similar has been seen before, gives rise to a powerful family of classifiers described as nearest neighbor classifiers. You are now comfortable with what a nearest neighbor does down to the level of being able to build one in Python. You even have seen a closely related concept not so often covered in a first semester ML class, namely an exemplar-based classifier.

19. In CS345 the nearest neighbor classifier was used as a jumping off point to begin the process of considering how a training versus test data split might best be defined. This means you are now comfortable with what it means to balance classes in a training-set and you understand how to gauge the relative value of increasing the amount of training data relative to test data.

20. Another concept reinforced in connection with nearest neighbor classifiers is that of a decision boundary. For linear classifiers you of course understand the simplicity of the decision boundary. Perhaps more important, you can describe the process of how a decision boundary arises in the context of a nearest neighbor classifier.

21. Simple sentences are of course desirable, but they do sometimes hide important details. In this light, consider the sentence "Samples A and B are similar." Before ML can be used this vague language has to be made precise, and that includes making a

commitment to a specific well-defined means of quantifying similarity. There are, at a minimum, three to four different approaches to this task, and you now can enumerate and code up these measures in Python.

22. Not all machine learning problems take the form of matching a label to a feature vector. There is another extremely common form of problem (the name should be jumping to your tongue right about now). You can instantly recognize and name this other form of problem and just as important write a succinct algebraic expression capturing the essence of the problem.

23. In Module 3 notebook 2 the connection between ML and optimization was made very explicit. This is one of the handful of places in CS345 where you should be thinking to yourself – that is why understanding basic calculus is important. Consequently, the connections between least-squared, derivatives, and fitting lines of best fit to data are all now related in how you think about linear regression models.

24. When fitting an ML model to sample data, another important concept is that of "outlier". Based upon what you have seen illustrated in this course through running notebook code you now have a clear and precise mental model for answering questions such as "What is an outlier?" and "How might I remove and outlier?"

25. How do you reach the top of a mountain – just keep walking up hill. So how does one optimize an ML model, with gradient descent(ascent) of course. In the module on fitting lines to data in CS345 effort was expended developing examples and hence motivating an intuition for the role that gradient descent plays in fitting models to data. Consequently, you now have a very pragmatic, and for simple examples, intuitive feel for how a gradient descent algorithm is constructed and behaves.

26. In CS345 we worked through in some detail a multivariate linear regression example involving return-on-investment in terms of sales relative to the purchase of advertising on different media, namely TV, radio and Newspaper. Not only are you familiar with this example, but you are also comfortable using it to discuss the basics of multivariate linear regression. You can also go a bit further and consider what it means to try to answer questions about relative importance of different features for making predictions.

27. An incredibly common theme in machine learning is that of taking an initial labeling or regression task which is non-linear and figuring out a way of transforming it into a linear problem. In CS345 this general idea is introduced in very specific terms and in the context of polynomial basis regression. Your knowledge and facility with the examples presented in class for this topic will serve you well both in terms of problems for which polynomial basis regression is a good answer and more generally to give you a mental model of the broader of approach of how to transform data to make ML easier/better.

28. One of the first times we encounter specific examples of overfitting in CS345 is in relationship to polynomial regression. The high-level lesson you now appreciate and can explain from this topic is how the ML optimization task can be extended to take on two – not one – constraint. Namely, a regularizing term (for example seek smaller coefficients rather than larger coefficients) can be added to the original mandate to fit the data well. It will do you well to study and understand the actual examples presented in class so that you can draw upon these to make sense of the broader concept of regularization in ML.

29. Put simply, when you are asked to describe k-fold cross validation with stratification at a job interview, you now can do so clearly and with concrete examples.

30. There is a jargon associated with reporting success and failures of an ML model that you must master. Consequently, the following terms are all now in your vocabulary: accuracy, true positive rate, false positive rate and confusion matrix. To go a bit further, as a person literate in ML you need to help others avoid some common mistakes when looking at summary statistics such as accuracy. Being even more concrete, how might two ML models each achieving an accuracy of 90%, turn out to have very different and possibly better/worse behavior for a given real world problem?

