example 1 : cats VS DOGS

we got 90%

solutions to improve the model :

- 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 L2 regularization
- change network architecture
  - activation functions
  - # hidden units

---

chain of assumptions of ML

1/ fit training set well on cost function

↓

2/ fit dev set well on cost function

↓

3/ fit test set well on cost function

↓

4/ performs well in real world

---

the algorithm fails in step 1

bigger network

better optimization algorithm

if the algorithm fails on dev set

- regularization
- bigger training set

if the algorithm fails on test set

- get bigger dev set : the algorithm overfits to the dev set

if it fails in real world

- change dev set
- change the cost function

1/ Choisir une archi simple

perceptron multi couche - LeNet - LSTM

+

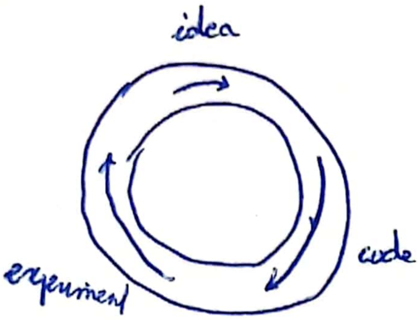
fonction d'activation par defaut

+

normalisation des données

why AI wasn't available :

- limited computational power
- interesting applications such as image recognition require large amounts of data that were not available



1 - large amounts of data allow researchers to try more ideas (better models) but require a longer time to train

2 - improvements in the CPU/GPU hardware enable the discovery of better deep learning algorithm

3 - better algorithms can speed up the iterative process by the reducing the necessary computation time

the two fundamental assumptions of supervised learning:

1/ you can fit the training set pretty well (low avoidable bias)

2/ the training set performance generalizes pretty well to the dev/test set (low variance)

Reducing

uman-level error as a proxy for Bayes error.

xample: medical image classification

suppose
- typical human : 3% error
- typical doctor : 1% error
- experienced doctor : 0,7% error
- team of experienced doctors : 0,5% error

what is human level error ?

Bayes error $\leqslant$ 0.5%

$\Rightarrow$ human level error : 0,5%

---

human level error :	1%	0.7%	0.5%
training error :	5%	5%	5%
dev error	6%	6%	6%

the choice doesn't matter : focus on bias

1%	0.7%	0.5%
1%	1%	1%
5%	5%	5%

the choice of human level doesn't matter :
focus on variance

1%	0.7%	0.5%
0.7%	0.7%	0.7%
0.8%	0.8%	0.8%

Here we need to choose 0.5% as a human level

surpassing human level performance

team of humans	:	0.5%	0.5%
one human	:	1%	1%
training error	:	0.6%	0.3%
dev error	:	0.8%	0.4%
avoidable bias	:	0.1%	
variance		0.2%	

problems where ML outperforms human level performance

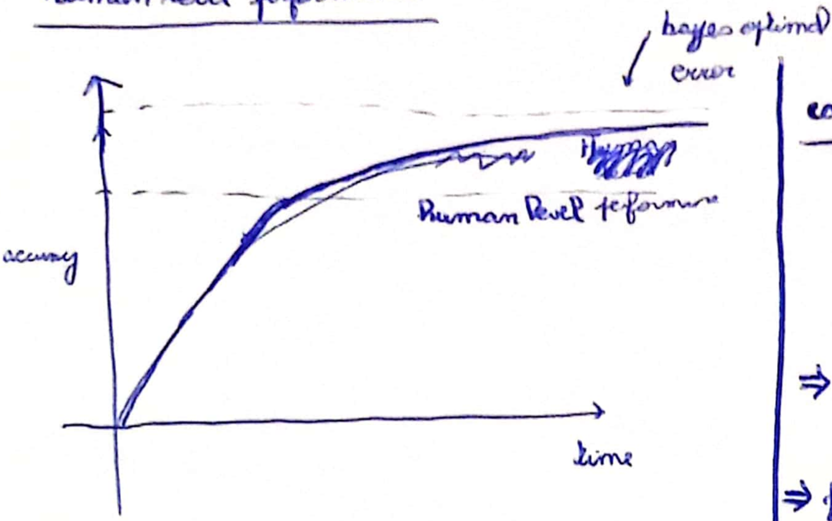
- online advertising
- product recommendations
- logistics (predicting transit time)
- loan approvals

} structured data

$\Rightarrow$ not natural perception tasks

- speech recognition
- some image recognition tasks
- medical tasks

Human-level performance



Bayes optimal error : best possible error

why progress slows down after surpassing human level performance?

1/ human level performance is not so far from Bayes optimal error in some cases.

