Course 3 :- Structuring Machine Learning Projects. Start date. 15-09-19.

Week 1 :- ML Strategy (1).

P1 : Introduction to ML Strategy.

L1 : Why ML Strategy. [10:30 am 15-01-19]

Some ideas of machine learning. (deep learning)

1. Collect more data.
2. Collect more diverse training set.
3. Train algorithm longer.
4. Try adam instead of G.d.
5. Try bigger network
6. Try smaller network.
7. Try dropout.
8. Add L2 regularization.
9. Change network architecture.
   - # activation function.
   - # hidden units.

[we will learn how to effectively use these ideas to solve our problem.

L2 : Orthogonalization [10:35 am 15.01.19].

→ Orthogonalization is a system design property that ensures that modifying an instruction or a component of an algorithm will not create side effects to other components of the system.

When a supervised learning system is designed these 4 assumptions are needed to be true and orthogonal.

1. Fit training set well on cost function.

   [if doesn't fit well, (i) use bigger neural network
   (ii) use better optimization algorithm.

2. Fit development set well on cost function.

→ If doesn't fit well, regularization or using bigger network might help.

3. Fit test set well on cost function.

→ [the use of bigger development set might help]

4. Performs well in real world.

P2: Setting up your goal.

L1: Single number evaluation metric. [8:10 pm 15-01-18]

[Precision, Recall, F_1-Score] → average of precision and recall.

→ we have to chose a single number evaluation metric by which we can correctly separate results of our classifiers.

[Such as using F_1-Score as evaluation metric which averages precision and recall].

** To chose a classifier from a set of classifiers,

i) a well defined development set

ii) a single number evaluation metric

are useful to speed up iteration process.

L2 :- Satisfying and optimizing metric. [8.20 pm 15.09.19]

Evaluation metrices. [used to evaluate performance of a classifier]

optimizing metrices

[The evaluation metric that we have to optimize]

[Accuracy , Cost minimization etc]

satisficing metrices.

[ok with some evaluation metric has satisfactory results]

[Running time, e.t.c] .

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

Accuracy → optimizing metric.

Running time → satisficing metric.

$N_{metric} = \begin{cases} 1 & \text{optimizing metric} \\ N_{metric}-1 & \text{satisficing metric} \end{cases}$

[8.35 pm 15.09.19]

L3 :- Train/dev/Test distributions.

v.v.g ** choose the development and test sets from the same distribution. It must be taken randomly from all data =

→ [আমার কাছে Data আছে এখন predict করতে হবে মানুষটি কি loan pay করবে কিনা].

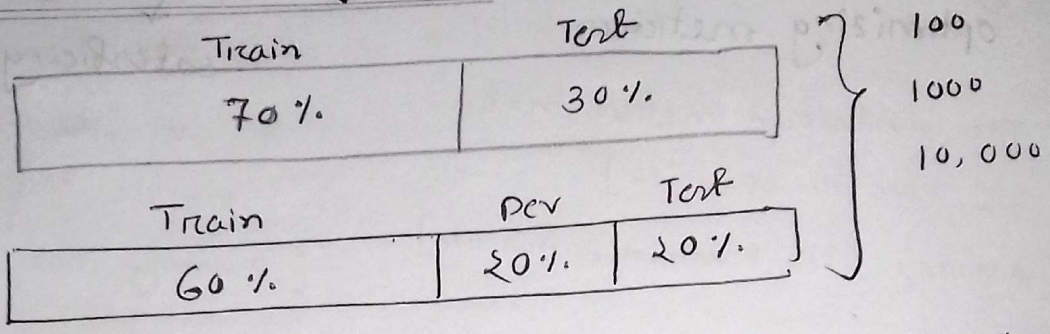
Guideline :-

Choose a dev set and test set (same distribution) to reflect data you expect to get in the future and consider important to do well on.

L4 : Size of the dev and test sets. [8.50 pm 15.01.19]

[From large amount of data we can use small dev and test set]

old way of splitting data.



guidelines.

** set your test set to be enough to give high confidence in the overall performance of your system.

** Test set helps evaluate the performance of the final classifier which could be less 30% of the dataset.

** The development set has to be big enough to evaluate different ideas.

** modern era - Big data.

Because a large amount of data is available, we don't have to compromised as much and can use a greater portion to train the model.

