(x, coefficients y)

Structuring ML Projects:

Introduction to ML strategy.

* orthogonalization
  - what to tune in order to acheive one effect, people are clear about

~~measure~~ tune different parameters differently

chain of assumptions in ML:

different tuners for different assumptions, independent of each other

- Fit training set well on cost func^n. ⇄ bigger N/W; Better opt. algo
  ↓
- Fit dev set well on cost func ⇄ Regularization; Bigger training set
  ↓
- Fit test set well on cost func^n. ⇄ bigger dev set
  ↓
- Performs well in real word. ⇄ change dev set or cost fun

⇓

orthogonalisation.

* Single number evaluation metric.

idea → code → exp → idea

	Precision	recall
A	95%	90%
B	98%	85%

Precision → of examples as cats, what % are cats

recall → what % of actual cats recognise

F1 Score = "Avg" of precision P & recall R

$\left(\frac{2}{\frac{1}{P}+\frac{1}{R}}\right.$ "Harmonic mean")

Scanned by CamScanner

* satisficing & optimizing metric :

	Accuracy	Run time
A	90%	80ms
B	92%	95ms
C	95%	1500 ms

Accuracy → optimizing
Run time → satisficing

let cost = accuracy - 0.5 x running time

maximise accuracy

$t \le 100ms$

⇓

has to be satisfied.

N metrics : 1 optimizing, N-1 satisficing } best.

wakewords/triggerwords

Alexa, ok google, Hey siri , ...

accuracy ⟶ maximize

false positive ⟶ atmost ≤ 1 per 24 hrs. (satisficing)

* Train , dev, test set distributions :

dev/test

⇓

develop set

holdout cross validⁿ set

Regions :

US
UK
Other Europe
South America } dev

India
China
Other Asia
Australia } Test.

dev set + metric

due to diff in data, optimisation for 1 can degrade other.

→ Randomly shuffle data, so that better distribution exists.

- Choose dev set and test set to reflect data you expect to get in future & consider important to do well on

dev & test set shud be from same distribution.

Train of dev & test set :

How long should they be.

- old way of splitting data.

70%	30%
Train	Test

reasonable if ~ 1000 examples.

60%	20%	20%
Train	Dev	Test

- million T.examples

98%	1%	1%
TS	DS	TeS

set big enough to give high confidence in overall performance

Some appln : Train + Dev, no Test set.

* **When to change dev / test set metrics :**

Metric : classifn error

A : 3% error → has porn (better on eval metric, but worse)

B : 5% error → doesn't have porn (poor, but good as no porn)

metric + eval → A

user → B

∴ change evaluation metric.

$$\boxed{\text{Error} = \frac{1}{m_{dev}} \sum_{i=1}^{m_{dev}} w^{(i)} I\{ \underbrace{y_{pred}^{(i)} \neq y^{(i)}}_{(0/1)} \}}$$

$$w^{(i)} = \begin{cases} 1 ; & x^{(i)} \text{ not porn} \\ 100 ; & x^{(i)} \text{ porn} \end{cases}$$

∴ when weights introduced.

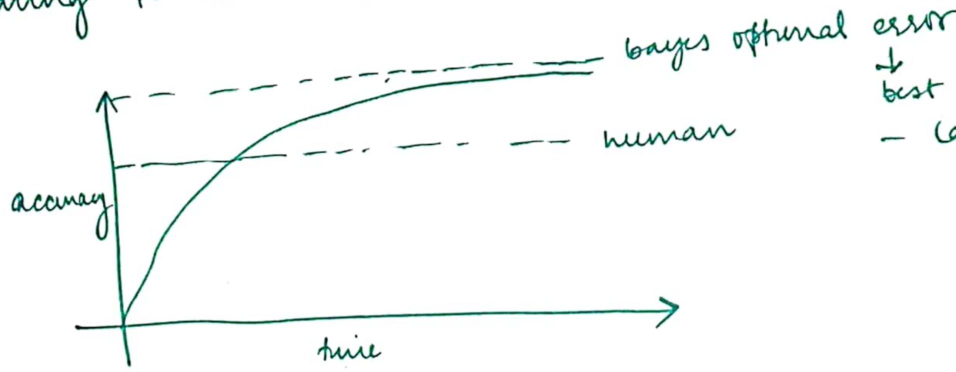
new metric defined.

