¿Cómo estructurar un buen proyecto de *Machine Learning*?

Anna Bosch Rue
VP Data Intelligence @ Launchmetrics
annaboschrue@gmail.com

Motivating example

90% Accuracy and you want to do better

IDEAS:

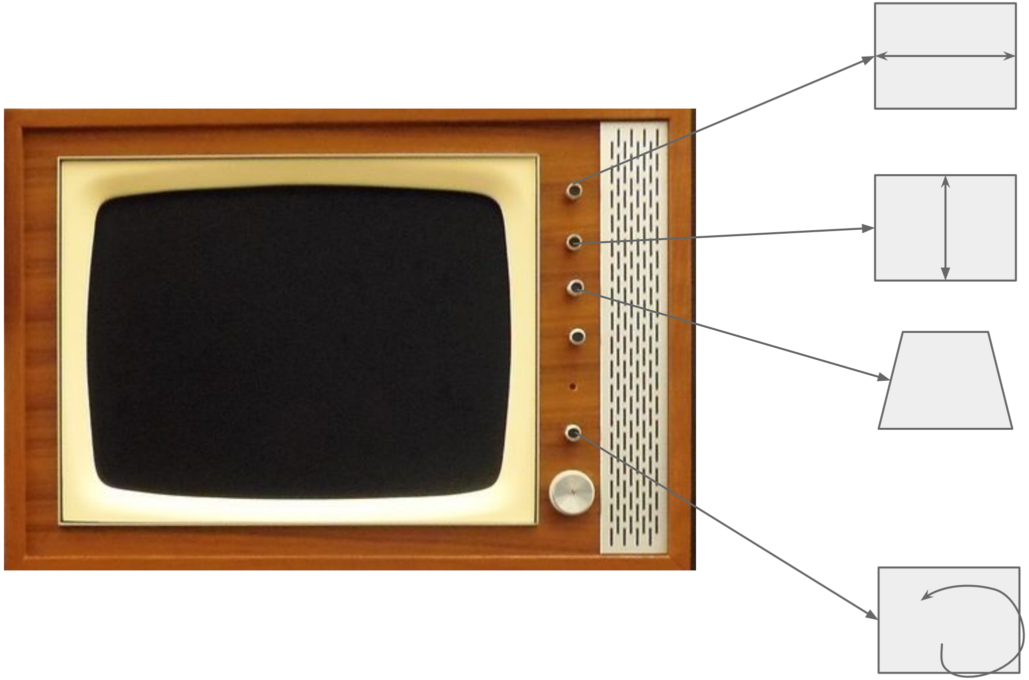
- Collect more data
- Collect more diverse training set
- Train algorithm longer with gradient descent
- Try Adam instead of gradient descent
- Try bigger network
- Try smaller network
- Try dropout
- Add regularization
- Network arquitecture:
    - Activation functions
    - # hidden units
    - ...

CONTENT

Part 1

Introduction to ML strategy

ORTHOGONALIZATION (I)



0.1 x width

+

0.3x height

-

1.7x trapez

+

0.8x rotation

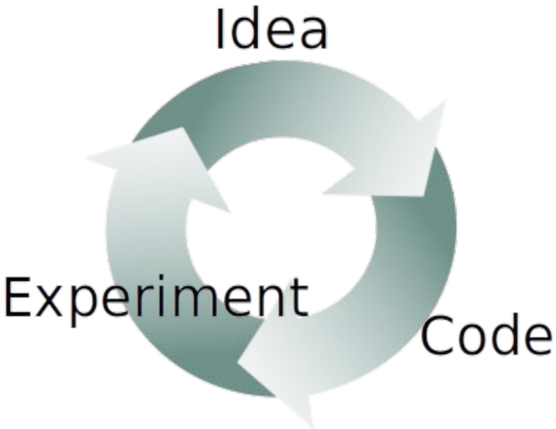
ORTHOGONALIZATION (II)

Chain of assumptions in ML

Fit **training set** well on cost function	Bigger network Adam
Fit **dev set** well on cost function	Regularization Bigger training set
Fit **test set** well on cost function	Bigger dev set
Performs well in **real world**	Change dev set Change cost function

Setting up your goal

SINGLE NUMBER EVALUATION METRIC



Classifier	Precision	Recall	F1 Score
A	95%	90%	92.4%
B	98%	85%	91.0%

Algorithm	US	China	India	Other	Average
A	3%	7%	5%	9%	6%
B	5%	6%	5%	10%	6.5%
C	2%	3%	4%	5%	3.5%
D	5%	8%	7%	2%	5.25%
E	4%	5%	2%	4%	3.75%
F	7%	11%	8%	12%	9.5%

