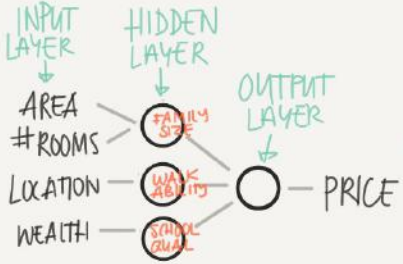
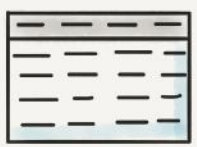
INTRO TO DEEP LEARNING



SUPERVISED LEARNING

INPUT: X	OUTPUT: y	NN TYPE
HOME FEATURES	PRICE	STANDARD NN
AD + USER INFO	WILL CLICK ON AD (0/1)	
IMAGE	OBJECT (1...1000)	CONV. NN (CNN)
AUDIO	TEXT TRANSCRIPT	RECURRENT NN (RNN)
ENGLISH	CHINESE	
IMAGE/RADAR	POS OF OTHER CARS	CUSTOM/HYBRID

NNs CAN DEAL WITH BOTH STRUCTURED & UNSTRUCTURED DATA



STRUCTURED

"THE QUICK BROWN FOX"

UNSTRUCTURED

HUMANS ARE GOOD AT THIS

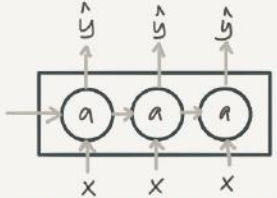
WHY NOW?

ONE OF THE BIG BREAKTHROUGHS HAS BEEN MOVING FROM SIGMOID TO RELU FOR FASTER GRADIENT DESCENT

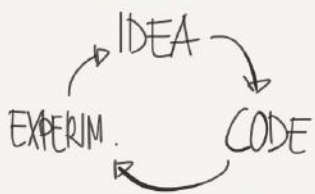
STANDARD NN

NETWORK ARCHITECTURES

CONVOLUTIONAL NN



RECURRENT NN



FASTER COMPUTATION IS IMPORTANT TO SPEED UP THE ITERATIVE PROCESS

LOGISTIC REGRESSION

AS A NEURAL NET

Slope = Derivatives

BINARY CLASSIFICATION

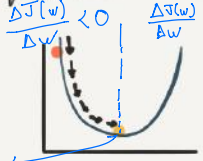
1 : CAT
0 : NOT CAT

X y

Calculate current loss (forward propagation)
Calculate current gradient (backward propagation)
Update parameters (gradient descent)

FINDING THE MINIMUM WITH GRADIENT DESCENT

$w := w - \alpha \frac{\Delta J(w)}{\Delta w}$

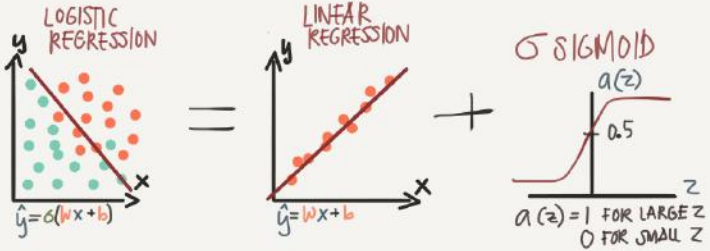


$\frac{\Delta J(w)}{\Delta w} < 0$ $\frac{\Delta J(w)}{\Delta w} > 0$

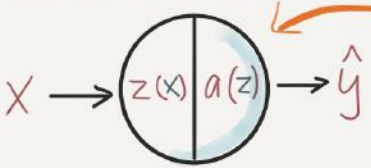
1. FIND THE DOWNHILL DIRECTION (USING DERIVATIVES)
2. WALK (UPDATE $W \in B$) AT A α LEARNING RATE

REPEAT UNTIL YOU REACH BOTTOM (CONVERGE)

Global Optimum find the minimum $J(w,b)$ here



PUTTING IT ALL TOGETHER



MINI NEURAL NET

$z(x) = wx + b$

$\hat{y} = a(z) = \sigma\,\text{SIGMOID}(z)$

1. FORWARD PROPAGATION • CALCULATE $\hat{y}$
2. BACKWARD PROPAGATION • GRADIENT DESCENT + UPDATE W & b

$\sigma(z) = \frac{1}{1+e^{-z}}$

REPEAT UNTIL IT CONVERGES

THE TASK IS TO LEARN W & b BUT HOW?

A: OPTIMIZE HOW GOOD THE GUESS IS BY MINIMIZING THE DIFF BETWEEN GUESS ($\hat{y}$) AND TRUTH (y)

$\text{LOSS} = \mathcal{L}(\hat{y}, y)$

$\text{COST} = J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

COST = LOSS FOR THE ENTIRE DATASET

The lower the better

$w := w - \alpha \frac{\Delta J(w)}{\Delta w}$

$b := b - \alpha \frac{\Delta J(b)}{\Delta b}$

} to change the COST funti

SHALLOW NEURAL NETS

2 LAYER NEURAL NET

We do this each node

$z = w^T x + b$

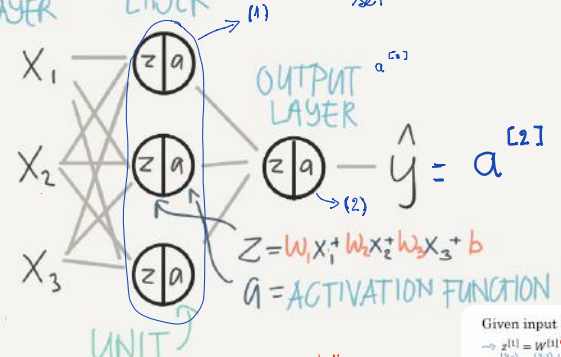
$a = \sigma(z)$

$a^{[0]} = x$

$a^{[1]}$

INPUT LAYER — HIDDEN LAYER — OUTPUT LAYER

you don't see what they should be in the training set



$a^{[2]}$ ← layer

$a^{[2]}_1$ ← node in layer

- Node 1 layer 1: $z^{[1]}_1 = w^{[1]T}_1 x + b^{[1]}_1$, $a^{[1]}_1 = \sigma(z^{[1]}_1)$
- Node 2 layer 1: $z^{[1]}_2 = w^{[1]T}_2 x + b^{[1]}_2$, $a^{[1]}_2 = \sigma(z^{[1]}_2)$

After calculate all the nodes' values above, we compile them to get the value of layer: $w^{[1]}, z^{[1]}, b^{[1]}, a^{[1]}$

★ the 4 lines of codes to solve the issue

Given input x:

$\rightarrow z^{[1]}_{(n_1,1)} = W^{[1]}_{(n_1,n_0)} a^{[0]}_{(n_0,1)} + b^{[1]}_{(n_1,1)}$

$\rightarrow a^{[1]}_{(n_1,1)} = \sigma(z^{[1]}_{(n_1,1)})$

$\rightarrow z^{[2]}_{(n_2,1)} = W^{[2]}_{(n_2,n_1)} a^{[1]}_{(n_1,1)} + b^{[2]}_{(n_2,1)}$

$\rightarrow a^{[2]}_{(n_2,1)} = \sigma(z^{[2]}_{(n_2,1)})$

WHY ACTIVATION FUNCTIONS?

EX. WITH NO ACTIVATION — $a = z$

$a^{[1]} = z^{[1]} = W^{[1]} x + b^{[1]}$ LAYER 1

$a^{[2]} = z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$ LAYER 2

PLUG IN $a^{[1]}$

$a^{[2]} = W^{[2]}(W^{[1]} x + b^{[1]}) + b^{[2]}$

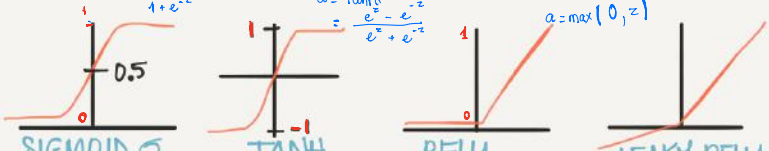
$= W^{[2]} W^{[1]} x + W^{[2]} b^{[1]} + b^{[2]}$

$W' \times + b'$ ← LINEAR FUNCTION

WE COULD JUST AS WELL HAVE SKIPPED THE WHOLE NEURAL NET & USED LIN. REGR.

Activation Function is the foundation of neural network

ACTIVATION FUNCTIONS



BINARY CLASSIFIER - ONLY USED FOR OUTPUT LAYER

SLOW GRAD DESCENT SINCE SLOPE IS SMALL FOR LARGE/SMALL VAL

NORMALIZED ⇒ GRADIENT DESCENT IS FASTER

Better than Sigmoid for sure

DEFAULT CHOICE FOR ACTIVATION

SLOPE = 1/0

AVOIDS UNDEF SLOPE AT Ø BUT RARELY USED IN PRACTICE

your output	Output layer
Binary Classification (0, 1)	Sigmoid
All other types	Relu

⇒ Hidden layer : Relu

Logistic Regression is ok with assigning with 0 but should not for neural network.

INITIALIZING W+B

WHAT IF: INIT TO Ø

THIS WILL CAUSE ALL THE UNITS TO BE THE SAME AND LEARN EXACTLY THE SAME FEATURES

SOLUTION: RANDOM INIT

BUT ALSO WANT THEM SMALL SO RAND * 0.01 ← HYPERPARAM

NEURAL NETWORKS & DEEP LEARNING • COURSERA • ANDREW NG • WK 3

© TessFerrandez

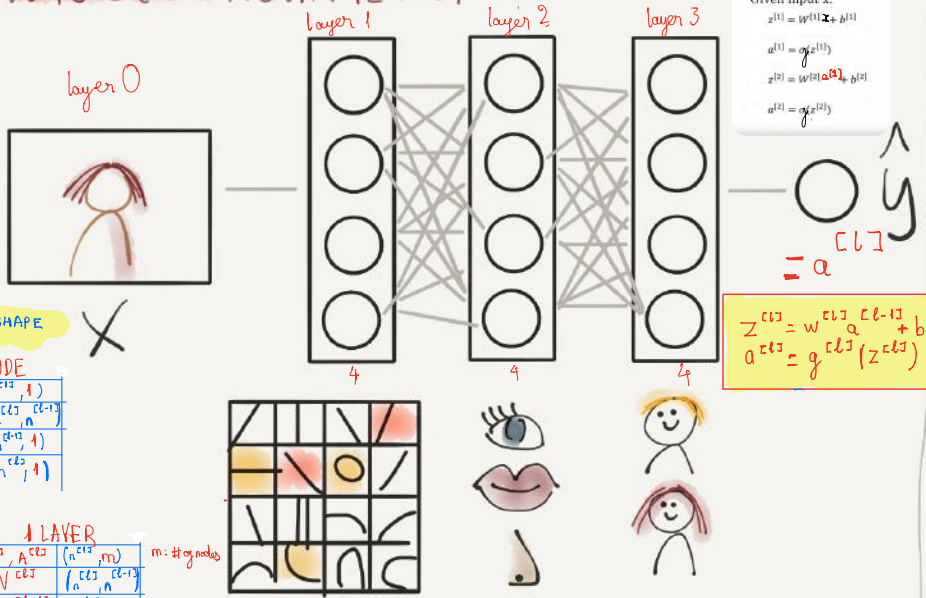
DEEP NEURAL NETS

- L = # of layers
  $L = 3$
- $n^{[l]}$: # units in layer l
  $n^{[0]} = n_x = n^{[0]} = 4$
  $n^{[L]} = n_y = 1$
- $a^{[l]}$: activation in layer l
  $a^{[l]} = g(z^{[l]})$
- $w^{[l]}$: weights for $z^{[l]}$

THERE ARE FUNCTIONS A SMALL DEEP NET CAN COMPUTE THAT SHALLOW NETS NEED EXP. MORE UNITS TO COMP.

Layer l: $w^{[l]}, b^{[l]}$
Forward: Input $a^{[l-1]}$, Output $a^{[l]}$
Cache $z^{[l]}$
$z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$ | $a^{[l]} = g^{[l]}(z^{[l]})$
Backward: Input $da^{[l]}$, output $da^{[l-1]}$, $dw^{[l]}$, $db^{[l]}$
Cache ($z^{[l]}$)

WHY DEEP NEURAL NETS?



