COURSE 3
WEEK 1

STRUCTURING MACHINE LEARNING PROJECTS

COURSE 3
WEEK 1

Why ML Strategy

When %90 accuracy is not enough for you :

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 L_2 regularization
- Network Architecture
  - Activation function
  - # hidden units
  - ...

Orthogonalization

Process of Being very clear-eyed about what to tune in order to try to achieve one effect is orthogonalization.

Fit training set well on cost function (≈ human level performance)
↓
Fit dev set well on cost function
↓
Fit test set well on cost function
↓
Performs well in real world (Happy cat pic. app users)

Solutions

bigger network
Adam opt. alg.

Regularization
Bigger training set

Bigger dev set

Change dev set or cost function

early stopping — affects — affects

! "When I train a neural network, I tend not to use early stopping."

Single Number Evaluation Metric

! "When teams are starting on a machine learning project, I often recommend that you set up a single real number evaluation metric for your problem"

One reasonable way to evaluate the performance of your classifier is to look at its precision and recall.

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

Precision → of the examples that your classifier recognizes as cats
What % actually are cats?

Recall → of all images that are really cats
What % of actual cats are correctly recognized?

Rather than using two numbers, precision and recall, to pick a classifier, you instead have to find a evaluation metric that combines precision and recall.

F1 Score

F1 score = Average of P and R

$$\frac{2}{\frac{1}{P}+\frac{1}{R}}$$ harmonic mean of P and R

Dev set + single number evaluation metric

Speed up iterative process of improving ML algorithm.

Satisficing and Optimizing Metrics

Another cat classification example:

Classifier	Accuracy (optimizing)	Running Time
A	90 %	80 ms
B	92 %	95 ms
C	95 %	1500 ms

Running Time → satisficing (It just has to be good enough)

maximize accuracy
subject to runningTime ≤ 100ms

N matrix : 1 optimizing
N-1 satisficing

For ex; maximize accuracy (optimizing metric)
subject to atmost 1 false positive every 24 hours operation
(satisficing metric)

Train / dev / test Distributions

cat classification dev/test sets
↳ development set, hold out cross validation set

Regions:

- US
- UK
- Other Europe
- South America

} Dev

- India
- China
- Other Asia
- Australia

} Test

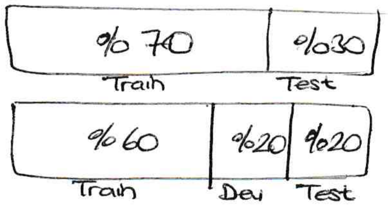
Bad idea
Dev and Test set should come from same distribution

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.

As a solution, randomly shuffle the data into dev/set. So that both the dev and test sets have data from all eight regions.

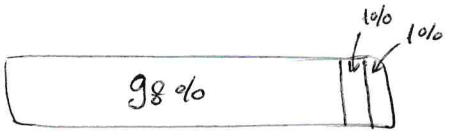
Size of dev and test sets

Old way of splitting data:



were reasonable when data sizes were just smaller

New:



Because data sizes are too big.

Size of Test Set:

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

"Not having a test set might be OKAY. But I don't recommend."
Purpose of test set is to help you evaluate your final cost buys.

When to change dev/test sets and metrics

Metric: classification error
Algorithm A: 3% error
Algorithm B: 5% error

Metric + Dev: Prefer A
You/Users: Prefer B

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

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

Orthogonalization for cat pictures: anti-porn

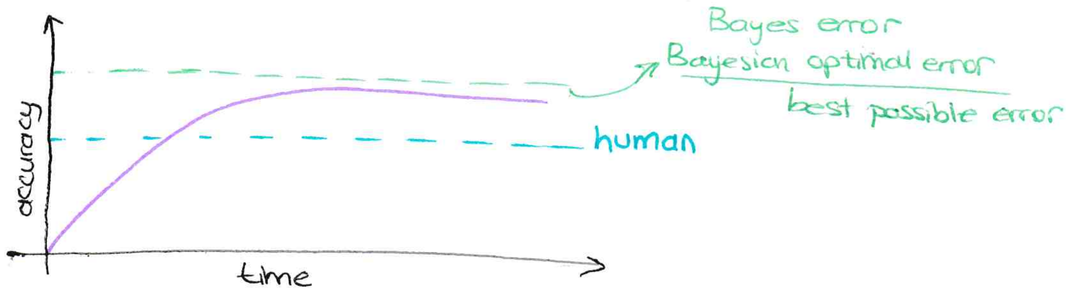
1. So far we've only discussed how to define a metric to evaluate classifiers. Placing target
2. Worry seperately about how to do well on this metric. Aim/shoot at target

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.