SATISFICING AND OPTIMIZING METRIC

Classifier	Accuracy	Running time
A	90%	80ms
B	92%	95ms
C	95%	1,500ms

Cost = Accuracy - 0.5 Running time

Maximize Accuracy
Subject to Running time <= 100ms

N Metrics: 1 Optimizing
N-1 Satisficing

TRAIN/DEV/TEST DISTRIBUTIONS

REGIONS:

- US
- UK
- Other Europe
- South America
- India
- China
- Other Asia
- Australia

Dev

Test

Randomly shuffle into dev/test

True story (By Andrew NG):

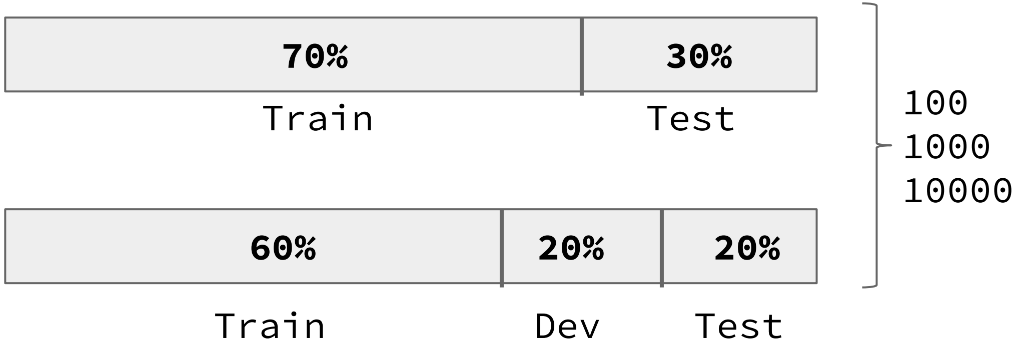
- Optimizing on dev set on loan approvals for medium income zip codes
- Tested on low income zip codes

Guideline:

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

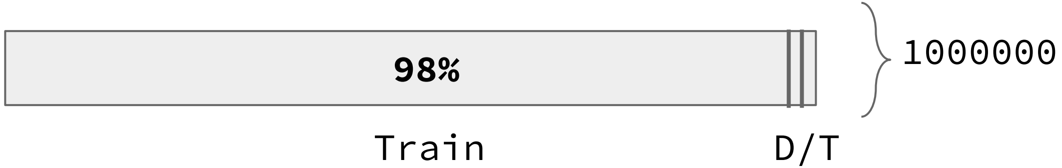
SIZE OF THE DEV & TEST SETS

OLD WAY OF SPLITTING DATA



Set your **dev set** to be big enough to detect differences in algorithm/models you're trying out

NEW WAY OF SPLITTING DATA



Set your **test set** to be big enough to give high confidence in the overall performance of your system

WHEN TO CHANGE DEV/TEST SETS AND METRICS

CAT DATASET EXAMPLES

Metric: classification error

Algorithm A: 3% → **Pornografic**
Algorithm B: 5%

Error:

$$\frac{1}{\sum_i w(i)} \cancel{\frac{1}{M_{dev}}} \sum_{i=1}^{M_{dev}} w(i)(\hat{y} - y)$$

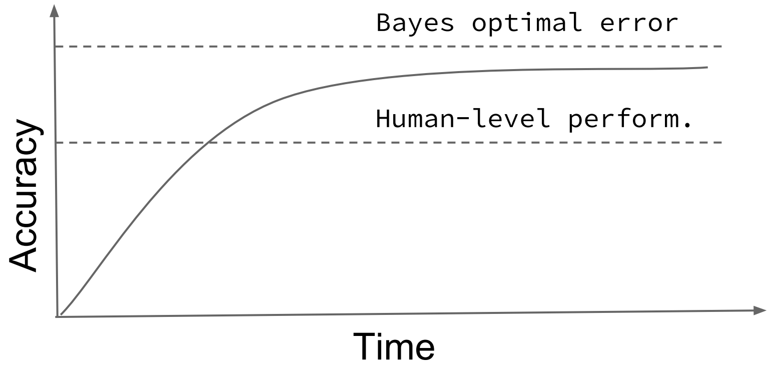
$$w^{(i)} = \begin{cases} 1 \text{ if } x(i) \text{ is non-porn} \\ 10 \text{ if } x(i) \text{ is porn} \end{cases}$$

Orthogonalization: (i)So far we've discussed how to define a metric to evaluate classifiers, (2)worry separately about how to do well on this metric

If doing well on your metric + dev/test set does not correspond to doing well on your application, change your metric and/or dev/test set

Comparing to human-level performance

WHY HUMAN-LEVEL PERFORMANCE?