$$\boxed{\text{Error} : \frac{1}{\sum w^{(i)}} \sum_{i=1}^{m_{dev}} w^{(i)} I\{ y_{pred}^{(i)} \neq y^{(i)} \}}$$

$$\boxed{J = \frac{1}{\sum w^{(i)}} \sum_{i=1}^{m} w^{(i)} L(\hat{y}^{(i)}, y^{(i)})}$$

- if doing well on metric + dev/test set doesn't correspond on doing well on your application, change your metric.

⊕ Comparing to human level performance :-



bayes optimal error → best possible error.
- can never be surpassed

X → y
audio → transcript.

- progress slows down after human level performance.
- progress never exceeds bayes optimal error.

* Avoidable bias

Human	1%	↕ to reduce
Training error	8%	
Dev error	10%	

Focus on fixing bias.

7.5%	↕ Avoidable bias
8%	↕ to reduce variance.
10%	

Focus on variance + by regularizⁿ, more training data.

Human level error as a proxy for bayes error.

* Understanding human level performance :

Ex: Typical human --- 3% error
" Doctor --- 1% error
. experienced doctor --- 0.7% error
Team of exp. doctors --- 0.5%.

~~what is~~ "human level" → Baye's error ≤ 0.5%.
error : 0.5%

Human (proxy for bayes error) = 1%, 0.7%, 0.5%.

Avoidable bias — Training error = 5%. Focus on bias redⁿ:

Variance — Dev error = 6%.

Human error = 0%. (bayes error) 1%. 0.7%. 0.5%. TE: 1%. DS: 5%.

Human Available bias becomes bias.

↓se the variance (regulariz", more Train data)

Surpassing human level performance:-

Team of humans 0.5%.

One human A B 1%.

0.6%.

TE 0.8%.

DE

0.5% → progress options less clear.

1%

0.3%

0.4%

⇓

better performance than humans.

online advert, product recommendⁿ, logistics, loan approvals.

Improving model performance:

1. Fit TS well to acheive low avoidable bias
2. TS performance generalises well to dev & test set to Variance

Human level

Avoidable bias → { Train bigger model; Train longer / better optimisⁿ algo; NN architecture, ...

Training error

Variance → { More data; Regularisation.

Dev error

Error analysis:

* Carrying out error analysis:

Ex: 90% accuracy
10% error.

Error analysis: "gives a ceiling as to how best can algo work"

manually takes 5-10%.

- get ~100 mislabeled dev set examples
- count how many dogs.

out of 100 50% → dog 10% → 5%.

Evaluate multiple ideas in ||el :

Img	Dog	Cat	Great Cats	Blurry	insta	Comments
1	✓			✓		Pitbull.
2			✓	✓		Rain at zoo
3						
⋮						⋮
% of total	8%		43% ...	61%	12%	

can improve performances through these.

- find a set of mislabled examples in dev set & look for false +ve & false -ve.

* cleaning up mislabeled data :
- if errors are random, then its OK (% not too high)
- robust to random errors
- Systematic label errors are bad.

⇒ Add one "incorrect label examples" column.
"6%."

if makes significant difference, then fix the error, else don't bother.

Ex: overall dev set error 10%.
incorr labls = 6% of 10% = 0.6%.
other : 9.4%.
∴ don't fix labels.

Correcting dev set examples (which r incorrect)
- dev & test set should be processed similarly from same distribution (apply same process to both)
- consider examples that algo got right, as well as wrong. (isn't always done)
- Train & dev/test may come from different distribution.