$z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$
$a^{[l]} = g^{[l]}(z^{[l]})$

VECTOR SHAPE

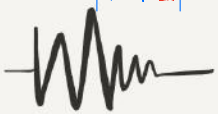
1 NODE

$z^{[l]}, a^{[l]}$	$(n^{[l]}, 1)$
$w^{[l]}$	$(n^{[l]}, n^{[l-1]})$
$a^{[l-1]}$	$(n^{[l-1]}, 1)$
$b^{[l]}$	$(n^{[l]}, 1)$

1 LAYER

$Z^{[l]}, A^{[l]}$	$(n^{[l]}, m)$
$W^{[l]}$	$(n^{[l]}, n^{[l-1]})$
$A^{[l-1]}$	$(n^{[l-1]}, m)$
$b^{[l]}$	$(n^{[l]}, 1) \rightarrow (n^{[l]}, m)$

m: # of nodes



LOW LEVEL AUDIO WAVE FEATURES ↗↘ PITCH — PHONEMES — WORDS — SENTENCES

C A T

VERY DATA HUNGRY

NEED LOTS OF COMPUTER POWER

ALWAYS VECTORIZE
VECTOR MULT. CHEAPER THAN FOR LOOPS

COMPUTE ON GPUs

LOTS OF HYPERPARAMS (W, b) ← Influence the parameters

- LEARNING RATE α
- # ITERATIONS
- # HIDDEN LAYERS
- # HIDDEN UNITS
- CHOICE OF ACTIVATION
- MOMENTUM
- MINI-BATCH SIZE
- REGULARIZATION

L layer neural net work.
[LINEAR → tanh] (L - 1 times) → LINEAR → SIGMOID

© TessFerrandez

NEURAL NETWORKS & DEEP LEARNING • COURSERA, ANDREW NG • WK 4

2 SETTING UP YOUR ML APP

CLASSIC ML

100 - 10000 SAMPLES

TRAIN	DEV	TEST
60%	20%	20%

ALL FROM SAME PLACE → DISTRIBUTION

DEEP LEARNING

1M SAMPLES

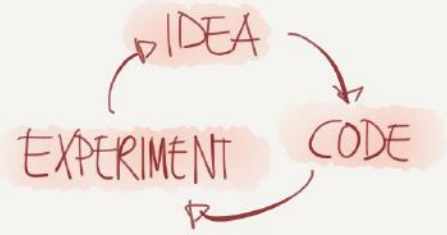
TRAIN	D	T
98%	1%	1%

EX: TRAIN  PRO CAT PICS FROM INTERNET

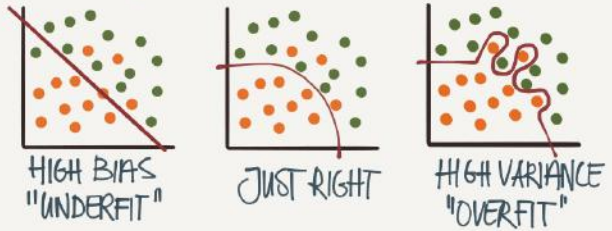
DEV/TEST  BLURRY CAT PICS FROM APP

This is a really bad scenario that you should avoid

TIP
DEV & TEST SHOULD COME FROM SAME DISTRIBUTION



BIAS/VARIANCE



	ERROR			
TRAIN	1%	15%	15%	0.5%
DEV / TEST	11%	16%	30%	1%
	HIGH VARIANCE (Over fit)	HIGH BIAS (Under fitting)	HIGH BIAS & VARIANCE	LOW BIAS & VARIANCE (Golden ratio)

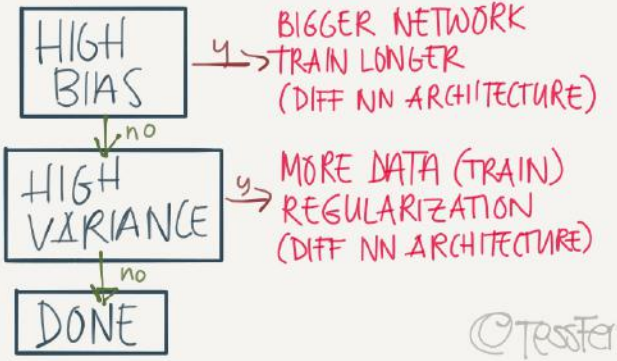
the error of fitting is too large, meaning the model is wrong.

you don't really evaluate well the model so the difference in error % is too large.

ASSUMING HUMANS GET 0% ERROR

Cases of no bias variance trade off can happen in deep learning as we can have both happen at the same time

THE ML RECIPE



REGULARIZATION

PREVENTING OVERFITTING

λ · regularization parameter

"Weight decay"

L2 REGULARIZATION

$$\|w\|_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w$$

$$\text{COST: } J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y},y) + \frac{\lambda}{2m}\|w\|_2^2$$

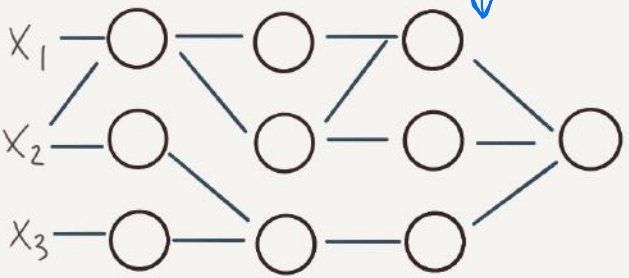
regularization function

EUCLIDEAN NORM

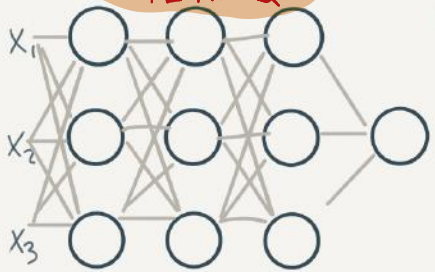
L1 REGULARIZATION

$$\text{COST: } J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y},y) + \frac{\lambda}{m}\|w\|_1$$

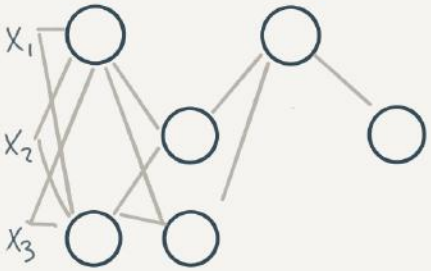
BOTH PENALIZE LARGE WEIGHTS ⇒ SOME WILL BE CLOSE TO 0 ⇒ SIMPLER NETWORKS



DROPOUT TECHNIQUE



FOR EACH ITERATION & SAMPLE SOME NODES ARE RANDOMLY DROPPED (BASED ON KEEP-PROB)

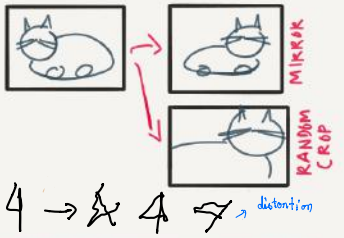


WE GET SIMPLER NWs & LESS CHANCE TO RELY ON SINGLE FEATURES

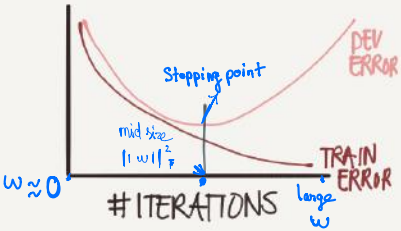
OTHER REGULARIZATION TECHNIQUES

DATA AUGMENTATION

GENERATE NEW PICS FROM EXISTING



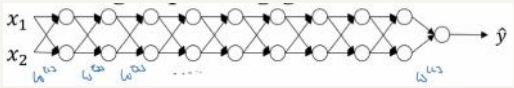
EARLY STOPPING



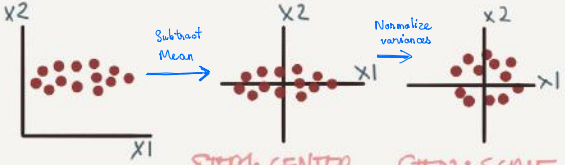
PROBLEM: AFFECTS BOTH BIAS & VARIANCE

IMPROVING DEEP NEURAL NETS · COURSERA

OPTIMIZING TRAINING



NORMALIZING INPUTS



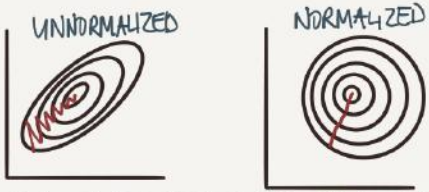
STEP1: CENTER AROUND 0,0

STEP2: SCALE SO VARIANCE IS SAME $\sigma x \to 1$

TIP

USE SAME AVG/VAR TO NORMALIZE DEV/TEST

WHY DO WE DO THIS?



IF WE NORMALIZE, WE CAN USE A MUCH LARGER LEARNING RATE α

↳ we don't need that much time to reach to the center like the left

DEALING WITH VANISHING/EXPLODING GRADIENTS

↗ This makes training really difficult because the more layers of your deep NN, the bigger or smaller your activation ($a^{[L]} = g(z^{[L]}) = g(w^{[L]} a^{[L-1]} + b^{[L]})$)

EX: DEEP NW (L LAYERS)

$\hat{y} = W^{[L]} W^{[L-2]} \cdots W^{[2]} X + b$ ← proven already

IF $W = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \Rightarrow 0.5^{L-1} \Rightarrow$ VANISHING ← $\approx 0.5^{L-1}$ every layer ⇒ making it smaller rapidly

OR $W = \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix} \Rightarrow 1.5^{L-1} \Rightarrow$ EXPLODING ← $\approx 1.5^{L-1}$ every layer ⇒ making it bigger rapidly

IN BOTH CASES GRADIENT DESCENT TAKES A VERY LONG TIME

PARTIAL SOLUTION: CHOOSE INITIAL VALUES CAREFULLY

$w^{[l]} = rand * \sqrt{\frac{2}{n^{[l-1]}}}$ (FOR RELU)

#inputs

XAVIER $\sqrt{\frac{1}{n^{[l-1]}}}$ (FOR TANH)

↳ SETS THE VARIANCE

GRADIENT CHECKING

IF YOUR COST DOES NOT DECREASE ON EACH ITER YOU MAY HAVE A BACKPROP BUG.

GRADIENT CHECKING APPROXIMATES THE GRADIENTS SO YOU CAN VERIFY CALC.

NOTE ONLY USE WHEN DEBUGGING SINCE IT'S SLOW

IMPROVING DEEP NEURAL NETS · COURSERA

OPTIMIZATION ALGORITHMS

$X = [x^{(1)}, x^{(2)}, \ldots, x^{(1000)} \mid x^{(1001)}, \ldots, x^{(2000)} \mid \ldots \mid \ldots x^{(m)}]$

(n_x, m)

$X^{\{1\}}$ $X^{\{2\}}$

m = 5 millions, each mini batch has 1,000 each

$X^{\{t\}}$ → the batch destination in the vector

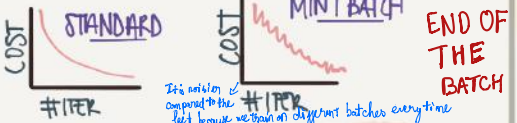
$X^{(i)}$ → the X in X vector

MINI-BATCH GRAD. DESCENT

Mini batch t: $X^{\{t\}}, Y^{\{t\}}$ → $(n_x, 1000)$, $(1, 1000)$

SPLIT YOUR DATA INTO MINI-BATCHES & DO GRAD DESCENT AFTER EACH BATCH

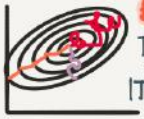
THIS WAY YOU CAN PROGRESS AFTER JUST A SHORT WHILE WITHOUT WAITING TO THE END OF THE BATCH



CHOOSING THE MINIBATCH SIZE

SIZE = m → BATCH GRAD DESC. $(X^{\{t\}}, Y^{\{t\}}) = (X, Y)$

SIZE = 1 → STOCHASTIC GRAD DESC every example is its own mini batch



BATCH: TOO LONG PER ITERATION

MIDDLE

STOCHASTIC: LOSE ALMOST ALL SPEED FROM VECTORIZATION

- SHORT ITER.
- USES VECT.

TIP

IF YOU HAVE < 2000 SAMPLES USE SIZE = 2000

OTHERWISE, USE 64, 128, 256...

SO X+Y FITS IN CPU/GPU CACHE

GRADIENT DESCENT W. MOMENTUM

↘ Speed up gradient descent

WE WANT TO REDUCE OSCILLATION ↕ SO WE GET TO THE GOAL FASTER

SOLUTION: SMOOTH OUT THE CURVE BY TAKING AN EXPONENTIALLY WEIGHTED AVERAGE OF THE DERIVATIVES (i.e. LAST ONE HAS MORE IMPORTANCE)

RMSProp - ROOT MEAN SQUARED

NORMALIZE GRADIENT USING A MOVING AVG.

