Course 3: Structuring ML Projects

Week 1

Why ML Strategy?

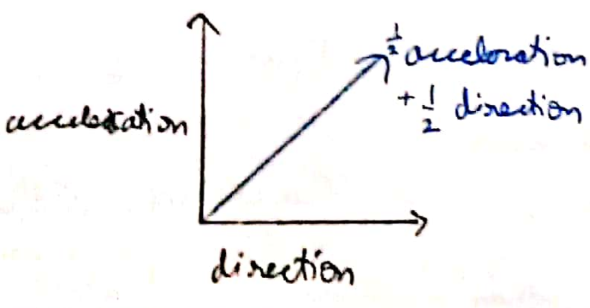
You may have different ideas on how to optimise your algorithm era: collect more data, try adam, add L_2 regularization, etc.

You may spend 6 months on an idea (like collecting more data) only to later realise that it doesn't affect the algo much.

So we will learn strategies to choose which idea to spend time on.

Orthogonilization

Orthogonilization is having controls that only control a single factor (like in a car steering controls direction, accelerator controls acceleration and brake controls deceleration). Imagine having a joystick that controls acceleration and direction, this'll be harder to control.



Orthogonal means 90° to each other

So how does this pertain to ml?

You try to orthogonalise (Use one idea to fix one problem ~~cost~~ while building your model.

Chain of assumption in ML:

→ Fit training set well on cost function

If it doesn't, try a larger network, adam...

→ Fit dev set well on cost function

If it doesn't, try regularisation, bigger train set

→ Fit test set well on cost function

If it doesn't, try bigger dev ~~to~~ set

→ Performs well in real world

If it doesn't, try to change dev set or cost function

Exa: Using early stopping isn't recommended because ~~it~~ it affects step 1 & 2 - ie its like having a joystick that controls acceleration and decelaration

Scanned with CamScanner

Single Number Evaluation metric

Precision - of all the examples recognised as cats, what % actually are cats

Recall - what % of actual cats are correctly recognised

Classifier	Precision (P)	Recall (R)
A	95%	90%
B	98%	85%

Since there are 2 evaluation metrics, its hard to choose which is better. So we use F1 score which is the harmonic mean of P and R.

$$\text{F1 score} = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

~~Satisfyi~~ Satisficing and Optimizing metrics

If we have N metrics, 1 metric will be an optimizing metric while N-1 will be satisficing (needs to just satisfy a condition)

classifier	Accuracy	Running Time
A	90%	80ms
B	92%	95ms
C	95%	1500 ms

maximise accuracy - optimizing

running time $\leq$ 100ms - satisficing

As long as running time $\leq$ 100ms we consider it and choose the one with best accuracy

Scanned with CamScanner

Train / dev / test data distribution

1. Make sure the dev and test sets come from the same distribution
   For example, if you have data from the following regions,
   - US
   - UK
   - Other Europe
   - South America

   ~~Dev~~

   - India
   - China
   - Other Asia
   - Australia

   ~~Test~~

   Don't divide them like this. Instead randomise all the data and then divide into dev and train sets

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

3. Set your test set to be big enough to give high confidence in the overall performance of your system.
   In the olden days when data size was small ($\leq$ 10,000), then they would divide as 70% - 30% or 60%-30% 20%.. Now since we have more data (lets say 1 million), then 98% -1% -1% is still good cuz 1% is 10,000.

Scanned with CamScanner

n. If doing well on your metric + dev/test set doesn't correspond to doing well on your application, change your metric and/or dev/train set