L5 :- when to change dev/test sets and metrics. [9.05 pm 15.01.19]
(2 of 12)

The misclassification error metric.

$$\text{Error} : \frac{1}{m_{dev}} \sum_{i=1}^{m_{dev}} \mathcal{L}\{\hat{y}^{(i)} \neq y^{(i)}\}.$$

This function counts up the number of misclassified examples.

→ If evaluation metric fails to correctly rank preferences between algorithms. The evaluation metric can be changed by adding up the weight term.

$$w^{(i)} = \begin{cases} 1 \\ 10 \end{cases}$$ - Adding different weight to rank preferences correctly.

→ Final function :-

$$\text{Error} : \frac{1}{\sum w^{(i)}} \sum_{i=1}^{m_{dev}} w^{(i)} \mathcal{L}\{\hat{y}^{(i)} \neq y^{(i)}\}.$$

[Adding weight to change evaluation metric]

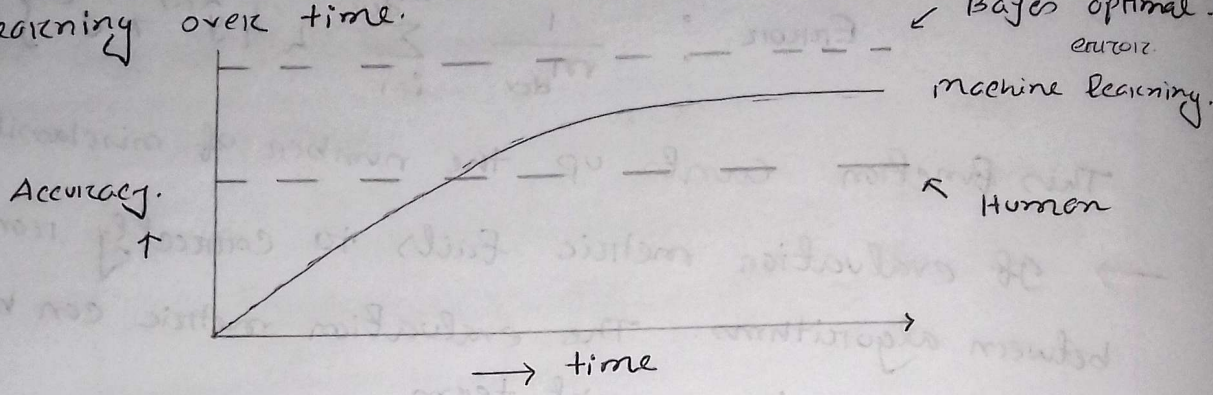
Guideline :

1. Define correctly an evaluation metric that helps better rank order classifiers.
2. optimize evaluation metric.

P3 :- Comparing to human-level performance.

L1 :- why human level performance. [11.25 pm 15.01.19]

This graph shows performance of humans and machine learning over time.



→ Bayes optimal error is defined as the best possible error. Any function mapping from x to y can't surpass this. level of accuracy. (Algorithm that always correctly predict.

** machine learning progresses when it surpasses human-level performance.

** Humans are quite good at a lot of tasks. So long as ML is worse than human, you can:

- Get labeled data from humans.
- Gain insight from manual error analysis: why did a person get this right?
- Better analysis of Bias and variance.

L2 :- Avoidable Bias. [11.40 PM 15-01-19]

	A	B
Human	1 %	7.5 %
Training error	8 %	8 %
Dev error	10 %	10 %

** The difference between training set and human level error is called avoidable bias.
↳ Try to reduce it.

(A) An gap between human error and training error is high use bias reduction technique. [Train a bigger neural network, Running the training set longer]

(B) An difference between human error and training error is low use variance reduction technique. [Regularization, Have a big training set]

L3 :- Understanding Human-level performance. [10.15 Am 16.01.19]

Human level error given an estimate of Bayes error.
The definition of human-level error depends on the purpose of the analysis. [eg my machine can work as good as human then it is really good]

Error Analysis Example :-

Human (Proxy for Bayes Error)
↕ Avoidable bias.
Training error
↕ variance
Dev error.

Human	1 %	1 %	0.7 %
		0.7 %	0.5 %
Avoidable bias		0.5 %	0.2 %
Training error	8 %	5 %	1 %
variance		1 %	0.7 %
Dev error	10 %	6 %	0.8 %
		Focus on Bias reduction technique	Focus on variance reduction technique.