$S_{dw} = \beta S_{dw} + (1-\beta)\, dw^2$

$S_{db} = \beta S_{db} + (1-\beta)\, db^2$

$w = w - \alpha \frac{dw}{\sqrt{S_{dw}}} \quad b = b - \alpha \frac{db}{\sqrt{S_{db}}}$

ADAM OPTIMIZATION

COMBO OF GD W MOMENTUM & RMSProp

LEARNING RATE DECAY

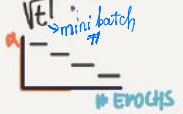
IDEA: USE A LARGE α IN THE BEGINNING, THEN DECREASE AS WE GET CLOSER TO GOAL

OPTION 1: $\alpha = \frac{1}{1 + \text{DECAYRATE} \cdot \text{EPOCH}} \alpha_0$

EXPONENTIAL: $\alpha = 0.95^{\text{EPOCH}} \alpha_0$

OPTION 3: $\alpha = \frac{k}{\sqrt{\text{EPOCH}}} \alpha_0$

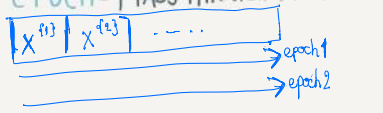
OPTION 4: $\alpha = \frac{k}{\sqrt{t}} \alpha_0$



OPTION 5: DISCRETE STAIRCASE

OPTION 6: MANUAL

EPOCH = 1 PASS THROUGH THE DATA



IMPROVING DEEP NEURAL NETS · COURSERA

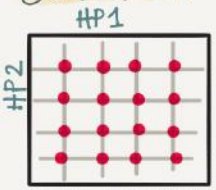
HYPERPARAM TUNING

WHICH HYPERPARAMS ARE MOST IMPORTANT?

- α LEARNING RATE
- # HIDDEN UNITS
- MINIBATCH SIZE
- β MOMENTUM TERM = 0.9
- # LAYERS
- LEARNING RATE DECAY
- $\beta_1 = 0.9$ $\beta_2 = 0.999$ $\varepsilon = 10^{-8}$ (ADAM) — very rare

TESTING VALUES

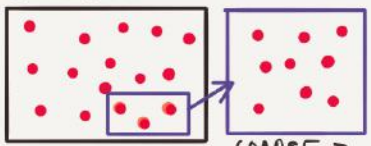
CLASSIC ML



GRID SEARCH

PROBLEM: ONE ITERATION TAKES A LONG TIME & IN 16 GO'S WE HAVE ONLY TRIED 4 α - BUT 4 DIFF ε ← NOT AS IMPORTANT

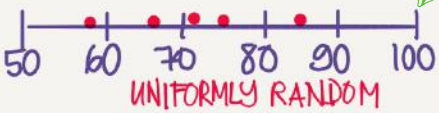
SOLUTION



RANDOM SEARCH + COARSE → DENSE

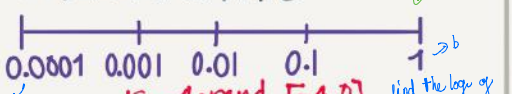
USE AN APPROPRIATE SCALE

HIDDEN UNITS



UNIFORMLY RANDOM

α LEARNING RATE

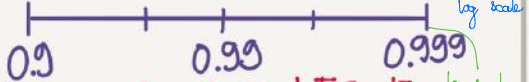


$r = -4 \cdot \text{rand}[-4, 0]$

$\alpha = 10^r$

find the logs of a and b, then try to sample from 10 to the a to the b on the log scale

β EXP WEIGHT AVE



$r = -3 \cdot \text{rand}[-3, -1]$

$\beta = (1 - 10^r)$

TIP

RE-EVALUATE YOUR HYP-PARAMS EVERY FEW MONTHS

PANDA VS CAVIAR

MY PANDA IS ACTUALLY A MISCLASSIFIED CAT BECAUSE I CAN'T DRAW PANDAS

BABYSIT ONE MODEL & TUNE

GOOD IF YOU HAVE LOTS OF SHARE COMP POWER

SPAWN LOTS OF MODELS W DIFF HP

MISC. EXTRAS

BATCH NORMALIZATION

NORMALIZE LAYER OUTPUT

- SPEEDS UP TRAINING
- MAKES WEIGHTS DEEPER IN NW MORE ROBUST (COVARIATE SHIFT)
- SLIGHT REGULARIZING EFFECT

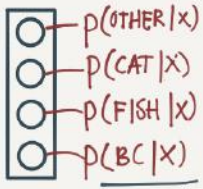
MULTICLASS CLASSIFIC.

$C = \#\text{CLASSES} = 4 \quad (0, \ldots, 3)$

SOFTMAX ACTIVATION

$t = e^{(z^{[L]})}$

$a^{[L]} = \frac{t}{\sum t_i}$



SUM: 1

EX: $z^{[L]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix}$ $t = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix} = \begin{bmatrix} 148.4 \\ 7.4 \\ 0.4 \\ 20.1 \end{bmatrix}$ $\sum t_i = 176.3$

$\Rightarrow a^{[L]} = \frac{t}{176.3} = \begin{bmatrix} 0.842 \\ 0.042 \\ 0.02 \\ 0.114 \end{bmatrix}$ 11.4% PROB IT'S A BABY CHICK

3 STRUCTURING YOUR ML PROJECTS

SETTING YOUR GOAL

★ GOAL SHOULD BE A SINGLE #

	PRECISION	RECALL
A	95%	90%
B	98%	85%

IS A OR B BEST?

	PRECISION	RECALL	F1
A	95%	90%	92.4%
B	98%	85%	91%

A IS BEST

F1 = HARMONIC MEAN BETW. RECALL & PRECISION → This is the best one ★

★ DEFINE OPTIMIZING VS SATISFICING METRICS

optimizing ↓ satisfying ↓

	ACCURACY	RUNTIME
A	90%	80ms
B	92%	95ms
C	95%	1500ms

If satisfying below a threshold then we choose the optimizing

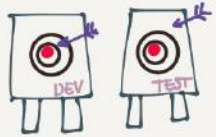
MAXIMIZE ACC.
GIVEN TIME ≤ 100ms

ACCURACY = OPTIMIZING
RUNTIME = SATISFICING

SELECTING YOUR DEV/TEST SETS

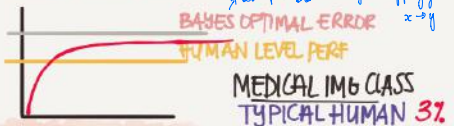
DATA
US
UK
EURDAE
S.AM
INDIA
CHINA
AUST.

OPTION1:
DEV = UK, US, EUR
TEST = REST



IF DEV & TEST ARE DIFF & WE OPTIMIZE FOR DEV WE WILL MISS THE TEST TARGET

HUMAN LEVEL PERF



MEDICAL IMG CLASS
TYPICAL HUMAN 3%
TYPICAL DOCTOR 1%
EXPERIENCED DR. 0.7%
TEAM OF EXP DRs. 0.5%
↑ HUMAN LEV PERF (PROXY FOR BAYES)

WHY DOES ACC SLOW DOWN WHEN WE SURPASS HUMAN LEVEL PERF?

1. OFTEN CLOSE TO BAYES
2. A HUMAN CAN NO LONGER HELP IMPROVE (INSIGHTS)
3. DIFFICULT TO ANALYSE BIAS/VARIANCE

Avoidable bias: the bias is as closed to human as possible so there is no need to actually do more about it ★

CAT CLASSIFICATION

Error

	A	B
HUMAN	1%	7.5%
TRAIN ERR	8%	8%
DEV ERR	10%	10%
	FOCUS ON BIAS	FOCUS ON VARIANCE

Proxy for Bayes error (0.5%)

BLURRY

AVOIDABLE BIAS

VARIANCE

HUMAN
| AVOIDABLE BIAS
TRAIN
| VARIANCE
DEV

- TRAIN BIGGER NETW.
- TRAIN LONGER / BETTER OPT. (RMSProp & ADAM)
- CHANGE NN ARCH OR HYPERPARAMS
- MORE DATA (TRAIN)
- REGULARIZATION
- NN ARCHITECTURE

	A	B
HUMAN	0.5	0.5
TRAIN ERR	0.6	0.3
DEV ERR	0.8	0.4
AVOID. BIAS	0.1	?

AVOIDABLE BIAS

VARIANCE

DON'T KNOW IF WE OVERFIT OR IF WE'RE CLOSE TO BAYES

OPTIONS TO PROCEED ARE UNCLEAR

STRUCTURING ML PROJECTS • COURSERA

ERROR ANALYSIS

Ways to determine if needed to go through mislabel
- Overall Dev set error
- Errors due incorrect labels
- Errors due to other causes

Correcting incorrect dev/test set examples
- Apply same process to your dev + test
- Consider examining examples algorithm got right/wrong
- Train and dev/test data may now come from slightly different distributions

YOU HAVE 10% ERRORS, SOME ARE DOGS MIS-CLASSIFIED AS CATS. SHOULD YOU TRAIN ON MORE DOG PICS?

1. PICK 100 MIS-LABLED
2. COUNT ERROR REASONS

	DOG	BLURRY	INSTA FILTER	BIG CAT	...
1	1		1		
2				1	
3		1			
...					
100			1		
	5%	43%	61%	12%	

5% OF ALL ERRORS

FOCUSING ON DOGS. THE BEST WE CAN HOPE FOR IS 9.5% ERROR

YOU FIND SOME INCORR. LABLED DATA IN THE DEV SET. SHOULD YOU FIX IT?

DL ALGORITHMS ARE PRETTY ROBUST TO RANDOM ERRORS.

BUT NOT TO SYSTEMATIC ERR. (EX. ALL WHITE CATS INCORR LABLED AS MICE)

ADD EXTRA COL. IN ERROR ANALYSIS AND USE SAME CRITERIA

NOTE IF YOU FIX DEV YOU SHOULD FIX TEST AS WELL.

Ideas to improve cat-detection:
- Fix pictures of dogs being recognized as cats
- Fix great cats (lion, panthers, ...) being misrecognized
- Improve performances on blurry pics

Base on this we can decide which is the worthiest of our time to do

FOR NEW PROJ. BUILD 1st SYSTEM QUICK & ITERATE

EX: SPEECH RECOGNITION

WHAT SHOULD YOU FOCUS ON?

NOISE
ACCENTS
FAR FROM MIKE

1. START QUICKLY
   DEV/TEST METRICS
2. GET TRAIN-SET
3. TRAIN
4. BIAS/VARIANCE ANAL
5. ERROR ANALYSIS
6. PRIORITIZE NEXT STEP

STRUCTURING ML PROJECTS • COURSERA

STRUCTURING ML PROJECTS · COURSERA 4

TRAIN vs DEV/TEST MISMATCH

AVAILABLE DATA

200 k PRO CAT PICS FROM INTERNET

10 k BLURRY CAT PICS FROM APP (WHAT WE CARE ABT)

HOW DO WE SPLIT → TRAIN/DEV/TEST?

OPTION 1: SHUFFLE ALL ✗

PROBLEM: DEV/TEST IS NOW MOSTLY WEB IMG (NOT REPRES. OF END SCENARIO)

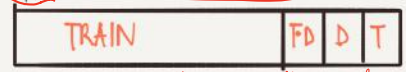
SOLUTION: LET DEV/TEST COME FROM APP · THEN SHUFFLE 5k OF APP PICS W WEB FOR TRAIN

BIAS & VARIANCE W MISMATCHED TRAIN/DEV

HUMANS	~0%
TRAIN	1%
DEV ERR	10%

IS THIS DIFF DUE TO THE MODEL NOT GENERALIZING OR IS DEV DATA MUCH HARDER

A: CREATE A TRAIN-DEV SET THAT WE DON'T TRAIN ON



Human error: 0% — This is too large

	A	B	C	D
TRAIN	1%	1%	10%	10%
TRAIN-DEV	9%	15%	11%	11%
DEV	10%	10%	12%	20%
	VARIANCE	TRAIN/DEV MISMATCH	BIAS	BIAS + DATA MISMATCH

Human level : 4%
↕ avoidable bias
Training : 7%
↑ variance
} evaluate on the training distribution
Train-dev : 10%
↑ data mismatch
Dev : 12%
↑ degree of overfitting to the dev set
} evaluate on the dev-test set distribution
Test : 12%

ADDRESSING DATA MISMATCH

EX. CAR GPS · TRAINING DATA IS 10.000H OF GENERAL SPEECH DATA

1. CARRY OUT MANUAL ERROR ANALYSIS TO UNDERSTAND THE DIFFERENCE (EX NOISE, STREET NUMBERS)
2. TRY TO MAKE TRAIN MORE SIMILAR TO DEV OR GATHER MORE DEV-LIKE TRAIN-DATA (ex: artificial data synthesis)

NOTE BE CAREFUL · IF YOU ONLY HAVE 1 HR OF CAR NOISE & APPLY IT TO 10K HR SPEECH YOU MAY OVERFIT TO THE CAR NOISE

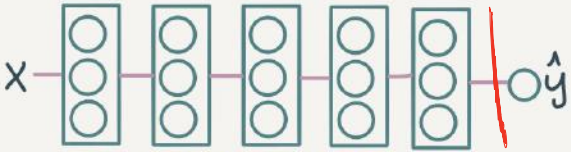
EXTENDED LEARNING

pre-train the weights and parameters first on a different dataset before applying on the one we need

TRANSFER LEARNING

PROBLEM: YOU WANT TO CLASSIFY SOME MEDICAL IMG. YOU HAVE AN NN THAT CLASSIFIES CATS

pretraining / fine tunning



NN TO CLASSIFY CATS/DOGS/CARS

OPTION 1: YOU ONLY HAVE A FEW RADIOLOGY IMAGES

SOLUTION: INIT W. WEIGHTS FROM CAT NN
ONLY RETRAIN LAST LAYER(S) ON RADIOLOGY IMAGES

OPTION 2 YOU HAVE LOTS OF RADIOLOGY IMG.