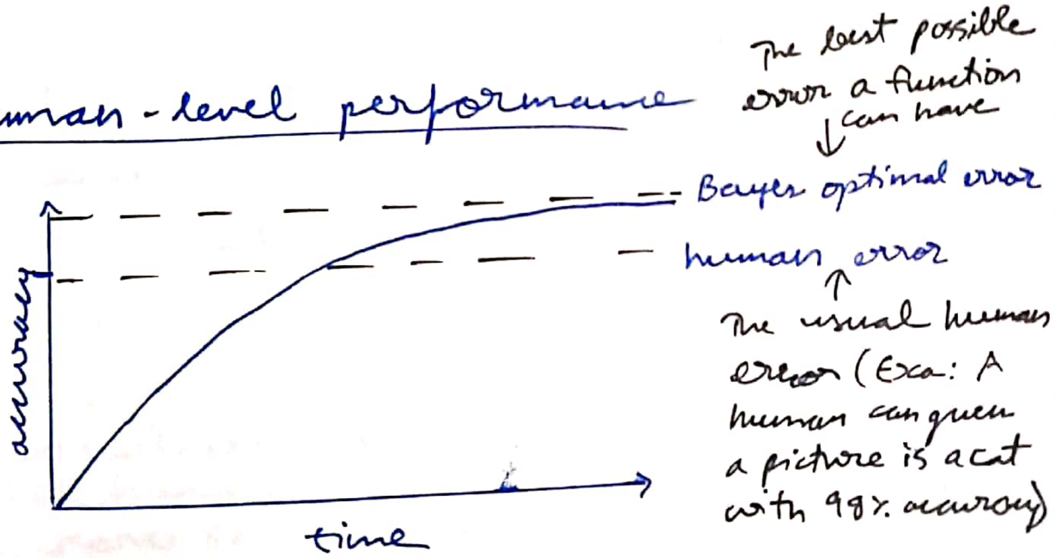
For example if:

Algorithm A: 3% error - allows porn

Algorithm B: 5% error - no porn

The old metric will choose A even though the users won't like it. So we need to use a new metric that chooses B (by punishing the algorithm if it classifies porn as cat.

Why human-level performance



As long as the accuracy hasn't surpassed the human error, there is room to improve it:

→ 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

Consider 2 examples:

	I	II
Human (≈ Bayes) error	1%	7.5%
	7%	0.5% Avoidable bias
Training error	8%	8%
	2%	2% Variance
Dev error	10%	10%
	Focus on bias	Focus on variance

- The human level error is a proxy for bayes error (since in cases like computer vision, a human does very well in recognising images)
- The difference between the human and training error is avoidable bias. In I, since avoidable bias is more, we can use a more complicated network
- The difference between training & dev error is variance. In II, since variance is more, it is overfitting. Regularisation can help.

Example:

Consider a medical image classification example:

a) Typical human .... 3% error

b) Typical doctor .... 1% error

c) Experienced doctor .... 0.7% error

d) Team of experienced doctors .... 0.5% error

What is human level error in this case?

Scanned with CamScanner

We saw before that human level error is a proxy for bayes error (best possible error). Hence we can consider d) 0.5% error.

However if you are writing a paper or doing a project, b) 1% error can be chosen since it can then be widely used.

	I	II	III
Human error	{1%, 0.7%, 0.5%}	{1%, 0.7%, 0.5%}	{0.7%, 0.5%}
	↕ 4%, 4.5%	↕ 0%, 0.5%	↕ 0.2%, 0%
Training error	5%	1%	0.7%
	↕ 1%	↕ 1%	↕ 0.1%
Dev error	6%	5%	0.8%
	↑ Bias	↑ Variance	↑ As we near the bayes error, it gets harder to optimise (can't decide if bias or variance)

Surpassing human-level performance

Consider the example:

Team of humans - 0.5%

One human - 1%

Training error - 0.3%

Dev error - 0.4%

here its unsure whether its a bias or variance problem.

- bayes error maybe 0.1%, 0.2%, 0.3%..
- we can't ask humans to see where the algo is doing bad, since the algo is better than humans

Therefore there is very little that one can do to improve an algo after its surpassed the human level

Problems where ML significantly surpasses human-level performance:

→ Online Advertising

→ Product Recommendations

→ Logistics (predicting transit time)

→ Loan approvals

Improving your model performance

Putting together everything we have learnt this week, there are 2 fundamental assumptions of supervised learning,

1. You can fit the training set pretty well
   - Avoidable bias
2. The training set performance generalizes pretty well to the dev/test set.
   - Variance

Reducing avoidable bias & variance:

Human-level

↕ Avoidable bias →
- → Train bigger model
- → Train longer/better optimization algorithm (momentum, RMSprop, adam)
- → NN architecture/hyperparameter search

Training error

↕ variance →
- → More data
- → Regularisation (L2, dropout, data augmentation)
- → NN architecture/hyperparameter search

Dev error

Scanned with CamScanner