L-4 Surpassing Human-level performance. [10:30 Am 16.01.19]

→ when training error is less than human level error or avoidable bias is (≤ 0) then we can call our machine has surpassed human level performance.

→ There are many problems where machine learning significantly surpass human level performance, espicialy with structure data.

- online advertising.
- Product recommendations.
- Logistics (predict transit time).
- Loan approvals. etc.

** structure data.
** un-structure data.

L-5 Improving your model performance [10.40 Am 16.01.19]

Two fundamental assumptions of supervised learning.

1. You can fit training set pretty well.
   (or have low avoidable bias).

2. The training set performance generalizes pretty well to the dev/test set.
   (Have low or acceptable variance).

Reducing (avoidable) bias and variance.

Human-level.

↕ Avoidable bias.

Training error.

↕ variance

Dev error.

1. Train bigger model.
2. Train longer / better optimization algorithm
   → momentum, RMSProp, Adam.
3. NN architecture / hyperparameter search.

1. Get more data.
2. Use Regularization.
   → L2, dropout, data augmentation.
3. NN architecture / hyperparameter search.

** change learning rate, activation functions. etc.

quiz :-

Week 2 : ML Strategy (2)

P1 :- Error Analysis.

L-1 :- Carrying out error analysis. [10.30 Am 19.01.19]

→ Look at development examples to evaluate new ideas.

→ Evaluate multiple ideas in parallel.

L-2 : Cleaning up incorrectly labeled data. [10.45 Am 19.01.19]

→ DL algorithms are quite robust to random errors.

Overall dev set error	10%	2%
Errors due incorrect labels	0.6%	0.6%
Errors due to other causes	9.4%	1.4%

Correcting incorrect dev/test set examples.

1. Apply same process to your dev and test sets to make sure they continue to come from same distribution.

2. Consider examining examples your algorithm got right as well as ones it got wrong.

3. Train and dev/test data may now come from different distributions.

L3: Build your first system quickly, then iterate. [12.00 Am 19.01.19]

Build your first system quickly,

1. Set up development/test set and metrices.
   - set up a target.
2. Build an initial system quickly.
   - Train training set quickly : Fit the parameters.
   - Development set : Tune the parameters.
   - Test set : Asses the performance.
3. Use Bias/variance analysis and Error analysis to Prioritize next steps.

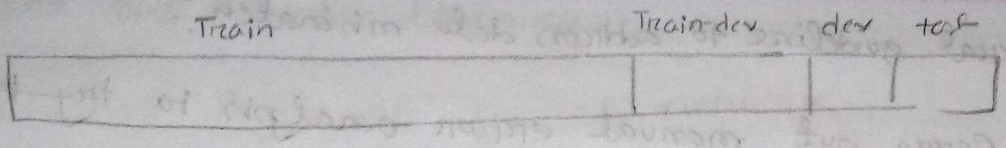
P2: mismatched training and dev/test set.

L1: Training and Testing on different distributions. [1.15 PM 19.01.19]

→ we have to choose a development set and test set to reflect data you expect to get in the future and consider important to do well.

- Training set test/dev set

L2: Bias and Variance with mismatched data distributions. [4.20 pm 19.01.19]



** Train-dev set and train set comes from same distribution.

→ Train-dev set is used find whether there is Bias, variance or data mismatch.

General formulation of error analysis.

Human level (Bayes error)
↕ → avoidable bias.
Training set error.
↕ → Variance.
Training-dev set error.
↕ → Data mismatch.
Dev set error.
↕ → Degree of overfitting to the dev set
Test set error.

we learn how to analyze bias and variance changes efficiently when training set is from a different distribution than the development and test sets.

[we have to take decisions by keeping in mind different error analysis techniques.

L-3 Addressing Data mismatch. [7.45 pm 19.01.19]

General guideline to address data mismatch:-

→ Carry out manual error analysis to ~~try to~~ understand the error differences between training, dev/test sets.

→ make training data or collect data similar to development and test sets.

** Artificial data synthesis can be used to make training data more similar to development set.

P3: Learning from multiple tasks.

L-1 Transfer Learning. [11.20 pm 19.01.19]

When neural network knowledge of one application used for another application it is called transfer learning.