SOLUTION: INIT WITH WEIGHTS FROM CAT NN
RETRAIN ALL LAYERS

THIS IS MICROSOFT CUSTOM VISION

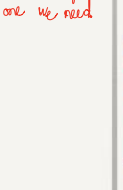
It is useful for the system to learn the initial weights so that they can perform better on harder dataset regardless of new training size

★ Never train in reverse (initial size < target size, Initial low level feature could be helpful for target learning)

MULTI TASK LEARNING

TRAINING ON MULT. TASKS AT ONCE

DETECT
CAR
STOP SIGN
PEDESTR.
TRAFFIC LIGHT



$\Rightarrow \hat{y} = (4, m)$ instead of $(1, m)$ before that

UNLIKE SOFMAX · MANY THINGS CAN BE TRUE

usual logistic loss

$$\text{COST: } J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{4} \mathcal{L}(\hat{y}_j^{(i)}, y_j^{(i)})$$

SUMMING OVER ALL OUTP OPTIONS

WE COULD HAVE JUST TRAINED 4 NNS INSTEAD BUT... MT LEARNING MAKES SENSE WHEN

A. THE LEARNING DATA YOU HAVE FOR THE DIFF TASKS IS QUITE SIMILAR – & THE AMOUNTS (EG. 1K CARS, 1K STOP SIGNS)

B. THE SUM OF THE DATA ALLOWS YOU TO TRAIN A BIG ENOUGH NN TO DO WELL ON ALL TASKS

C. Usually: amount of data you have for each task is quite similar

IN REALITY TRANSFER LEARNING IS USED MORE OFTEN

END-TO-END LEARNING

FROM X-RAY OF CHILDS HAND TELL ME THE AGE OF THE CHILD

TYPICAL SOLN:

1. LOCATE BONES TO FIND LENGTHS USING ML
2. TRAIN MODEL TO PREDICT AGE BASED ON BONELENGTH

END-TO-END

RADIOLOGY IMG → CHILD AGE

PROS:

- LET'S THE DATA SPEAK (MAYBE IT FINDS RELATIONS WE'RE UNAWARE OF)
- LESS HAND-DESIGNING OF COMPONENTS NEEDED

CONS:

- NEEDS LARGE AMTS OF LABELED DATA ($x \to y$)
- EXCLUDES POTENTIALLY USEFUL HAND-MADE COMPONENTS

STRUCTURING ML PROJECTS · COURSERA

TESS NOTES

@Tessferrandez

CONVOLUTION
FUNDAMENTALS

COMPUTER VISION

PROBLEM: IMAGES CAN BE BIG
1000×1000×3 (RGB) = 3M

WITH 1000 HIDDEN UNITS WE NEED 3M * 1000 = 3B PARAMS

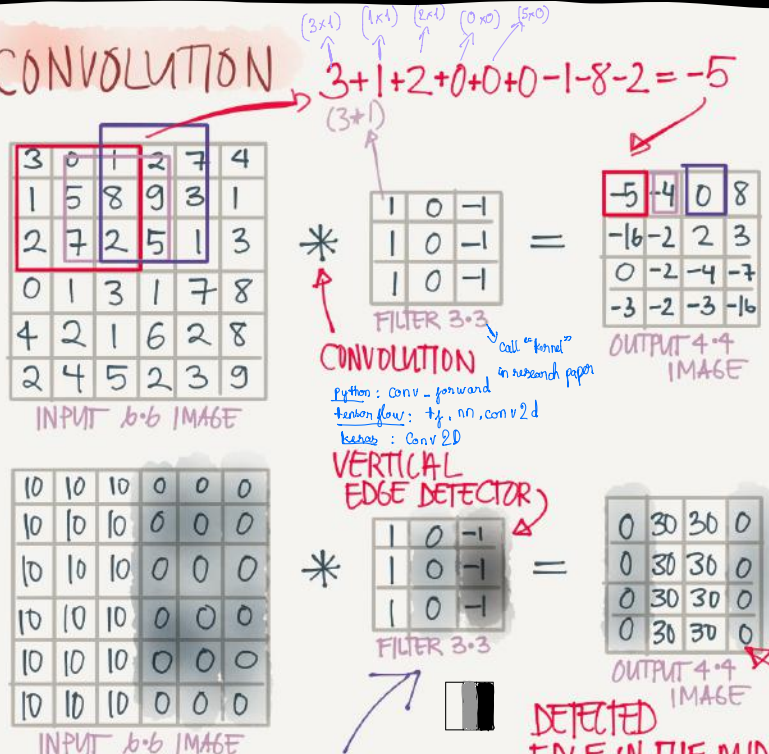
SOLUTION: USE CONVOLUTIONS

IT'S LIKE SCANNING OVER YOUR IMG WITH A MAGNIFYING GLASS OR FILTER

TESS NOTE

ALSO SOLVES THE PROBLEM THAT THE CAT IS NOT ALWAYS IN THE SAME LOCATION IN THE IMG

CONVOLUTION



WE COULD HARD·CODE FILTERS · JUST LIKE WE CAN HARD·CODE HEURISTIC RULES... BUT.... A MUCH BETTER WAY IS TO TREAT THE FILTER # AS PARAMS TO BE LEARNED

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$(6\times 6)_{n\times n} * (3\times 3)_{f\times f} = (4\times 4)_{[(n-f+1)\times(n-f+1)]}$

Most deep learning engineers don't need to bother with the details of the backward pass

CONVOLUTIONAL NEURAL NETS · COURSERA

PADDING

PROBLEM: IMAGES SHRINK
6x6 → 3x3 → 4x4
PROBLEM: EDGES GET LESS 'LOVE'

SOLUTION: PAD W. A BORDER OF 0s BEFORE CONVOLVING

0	0	0	0	0	0	0	0
0	3	0	1	2	7	4	0
0	1	5	8	9	3	1	0
0	2	7	2	5	1	3	0
0	0	1	3	1	7	8	0
0	4	2	1	6	2	8	0
0	2	4	5	2	3	9	0
0	0	0	0	0	0	0	0

TWO COMMONLY USED PADDING OPTIONS
(HOW MUCH TO PAD)

'VALID' ⇒ P = 0 NO PADDING

'SAME' ⇒ $P = \frac{f-1}{2}$ OUTPUT SIZE = INPUT SIZE

FILTER SIZE

$(n-f+1)\times(n-f+1)$
$(n+2p-f+1)\times(n+2p-f+1)$

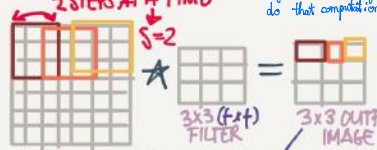
NOTE: ALL CONVOLUTION IDEAS CAN BE APPLIED TO 1D AS WELL LIKE EKG SIGNALS · AND 3D LIKE CT-SCANS

STRIDE

WHAT PACE YOU SCAN WITH

2 STEPS AT A TIME ↓ S=2

If some part of the box is outside, then you don't do that computation



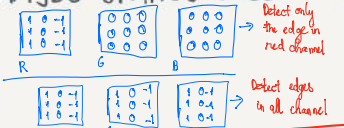
$\frac{n+2p-f}{s}+1$

The result not integers ⇒ round down value: $\lfloor z \rfloor$

CONVOLUTIONS OVER VOLUMES

THIS ALLOWS US TO DETECT FEATURES IN COLOR IMAGES FOR EXAMPLE

MAYBE WE WANT TO FIND ALL EDGES OR MAYBE ORANGE BLOBS

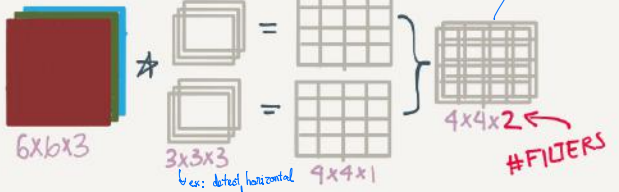


Parameters: $((n_x \times n_y \times d_y) + 1) \times k)$

MULTIPLE FILTERS

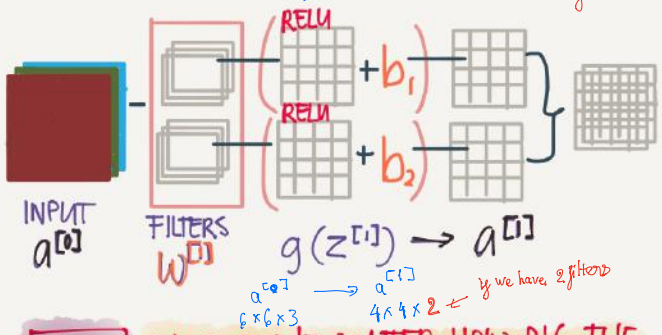
DETECTING MULTIPLE FEATURES AT A TIME

$(n-f+1)\times(n-f+1)\times n_c$



ONE CONV. NET LAYER

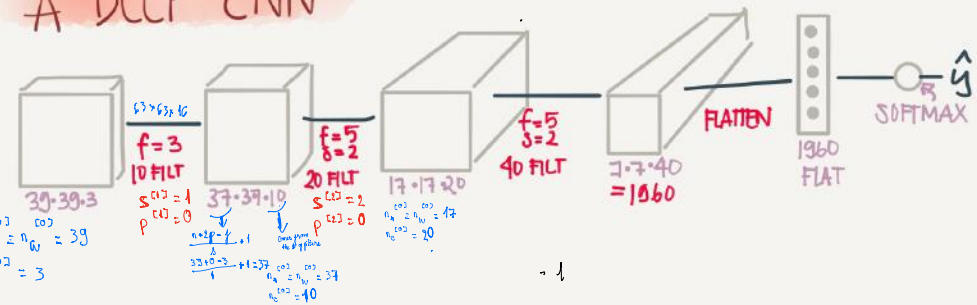
$z^{[1]} = w^{[1]}a^{[0]} + b^{[1]}$
$a^{[1]} = g(z^{[1]})$



NOTE IT DOESN'T MATTER HOW BIG THE INPUT IS - THE LEARNABLE PARAMS W & b ONLY DEPEND ON THE # OF FILTERS AND THEIR SIZES.

$W = 3\cdot 3\cdot 3\cdot 2 = 54$
$b = 2$
} 56 PARAMS TO LEARN

A DEEP CNN



A LOT OF THE WORK IS FIGURING OUT HYPERPARAMS = #FILTERS, STRIDE, PADDING ETC

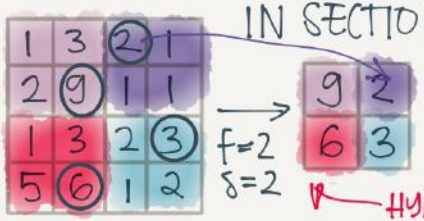
TYPICALLY SIZE → TREND DOWN
FILTERS → TREND UP

TYPICAL CONV.NET LAYERS

CONVOLUTION
POOLING
FULLY CONNECTED

POOLING (MAX)

FIND MAX VAL IN SECTION



HYPERPARAMS

There is also average pooling

- ★ REDUCES SIZE OF REPRES.
- ★ SPEEDS UP COMPUTATION
- ★ MAKES SOME OF THE DETECTED FEAT. MORE ROBUST

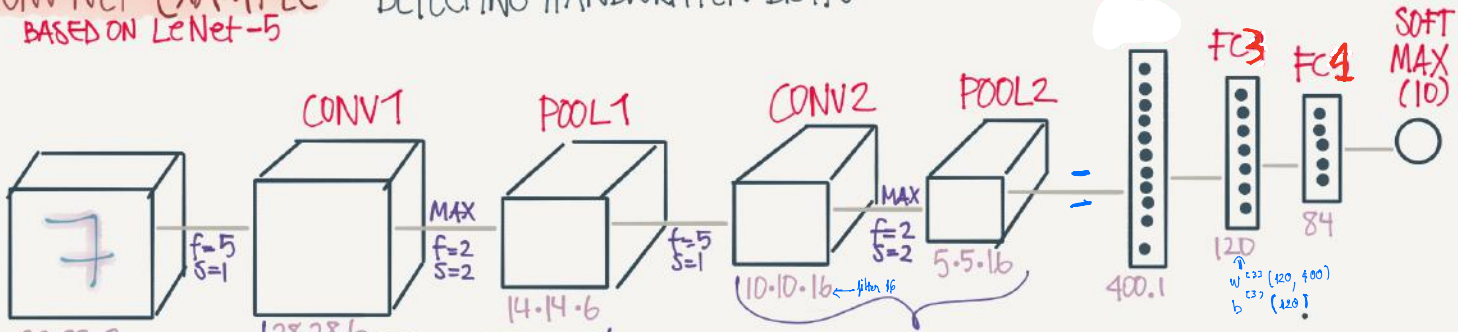
$n_H \times n_W \times n_C$

$\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor * \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \times n_C$

