3 ML strategies

Andrew Ng

Note by:
Hossein Mamaghanian
amirhm@gmail.com

"ML Strategies"

better to define an evaluation metrics

Precision: recognized as cat what % are really cat

Recall: what % of all cats are correctly recognized.

→ F1 score: (standard way of combining Precision and Recall)

$$= \frac{2}{\frac{1}{P} + \frac{1}{R}}$$: "hermonic mean."

having a single evaluation metrics speed up iterating

Or in other situation may be an average could be solution for a simple metrics.

---

example :

	Accuracy	Running time
A	92	80ms
B	93	95ms
C	95	1500ms

Cost = Acc − 0.5 × running Time

Maximize Accuracy subject to running Time < 100ms

optimizing metrics. satisfying metrics

if we have N metrics | : 1 optimizing
better to define → | : (N-1) satisfing

→ example of google assistor Amazon Alexa:

→ wake words / Trigger words.

* Alexa, ok google, hey siri

(optim) accuracy:

(satisfy) # false positive:

Maximize accuracy s.t ≤ 1 false positive in 24 h

Train / Dev / test distributions

↳ developement set / hold out cross validation set.

for example if we have data set from different geographic regions:

US
UK
eu.
S.A
India
china

} dev
} test

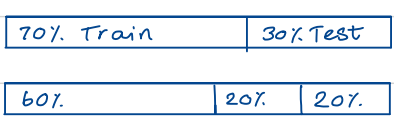
(Dev and test should come from same distribution)

bad idea / better idea would be to shuffle into Dev/test.

Dev/Test should reflect data you expect to get in the future.
(should be from same distribution)

---

old way of splitting data



} if we have less data (old way)
100 ~ 10K

In modern era of ML, where we need huge data for training

size of test set.

big enough to give high confidence in the overall performance of system.

→ it is ok if only Train + Dev is used. (No test).

(Not usally recommended, better to keep test as an seperate set).

→ when to change dev/test sets and metrics.

some times, since the metric does not reflect the what is prefend to rank different algorithms.

→ Alg A : 03% error (cat) → but higher F.p (Pornographic images).

Alg B : 05% error (cat)

→ then: metric + Dev → Algo A

You/user → prefer B

→ if metrics defined as:

$$\frac{1}{m_{dev}} \sum^{m_{dev}} \mathcal{L}(y^{(i)}_{pred}, y^{(i)})$$

Take away: if metrics you are using actually is not ranking on the preference order, define new metrics

→ adding a weight to penalize the F.P for Pornographic

$$\text{error} = \frac{1}{\sum w_i} \sum^{m_{dev}}_{i=1} w_i \mathcal{L}(y^{(i)}_{pred}, y^{(i)})$$

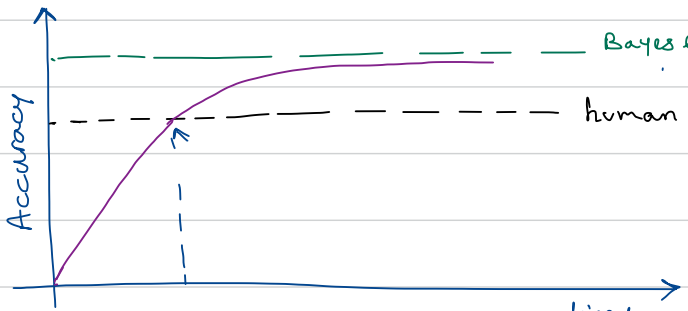
$$w_i \begin{cases} 1 \rightarrow x^{(i)} \text{ non porn} \\ 10 \rightarrow x^{(i)} \text{ porn} \end{cases}$$

"Orthogonalization"

① Define a better metrics.

② Worry seperately, how to do well on this metric.

Human level / Baysian error.



Bayes error, Bayse optimal error. (best possible error).

easy to reach to the level of human on some task but gets difficult to surpass it.

if performance is bellow the human level :

1. get more labeld data from humans.
2. get insight from manual error Analysis (why human get this right;)
3. Bias/ Variance Analysis (considering human level).

Bias/ Variance considering human level

suppose:

space for improvement

human err	1 %	7.5%	0.5% avoidable bias
Train err	8 %	8%	
Dev err	10%	10%	space for improvement.

→ Focus on bias since still there is room for better training

→ focus on variance

→ human level error as a proxy for Bays error.