2/ when we are bellow human level performance there are many tools to use to surpass ~~human~~ it.
but harder to use after surpassing human level performance

So long as ML is worse than humans, you can:

- get labeled data from humans
- gain insight from manual error analysis (manual checking)
- Better analysis of bias / variance

cat classification example 1

Humans : 1%
training error : 8%
dev error : 10%

⇒ we want to do better on train
(the algo is not fitting well the train dataset)

⇒ focus on reducing bias

example 2

Humans : 7.5%
training error : 8%
dev error 10%

⇒ focus on reducing variance

⚠ for computer vision task, we think of human level error as a proxy for Bayes error.

1/ difference between bayes error and train error :

avoidable bias

⚠ in overfitting, we can achieve ~~less~~ ~~error~~ error than bayes error

2/ difference between train error and dev error :

variance

developpement set:

you try different ideas, you train on train set
use the dev set to pick one model
keep iterating to improve the model performance
until we find the classifier we are happy with
test it on the test set.

example : cat classifier

Region: US - UK - other Europe - South America
India - China - other Asia - Australia

1/ first way:

use US - UK - other Europe - South America :
Dev Set
the others
Test set

⇒ bad idea because Dev and Test come from different distribution

2/ second way : the best

Randomly shuffle the data into dev and test sets.

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

- - - - -

sometimes we need to train the model on the data that is available and its distribution may not be the same as the data that will occur in production.

- - - - - - -

adding training data that differs from the dev set may still help the model improve the performance on the test set.
what matters is dev and test sets have the same distribution

size of the test set

it must be big enough to give high confidence in the overall performance of your system

When changing dev and test sets / metric ?

metric error

A : 3% error + shows inappropriate images
B : 5% error + no inappropriate images

metric + dev set : prefer A (lower error)
we / users : prefer B (no inappropriate images)

⇒ $\boxed{\text{we need to change the metric or the dev set}}$

$$Error\_initial = \frac{1}{m_{dev}} \sum_{i=1}^{m_{dev}} \mathbb{1}\{y_{pred}^{(i)} \neq y^{(i)}\}$$

$$\Downarrow$$

$$Error\_changed = \frac{1}{\sum w^{(i)}} \sum_{i=1}^{m_{dev}} w^{(i)} \, \mathbb{1}\{y_{pred}^{(i)} \neq y^{(i)}\}$$

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

---

algorithm A : 3% on dev set
B : 5% on dev set

But algorithm B is doing better on the production

in dev : high quality images
in prod : low quality images

⇒ change the dev/test set

... match

4/. Note

- Having 3 evaluation metrics makes it harder for you to quickly choose between 2 different algorithms and will slow down the speed with which you team can iterate

⇒ solution:

define which criterion is most important then set thresholds for the other two

⇓

stakeholders must define thresholds for satisficing metrics

and must leave the optimizing metric unbounded

adressing data mismatch

1/ carry out manual error analysis to try to understand the differences between traingng and dev/test sets

to avoid overfitting the dev set : we should manually examingng only the dev set.

2/ make training data more similar or collect more data similar to dev/test sets
   i/ collect more data
   ii/ artificial data synthesis

⚠ if we have

10000 hours of speech

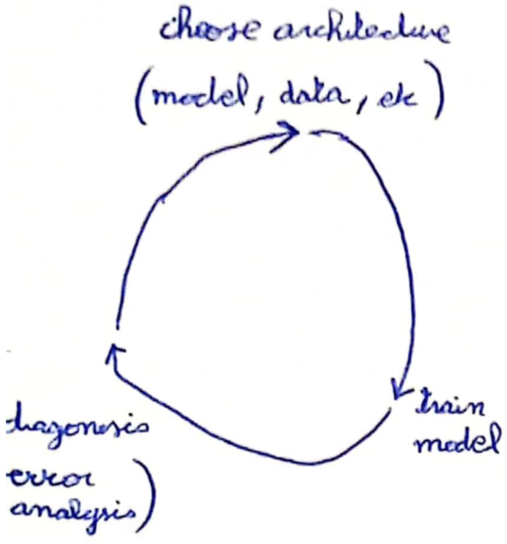
1 hour of noise

if we repeat the noise 10000 times

⇒ we may overfit to the noise

the synthetic data can help to train the model to get better performance at the dev set

but shouldn't be added to the dev or test sets because they don't represent our target in a completely accurate way



$n_{cv} = 500$

he algorithm missclassifies 100

manually examine them

machine learning projed

How to add data ?

1/ add more data of everything

2/ add more data of the types where error analysis has indicated it might help.

