Structuring ML Projects

Why ML Strategy? → save time (and not lose months...)

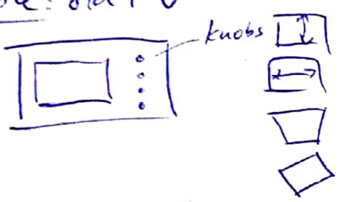
→ Cat classifier → 90% Accuracy → want to be better!
↳ Ideas how to improve

- more data
- more diverse training set
- train longer with GD
- try Adam instead of GD
- try bigger network
- try smaller network
- try dropout
- try L2 regularization
- NN architecture
  - activation functions
  - # hidden units

} how can we measure our improvement?

Orthogonalization → "which HP to tune to get which effect?"

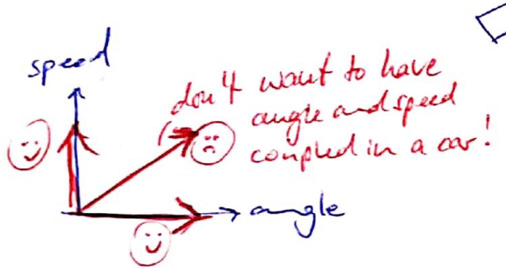
example: old TV



or car
steering
accelerating
braking

} we want each "knob" or steering feature to have an isolated effect (on the TV's picture or on the car's steering angle (or decoupled effect)
↳ Orthogonalization!

90°

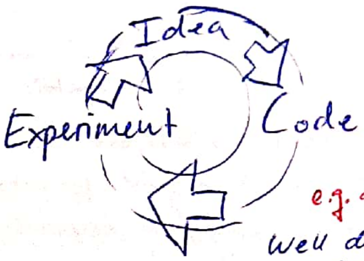


Chain of assumptions in ML (for a supervised system) (Orthogonalization process)

1) Fit training set well on cost function (≈ human-level performance) (want one knob to adjust) → tune algorithm well to fit on training set → bigger NN, better optimizer...
↓
2) Fit dev set well on cost function → set of knobs: → Regularization → Bigger training set
↓
3) Fit test set well on cost function → Bigger dev set (overtuned ~~the~~ former dev set)
↓
4) Performs well in real world → change either dev set or cost function

→ not use early stopping → affecting multiple things at the same time

Using a single number evaluation metric



what % was classified correctly? (of examples)
→ what % is actually true, will be correctly recognized by our classifier

Classifier	Precision	Recall
A	95%	90%
B	98%	85%

→ often a trade-off

e.g. after HP change

→ | F1 Score |
|---|
| 92.4% |
| 91.0% |

≈ average of P. and R.

$$F1 = \left(\frac{2}{\frac{1}{P}+\frac{1}{R}}\right)$$ "Harmonic mean"

normal average is also quick to evaluate

Well defined Dev Set + Single real number eval metric
↳ speeds up iterative process of ML!

Satisficing and Optimizing metric → When a single value evaluation metric is not possible

ex1

optimizing metric — satisficing metric

Classifier	Acc.	Run time
A	90%	80ms
B	92%	95ms
C	95%	1500ms

- maximize accuracy
- subject to run time ≤ 100 ms

satisficing has only to do sufficiently

N metrics: 1 optimizing, N-1 satisficing

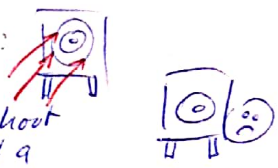
ex2: wake/trigger words
→ Alexa, OK Google, "Hey Siri"...
Interested in accuracy
• # false positives (wake up without command)

maximize accuracy — optimizing metric
subject to ≤ 1 false positive / 24 hours — satisficing metric

Train/dev/test distributions

classification dev/test sets
(hold out cross validation set)

Analogy:



learn to shoot arrows at a target, → then ask model to "shoot" at a different target with same accuracy

ex1:
Regions
- US
- UK
- oth. Europe
- South America
} Dev
- India
- China
- oth. Asia
- Australia
} Test

bad because they are from different distributions!!
Teams lose time by doing well on dev set only to realize that test set will behave completely different

Solution → randomly shuffle into dev/test sets!
↳ same distribution, from mixed together

ex2:
loan approval: dev set was optimized to medium income ZIP codes
↳ was then tested on low income ZIP codes

↳ Guideline — 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.