** pre-training ** fine-tuning.

→ Transfer learning is useful when we have a lot of data for the problem we transfer from and less data for the problem we transfer to.

when to use transfer learning. [Transfer A → B].

1. Task A and B have the same input x.
2. You have a lot more data for Task A than Task B.
3. Low level features from task A could be helpful for task B.

Guidline for transfer learning.

→ delete last layer of neural network.

→ delete weights feeding into the last output layer of the neural network.

→ create a new set of randomly initialized weights for the last layer only.

→ New data (x, y).

L-2 multi-task learning. [11:40 pm 10.01.19]

- multitask learning refers to having one neural network to do simultaneously several tasks.

Example :- Such as train a neural network to detect pedestrain, car, road sign, traffic lights etc.

when to use multi-task learning.

1. Training on a set of tasks that could benefit from having shared lower-level features.

2. usually amount of data you have for each task is quite similar.

3. Can train a big enough neural network to do well on all the tasks.

[Transfer learning use more often than transfer learning]

P4 : End to end deep learning.

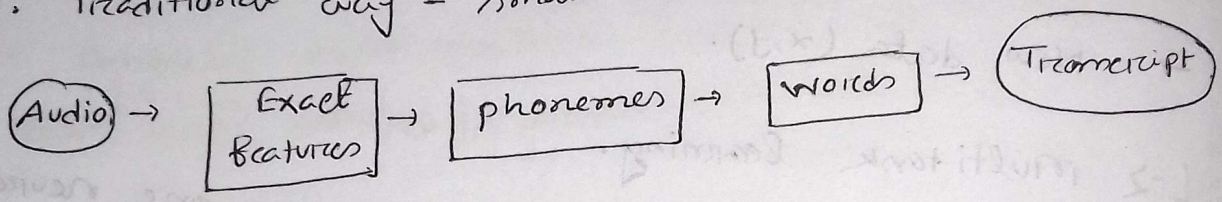
L-1 : what is end to end deep learning? [1.20 Am 20.01.19

End to end deep learning is the simplification of a pipeline or learning systems into one neural network.

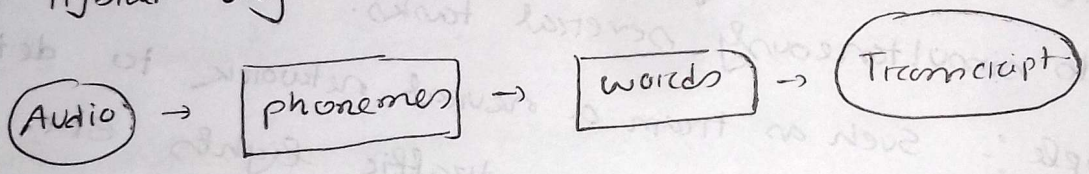
** End to end deep learning can not be used for every problem because it needs a lots of labeled data.

Example :- Speech recognition model.

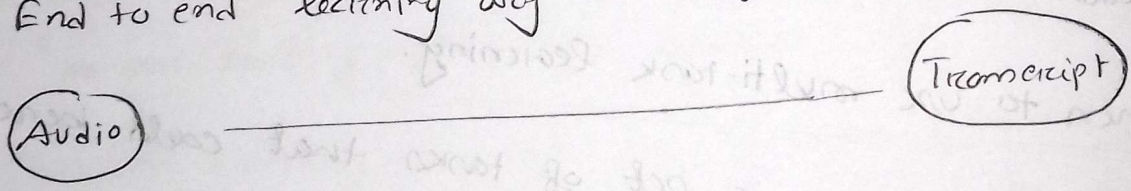
1. Traditional way - small data set



2. Hybrid way - medium data set.



3. End to end learning way — Large data set



Applications :-

1. Audio transcript / speech recognition.
2. image captures.
3. image synthesis.
4. machine translation.
5. face recognition.
6. Self-driving car etc.

L2:- whether to use end to end deep learning. [11:15 Am 20.01.19]

If one has enough data to learn a function then he can use end to end deep learning.

Pros:-

1. Let data speak:- machine will able to find which statistics are in the data, rather than being forced to reflect human preconceptions.

2. Less hand-designing of components needed:- It simplifies design work flow.

Cons:-

1. Require large amount of labeled data.:-

→ It can not be used for every problem as it needs a lot of labeled data.

2. Excludes potentially useful hand-designed components.