* Build your first system quickly, then iterate :
- set up dev/test set & metric
- Build initial system quickly.
- Use Bias/variance analysis & error analysis to prioritise next step.

→ Build first system quickly, then iterate.

Scanned by CamScanner

Training & testing on different distributions:

Ex: webpage : 200,000 (good qlty) } mix them
phone app: 10,000 (bad qlty) } 210,000
↓
shuffle,
split them randomly
in train, dev, test set.

Option 1 Ex: dev – 2500
test – 2500

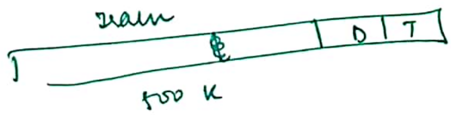
don't use this, as algo will optimise for web images most of the time.
Instead,

Training 200,000 → web + 5000 → app.
dev/test: all mobile app.
advantage: aiming the tgt.
as we have to make mobile app
∴ this is better.

· Speech recog examples.

Training
Purchased data
smart speaker control
Voice keyboard.
...
500,000 occurrences.

Dev/Test
speech activated }
Rearview mirror }
→ 20000



or

↓
from
rearview
mirror → also a good thing.

* Bias and Variance with mismatched data distribution.

Ex: Assume humans ≈ 0%.
Train error : 1% ↓ distribution is difference ∴ difficult to analyse error.
Dev error : 10%

define ↓ Training-dev set : Same dist as Tr.set but not used for training.

randomly shuffle Tr.S and
carve out small
tr.dev set.

Ex: Training error = 1%
Training-dev err = 9%
Dev error = 10%

↕ variance prob (as same distrib.)

↕ 0% (Bayes)
10%
11%
12%

↕ Avoidable bias prob.

Tr 1%
Tr-Dev 1.5%
Dev 10%

↕ low variance prob
↕ data mismatch prob.

0%
10%
11%
20%

↕ Avoidable bias as well as
↕ large data mismatch.

Thus,

Human level error	4%	↕ Avoidable bias
Tr set error	7%	↕ variance
Tr-Dev set error.	10%	↕ mismatch
Dev error	12%	↕ degree of overfitting
Test error	12%	

{ → same distribution.

ut.

Human	---	4%	
Tr set	---	7%	
Tr-Dev	---	10%	
Dev	---	6%	if dev/test set examples are much easier.
Test	---	6%	

more general formulation: (Rear view mirror)

	general speech recog.	Rearview mirror speech data
Human level	4%	6%
	↕ avoidable bias	
Error on trained exa	"Tr error" 7%	6%
	↕ variance	
Error on not trained on ex.	"Tr dev set" 10%	"Dev/Test error" 6%

data mismatch

Already doing quite well.

* Addressing data mismatch:
- carry out manual error analysis to try to understand difference between training & dev/test set.
- Make tr data more similar; or collect more data similar to dev/test set.

Artificial data synthesis :

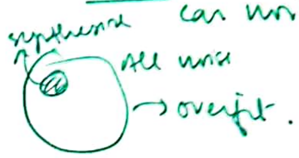
"The quick brown fox jumps over the lazy dog" + " car noise " = synthesised in-car audio .

↑ 10,000 hrs

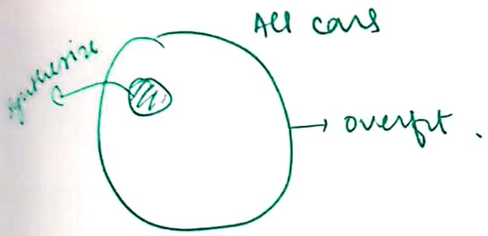
↑ 1 hour

opt 1 : 1 hr, 10 000 times repeated. learning algo can overfit to 1 hr car noise.

sol^n : use 10,000 hrs car noise (possible) .



car recognition

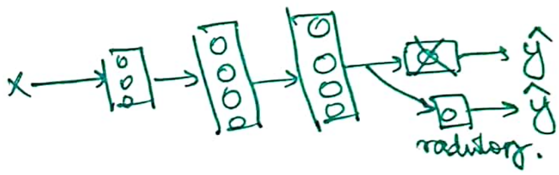


if data mismatch, do error analysis, get more Tr. data, etc...