CONV NET EXAMPLE

BASED ON LeNet-5

DETECTING HANDWRITTEN DIGITS



2 naming methods { conv + pool = layer / 1 conv = 1 layer, 1 pool = 1 layer

$n_H, n_W \downarrow$
$n_C \uparrow$

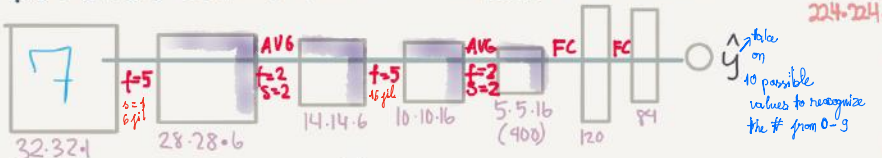
CONVOLUTIONAL NEURAL NETS · COURSERA

CONVOLUTIONAL NEURAL NETS · COURSERA

CLASSIC CONV. NETS

LeNet-5 : to recognize hand written digits

DOCUMENT CLASSIFICATION ≈60k PARAMETERS



TRENDS: HEIGHT/WIDTH GO DOWN n_H, n_W ↓
CHANNELS GO UP n_C ↑

COMMON PATTERN: A COUPLE OF CONV(1+)/POOL LAYERS FOLLOWED BY A FEW FC

OLD STUFF: USED AVG POOLING INST. OF MAX
PADDING WAS NOT VERY COMMON
IT USED SIGMOID/TANH INST OF RELU

VGG-16

- ALL CONV. LAYERS HAVE SAME #. PARAMS
  f=3x3 S=1 P=SAME
- AND POOLING LAYER 2x2 S=2

≈138 M PARAMETERS

- VERY DEEP
- EASY ARCHITECTURE
- # FILTERS DOUBLE 64, 128, 256, 512

AlexNet

IMAGE CLASSIFICATION ≈60M PARAMETERS

- SIMILAR TO LeNet BUT MUCH BIGGER
- USES RELU
- THE NN THAT GOT RESEARCHERS INTERESTED IN VISION AGAIN

CONVOLUTIONAL NEURAL NETS · COURSERA

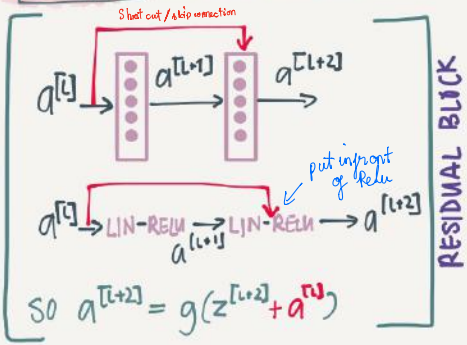
SPECIAL NETWORKS

ResNets

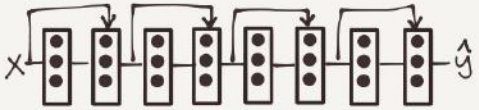
: an experiment when training error is too high (add in the end of neural network)

PROBLEM: DEEP NN OFTEN SUFFER PROBLEMS W VANISHING OR EXPLODING GRADIENTS

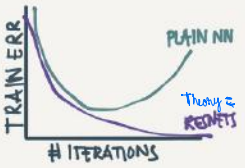
SOLUTION: RESIDUAL NETS



So $a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$



VERY DEEP NN



$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$

$= (w^{[l+2]} a^{[l+1]} + b^{[l+1]} + a^{[l]})$

WORST CASE: w & b HAVE VANISHED TO $\emptyset$ $\Rightarrow$

$a^{[l+2]} = g(a^{[l]}) = a^{[l]}$

SO EXTRA LAYERS LEARNED NOTHING BUT DIDN'T HURT

BEST CASE: LAYERS LEARNED SOMETHING

NETWORK IN NETWORK (1×1 CONVOLUTION)

6	5	3	2
4	1	9	5
5	8	2	4
0	3	6	1

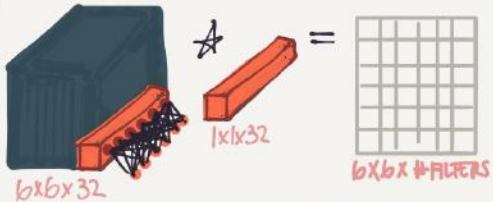
$\ast$ [2] =

12	10	6	4
8	2	18	10
10	16	4	8
0	6	12	2

1×1 CONVOLUTION

IT SEEMS PRETTY USELESS, BUT IT ACTUALLY SERVES 2 PURPOSES

1. NETWORK IN A NETWORK



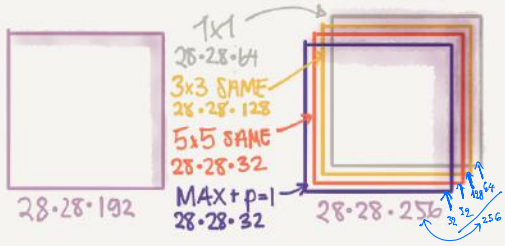
LEARNS COMPLEX, NON-LINEAR RELATIONSHIPS ABOUT A SLICE OF A VOLUME

2. REDUCING # CHANNELS

INCEPTION NETWORKS

called "GoogLeNet" by the google employees to pay homage to the LeNet we talked above

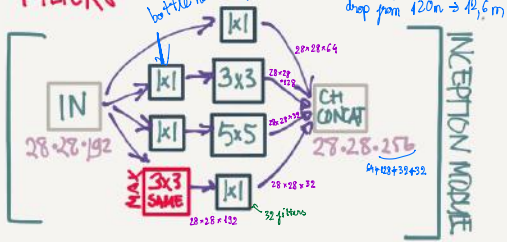
INSTEAD OF CHOOSING A 1x1, 3x3, 5x5 OR A POOLING LAYER - CHOOSE ALL



PROBLEM: VERY EXPENSIVE TO COMPUTE

SOLUTION: SHRINK THE #CHANNELS W A 1x1 CONV BEFORE APPLYING ALL THE FILTERS

The computation cost drop from 120m ⇒ 12,6 m



TO BUILD AN INCEPTION NETWORK YOU MAINLY STACK A BUNCH OF INCEPTION MODULES

INCEPTION THE MOVIE

CONVOLUTIONAL NEURAL NETS · COURSERA

PRACTICAL ADVICE

USE OPEN SOURCE IMPLEMENTATIONS

SOME OF THE PAPERS ARE HARD TO IMPLEMENT FROM SCRATCH - USING OS YOU CAN REUSE OTHER PPLS WORK

DON'T FORGET TO CONTRIBUTE

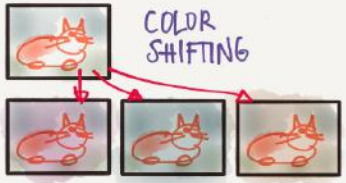
DATA AUGMENTATION

WE ALMOST ALWAYS NEED MORE DATA TO TRAIN ON

MIRRORING

RANDOM CROPPING

ROTATION
SHEARING
LOCAL WARPING
...

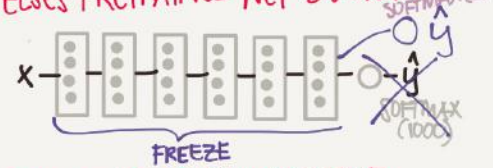


COLOR SHIFTING

TRANSFER LEARNING

WANT TO TRAIN A CLASSIFIER FOR YOUR CATS BUT DON'T HAVE ENOUGH PICTURES

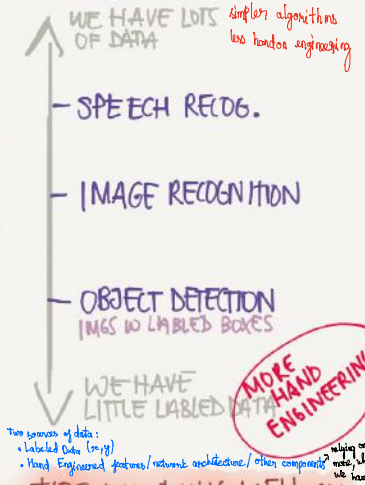
SOLUTION DOWNLOAD SOMEONE ELSES PRETRAINED NET & WEIGHTS



FREEZE THE PARAMS, AND JUST REPLACE THE SOFTMAX LAYER WITH YOUR OWN & TRAIN

IF YOU HAVE MORE PICS · RETRAIN A MORE OF THE LATER LAYERS (MAYBE INITIALIZING WITH THE PRETRAINED WEIGHTS)

STATE OF COMPUTER VISION



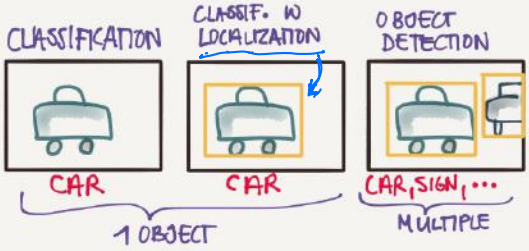
TIPS FOR DOING WELL ON BENCHMARKS/COMPETITIONS

+ ENSEMBLING.
  AVG OUTPUTS FROM MULT NN
+ MULTI-CROP AT TEST TIME
  AVG OUTPUTS FROM MULTIPLE CROPS OF THE IMAGE

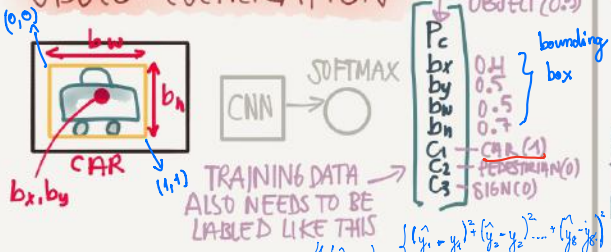
IN PRACTICE THEY ARE NOT USED IN PRODUCTION BECAUSE THEY ARE COMPUTE & MEM EXPENSIVE

CONVOLUTIONAL NEURAL NETS · COURSERA

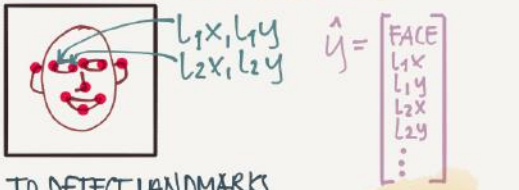
DETECTION ALGORITHMS



OBJECT LOCALIZATION



LANDMARK DETECTION



TO DETECT LANDMARKS IN THE FACE (CORNER OF MOUTH ETC.) LABEL THE X,Y COORDS OF THE LANDMARK

USED FOR SENTIMENT ANALYSIS & FOR EFFECTS LIKE PLACING CROWN ON HEAD ETC.

SLIDING WINDOWS DETECTION

1. CREATE TIGHTLY CROPPED IMG OF CARS (1OTS)
2. SLIDE A WINDOW OVER THE IMG. & CLASSIFY THIS WINDOW CAR(1/0) AGAINST YOUR OTHER CARS
3. REPEAT WITH SLIGHTLY LARGER WINDOW SIZE

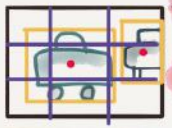
PROBLEM: VERY EXPENSIVE (TO COMPUTE)

SINCE ADJ WINDOWS SHARE A LOT OF THE COMPUTATIONS WE CAN DO THIS MUCH CHEAPER W CONVOLUTIONS

EACH OF THE 4 VALS ARE RESULTS FOR EACH OF THE 4 WINDOWS

NOW WE JUST PASS THROUGH ONCE AND CALC ALL AT THE SAME TIME

YOLO: You Only Look Once



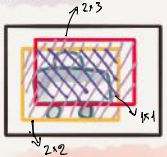
1. SPLIT IMG INTO X(9) GRID CELLS
2. FOR EACH CELL, SAY IF IT CONTAINS CAR + BOUNDING BOX (IF CELL CONTAINS THE MID POINT)

IN PRACTICE WE MIGHT HAVE A 19x19 GRID

SAME AS OBJECT LOCALIZATION

$X \in \hat{y} = 3 \times 3 \times 8$

HOW DO YOU KNOW HOW GOOD IT IS?