31. Decision trees represent a distinctive approach to solving classification problems relative to what was presented earlier, for example nearest neighbor classifiers. It goes without saying you understand what a decision tree looks like and how it operates. Perhaps more important, you now understand and can explain what information may be revealed by attempting to construct a decision tree relative to a particular problem/dataset. Finally, you probably have worked out that in CS345 we really have not had the time and resources to learn exactly what happens inside a modern decision tree construction algorithm; that is OK.

32. The "wisdom of the crowd" is a popular and intriguing topic even reaching into mainstream media. The more formal version of this phenomena arises in what are commonly called ensemble classifiers. Starting with a simple coin toss game, the lectures and notebooks in CS345 carefully establish the basics of how it can come to pass that a relatively unreliable algorithm may – when run repeatedly – come to be the basis for a highly reliable algorithm. This concept, indeed how the introduction of randomness in the outcome of a classifier, and help us (not hurt us) is so important that all aspects of it are now understood by you to such an extent you could convince a skeptic should the need arise.

33. The random forest approach to building a classifier unites to key ideas, decision tree and ensemble methods. Be clear in your mind how you would explain a random forest classifier to someone else relative to each concept as well as how they are put together.

34. The connection between Rosenblatt's Perceptron and modern neural networks should now be clear to you, including specifically the decision in CS345 to first introduce two-layer neural networks as ways of solving the XOR problem. Along the way, you now have at your disposal a complete understanding terms such as: single layer network, hidden layer, back propagation. Also, going all the way back to dot products, you can explain for the earliest networks presented in CS345 exactly where dot products are utilized in the calculation of a node's excitation.

35. Another feature of the early example of a neural network in CS345 was to underscore an often-neglected issue – namely that a single attempt to train a network from randomly selected weights may or may not find a good solution to your problem. Be ready to explain this very clearly perhaps using the example of 2D data in which one class of points lies in a ring surrounding the other class.

36. The original inspiration for a neural network node was a single neuron. The neuron receives excitations through is dendritic tree and when that excitation exceeds a threshold the neuron fires a signal down its axon. Hence was born the notion of an activation function in each cell of a neural network. However, as you can clearly explain, the family of possible activation functions in neural network models for ML expanded to include sigmoid style functions and ultimately ReLu. Be particularly aware and able to discuss the role of ReLu in modern ML.

37. When moving to modern ML models based upon neural networks the existence of modern API (for example TensorFlow) is a mostly good-news story with a catch. The good news is obvious, form CS345 you have gained facility at rapidly building potentially very comples and power neural network and fitting them to your data. The catch is understand as best possible what going on when training and subsequently using the models. Note almost all of CS345 has been planned to give you mental tools to help in this process of understanding. So, in this light, consider and be able to tell someone why CS345 carefully address the following: linear functions, non-linear functions, dot products, gradient descent, training versus test data, measures of performance, and finally the value of randomness.

38. In using Keras as a tool for building training and evaluating ML models there is a highly templated style of coding. At a minimum, you have come out of CS345 able to teach others about this style of coding and what the various parts of the code accomplish. For example, setting up the architecture as distinct from training the ML model.

39. Performance of an ML model on the training data is never of importance when it is time to convince yourself – or anyone else – that your model has generalized to the task as expressed through your data. For that performance on the TEST data is all that matters. However, when training models, it is common to plot both training and test data accuracy in a single plot. Be clear that you can explain why this is common and helpful.

40. Recognizing handwritten digits is not hard when compared to general objects, you know this to be true. To be more specific, in CS345 you have had the chance to work through in lecture an on your own a sequence of neural network ML models applied to the MNIST and CIFAR10 datasets. To begin, this means you are familiar with both datasets and how they can be used in conjunction with neural networks of different architectures. Also, you can discuss questions of the form "How much does it help to add another hidden layer?" or "What happens if the hidden layer is made wider?"

41. It may not seem like much, but when an ML person says they our using a one hot encoding, they are not saying it is cool/popular etc. More seriously, you can now convert a problem to a one hot encoding as well as understand the value in doing so.

42. Yann LeCun played a critical role in the modern ML revolution by introducing convolutional neural networks. You now have a concrete example of just how much better a convolutional neural network can perform and an object recognition task, and further you can explain the key elements including, what is a kernel, what is convolution, what is pooling, how deep is a CNN, and finally how many trainable parameters arise when compared to a brute force multilayer neural network.