SUMMARY OF COURSERA COURSE

STRUCTURING MACHINE LEARNING PROJECTS (2 WEEKS)

RATINGS: 4/5

WEEK 1 – ML STRATEGY PART 1

UNIT 1: Why ML Strategy

- How to much more quickly and efficiently get your machine learning systems working
- Learn some strategies in analyzing ML problems
- ML strategies are changing in the era of Deep Leaning

UNIT 2: Orthogonalization

- So many you could things to try when building a ML system
- Orthogonalization: being clear eyed in what to tune to achieve one effect
- Chains of assumptions in ML: Fit training set well on cost function (knobs = bigger network, different optimization algo) > Fit dev set well on cost function (knobs = regularization, bigger training set) > fit test set well on cost function (knobs = bigger dev set) > Performs well in real world (knob = change dev set or cost function)
- He tries to avoid using "early stopping"

UNIT 3: Single Number Evaluation Metric

- Your progress will be much faster if we have a real number evaluation metric
- There is a trade-off between "precision" and "recall"
- The "f1 score" averages the precision and recall scores
- Having a well-defined "dev set" and a single number evaluation metric helps to speed up the iterative process

UNIT 4: Satisficing and Optimizing Metric

- It is not always easy to combine all the things you care about into a single evaluation metric
- "maximize Accuracy and subject to running time < 100ms": Optimizing Metric = Accuracy, Satisficing Metric = Running Time
- If there are multiple things you care about you will want to set them to either "Optimizing" or "Satisficing" metrics

UNIT 5: Train/dev/test distributions

- The setup of the train/dev/test sets are very important
- Try to make sure the dev and test sets come from the same distribution
- Choose a dev set and test set to reflect data you expect to get in the future and consider important to do well on

UNIT 6: Size of the Dev and Test Sets

- Old way of splitting data: 70/30, 60/20/20
- In modern deep learning era with large dataset we can use smaller % (< 1%) for the dev/test sets
- Your test/dev set should be big enough to give high confidence in the overall performance of your system
- “train / dev sets” is the proper way of calling it in the case where there is no “test set”

UNIT 7: When to change dev/test sets and metrics

- Sometimes part way through a project you might realize you put your target in the wrong place, define a new evaluation metric if you are satisfied with the current metric
- “define a metric to evaluate classifiers as a step” > “how to do well on this metric as another step”
- If doing well on metric and dev/test sets but not doing well in actual application, try to change your metric and/or dev/test sets

UNIT 8: Why human-level performance

- Comparing Machine Learning systems to human level performance because: advances in deep learning is suddenly working better, the workflow of a machine learning system is much more efficient when you are trying to do something that humans can also do
- “Bayes Optimal Error” can be seen as the best possible error a model can achieve
- For most task the human level performance is not far from the Bayes Optimal Error
- While your algorithm is doing worse than humans you can: get labelled data from humans, gain insight from manual error analysis, better analysis of bias/variance
- Comparing to human level performance especially on task humans (image recognition) do well is a very good idea

UNIT 9: Avoidable Bias

- Human level performance can help to determine if our model is doing well or doing too well (determine if it is under fitting or overfitting)
- Human level error as a proxy for Bayes Error, very true for Computer Vision
- Difference between Bayes Error and training error as “Avoidable Bias”
- Knowing your “estimate Bayes Error” is very essential

UNIT 10: Understanding Human Level Performance

- Human level error helps to estimate our Bayes Error
- You have to be clear on what your purpose his for surpassing Human Level Error
- Making progress in ML systems gets harder as we approach Human Error performance
- Sometimes “Bayes Error” is not “0”

UNIT 11: Surpassing Huma-Level Performance

- People get excited when they surpass human level performance on a specific task
- When you surpass human level performance your options for improving your model is no more clear
- Problems where ML significantly surpasses human-level performance: Online Advertising, Product recommendations, Logistics (predicting transit time), loan approvals, they are mostly "Structured Data" cases
- Humans are very good at "Natural Perception" tasks and it is hard for Computer to surpass human level performance in this tasks

UNIT 12: Improving your model performance

- Two fundamental assumptions of supervised learning: You can fit the training set pretty well (reduce avoidable bias), the training set performance generalizes pretty well to the dev/test set (reduce variance)
- To reduce avoidable bias: train bigger model, train longer/better optimization, better NN architecture/hyper parameters
- To reduce variance: more data, regularization, NN architecture

UNIT 13: Machine Learning Flight Simulator

UNIT 14: Quiz

UNIT 15: Andrej Karpathy Interview

- He was the reference human level performance error holder in some image classification
- Unsupervised learning is not yet as promising as expected
- Try to know what is happening under the hood before diving into Frameworks

