COURSE 3

STRUCTURING MACHINE LEARNING PROJECTS

WEEK 01

- ML Strategy, Introduction to ML Strategy

① Why ML strategy?

To see which ideas are worth pursuing or not?

② Orthogonalization:

Having orthogonal means that if we want to increase or decrease something then other should not affected until unless we want them to work together like having two knobs different for different work

Chain of assumptions in ML

- fit training set well on cost function
- fit dev set well on cost
- fit test set well
- Performs well in real world.

If above does not work properly so we do need knob or parameters for each like training set parameter does not effect test set parameters.

if Training set is not doing well

- Bigger Network
- Adam

if dev set is not doing well.

- Regularization

- Bigger training set

If dev set does well but not test set then

- Bigger dev set

If it does not perform well in real world then (dev set is doing well)

- Change dev set or
- cost function

Early stopping which affects all of them.

- Setting Up your Goal

① Single Number Evaluation Metric

F1 score is that single number evaluation metric

This is called Harmonic Mean of Precision and Recall.

$\left(\frac{2}{\frac{1}{P}+\frac{1}{R}}\right)$

② Satisficing and Optimising Metric

Suppose

Cost = accuracy − 0.5 × running Time

maximize accuracy and

running time ≤ 100ms

That means optimizing accuracy and satisficing running time

Optimizing → Accuracy
Satisficing → Running Time

Classifier	Accuracy	Running Time
A	90%	80ms
B	92%	95ms
C	95%	1500ms

③ Train/Dev/Test Distributions

Dev/Test sets:

↳ Hold out cross validation

Dev set and test set should come from the same distribution

④ Size of Dev and Test sets

Old way

Train: 70% Test: 30%

This was good for smaller dataset but now the dataset is larger then 98% is enough for training data.

Set the test set big enough to give you enough confidence.

Sometimes, we just need train and dev set.

⑤ When to change dev/test set and metrics

Example

Metric: Classification Error

Algorithm A: 3% error

Algorithm B: 5% error

↳ It does not have pornographic images

It have pornographic images

Error: $\frac{1}{m_{dev}} \sum_{i=1}^{m_{dev}} \mathcal{L}\{y^i_{pred.} \neq y^i\}$

- One way is to change or add the weight
- Otherway is to labeling correctly or defining new evaluation metric

- Comparing to Human Level Performance

① Why human-level performance?

Bayes optical error is the best possible error so that it can not reduce that error.

② Avoidable Bias:

Human level 1% } → We showed focus
Training error 8% } on bias
Dev Error

When human level error is 7.5% then we will focus on variance for Dev.

③ Understanding Human Level Performance

Least error which can be defined can be taken as proxy for the Bayes level error.

④ Surpassing Human Level Performance

Problems where ML significantly surpasses human-level-performance

- Online advertising
- Product Recommendations
- Logistics
- Loan approvals.

⑤ Improving your model performance

The two fundamental assumptions of supervised learning

(A) You can fit training set pretty well.

(B) Training set performance generalizes pretty well to dev/test set.

Human Level.

↕ Avoidable Bias →

Training Error

↕ Variance →

Dev Error.

→ Train Bigger Model
→ Train longer / better optimization algo
  - Momentum, RMSprop, Adam
→ NN. architecture / hyperparameters search

→ More data
→ Regularization
  - L2
  - Dropout
  - Data Augmentation.

WEEK 2

- Error Analysis

① Carrying out Error Analysis

Ceiling:

What is best u can get?

② Cleaning up Incorrectly Labeled Data.

DL algorithms are quite robust to random errors in the training set. They are less robust to consistently labeled.

If incorrectly labeled data makes a significant impact then we should do it.

③ Build your first system quickly, then iterate.

- Set up dev/test set and metric
- Build initial system quickly
- Use Bias/Variance analysis & Error to prioritize next steps.

- Mismatched Training and Dev/Test set

① Training and Testing on Different Distributions

② Bias and Variance with Mismatched Data Distributions

Let's say

humans ≈ 0% error

Training error — 1% error

Dev error — 10% error

9% increase error can be because of 2 reasons

(A) Either their distribution is different

(B) Algorithm saw training data not in dev set.