Why human-level performance?

2 main reasons:

1. advances in deep learning.
2. workflow is much more efficient.



Why compare to 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 analysis:
  Why did a person get this right?
- Better analysis of bias/variance

Avoidable Bias

Humans (≈ Bayes)	1%		7.5%		
Training error	8%	↕ 7%	8%	→ 0.5% ↕	Avoidable Bias
	10%	↕ 2%	10%	→ 2% ↕	Variance
	Focus on bias		Focus on variance		

Human level error as a proxy for Bayes error...

Understanding Human-Level Performance

Human-level error as a proxy for Bayes error

Bayes error → Best possible error any function could.

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 "human level" error?

Bayes error ≤ 0.5

Error Analysis Example (Medical Imaging Diagnosis Ex)

Human (proxy for Bayes error)	1% / 0.7% / 0.5%		1% / 0.7% / 0.5%		
↕ Avoidable bias		4% / 4.5%		0% / 0.5%	
Training error	5%		1%		
↕ variance		1%		4%	
Dev error	6%		5%		

⇒ You should focus on bias reduction techniques such as train bigger network.

⇒ You should focus on variance reduction techniques such as regularization or getting bigger training set.

~~0.7~~ % / 0.5 %	
	0.2 % / ~~0.0 %~~
0.7 %	
	0.1 %
0.8 %	

Problem gets harder as you achieve or approach human level performance.

⇒ This problem arose only when you're doing very well on your problem already.

Summary of bias / variance with human-level performance

Human-level error (proxy for Bayes error)
↕ "Avoidable bias"
Training error
↕ "Variance"
Dev error

This techniques will tend to work well until you surpass human-level performance.

Surpassing human-level Performance

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

here we can't know what bayes error is. It could be 0.1 % / 0.2 % / 0.3 %

What is avoidable bias?

Problems where ML significantly surpasses human-level performance?

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

Error Analysis

Carrying out error analysis

Look at dev examples to evaluate ideas

Should you try to make your cat classifier do better on dogs?

%90 accuracy
%10 error

Error analysis:

- Get ~100 mislabeled dev set examples
- Count up how many are dogs

"ceiling" → what's in the best case? How well could working on dog problem help you?

→ If there is 5 mislabeled dogs images and when we correct them it just decreases 0.5 from 10% error. (It affects %5, means decreases 0.5)

10
↓
9.5

50%
50 mislabeled dogs images affect 50% and 10% error goes down 5% error.

NOTE!
During error analysis, you're just looking at dev set examples that your algorithm has misrecognized

Evaluate multiple ideas in parallel

Ideas for cat detection:

- Fix pictures of dog being recognized as cats
- Fix great cats (lions, panthers, etc...) being misrecognized
- Improve performance on blurry images

Image	Dog	Great Cats	Blurry	Instagram filters	Comments
1	✓				
2					pitbull
3			✓		
⋮		✓	✓		Rainy day at zoo
% of total	8%	43%	61%		

To carry out error analysis, you should find a set of mislabeled examples

Cleaning up Incorrectly Labeled Data

DL algorithms are quite robust to random errors in the training set.
They are less robust to systematic errors.

Random errors or near random errors are not too bad for most deep learning algorithms.

Error Analysis.

Image	. . . . .	Incorrectly labeled	Comments
...			
98		✓	Labeler missed cat in background
99			
100		✓	Drawing of a cat; Not a real cat
% of total	8% 43% 61%	6 %	

Overall dev set error	10%	2 %
Errors due ~~to~~ incorrect labels	0.6%	0.6% → it affects more
Errors due to other causes	9.4 %	1.4 %

Goal of dev set is to help you select between two classifiers A & B.

for ex: 2.1 % 1.9 %

you choose B, because you know where error comes from and when you correct labels, it will be less.

Correcting Incorrect dev/test set examples:

- Apply same process to your dev and test sets to make sure they continue to come from the same distribution
- Consider examining examples your algorithm got right as well as ones it got wrong
- Train and dev/test data may now come from slightly different distributions → Training can be different distribution than dev/test distribution, learning algorithms are quite robust to that.

Build your first system quickly, then Iterate

Speech recognition example:

- Noisy background → cafe noise, Car noise
- Accented speech
- Far from microphone
- Young children's speech
- Stuttering uh, ah, um
- ...