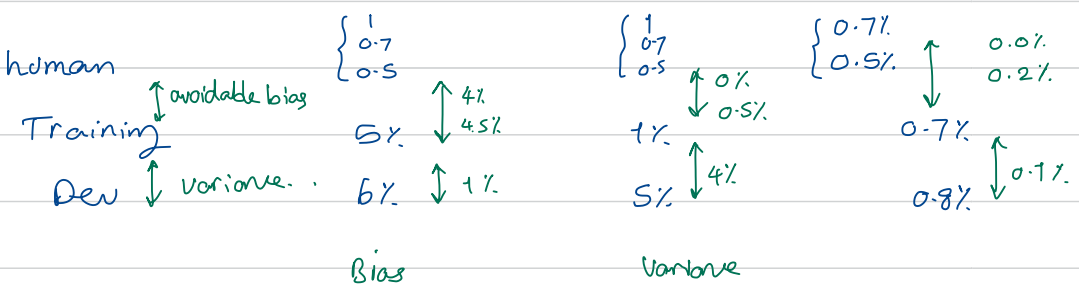
Imagine a radiology task:

Typical human	3%
Doctor	1%
Experienced doctor	0.7%
team of ...	0.5%

what is the human-level ?

→ Bayes error ⩽ 0.5 %

→ for the purpose of publication, maybe other options could be used.



Regardless of which human-level is used.

here, as it gets better it's relevant which one to choose.

Surpassing human-level.

→ team of human	0.5%		0.5 ← when it's surpassed,
one human	~~1%~~	0.1%	1% it's difficult to rely
Train error	0.6		0.3 on human intuition.
Dev error	0.8	0.2	0.4

Problems where ML significantly surpasses human-level

→ online ads
→ product recommendation
→ logistics.
→ loan approvals.

① . all learned from structured data
② . No natural perception (if there is, human does better)
③ . lots of data are available.

Some other perception task which ML is close or surpassing human-level.

- speech recognition
- some image recognition
- medical. (ecg, skin images, ...)

Fundemental assumption of unsupervised learning

1. You can fit the traing set, pretty well. ~ Avoidable bias
2. Training set performance generalize pretty well to the dev/test sets. ~ variance

human
↓ Avoidable bias
Training
↕ variance
Dev

Avoidable bias:
1. Train bigger model.
2. Train longer/better opt. NN architecture/h.p search.

variance:
1. More data 2. using regularization (l2, dropout, data-aug.)
3. NN architecture 4. h.p search

"error Analysis"

For example using error Analysis to check weather an idea worth to work on

→ Imagine → 90% Accuracy in cat/dog classifier

10% error

→ ex. idea: You want to improve the cases that cats are wrongly classified

→ Take 100 ex from Dev set, check how many of mislabels were dog

5% → very little effect

50% → worth to check.

→ evaluate several ideas in parallel.

① Fix pic of dog recognized as cat

② Fix for big cats.

③ Improvement of performance on blurry images.

Image	idea 1	idea 2	idea 3	comments
1.				
2.	✓			Pitbulls,...
3.		✓		
4.			✓	
⋮				⋮
		✓		
% of total	8%	43%	61%	

helps to see which idea worth to work and has the greatest impact.

cleaning up incorrectly labeled data.

→ if the error exist in labeled data (Train/Dev/Test) data sets.

Normally DL. algorithms are robust to random errors in the training set (Most of the time it's ok, to leave it and not spend time to fix as long as it's random, small and size of data set is big).

"But Not systematic errors"

↳ example if in cat/Dog problem all the white dogs are mislabeled as cat (This is called systematic error).

How about Dev/Test set

→ "Do error analysis"

Image	Dog	Great Cat	Blurry	Incorrectly labeled	Comments
...					
98				✓	Labeler missed cat in background
99		✓			
100				✓	Drawing of a cat; Not a real cat.
% of total	8%	43%	61%	6%	

does it make sense to go and Fix?

	case1	case2
all errors	10%	2%
other cause	9.4%	1.4%
mislabled	0.6%	0.6%
	Not a big impact	Big impact

→ Make sense to go and fix mislables

If this was in dev set and goal was to select between two classifier A & B and range of errors between two were in affected range then make sense to fix mislabled info.

Strategies for fixing incorrect Dev/test examples.

→ Apply same Process (To make sure they continue to come from same dis)

→ consider examining both examples algo got right and as well wrong.

o Now after fixing Dev/Test, Training set is slightly from different distribution (This is less problematic, But O.K. since Traing Size is huge and Maybe not possible or less priority).

→ Build your system quickly, then iterate.

* Set up Dev/Test set and metrics.
* Build system quickly,
* Use Bias/Variance analysis & error analysis To prioritize next steps.
  ↳ For example in speech problem we may find that mone error is due to "far from Mic" then we can work on that.

example speech project

1. Noisy background
   → cafe/car
2. Accented speech
3. Far from Mic
4. Young children's speech.
5. Stuttering

→ Build Fast quick/Dirty, Not over think just to do analysis and prioritize.

→ less strongly if there are significant prior experience/or literature!