HOW GOOD IS THE RED SQUARE?

$IOU = \frac{\text{SIZE OF}}{\text{SIZE OF}} = \frac{1 \times 1}{2 \times 3 + 1 + 2} = \frac{1}{9}$

INTERSECTION OVER UNION

GENERALLY · IF $IOU \geq 0.5$ IT IS REGARDED AS CORRECT ⇒ if lower than 0.5, we keep both box

WHAT IF MULTIPLE SQUARES CLAIM THE SAME CAR?

NON-MAX SUPRESSION

IF TWO BOUNDING BOXES HAVE A HIGH IOU – PICK THE ONE W HIGHEST P_c – GET RID OF THE REST.

ANCHOR BOXES

ANCHOR BOXES LET YOU ENCODE MULTIPLE OBJECTS IN THE SAME SQUARE

CONVOLUTIONAL NEURAL NETS · COURSERA

- Verification:
  - Input: image, name/ID. (1 : 1)
  - Output: whether the input image is that of the claimed person.
  - "is this the claimed person?"
- Recognition:
  - Has a database of K persons
  - Get an input image
  - Output ID if the image is any of the K persons (or not recognized)
  - "who is this person?"

FACE RECOGNITION

FACE VERIFICATION

IS THIS PETE?

99% ACC => PRETTY GOOD

FACE RECOGNITION

WHO IS THIS? (OUT OF K PERSONS)

IF K = 100 NEED MUCH HIGHER THAN 99% (99.9%, something)

ONE-SHOT LEARNING

NEED TO BE ABLE TO RECOGNIZE A PERSON EVEN THOUGH YOU ONLY HAVE ONE SAMPLE IN YOUR DB.

YOU CAN'T TRAIN A CNN WITH A SOFTMAX (EACH PERSON) BECAUSE

A) YOU DON'T HAVE ENOUGH SAMPLES

B) IF A NEW PERSON JOINS YOU NEED TO RETRAIN THE NETWORK

SOLUTION LEARN A SIMILARITY FUNCTION

$d(\text{img1}, \text{img2}) = \text{degree of difference}$

BUT HOW DO YOU LEARN THIS?

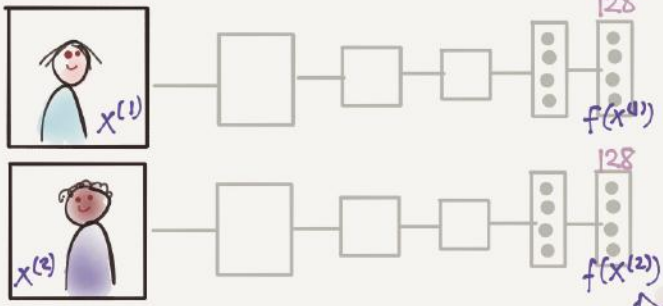
threshold τ

"small" if same people

"big" if different people

SIAMESE NETWORK

DeepFace



$$d(x^{(1)}, x^{(2)}) = \|f(x^{(1)}) - f(x^{(2)})\|_2^2$$

LEARN THE PARAMS OF THE NN SUCH THAT

- IF $x^{(i)}, x^{(j)}$ ARE THE SAME PERSON · $d(x^i, x^j) \Rightarrow$ SMALL
- IF $x^{(i)}, x^{(j)}$ ARE DIFFERENT PEOPLE · $d(x^i, x^j) \Rightarrow$ LARGE

WE CAN ACCOMPLISH THIS WITH THE TRIPLET LOSS FUNCTION

TRIPLET LOSS

FaceNet

(because we always look at 3 pictures at a time)

WANT $\underbrace{\|f(A) - f(P)\|^2}_{d(A,P)} \leq \underbrace{\|f(A) - f(N)\|^2}_{d(A,N)} \Rightarrow d(A,P) - d(A,N) \leq 0$

BUT WE WANT A GOOD MARGIN, SO...

$d(A,P) - d(A,N) + \alpha \leq 0$

α: push the $d(A,P)$ and $d(A,N)$ further away from each others

TIP

PRECOMPUTE ENCODINGS FOR PPL IN YOUR DB, SO YOU DONT HAVE TO SAVE IMAGES & COMPUTE ENCODINGS AT RUN-TIME

HOW DO WE CHOOSE TRIPLETS (A,P,N) TO TRAIN ON?

- IF A/P ARE VERY SIMILAR, & A/N ARE VERY DIFFERENT TRAINING IS VERY EASY.

SELECT A/N THAT ARE PRETTY SIMILAR TO TRAIN A GOOD NET

SOME BIG COMPANIES HAVE ALREADY TRAINED NETWORKS ON LARGE AMTS OF PHOTOS SO YOU MAY JUST WANT TO REUSE THEIR WEIGHTS

Traing set: 10k pictures of 1k persons

NEURAL STYLE TRANSFER

CONTENT (C) + STYLE (S) = GENERATED (G)

PICTURE OF CAMBRIDGE | STARRY NIGHT - VAN GOGH | CAMBRIDGE IN THE STYLE OF VAN GOGH

WE CAN VISUALIZE WHAT A NETWORK LEARNS BY LOOKING AT WHAT IMAGES (PARTS) ACTIVATED EACH UNIT MOST

LAYER 1 · SIMPLE FEATURES LIKE LINES/COLORS

LAYER 3 · MORE COMPLEX FEATURES · BIRD LEGS, HONEY COMB PATTERNS

LAYER 5 · EVEN MORE COMPLEX · MAYBE ONE NODE IS A CAT DETECTOR

BUT HOW DOES THIS HELP US GENERATE AN IMAGE IN THE STYLE OF ANOTHER?

IDEA:

1. GENERATE A RANDOM IMG
2. OPTIMIZE THE COST FUNCTION

$$J(G) = \alpha J_{CONTENT}(C,G) + \beta J_{STYLE}(S,G)$$

HOW SIMILAR ARE C & G — HOW SIMILAR ARE S & G

find the G that minimize the J(G)

3. UPDATE EACH PIXEL

CONTENT COST FUNCTION

- USE A PRE-TRAINED CONVNET (EX VGG)
- SELECT A HIDDEN LAYER SOMEWHERE IN THE MIDDLE (not too deep or shallow)
  LATER ⇒ COPIES LARGER FEATURES
- LET $a^{[l](C)}$ & $a^{[l](G)}$ BE THE ACTIVATIONS
- IF $a^{[l](C)}$ & $a^{[l](G)}$ ARE SIMILAR THEY HAVE SIMILAR CONTENT
  BECAUSE THEY BOTH TRIGGER THE SAME HIDDEN UNITS

HOW DO WE TELL IF THEY ARE SIMILAR?

$$J_{CONTENT}(C,G) = \frac{1}{2} \| a^{[l](C)} - a^{[l](G)} \|^2$$

CAPTURING THE STYLE

USING THE STYLE IMG AND THE ACTIVATIONS IN A LAYER · LOOK THROUGH THE ACTIVATIONS IN THE DIFFERENT CHANNELS TO SEE HOW CORRELATED THEY ARE

WHEN WE SEE PATTERNS LIKE THIS

DO WE USUALLY SEE IT WITH PATCHES LIKE THESE?

⇒ if "yes" then they are "correlated"
if "no" then they are "uncorrelated"

$a^{[l]}_{ijk}$ = activation at (i,j,k) — $G^{[l]}: n_c^{[l]} \times n_c^{[l]}$

STYLE MATRIX → "Gram matrix"

CREATE A MATRIX OF HOW CORRELATED THE ACTIVATIONS ARE, FOR EACH POS (x,y) & CHANNEL PAIR (K, K') FOR THE STYLE IMG & GENERATED

$$G^{[l](S)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l](S)}_{ijk} \cdot a^{[l](S)}_{ijk'}$$

$k = 1, \dots, n_c^{[l]}$

THE STYLE COST FUNCTION

$$= \sum_{k=1}^{n_c^{[l]}} \sum_{k'=1}^{n_c^{[l]}} \left(G^{[l](S)}_{kk'} - G^{[l](G)}_{kk'}\right)^2 \cdot \frac{1}{\left(2 n_H^{[l]} \cdot n_W^{[l]} \cdot n_c^{[l]}\right)^2}$$

$$J_{style}(S,G) = \| G^{(S)} - G^{(G)} \|_F^2$$

FROBENIUS NORM

TO GET MORE VISUALLY PLEASING IMAGES IF YOU CALC $J(S,G)$ OVER MULTIPLE LAYERS

$$J_{style}(S,G) = \sum_l \lambda^{[l]} J^{[l]}_{style}(S,G)$$

CONVOLUTIONAL NEURAL NETS · COURSERA

5

RECURRENT NEURAL NETWORKS

SEQUENCE MODELS · COURSERA

SEQUENCE PROBLEMS

IN	OUT	PURPOSE
"THE QUICK BROWN FOX JUMPED..."		SPEECH RECOGNITION
∅		MUSIC GENERATION
"THERE IS NOTHING TO LIKE IN THIS MOVIE"	★★☆☆☆	SENTIMENT CLASSIFICATION
AGCCCCTGTG AGGAACTAG	AGCCCCTGTG AGGAACTAG	DNA SEQUENCE ANALYSIS
Voulez-vous chanter avec moi?	Do you want to sing with me?	MACHINE TRANSLATION
	RUNNING	VIDEO ACTIVITY RECOGNITION
Yesterday Harry Potter met Hermoine Granger	Yesterday Harry Potter met Hermoine Granger	NAME ENTITY RECOGNITION

Because of this, x and y might have different length

NAME ENTITY RECOGNITION

output per input words

X = HARRY POTTER AND HERMOINE GRANGER INVENTED A NEW SPELL

$X^{<1>}$ $X^{<2>}$ ...

T_x = length of the sequence (9 words)

y = 1 1 0 1 ... $T_y = T_x$

$y^{<1>}$ $y^{<2>}$

1 0 0 0 0

$x^{(i)<t>}$: t^{th} Element in the i^{th} input sequence

$y^{(i)<t>}$: t^{th} element in the i^{th} output sequence

$T_x^{(i)}$: input sequence length for i^{th} training example

$T_y^{(i)}$

EXAMPLE OF A PROBLEM WHERE EVERY $x^{<t>}$ HAS AN OUTPUT $y^{<t>}$

HOW DO WE REPRESENT WORDS?

CREATE A VOCABULARY (EG 10K MOST COMMON WORDS IN YOUR TEXTS · OR DOWNLOAD EXISTING)

a	1
aaron	2
and	367
harry	4075
potter	6830
zulu	10000

EACH WORD IS A **ONE-HOT** VECTOR

$x^{<1>}$ HARRY = [0 ... 1 ... 0] (1 at 4075)

$x^{<2>}$ zulu = [0 ... 1] (1 at 10000)

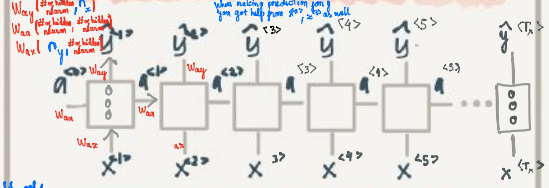
10k vector

WE COULD USE A STANDARD NETWORK BUT...