Size of the dev and test sets

70% train | 30% test
60% train | 20% dev | 20% test
} # m 100 – 10.000

Tr 98% | DT 1% 1% — # m ~1.000.000

Size of test set
→ big enough to give high confidence in overall performance

Sometimes OK to have only train + dev set and no test set

When to change dev/test sets and metrics

ex: Metric: classification error
Algorithm A: 3% error → let through pornographic images
" B: 5% error → lets through no porno images
→ B is preferred although has higher error

adjust metric

$$\text{Error} = \frac{1}{\sum_i w^{(i)}} \sum_{i=1}^{m_{dev}} w^{(i)} \mathcal{L}\{ y_{pred}^{(i)} \neq y^{(i)} \}$$

"weight" $w^{(i)} \begin{cases} 1 & \text{if } x^{(i)} \text{ is no-porn} \\ 10\text{-}100 & \text{if } x^{(i)} \text{ is porn} \end{cases}$

Orthogonalization

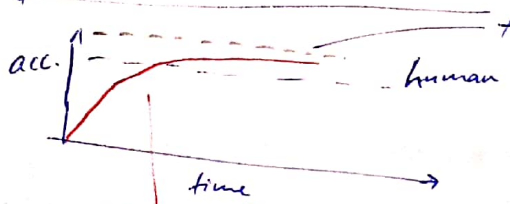
→ So far only how to define a metric to evaluate classifier

→ Worry separately about how to do well on this metric

If doing well on metric + dev/test set doesn't correspond to doing well on your application, change your metric and/or dev/test set
sharp images for training/dev/test but user images are blurry (cat classifier)

DL spec // course III // week 1

Why human-level performance?



"Bayes optimal error"
→ best possible error

progress fast until surpass human-level performance

Why compare to human-level performance

- Humans are quite good at a lot of tasks. As long as ML is worse we can:
  - → get labeled data from humans
  - → gain insight from manual error analysis: why did a person get this right?
  - → Better analysis of bias/variance

Avoidable Bias

Cat classifier

Humans 1%
train error 8%
dev error 10%
} try ~~[illegible]~~ reducing bias

focus on bias

different case

human 7.5% (maybe pics blurry)
8% — 0.5% avoidable bias!
10% ↕ 2% variance

focus on variance here

variance reduction techniques such as regularization or getting more data

Human level error as a proxy (estimate) for Bayes error

$\boxed{\text{diff: Bayes} - \text{train-error} = \text{"Avoidable bias"}}$

Understanding human-level performance

HLE as proxy for Bayes error

Medical image classification example:

	error
normal person	3%
" doctor	1%
experienced doctor	0.7%
team of experienc. doctors	0.5%

What is → "human-level" error?

Bayes error ≤ 0.5%

Human (proxy for Bayes)	1% / 0.7% / 0.5%	} 4-4.5%	→ higher, so focus on bias reduction techniques → bigger NN
Training error	5%	} 1%	
Dev error	6%		

avoidable bias
variance

H.er.	1% / 0.7% / 0.5%	} 0.1%
tr. er	1%	} 4%
Dev	5%	

→ use variance reduction techniques! (regularization or bigger training examples)

important case

here it matters which human error we take

Human error	0.5%	} 0.2% → twice as big! ↳ so we know that we actually can do better!
tr. err.	0.7%	} 0.1%
dev. err.	0.8%	

→ works until surpassing human-level performance

Surpassing human-level performance

Team humans	0.5%	avoidable bias 0.2%	0.5%
one human	1%		1%
training error	0.6%	variance 0.2%	0.3% ← does this mean we overfitted the model or are we actually above/better than human error?
dev error	0.8%		0.4%

examples where ML significantly > human level performance

- Online advertising
- Product recommendations
- Logistics (transit time prediction)
- Loan approvals

learned from structured data (not natural perception) (Lots of data)

humans are very good at natural perception:
- speech recog.
- image "
- medical
...

Improving your model performance

2 fundamental assumptions of supervised learning

1. You can fit training set well → low avoidable bias

2. training set performance generalizes pretty well to dev/test set
→ Variance not too bad

Human-level
↕ avoidable bias → bigger model; train longer / better optimization – momentum, RMSprop, Adam; NN architecture / HP search, try others like RNN / CNN

training-error
↕ variance

dev-error → more data; Regularization → L2, dropout, data augmentation; NN architecture / HP search

---

training set 10'000'000 images $y = \begin{cases} 0 & \text{no bird} \\ 1 & \text{bird} \end{cases}$

→ what is evaluation metric?

→ how structure data?