Mismatched Training and dev/test data,

example: We want to design app for cat/dog problem

200k image from web (clean/framed)

10k from cellphone camera

→ There are two way to use this data.

Option 1: shuffle all 210k image and draw train/dev/test from that

Train	205K	Pros: Train/Dev/Test are all from same distribution
Dev	2.5K	Cons: only $\frac{10}{210}$ % of Dev/Test set are from our target
Test	2.5K	distribution (cellphone). → Aiming at wrong direction

Option 2: Dev/Test all are from target application.

Train	205K (200k + 5K cellphone).	Cons: Train set is from different distribution
Dev	2.5K (all cellphone)	Pros: Dev/Test are from correct target distribution
Test	" " "	Aiming at right direction.

→ Option 2 is better in longer term, since Training is done for correct target distribution.

Bias / Variance analysis with mismatched data distribution.

→ This is very useful analysis to prioritize next steps but how if datasets are not from same distributions?

→ Imagine Cat/Dog problem

Human err	0%	if dev is not from same distributions now is not possible to say that is it really variance problem or data mismatch (simply Dev set is much difficult).
Training_err	1%	
Dev err	10%	9%

→ Now, we should introduce new Training-dev set. (drawn from Train set)

→ Same as Training distribution, but not trained on them, (not seen by algorithm)

Now Human error	0%	0%	0%	0%
Train error	1%	1%	10% (Avoidable bias)	10% (Avoidable bias)
Train-dev	9% (variance problem)	1.5% (0.5% bias)	11%	11%
Dev	10%	10% (8.5%)	12%	20% (data-mismatch)
	Main is variance Problem Not data mismatch	main Problem is data mismatch,	avoidable bias Problem.	both bias and data mismatch

Human Level	4%	avoidable bias
Train error	7%	variance
Train-Dev	10%	data-mis-match
Dev_err	12%	degree of overfitting to dev set
Test_err	12%	

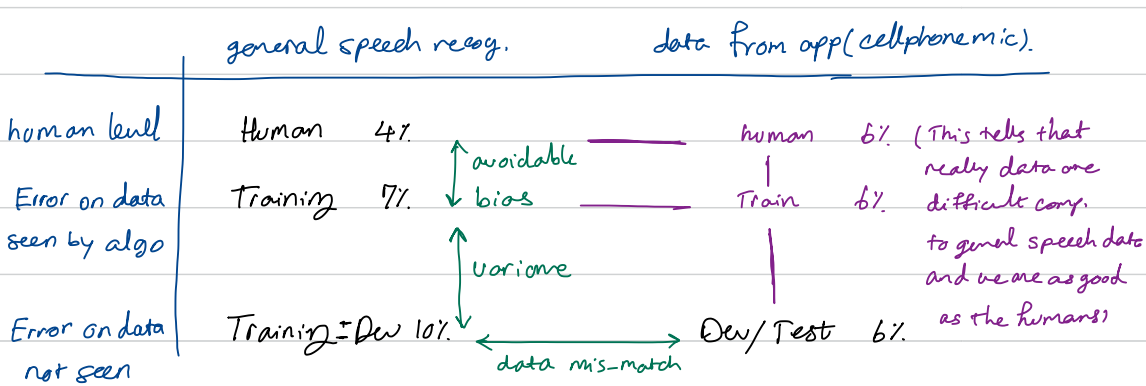
if this Problem exist means that you should go back and get bigger dev set.

generally this trends of errors getting worse as we go down but in some cases maybe Dev/Test is much easier than Train set and error for them might be lower.

More general Formulation

Imagine an application that we build a speech recognition system.

where Training is done on general data but target is from app.



Main Path for analysis is the green line but sometimes it is worth to fill the violet parts as well to get some more insight.

Addressing data mis-match.

→ carry out manual error analysis between Training, dev/Test sets

for example car noise

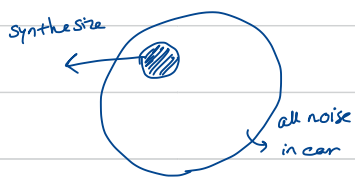
→ Make Training data more similar to dev/Test sets

add artificial generated car noise ..

Artificial data synthesize work but there is a catch that maybe we overfit to the artificial data → example

real speech + car noise

100,000 hr 1 hr



chance that learning algo overfits to this subset

→ Most of the time for human ear this look fine but not in reality.

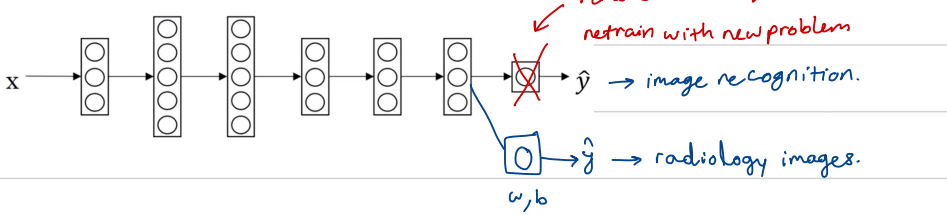
same for example of taking car images from games. (very small subset of cars exist in games, to human eyes is fine but not in reality).