- Set up dev/test set and metric
- Build initial system quickly
- Use Bias/Variance analysis & Error analysis to prioritize next steps

"I'd encourage you to build something quick and dirty"

Don't Forget!:
The dev set is tagging you into target and when you hit it, you want that to generalize to the test set.

MISMATCHED TRAINING AND DEV/TEST DATA

Training and Testing on different Distributions

Data from webpages — Data from mobile app ← Care about this

≈ 200 000 → 210 000 ← ≈ 10 000

~~Option 1~~ ✗

train | dev | test

205 000 — 2500 — 2500

Advantage: All come from same distributions so that makes it easier to manage.

Disadvantage: If you look at dev set, of these 2500 examples, a lot of it will come from the web page distribution of images, rather than what you actually care about, which is mobile app dist of images.

Option 2:

train: web 205000 + mobile app 5000 | dev: mobile app 2500 | test: mobile app 2500

Adv: You are aiming the target where you want it to be.

Speech recognition example:

Training

purchased data x, y (audio clip, transcript)

smart speaker control

voice keyboard

...

500 000 utterences

Dev/test

Speech activated rearview mirror

20 000

This distribution will be very different than training data

Train 500 k | D 10k | T 10k

510 k | 5k | 5k ← Better

Bias and Variance with mismatched data distributions

Cat Classifier example:

Assume humans get ≈ 0% error

Training error ... 1%

Dev error 10%

There may be 2 problem:

1. The algorithm saw data in training set but not in the dev set.
2. The distribution of data in the dev set is different.

Training-dev set: Same distribution as training set, but not used for training

train | train-dev | dev | test

same distribution (train, train-dev)

randomly shuffle and take some part of train set — wasn't trained

You won't do backprop here — ~~trained~~

Training error ... 1%

Training-dev error ... 9%

Dev error ... 10%

variance problem!

Training error ...	1%	
Training-dev error ...	1.5%	↕ data mismatch ~~error~~ problem!
Dev error	10%	

Human error	0%	↕ Avoidable bias	0%	↕ Avoidable bias
Training error	10%		10%	
Training-dev error ...	11%		11%	↕ Data mismatch
Dev error	12%		20%	
	Bias problem		Bias + Data mismatch problem	

Human level	4%	↕ avoidable bias	4%
Training set error	7%	↕ variance	7%
Training-dev set error	10%	↕ data mismatch	10%
Dev error	12%	↕ degree of overfitting to the dev set	6%
Test error	12%		6%

Rearview mirror ex:

	General speech recognition	Rearview mirror speech data	
Human level	"Human level" 4%	6%	↕ avoidable bias
Error on examples trained on	"Training error" 7%	6%	↕ variance
Error on examples not trained on	"Training-dev error" 10%	"Dev/Test error" 6%	

Addressing Data Mismatch

- Carry out manual error analysis to try to understand difference between training and dev/test sets

  E.g noisy - car noise   street numbers

- Make training data more similar; or collect more data similar to dev/test sets

  E.g simulate noisy in-car data

One technique is artificial data synthesis

Ex: "The quick brown fox jumps over the lazy dog" + Car Noise = Synthesized in-car audio

Possible problem

LEARNING FROM MULTIPLE TASKS

Transfer Learning

Keep parameters,

Initialize the last layer's weights, randomly. Retrain the neural network on the new radiology data set.

2 choice:

1. If you have small dataset, you might want to just retrain the weights of the last layer, keep the rest of parameters fixed.
2. If you have enough data you could also retrain all the layers of the rest of the neural network.

Initial phase of traning on image recognition is sometimes called **pretraining**
If you are updating all the weights afterwards, then training on the radiology data is sometimes called **fine tuning**

Generally in transfer learning, you are transferring from a problem with a lot of data to a problem with relatively little data.

It doesn't make sense in the opposite case.

When transfer learning makes sense

- Task A and B have the same input x.
- You have a lot more data for Task A than Task B. data of Task B are more valuable
- Low level features from A could be helpful for learning B.

Summarize; Transfer learning has been most useful if you're trying to do well on some Task B, usually a problem where you have relatively little data.

Multi-task Learning

Simplified autonomous driving example:

pedestrians	0
cars	1
stop signs	1
traffic lights	0

(4,1)

if you have m examples: (4, m)

Unlike softmax regression:

One image can have multiple labels.

Multi-task learning:

Does each image have each of these four objects in it?

When Multi-task learning makes 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.
- Can train a big enough neural network to do well on all the tasks.

Note: Transfer learning is more used than multi-task learning. Multi-task learning is used mostly computer vision such as object detection.