Humans are quite good at a lot of tasks. So long as ML is worse than humans, you can:

- Get labeled data from humans
- Gain insight from manual error
- Better analysis of bias/variance

Comparing to human-level performance

AVOIDABLE BIAS

Use human level error as a proxy for Bayes error

BIAS & VARIANCE

Cat classification

Human level (aprox):	0%	0%	0%	0%
Training set error:	15%	1%	15%	0.5%
Dev set error:	16%	11%	30%	1%
	High bias	high variance	high bias high variance	low bias low variance

Humans (Bayes) :	1%		7.5%		
		7%		0.5%	**Avoidable bias**
Training set error:	8%		8%		
		2%		2%	**Variance**
Dev set error:	10%		10%		
	Focus on **bias**		Focus on **variance**		

UNDERSTANDING HUMAN -LEVEL PERFORMANCE (I)

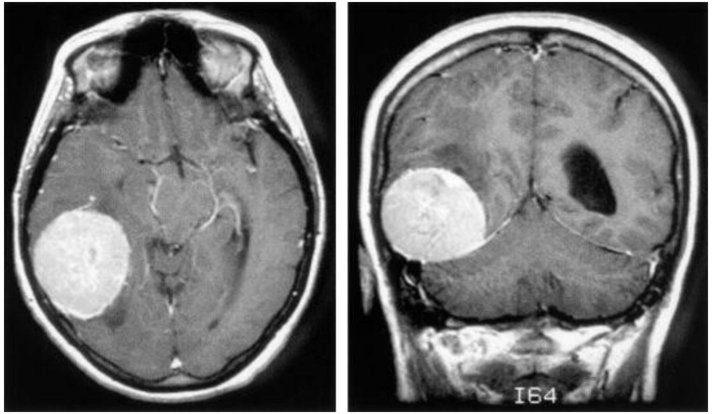
HUMAN-LEVEL ERROR AS A PROXY FOR BAYES ERROR

Medical image classification example:
Suppose:

(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 the “human-level” error?

Bayes error <= 0.5%



UNDERSTANDING HUMAN-LEVEL PERFORMANCE (II)

Human	1%/0.7%/0.5%	1%/0.7%/0.5%	0.7%/0.5%
↕ Avoidable bias	↕ 4%-4.5%	↕ 0%-0.5%	↕ 0.2%-0%
Training error	5%	1%	0.7%
↕ Variance	↕ 1%	↕ 4%	↕ 0.1%
Dev error	6%	5%	0.8%
	Bias	**Variance**	

UNDERSTANDING HUMAN-LEVEL PERFORMANCE (II)

Human

Avoidable bias

Training error

Variance

Dev error

1%/0.7%/0.5%	1%/0.7%/0.5%	0.7%/0.5%
Bias	**Variance**	

Guide:

- Compare with the optimal error
- Use the human error as a proxy
- Knowing the optimal error we will be more fast and efficient
- We will know if we need to focus on reducing bias or variance
- It will work until we reach human error
- After that it will be more difficult

SURPASSING HUMAN-LEVEL PERFORMANCE

Team of Humans		0.5%		0.5%
	0.1%		-0.2%	
One human		1%		1%
Training error		0.6%		0.3%
	0.2%		-0.1%	
Dev error		0.8%		0.4%

PROBLEMS WHERE ML SIGNIFICANTLY SURPASSES HUMAN-LEVEL PERFORMANCE:

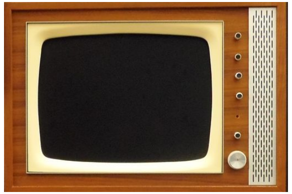
- Online advertising
- Product recommendations
- Logistics (predicting transit time)
- Loan approvals

- Speech recognition
- Some image recognition
- Medical (ECG, …)

- Structured data
- Not natural perception
- Lots of data

IMPROVING YOUR MODEL PERFORMANCE

THE TWO FUNDAMENTAL ASSUMPTIONS OF SUPERVISED LEARNING



(1) You can fit the training set pretty well
(2) The training set performance generalizes pretty well to the dev/test set

REDUCING (AVOIDABLE) BIAS & VARIANCE

Human level

Bias → Train bigger model
Train longer/better optimization algorithms (momentum, RMSprop, Adam)
NN architecture/hyperparameters search (RNN, CNN)

Training error

Variance → More data
Regularization: L2, dropout, data augmentation
NN architecture /hyperparameters search

Dev error

Q&A

Anna Bosch Rue
VP Data Intelligence @ Launchmetrics
annaboschrue@gmail.com