3/ data augmentation (images - sounds)

4/ create new synthesis examples

5/ transfer learning: when we don't have a lot of data

example: you want to predict numbers 0.... 9

option 1: only train output layers algorithm

train a model to predict ~~another~~ a lot of types of data (cats - dogs - peoples ....) supervised pretraining

take the parameters of the model

apply the model for the table

(change the parameters of the final layer)

fine tuninging

~~option~~

option 2: train all pramaters and the parameters of the hidden layers will be initialized with the old values

steps of transfer learning

1/ download neural network parameters train on a large dataset ~~......~~ with same input type as your application

2/ further train the network on your own data examples: cat classifier | data mismatch | adding data

90% accuracy   10% error

1/ if the algorithm misclassifies dogs and puts them into cats

⇒ add more dog images

add features specific to dogs

but should you do make your cat classifier ~~cats~~ better on dogs?

error analysis:

get ~100 mislabeled dev set examples

count up how many are dogs

if 5% are dogs:

⇒ "ceiling

if 50% are dogs

⇒ more optimistic to spend time on dogs

2/ evaluate multiple ideas in parallel

ideas for cat detection algorithm

1/ fix dogs pictures being recognized as cats

2/ fix great cats (lions - panthers ....) being misrecognized

3/ improve performance on blurry images

Image	idea 1	idea 2	blurry
1	✓		
2			✓
⋮		✓	✓
....			
% total	8%	43%	61%

cleaning incorrectly labeled data

1/ training set

deep learning algorithms are quite robust to random errors in the training set

⇒ if the errors are reasonably random, don't pay attention to them

but they are sensitive to systematic errors

⇒ if the errors are systematic, ~~...~~ (~~classify~~ all white dogs are annotated cats)

⇒ we need to fix it

2/ dev ~~set~~ and test set

in the error analysis, we add a column called incorrectly labeled

example 1

overall dev set error: 10%

errors due to incorrect labels: 6% of the errors

⇒ 0.6 % of all data

errors due to other causes: 9.4 % of all data

in this case: errors due to incorrect labels is a small fraction

example 2

overall error: 2%

errors due to incorrect labels: 30% of the errors

⇒ 0.6 % of all data

errors due to other causes: 1.4 % of all data

(70% of the errors)

⇒ in this case it is better to fix the incorrect labels

How to correct labels

1/ apply some process to your $\boxed{\text{dev and test sets}}$ to make sure that they come from the same distribution

2/ consider examining example your algorithm got right as well as ones it got wrong

(it isn't always done

3/ train and dev/test data may now come from slightly different distributions

Training and testing on different distributions

we want a $\boxed{\text{mobile app}}$ that classifies

we have : mobile pictures

~~we train with~~ pictures from web-pages

~~we test~~ on pictures from mobile app (less quality)

↘ "

≃ 10 000 mobile pictures

≃ 200 000 web- pictures

we want our algorithm do well on mobile images

we don't want only use the mobile pictures

(small dataset)

option 1 : put the 2 datasets together

shuffle them into train/dev/test set

advantages : train/dev/test sets come from the same distribution

disadvantage : the dev set : most of the images will come from web-pictures which is not representative of the use case

⇒ this option is not suitable

option 2 :

train : 200 K web + 5 K mobile

dev : 2,5 K mobile pictures

test : 2,5 K mobile pictures

advantage : we aim the objective of the app

disadvantage : the distributions are different

⇒ this option is suitable

II - bias and variance

cat classifier example :

human error	↕ bias
training error	↕ variance
training dev	↕ data mismatch
dev	↕ overfitting to dev
test	

assume humans get ≈ 0% error

training error : 1%

dev error : 10%

① if dev data came from the same distribution as the training set :

⇒ we have a variance problem

② if they don't come from the same distribution

⇒ we can't have this interpretation

(dev set may contain harder examples)

(we may also have variance but not only)

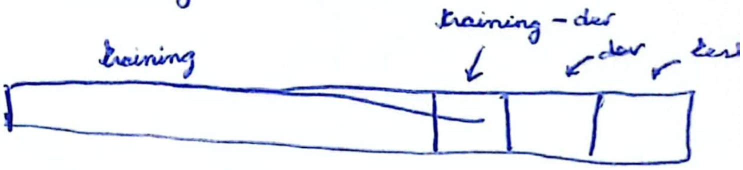
⇓

how to determine the cause ?

variance or different distributions ?

so we define another set :

training - dev set : same distribution as training set but not used for training



1 - train on training

training error : 1%

training dev error : 9%

dev error : 10%

⇒ here we can affirm that we have a variance problem.

2 - training error : 1%

training dev error : 1.5%

dev error : 10%

⇒ here we have a data mismatch error

human error ≃ 0%
3 - training error : 10%
training dev error : 11%
dev error : 12%

⇒ bias problem

4 - human error ≃ 0%
training error : 10%
training dev error : 11%
dev error : 20%

⇒ bias problem
data mismatch error

5 - human error : 4%
training error : 7%
training dev error : 10%
dev error 6%
test error 6%

⇒ here we can say that training set is much harder than dev set

~~human error~~