Why machine learning strategy.

Motivating e.g. You are working on your cat classifier.

Ideas:

- Collect more data
- Collect more diverse training set.
- Train algorithm longer with gradient descent.
- Try Adam instead of GD.
- Try bigger/smaller network.
- Try dropout.
- Add L_2 regularization.
- Network architecture.
  - ☆ Activation func.
  - ☆ # Hidden units.

You have a lot of ideas to try with.

So what is the strategy to analyze your network. And where to start modify your network.

Orthogonalization.

e.g: learning to drive a car. 3 controls: Steering, Accelerating. Break.

Each of them has only one functionality.

e.g: If we have controls such as: 0.3 x angel - 0.8 speed ...

(Hard to manipulate.)

- Chain of assumption in ML.

  - Fit training set well on cost func. ⇒ Bigger Network, Adam .....

Fit dev set well on dev set. ⇒ Regularization. Bigger training set. .....

Fit test set well on test set. ⇒ Bigger dev set.

Perform well in real world. ⇒ Change dev set or cost func.

* When I try to tune the network. Maybe not use earlystopping. Because it both affect train / dev set. Not ORTHOGONAL.

Single number evaluation metric.

There always a trade-off between.

e.g. Classifier.	Precision	Recall.	F1-score. → "Average" of P and R
A	95%	90%	92.4%
B	98%	85%	91.0%

$\frac{2}{\frac{1}{P}+\frac{1}{R}}$ "Harmonic mean".

Precision: Of all the e.g. your classifiers recognize as cats. what percentage are actually cats.

Recall: What percentage of cats your classifier recognize as cats.

Now A better with recall. B better with precision. But A has better F1-score.

* Standard strategy: Dev set + Single number evaluation metric.

Another e.g. Classifier with error rate.

Algorithm	US	China.	India	Other	Average
A	3%	7%	4%	9%	6%
B	5%	6%	5%	10%.	6.5%.
C					
D		....			
E					
F					

Maybe pick the lowest average error rate.

Satisficing and optimizing metrics.

Cat classification e.g.

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

You might want to maximize acc.
Subject to running time ≤ 100 ms.
Accuracy: Optimizing.
Running time: Satisficing.
Under this. we think B is the "BEST" classifier.

N metric: 1 optimizing. N-1 Satisficing.

Train/Dev/Test Distribution.

same distr.

Guideline: Choose a dev set and test set to reflect data you expect to get in the future and consider important to do well on.

When to change dev/test set and metrics.

Cat dataset e.g.

Metric: classification error.

A: 3% error → include some pornographic. (unacceptable).

B: 1% error.

From user view maybe. B is better. because algorithm A allow some unacceptable graphics to go through.

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

way to including e.g. pornographic. $\frac{1}{m_{dev}} \sum_{i=1}^{m_{dev}} w^{(i)} 1\{y_{pred} \neq y\}$.

$$w^{(i)} = \begin{cases} 1 & \text{if } x^{(i)} \text{ is non-porn} \\ 10 & \text{if } x^{(i)} \text{ is porn.} \end{cases}$$

Two steps: ① Define a target. ② How to hit the target.

Another e.g.

A: 3% error

B: 5% error.

When you deploy the model. you find that B actually doing better.

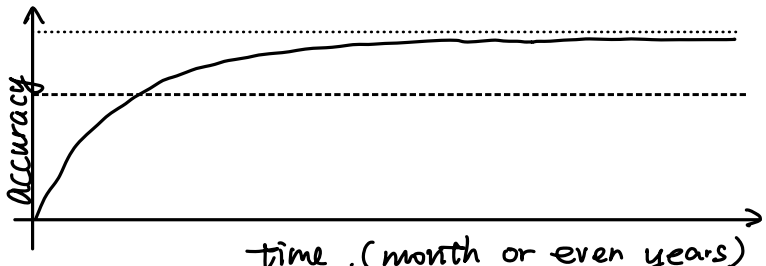
Dev/test set.

e.g. Testing with high resolution. User upload low resolution

⇛ Change your metric and/or dev/test set.

Why human-level performance.

e.g: Some so blur images. Can't be recognized using anything.



Bayes optimal error. Best possible error.

human level performance.

So fast before human performance.

Slow after surpacing human.

Human are quite good at a lot of tasks. So long as ML is worse than humans. you can: (Before surpassing human-level performance).

- Get labeled data from human.
- Gain insight from manual error analysis.
  - Why did a person get this right?
- Better analysis of bias/variance.

Avoidable bias.

Cat e.g.

Human level error: 1%.

Training error: 8%

Dev error: 10%.