(A) INPUT & OUTPUTS CAN HAVE DIFFERENT LENGTHS IN DIFF EXAMPLES

(B) WE DON'T SHARE FEATURES LEARNED ACROSS DIFFERENT POSITIONS

11 RECURRENT NEURAL NET (RNN)



PREVIOUS RESULTS ARE PASSED IN AS INPUTS SO WE GET CONTEXT: $a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$

$a^{<t>} = g(W_{aa}\, a^{<t-1>} + W_{ax}\, x^{<t>} + b_a)$ — Tanh / ReLu

$\hat{y}^{<t>} = g(W_{ya}\, a^{<t>} + b_y)$ — Sigmoid

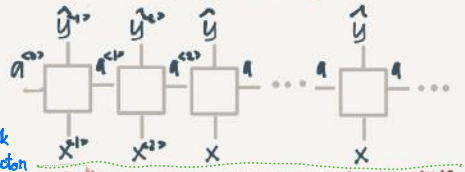
$\hat{y}^{<t>} = g(W_y a^{<t>} + b_y)$

THE SAME W & b ARE USED IN ALL TIME STEPS

THE LOSS WE OPTIMIZE IS THE SUM OF $\mathcal{L}(\hat{y}, y)$ FROM 1–T

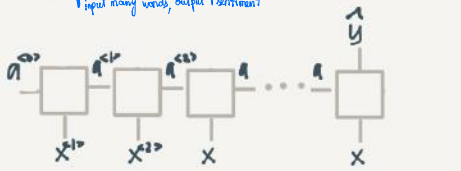
DIFFERENT TYPES OF RNN

MANY-TO-MANY $T_x = T_y$



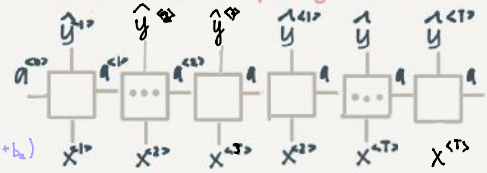
MANY-TO-ONE EX. SENTIMENT ANALYSIS

input many words, output 1 sentiment



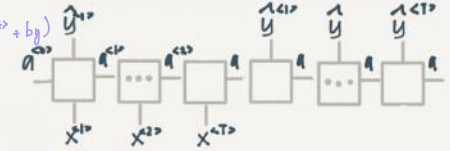
ONE-TO-MANY · MUSIC GENERATION

MANY-TO-MANY $T_x \neq T_y$ BASIC



OK

TRANSLATION



MORE ON RNNs

SEQUENCE MODELS · COURSERA

LANGUAGE MODELLING

HOW DO YOU KNOW IF SOMEONE SAID

a) THE APPLE AND PAIR SALAD OR $P(a) = 3,2 \times 10^{-13}$

b) THE APPLE AND PEAR SALAD ? $P(b) = 5,7 \times 10^{-10}$

⇒ Decide the P of sentence you hear

THE PURPOSE OF A LANG. MODEL IS TO CALCULATE THE PROBABILITIES

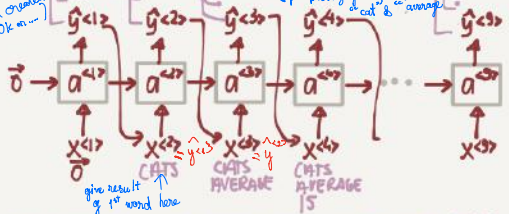
feed the data, it will calculate what is P for the next word to be (based on the corpus you created) (10K words...)

EX. CATS AVERAGE 15 HOURS OF SLEEP A DAY

P(a) P(aaron) ... P(cats) ... P(the) ... P(zulu) P(<EOS>)

probability of word comes after cat ⇒ average

probability of word after "cat is" ⇒ average



SO GIVEN: CATS AVERAGE 15 WHAT IS THE PROB. THE NEXT WORD IS HOURS?

SAMPLING SENTENCES

1. TRAIN ON ALL HARRY POTTER BOOKS.
2. RANDOMLY SELECT A WORD (ON OF THE TOP WORDS) (EX. THE)
3. PASS THIS INTO THE NEXT TIMESTAMP AND SAMPLE A NEW WORD
4. REPEAT UNTIL X WORDS OR YOU REACHED <EOS>

you can reject any word that come out as <UNK>

CAN DO AT CHARACTER LEVEL AS WELL

YAY! YOU ARE NOW YOUR OWN J.K. ROWLING

VANISHING GRADIENTS

THE CAT, WHO ALREADY ATE APPLES AND ORANGES AND A FEW MORE THINGS BLA BLA WAS FULL

THE CATS, WHO ALREADY ATE ... WERE FULL

NEED TO REMEMBER SING/PLURAL FOR A LONG TIME

SINCE LONG SENTENCE ⇒ DEEP RNN WE GET THE VANISHING GRADIENTS PROB WE HAVE IN STANDARD NNs - I.E THE GRADIENTS FOR CAT/CATS HAVE LITTLE OR NO EFFECT ON WAS/WERE. → then we need to figure how to retain that gradient from end → beg

NOTE SOMETIMES YOU SEE EXPLODING GRAD (AS OVERFLOW NaN) BUT THIS IS EASILY FIXED WITH GRADIENT CLIPPING

too big → make algorithm messed up

Ex: gradient vector > threshold → rescale some gradient vector so that it is not too big

GATED RECURRENT UNIT GRU

HELPS RECALL IF CAT WAS SING. OR PLURAL (go to dl 2 note)

THE GRU ACTS AS A MEMORY — AT EVERY TIMESTEP IT CALCULATES A NEW $\tilde{c}$ TO STORE AND A GATE Γ_u DECIDES TO UPDATE c TO $\tilde{c}$ OR NOT

LONG SHORT TERM MEMORY (LSTM)

(go to dl 2 note)

THE LSTM IS A VARIATION ON THE SAME THEME AS GRU BUT WITH AN ADDITIONAL Γ_F FORGET GATE

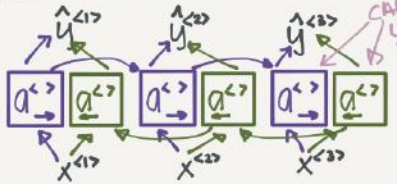
→ usually BRNN + LSTM is the best

2/ BI-DIRECTIONAL RNNs (BRNN)

HE SAID, 'TEDDY BEARS ARE ON SALE'

HE SAID, 'TEDDY ROOSEVELT WAS A GREAT PRESIDENT'

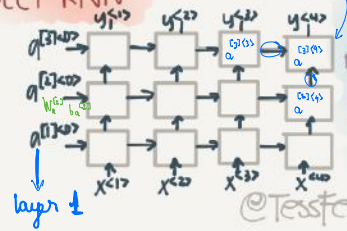
PROBLEM: WITHOUT LOOKING FORWARD WE CANT SAY IF TEDDY IS A TOY OR A NAME



ONE DISADVANTAGE IS THAT YOU NEED THE FULL SENTENCE BEFORE YOU BEGIN. SO NOT SUITABLE FOR LIVE SPEECH REC

3/ DEEP RNN

$a^{[l]<t>} = g(W_a^{[l]} [a^{[l]<t-1>}; a^{[l-1]<t>}] + b_a^{[l]})$



SINCE THEY ARE ALREADY TEMPORAL DEEP THEY USUALLY DON'T HAVE A LOT OF LAYERS

@Tessferrandez

SEQUENCE MODELS · COURSERA

NLP & WORD EMBEDDINGS

any product of the

MAN IS TO WOMAN AS KING IS TO QUEEN

Man $\begin{bmatrix}0\\1\\0\\\vdots\\0\end{bmatrix}$ Woman $\begin{bmatrix}0\\0\\1\\\vdots\\0\end{bmatrix}$

PROBLEM: THE ONE-HOT REPR O_{456} OF APPLE HAS NO INFO ABOUT ITS RELATIONSHIP TO O_{6257} ORANGE

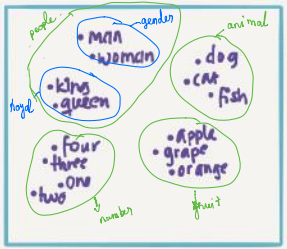
I WANT A GLASS OF ORANGE ____
I WANT A GLASS OF APPLE ____

SOLUTION: CREATE A MATRIX OF FEATURES TO DESCRIBE THE WORDS

WORD EMBEDDINGS

	MAN 5391	WOMAN 9853	KING 4914	QUEEN 7157	APPLE 456	ORANGE 6257
GENDER	-1	1	-0.95	0.97	0.00	0.01
ROYAL	0.01	0.02	0.93	0.95	-0.01	0.00
AGE	0.03	0.02	0.7	0.69	0.03	-0.02
FOOD	0.04	0.01	0.02	0.01	0.95	0.97
⋮						
	e_{5391}					

IN REALITY · THE FEATURES ARE LEARNED & NOT AS STRAIGHTFWD AS GENDER/AGE



algorithm that makes this happen

t-SNE

VISUAL REPRESENT OF 300D WORD EMBEDDINGS (300D → 2D)

USING WORD EMBEDDINGS

EX. NAME/ENTITY RECOGN

Remember this should be a "bi-directional" RNN

WITH WORD EMBEDDINGS WE UNDERSTAND THAT AN ORANGE FARMER IS A PERSON ⇒ SALLY SMITH = NAME

- APPLE ~ ORANGE ⇒ APPLE FARMER = PERSON
- USING WORD EMBEDDINGS TRAINED ON LOTS OF TEXT WE ALSO GET EMB. FOR MORE UNCOMMON WORDS (DURIAN, CULTIVATOR)

EX. MAN IS TO WOMAN AS KING IS TO ?

Analogies

E = EMBEDDING MATRIX

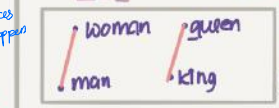
e_{man}

	MAN 5391	WOMAN 9853	KING 4914	QUEEN 7157
GENDER	-1	1	-0.95	0.97
ROYAL	0.01	0.02	0.93	0.95
AGE	0.03	0.02	0.7	0.69
FOOD	0.04	0.01	0.02	0.01

$e_{man} - e_{woman}$ $e_{king} - e_{queen}$

$\begin{bmatrix}-2\\0\\0\\0\end{bmatrix}$ VERY SIMILAR $\begin{bmatrix}-2\\0\\0\\0\end{bmatrix}$

try to find the word that has $e_{man} - e_{man} = e_{king} - e_{word}$



FIND MAN→WOMAN KING→ ?

$FIND(w): \arg\max_w \text{SIM}(e_w, e_{KING} - e_{MAN} + e_{WOMAN})$ (high degree of similarity between)

$\text{SIM}(u,v) = \frac{u^T v}{\|u\|_2 \|v\|_2}$ = COSINE SIMILARITY

After the t-SNE mapping, the links won't apply to them

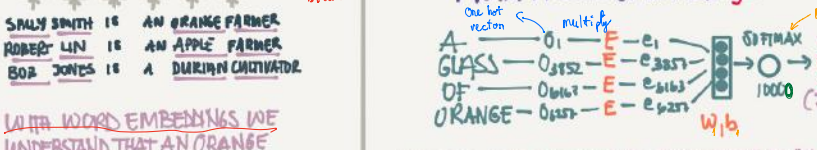
LEARNING WORD EMBEDDINGS

HOW DO WE LEARN THE EMBEDDING MATRIX E?

IDEA 1: USING A NEURAL LANG MODEL

I WANT A GLASS OF ORANGE ____ (6 words) → $\hat{y}$

If we have 300 dimensional word embedding then 300 × 6 words ⇒ 1800 dimensional vector



WE CAN USE DIFFERENT CONTEXTS THAN THE LAST 4 WORDS

- LAST 4 WORDS → to build a language model
- 4 WORDS LEFT+RIGHT
- LAST 1 WORD
- NEARBY 1 WORD

Learn a word embedding

SKIP-GRAM

RANDOM WITHIN EX 5 WORDS

because its taking as input one word like "orange" and then trying to predict nearby words, skipping a few words from the left or the right side. To predict what comes little bit before little bit after the context words

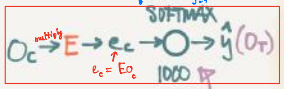
IDEA 2: SKIP-GRAMS — WORD2VEC

I WANT A GLASS OF ORANGE JUICE TO GO ALONG WITH MY CEREAL

PICK RANDOM CONTEXT/TARGET PAIRS (WITHIN EX 5 WORDS)

CONTEXT	TARGET
ORANGE	JUICE
ORANGE	GLASS
ORANGE	MY
...	...

$P(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10000} e^{\theta_j^T e_c}}$ ← this is very slow



NOTE: WHILE THIS SIMPLE NN PREDICTS OT OUR REAL GOAL IS TO LEARN E

THIS IS VERY COMPUTATIONALLY EXPENSIVE BUT WE CAN OPTIMIZE BY USING A HIERARCHICAL SOFTMAX CLASSIFIER

IDEA: NEGATIVE SAMPLING

You have a positive example ("orange" and "juice") then you will deliberately generate a bunch of negative examples, negative samplings, hence the name negative sampling, with which to train k more of these binary classifiers

1. PICK A CONTEXT/TARGET PAIR AS A POSITIVE EXAMPLE
2. PICK A FEW NEG EXAMPLES CONTEXT + RANDOM

CONTEXT (x)	WORD	TARGET (y)
ORANGE	JUICE	1 (positive)
ORANGE	KING	0 (negative)
ORANGE	BOOK	0
ORANGE	THE	0
ORANGE	OF	0

K

O_c → multiplied by E → e_c → 10000 BIN. CLASSIFIERS (JUICE, KING, ...)