"Transfer learning"

Taking the knowledge neural network learned from one task and apply to seperate task.

Imagine Problem of radiology images.

Take an image recognition model



→ depending on the amount of data we have for the problem at hand we may use the knowledge learned in image recognition task as initial value and Train full network or just the latest layers which are added

→ if full net is trained the training on image recognition is called "pre-training" and on radiology called "fine tunning"

→ The logic behind is that most of knowledge learned in early layers are general features in image like edge / curvature, ...) which is useful for radiology task

Trasfer learning from Task A → B make sense if

1. both have same input (image in radiology example).
2. We have alot more data for task A than B. (if reverse is true, it doesn't hurt but just would'nt make sense)
3. low-level features from A could be helpful for learning B

learning from multiple task.

→ In multi-task learning idea is to start off simultaneously, trying to learn several task at the same time.. each task helps other tasks.

example of self driving car

goal: { Pedstrains. Cars Stop Signs. Traffic lights. } $y = \begin{bmatrix} x \\ x \\ x \\ x \end{bmatrix}$

Output node is 4x1 vector or in context of Traing set $Y \in R^{4 \times m}$

The intuition is that all these tasks are related and knowledge learned could be useful for others as well.

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

↳ usual logistic loss

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

→ unlike soft max regression

one image could have multiple lables.

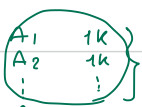
→ In multi-task learning, it is possible to train even if not all the lables are give for each image. $Y = \begin{bmatrix} 1 & 1 & \cdots \\ 1 & 0 & \\ 0 & ? & \\ ? & ? & \end{bmatrix}$ → in such case in loss calculation sum over j is done just for available labels.

"When it make sense?"

① . a set of task that could benefit from having shared low-level features

② . Amount of data for each task is quite similar.

in T.L. A 1, M
↓
B 1K



→ There are a lot more data to learn and augment on task A100

③ Can Train a big enough N.N. to do well on all.

→ Compared to alternative to multi-task learning which is independently Training different N.N for each task, — multi-task learning only hurts when the NN. is not big enough.

end to end deep learning

Traditional speech recognition:

X (audio) $\xrightarrow{\text{MFCC}}$ features $\xrightarrow{\text{ML}}$ Phonemes $\rightarrow$ words $\rightarrow$ Transcripts

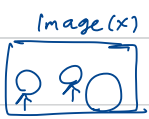
X (audio) $\rightarrow$ [end-to-end d.L] $\rightarrow$ Transcripts

depending on the data set size small : Traditional may work better

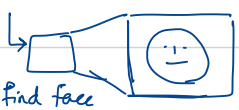
bigger : end-to-end system shines!

example on Intermediate solution

face recognition system



N.N → Identity (doesn't work well now).



N.N → Identity (works better).

→ face recognition N.N takes two images and tells are the same person or not. (This Task needs 100,Ms of images).

Machine Translation

english → text analysis ---- → french

english → Trained on Pair of (F, E) sentence → french

Estimating child age.

Image → extract bones → estimate age based on statistic
(does not require alot data).

Image → age (does not work now since There are not enough data)

Pros and Cons

Pros: ①. let the data speak.

if big enough N.N is trained, more probable to learn all representation necessary without human prejudjment.

②. less hand-designing of components needed.

Cons:

① May need large amount of data.

②. excludes potentially useful hand desiged components.

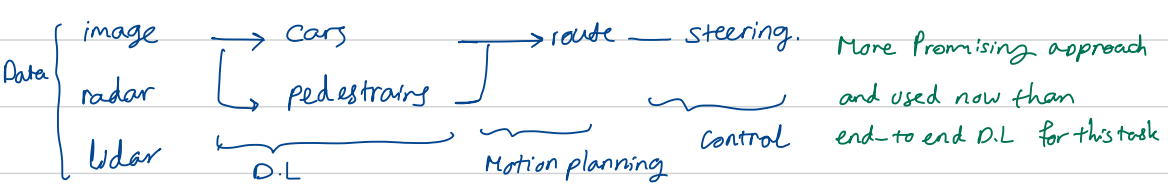
Data
hand-desiged features (Injecting human knowledge)

if less data exist, hand-desiged features are useful

→ Applying end-to-end deep learning.

→ Do you have enough data to learn a function of the complexity needed to map X to Y?

Example of self-driving car:



→ Use D.L to learn individual components.

→ Carefully choose X → Y depending what task you can get data for.