Learning from multiple tasks :

* Transfer learning :-

Ex: Trained on img recog.



↓ change last layer

radiology .

↘ (x, y) → radiology img
→ diagnosis .

$w^{[L]}$, $b^{[L]}$ retrain the NN.

⇓

retrain $w^{[L]}$ & $b^{[L]}$ or

if enough data , retrain whole model

→ have more radiology imgs

→ pretraining NN .

can add several new layers instead of L-1 layer & retraining them .

When Transfer learning makes sense?

Transfer A → B

* A & B have same i/p x .
* You have lot more data for A than B .
* low level features of B can be learned from A .

Scanned by CamScanner

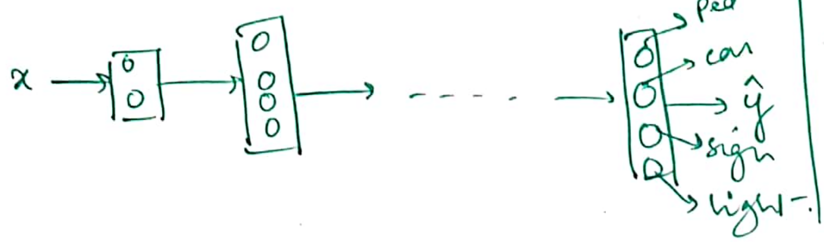
* Multi-task learning:

Ex: autonomous driving detector.

detect	$y^{(i)}$
Peds	0
cars	1
signs	1
lights.	0
⋮	⋮

$\Big\}$ (4,1).

$$Y = \begin{bmatrix} | & | & & | \\ y^{(1)} & y^{(2)} & \cdots & y^{(m)} \\ | & | & & | \end{bmatrix}$$



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

Sum only over val = 0/1

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

unlike softmax regr. one img can have multiple label.

→ Multitask learning as 4 things are done by 1 NN.

When makes sense?

- Training for shared low level features.
- Usually: Amt of data for each task is quite similar.
- Can train a big enough NN to do well on all tasks.

→ if NN isn't big enough, then multitask learning can hurt model performance.

* End to end deep learning:

Takes multiple stages and replaces with a NN.

- Speech recog example

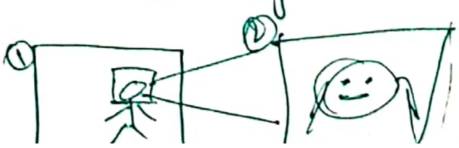
x audio $\xrightarrow{MFCC}$ features $\xrightarrow{ML}$ Phonemes → words → transcript y

audio ——————————→ transcript.

3000 h ✓ 10,000 h ⋮ 100,000 h

- Face recognition

Img(x) → y. Identity



Identify pic with body } same person?

∴ lot of data for 2 subtasks.

More examples.

- Machine translation

(x, y)
Eng   French.

English → text analysis → - - - → french
English ——————————→ french.

Estimating child's age : Image $\xrightarrow{①}$ bones $\xrightarrow{②}$ age.
Image ——————→ age.

* whether to use end to end deep learning :

- Pros & cons :

Pros : $x \to y$

- if big enough NN trained, it will figure it out.
- captures statistics in data.

Ex: phonemes क, आ, ट for cat. can be omitted

- less hand-design of comp. needed

Cons :

- need a lot of data for $x \to y$ (x, y)
  i/p end o/p end.
- Excludes useful hand-designed components
  by humans, can be useful or harmful.

Ex: autonomous car :

image $\xrightarrow{DL}$ cars → route $\xrightarrow{control}$ steering
radar $\xrightarrow{DL}$ peds   motion planning
⋮

- use DL to learn individual comp.
- Carefully chose X → Y depending what tasks you chose data for.