WEEK 2 – ML STRATEGY PART 2

UNIT 1: Carrying out error analysis

- Error analysis is the process of examining mistakes that your algorithm is making
- "ceiling performance"
- You can evaluate multiple ideas in parallel
- During error analysis we are looking at dev set example that our algorithm misrecognized
- It gives an estimate on how worthwhile it will be to work on an error

UNIT 2: Cleaning up incorrectly labelled data

- Deep Learning algorithms are quite robust to random errors in the training set, but less robust to systematic errors
- Add an extra column during error analysis for "incorrectly labelled" data
- Goal of dev set is to help you select between two classifiers A and B

- Correcting incorrect dev/test sets examples: apply same process to your dev and test sets to make sure they come from the same distribution, consider examining examples your algorithm got right as well as ones it got wrong (not easy to implement), train and dev/test data may now come from slightly different distributions (quite okay, learning algorithms are quite robust to this)
- People feel reluctant about manually carrying out error analysis

UNIT 3: Build your first system quickly, then iterate

- Quickly set up dev/test set and metric > build initial system quickly > use bias/variance analysis and error analysis to prioritize next steps
- Applies mostly for brand new machine learning application

UNIT 4: Training and Testing on different distributions

- Deep learning algorithms have a huge hunger for training data
- Make sure dev/test set data distribution are the same and reflect the distribution of data for future applications

UNIT 5: Bias and Variance with mismatched data distributions

- The way you analyze bias and variance changes when your training set comes from a different distribution than dev/test sets
- Define a "train-dev set": same distribution as training set, but not used for training
- "data mismatch problem": when algorithm works well on "train-dev set" but not on the "dev set"
- No super systematic ways to deal with data mismatch
- Using different data distribution on train and the dev/test sets helps to get more data

UNIT 6: Addressing Data Mismatch

- Carry out manual error analysis to try to understand difference between training and dev/test sets
- Make training data more similar; or collect more data similar to dev/test sets, you can sue artificial data synthesis to make training data more similar
- Artificial synthesis can make model over fit on a subset of data if it is synthesizing from only a subset of data

UNIT 7: Transfer Learning

- You can apply the knowledge the NN has learned from one task and apply that knowledge to a separate task
- Transfer Learning, if we have small data set just retrain the last one of two layers, change the output layers
- "pre-training" with initial application (e.g. image recognition) > "fine tuning" with new application (e.g. radiology diagnosis)

- Transfer Learning examples: using a model pre-trained on image recognition for radiology diagnosis, using a model pre-trained on speech recognition for wake words/trigger word detection
- We usually do Transfer Learning when we have a lot of data for the problem we are “transferring from” and relative less data for the problem we are “transferring to”, but situations where the opposite is true Transfer Learning will not be helpful
- Transfer Learning (from A to B) makes sense: When task A and B have the same input, when you have more data for A than B, when low level features from A will be useful in B

UNIT 8: Multi-task Learning

- In multi-task you start off simultaneously unlike in transfer learning where it is a sequential process
- Try to have one neural network do several things at the same time
- Unlike softmax regression in multitask learning an image can have more than one label
- Multitask learning can also be trained with images that have only a subset of the labels
- When multi-task learning makes sense: Training on a set of task that could benefit from having shared lower-level features, usually amount of data you have for each task is quite similar, if you can train a big enough neural network to do well on all the tasks
- Alternative to multi task is to train individual NN for each tasks
- Transfer Learning is used much more often than Multi-Task Learning
- Multi-task learning is used much in Computer Vision and Object Detection

UNIT 9: What is end-to-end learning?

- End to End Deep Learning: it takes multiple stages of data processing and replace them usually with a single NN, it obsoleted a lot of research, it requires a lot of data to work really well
- Speech Recognition can utilize End to End DL
- In cases where we don’t have enough data for End to End DL, breaking down the task into steps often works better
- End to End DL works very well with Machine Translation (e.g. English to French) because there is enough data for this
- End to End DL does not always works but when it does it helps to simplify the problem

UNIT 10: Whether to use End-to-End Deep Learning

- Pros: let’s the data speak (lets the learning algorithm learn by itself), less hand-designing of components needed
- Cons: May need large amount of data, excludes potentially useful hand-designed components
- When you don’t have much data, hand-designing components could be helpful for the algorithm
- Do you have sufficient data to learn a function of the complexity needed to map X to Y?

UNIT 11: Quiz

UNIT 12: Ruslan Salakhutdinov Interview

- He is currently director of research at Apple
- Increase in computing power has helped in the emergence of Deep Learning
- A lot of success nowadays is in Supervised Learning
- Try different things, don’t be scared to innovate
- It is important to know what is under the hood
- You can do amazing things in both “Industry” and “Academia”