The solution for these 2 problem include we create a training-dev set:

Same distribution as training set, but not used for training.

Training error	1%	1%
	↕ variance	
Training-dev error	9%	1.5%
	↕ data mismatch	
Dev-error	10%	10%

① There is no issue with distribution we have to train more.

Human Error	0%	→ Avoidable Bias
Training error	10%	10% → Avoidable Bias
Training_dev error	11%	11%
Dev - error	12%	20% ↕ Data mismatch.
	↓	
	High Biased.	High Biased + Data. mismaiched.

Human Error.	4%	→ Avoidable Bias.
Training "	7%	→ Variance.
Training - dev "	10%	→ Data Mismaiched.
Dev - Error.	12%	→ Degree of overfitting to devset.
Test Error.	12%	

More general informatim

	General Speech Recognition	Rearview mirror speech data.
Human level.	4% → Avoidable Bias	6%
Error on example Trained	7% → Training variance	6%
Error on example not trained.	10% → Training-dev error	
	Dev/Test	6%
	↓ data mismatch	

(3) Addressing Data Mismaich

Inle can try out

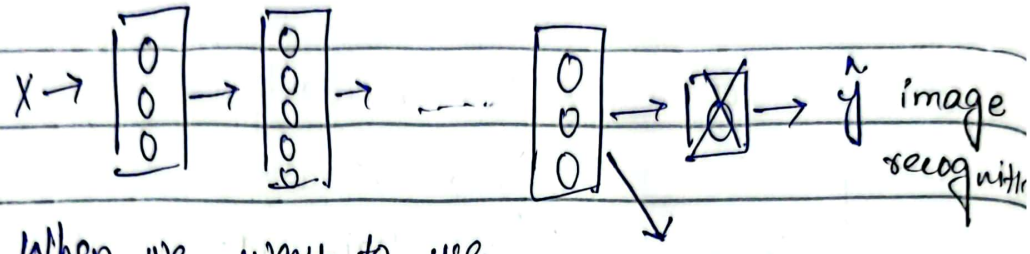
- Cassy out manual error analysis to try

to understand difference between training and dev/test sets.

- Make training data more similar; or collect more data similar to dev/test sets.

- Learning from multiple tasks :

① Transfer Learning



When we want to use the learning of one algorithm then we just have to discard last layer and weights and the remaining can be same. And, new data set

If we have large dataset then we can have or retrain all neural network otherwise just change or initialize weights of last layer.

Fine tuning data is that we again retrain the data.

It works when you have small data for 2nd problem.

When it will help?

- Task A and B have the same input x.

- You have a lot more data for Task A than Task B
- Low level features from A could be helpful for learning B

② Multi-Task Learning

When there are more than one output like in autonomous car we have to detect hurdles, other cars, stop sign and traffic signal etc.

Loss : $\hat{y}^{(i)}$ $(4,1)$

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{4}\left(\mathcal{L}(\hat{y}_j^{(i)}, y_j^{(i)})\right)$$

↓

usual logistic loss

$$-y_j^{(i)}\log\hat{y}_j^{(i)} - (1-y_j^{(i)})\log(1-\hat{y}_j^{(i)})$$

We can do this even if some labels are missing.

when multi-task learning makes sense

- Having shared low-level features
- Amount of data you have for each task is quite similar.
- Can train a big enough neural network to do well on all the task

Note: It only hurts if neural network is not much large.

- End-to-End Deep Learning.

① What is end-to-end deep learning?

X —MFCC→ features —ML→ Phonemes → words → transcript ŷ

audio

audio —end-to-end deep learning→ transcript

The only drawback is that we need a large dataset.

② Whether to use end-to-end learning

Pros:

- let the data to speak. (We don't force to use our logic)
- Less hand-designing of components needed

Cons:

- May need large amount of data.
- Excludes potentially useful hand-designed components.

Applying end-to-end deep learning

Do you have sufficient data to learn a function of the complexity needed to map x to y?