Too large.

Focus on reducing bias.

e.g. Too blurry.

7.5%.

Maybe we are just doing fine on training.

8%

10%

Focusing reducing variance.

e.g. Regularization.

Avoidable bias

Variance

Use human-level error as a proxy for Bayes error.

Difference between bayes (human) error and training error: Avoidable bias

Surpassing Human Level Performance.

If your algorithm surpass the human level performance.

Then it is hard to explain. hard to improve your performance.

Problems where ML significantly surpass human.

- Online advertising.
- Product Rec.
- Logistics.
- Loan approval.

  (Structural Data. Not natural perceptron, Lots of data).

- Speech recognition. } Human are good at these.
- Image recog. } Already surpass. but really hard.

Mismatched training, dev, and test data.

Say a mobile app. Data from websites. (high resolution professional).

Data from mobile app (low resolution. random image).

Say: 200,000 images from web. 10,000 from mobile.

Your NN work well on web image. you want to further focus on these mobile app

Option ①. Combine them all together, then split into train. dev. test set.

adv: All from same distribution.

disadv: On expectation, only a small part in dev and test are from

the mobile app. (Focusing on the wrong dataset).

Option ②. Train using 205,000. All 200,000 from web. and add 5000 from app. And dev & test only include images from the app.

adv: Now you are aiming the data you want.

disadv: train & test diff distribution.

In general. ② will give better result.

Bias and Variance of mismatched data distribution.

Cat classifier example.

Assume human error ≈ 0%.

Training error: ≈ 1%

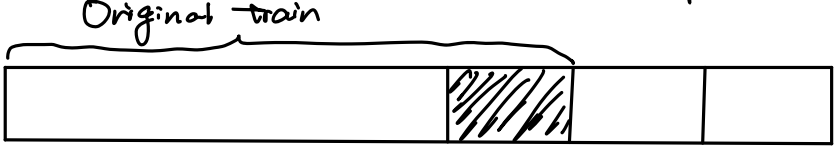
Dev error: ≈ 10%.

From train to dev: ① The algorithm see diff. data

② The dist. of two sets of data are diff.

We create a new set: Training-dev set. Same dist. as train set.

but not used for training.



e.g.

Train error ≈ 1%	1%	Human: 0%.	Human 0%
Train-Dev error ≈ 9%	1.5%	10%	10% ⇒ bias problem
Dev error ≈ 10%.	10%	11%	11% ⇒ Both bias & data mismatch problem
		12%	20%.

⇒ Data mismatch problem.

⇓

Variance problem, because they are from

the same distribution.

Key quantity:

① Human level error: } Avoidable bias problem
② Training set error: }
③ Training-dev set error: } Variance problem
④ Dev error: } Data mismatch problem
⑤ Test error: } degree of overfitting the dev set.

Addressing data mismatch.

→ Carry out manual error analysis to try to understand diff. between train & dev/test set.

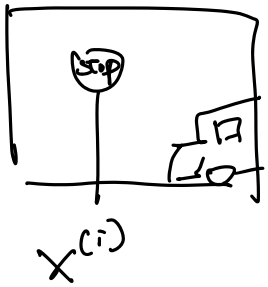
e.g. Noise.

→ Make train data more similar. Adding data similar to dev/test set.

Multi-task learning.

e.g. Autonomous driving.

Need to recognize: Pedestrian.
Cars
Stop signs.
Traffic lights.



$x^{(i)}$

$$y^{(i)} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$ → whether the thing exist

$$Y = \left[y^{(1)}, y^{(2)}, \cdots, y^{(m)} \right]$$

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

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

(4.1)

usual logistic loss

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

if the label sometimes missing.

Just ignore the term.

Unlike softmax.

One image can have multiple label.

Multi-task learning.

When multi-task learning make sense.

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

→ Usually: Amount of data you have for each task is quite similar

e.g. in transfer learning. we also need similar tasks.

→ Can train a big enough NN to do well on all the tasks.

*Alternative way: Train multi nns, each on only 1 task.

End to end deep learning.

What is that?

e.g. Speech recog example.

x → audio —MFCC→ features —ML→ Phonemes ("c a t") (ke a te) → words → transcript → y

end2end: x: audio ——————————→ y: transcript.

Currently. 3000h data works well for multi step work

Only large amount of data let end-to-end work better

Whether to use end to end learning.

Pros:

→ Let the data speaks.

→ Less hand-designing of components needed.

Cons:

→ May need large amount of data.

→ Excludes potential useful hand-designed component.

↳ human can inject useful knowledge.

Applying end-to-end learning.

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