YOU ONLY TRAIN 5 FOR EACH CONTEXT WORD ⇒ EFFICIENT TO TRAIN

NOTE: SOMETIMES BY CHANCE YOU PICK A POS PAIR · BUT IT DOESN'T MATTER

K = 5-20 (small dataset)
K = 2-5 (large dataset)

Selecting the negative samples:

$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{10000} f(w_j)^{3/4}}$

@Tessferrandez

SEQUENCE MODELS · COURSERA

IDEA: GLOVE WORD VECTORS

C, t (connect with each other)

X_{ij} = #TIMES WORD j (TARGET t) APPEARS IN THE CONTEXT OF i (CONTEXT c)

$X_{ij} = X_{ji}$ (HOW RELATED THEY ARE)

$$\text{MINIMIZE} \sum_{i=1}^{10k} \sum_{j=1}^{10k} f(X_{ij})(\theta_i^T e_j + b_i + b_j' - \log X_{ij})^2$$

$f(X_{ij})$: weighting term, $f(X_{ij}) = 0$ if $X_{ij} = 0$ (IF NO CONTEXT)

(ALSO HELPS WEIGHING VERY FREQ WORDS (THE, OF...) & VERY INFREQUENT (DURIAN))

θ_i, e_j are symmetric

EVERYTHING LED UP TO THIS VERY SIMPLE ALGORITHM

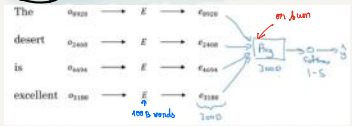
SENTIMENT CLASSIFICATION

X	y
THE DESSERT IS EXCELLENT	★★★★
SERVICE WAS QUITE SLOW	★★
GOOD FOR A QUICK MEAL BUT NOTHING SPECIAL	★★★
COMPLETELY LACKING IN GOOD TASTE GOOD SERVICE AND GOOD AMBIENCE	★

PROBLEM: YOU MAY NOT HAVE A LARGE DATASET

BUT YOU CAN USE AN EMBEDDING MATRIX E THAT IS ALREADY PRE-TRAINED

IDEA: SIMPLE CLASSIFICATION

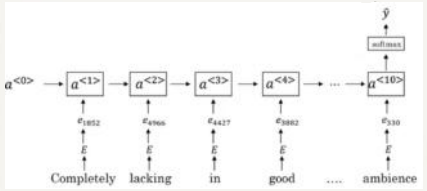


WORKS WELL FOR SHORT SENTENCES BUT DOESN'T TAKE ORDER INTO ACCOUNT

"COMPLETELY LACKING IN GOOD TASTE, GOOD SERVICE AND GOOD AMBIENCE"

THIS MAY BE SEEN AS A +++ REVIEW

IDEA: USE AN RNN



THIS CAN NOW TAKE INTO ACCOUNT THAT COMPLETELY LACKING NEGATES THE WORD GOOD

ELIMINATING BIAS IN WORD EMBEDDINGS

MAN IS TO COMPUTER PROGRAMMER ✓ AS WOMAN IS TO HOME MAKER ✗

SOMETIMES THE TEXT CONTAINS & ALGOS LEARN A GENDER, RACE, AGE... BIAS WE DON'T WANT OUR MODELS TO HAVE · EX. HIRING BASED ON GENDER, SENTENCING BASED ON RACE ETC.

ADDRESSING BIAS

1. IDENTIFY BIAS DIRECTION

$e_{he} - e_{she}$, $e_{male} - e_{female}$ → find the average

2. NEUTRALIZE

FOR EVERY WORD THAT IS NOT DEFINITIONAL (GIRL, BOY, HE, SHE...) PROJECT TO GET RID OF BIAS

3. EQUALIZE PAIRS

THE ONLY DIFF BETWEEN EX GIRL/BOY SHOULD BE GENDER

↑ making sure that words like "girl" or "boy" are both exactly the same similarity, are exactly same distance from words that are gender neutral ("doctor", "baby sitter", ...)

HOW DO YOU KNOW WHICH WORDS TO NEUTRALIZE?

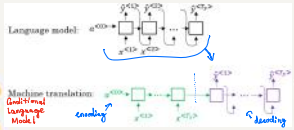
DOCTOR, BEARD, SEWING MACHINE?

A: BY TRAINING A CLASSIFIER TO FIND OUT IF A WORD IS DEFINITIONAL

TURNS OUT THE # OF PAIRS IS FAIRLY SMALL SO YOU CAN EVE HAND PICK THEM

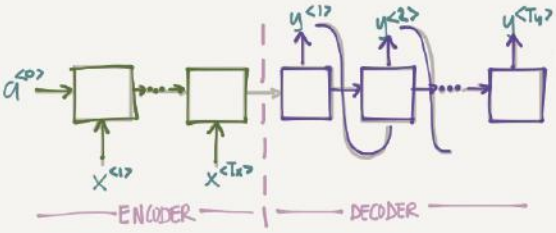
SEQUENCE MODELS · COURSERA

SEQUENCE TO SEQUENCE



BASIC MODELS

JANE VISITE L'AFRICE EN SEPTEMBRE → JANE IS VISITING AFRICA IN SEPTEMBER



 → THIS IS A CAT ON A CHAIR

CNN → RNN

HOW DO YOU PICK THE MOST LIKELY SENTENCE?

$$P(y^{\langle 1\rangle}, \dots, y^{\langle T_y\rangle} | x^{\langle 1\rangle}, x^{\langle 2\rangle}, \dots x^{\langle T_x\rangle})$$

(ENGLISH / FRENCH)

WE DON'T WANT A RANDOMLY GENERATED SENTENCE (WE WOULD SOMETIMES GET A GOOD, SOMETIMES BAD)
INSTEAD WE WANT TO MAXIMIZE

$$\arg\max_{y^{\langle 1\rangle},\dots,y^{\langle T_y\rangle}} P(y^{\langle 1\rangle}, \dots, y^{\langle T_y\rangle} | x)$$

IDEA: USE GREEDY SEARCH

1. PICK THE WORD WITH THE BEST PROBABILITY
2. REPEAT UNTIL <EOS>

The first ("Jane is") is ok but since "going" has the higher possibility, the first example will be chosen

WITH THIS WE COULD GET
- JANE IS GOING TO BE VISITING AFRICA THIS SEPTEMBER

INSTEAD OF
- JANE IS VISITING AFRICA THIS SEPTEMBER

SOLUTION OPTIMISE THE PROB OF THE WHOLE SENTENCE INSTEAD

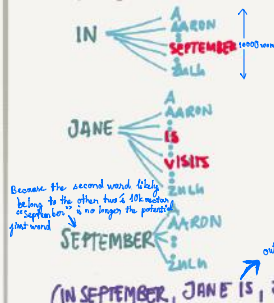
BEAM SEARCH

a heuristic search algorithm

1. PICK THE FIRST WORD
   Beam width
   PICK THE **B** (EX 3) BEST ALTERNATIVES (IN, JANE, SEPTEMBER) → the 3 probable first words in the translation
2. FOR EACH **B** WORDS PICK THE NEXT WORD AND EVALUATE THE PAIRS TO END UP W **B** PAIRS

$$P(y^{\langle 1\rangle}, y^{\langle 2\rangle} | x) = P(y^{\langle 1\rangle} | x)\, P(y^{\langle 2\rangle} | x, y^{\langle 1\rangle})$$

$\approx P(y^{\langle 2\rangle} | x, \text{"in"})$



NOTE: KEEP TRACK OF THE P FOR THE SENTENCES OF EACH LENGTH - AFTER X ITER (X MAX WORDS) PICK THE BEST

(IN SEPTEMBER, JANE IS, JANE VISITS)

out of all this we cut down to the **B** again

3. REPEAT TIL DONE

$$\text{ARG MAX} \prod_{t=1}^{T_y} P(y^{\langle t\rangle} | x, y^{\langle 1\rangle}, \dots y^{\langle t-1\rangle})$$

OVERFLOWS

PROBLEM: MULTIPLYING PROBABILITIES (<<<1) RESULTS IN A VERY SMALL NUMBER

PROBLEM II: IF WE OPTIMIZE FOR THE MULT WE WILL PREFER SHORT SENTENCES. SINCE EACH WORD WILL REDUCE PROB

INSTEAD WE CAN OPTIMIZE FOR THIS

$$\frac{1}{T_y^{\alpha}} \sum_{t=1}^{T_y} \log P(y^{\langle t\rangle} | x, y^{\langle 1\rangle}, \dots, y^{\langle t-1\rangle})$$

$\alpha = 1$ complete normalization by length
$\alpha = 0$ not normalizing at all
$\alpha = 0.7$ a better option

This is the most correct model

HOW DO WE PICK B?

(can get diminishing return)

LARGE B: BETTER RESULT, SLOWER
SMALL B: WORSE RESULT, FASTER

IN PROD YOU MIGHT SE B=10.
100 IS PROBABLY A BIT TOO HIGH - BUT ITS DOMAIN DEPENDENT

ERROR ANALYSIS IN BEAM S.

HUMAN: JANE VISITS AFRICA IN SEPT... y^*
ALGO: JANE VISITED AFRICA LAST SEPTEMBER $\hat{y}$

HOW DO WE KNOW IF ITS OUR RNN OR OUR BEAM SEARCH WE SHOULD WORK ON?

LET THE RNN GIVE $P(y^*|x) = P(y^*, x)$ & $P(\hat{y}|x) = P(\hat{y}, x)$

IF $P(y^*) > P(\hat{y})$: beam search chose $\hat{y}$. But y^* attains higher $P(y|x)$
⇒ Beam search is at fault
BEAM PICKED THE WRONG ONE - TRY A HIGHER B

ELSE: $P(y^*) \leq P(\hat{y})$ y^* is the better translation than $\hat{y}$. But RNN predicted $P(y^*|x) < P(\hat{y}|x)$
⇒ RNN is fault
THE RNN PICKED THE WRONG PROBS - SO FOCUS ON THE RNN

Human	Algorithm	$P(y^*\|x)$	$P(\hat{y}\|x)$	At fault?
Jane visits Africa in September.	Jane visited Africa last September.	1.5×10^{-10}	2.0×10^{-10}	B
Jane visits Africa in September.	Jane visited Africa last September.	1.5×10^{-10}	2.0×10^{-10}	RNN

fernandez

SEQUENCE TO SEQUENCE

SEQUENCE MODELS · COURSERA

FRENCH: LE CHAT EST SUR LE TAPIS
HUMAN1: THE CAT IS ON THE MAT
HUMAN2: THERE IS A CAT ON THE MAT

HOW DO YOU EVALUATE THE MACHINE TRANSLATION WHEN MULTIPLE ARE RIGHT?

BLEU SCORE (* optional)

IDEA: CHECK IF THE WORDS (MT) APPEAR IN THE REAL TRANSLATION

"THE THE THE THE THE THE THE"
SCORE: 7/7

IDEA: ONLY GIVE CREDIT FOR A WORD THE MAX # TIMES IT APPEARS IN A TARGET SENTENCE
SCORE: 2/7 — COUNT CLIP

"THE CAT THE CAT ON THE MAT"

	COUNT	COUNT CLIP
THE CAT	2	1
CAT THE	1	0
CAT ON	1	0
ON THE	1	1
THE MAT	1	1
BIGRAMS	6 (on the MT output)	4 (on the 2 references we have)

BI-GRAM SCORE: 4/6

COMBINED BLEU SCORE

$BP \cdot \exp\left(\frac{1}{4}\sum_{n=1}^{4} P_n\right)$

P_1 = SCORE SINGLE WORD
P_2 = SCORE BIGRAMS
...
BP = BREVITY PENALTY — PENALIZES SENTENCES SHORTER THAN THE TARGET

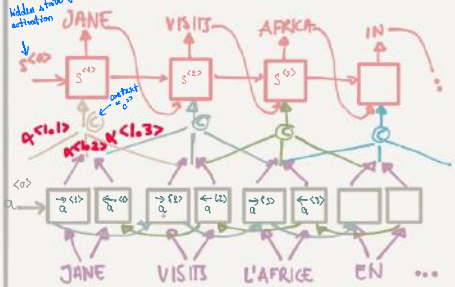
A USEFUL SINGLE NUMBER EVAL METRIC

ATTENTION MODEL

"BLA BLA BLA BLA SUPERLONG SENTENCE IN FRENCH"

"OOPS! SORRY I FORGOT WHAT YOU SAID" — TRANSLATOR

SOLUTION: TRANSLATE A LITTLE AT A TIME USING ONLY PARTS OF THE SENTENCE AS CONTEXT



$\alpha^{<t,t'>}$ = HOW MUCH ATTENTION $y^{<t>}$ (English word) SHOULD PAY TO $a^{<t'>}$ (French word)

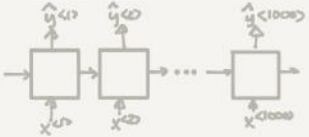
$C^{<2>} = \sum_{t'} \alpha^{<2,t'>} \cdot a^{<t'>}$ (attention weight) $\quad \sum_{t'} \alpha^{<1,t'>} = 1$

α IS CALCULATED USING A SMALL NEURAL NETWORK

$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$

SPEECH RECOGNITION

→ THE QUICK BROWN FOX...



PROBLEM: 10s CLIP AT 100Hz = 1000 INPUTS BUT ONLY ≈ 20 OUTPUTS

SOLUTION: USE CTC COST (CONNECTION TEMPORAL CLASSIFICATION)

"space character" — "blank" character

ttt_h_eee___U___qqq

COLLAPSE REPEATED CHARS NOT SEP BY BLANK (enated)

TRIGGER WORD DETECTION