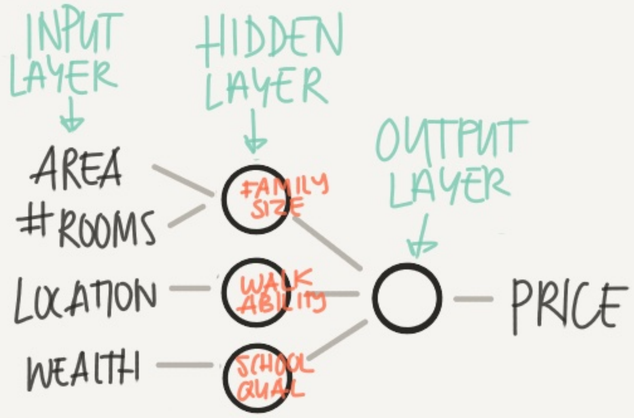
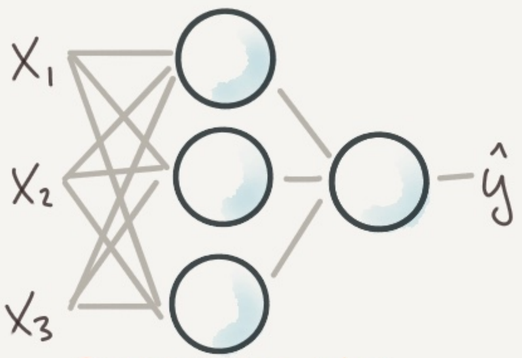
INTRO TO DEEP LEARNING

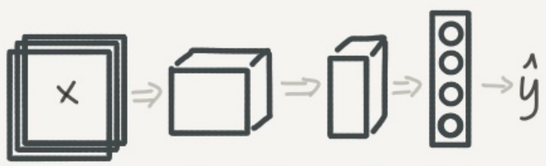


SUPERVISED LEARNING

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

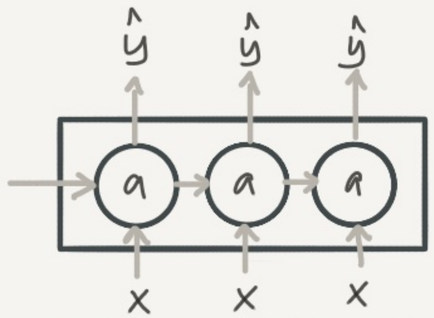


STANDARD NN



CONVOLUTIONAL NN

NETWORK ARCHITECTURES



RECURRENT NN

NNs CAN DEAL WITH BOTH STRUCTURED & UNSTRUCTURED DATA

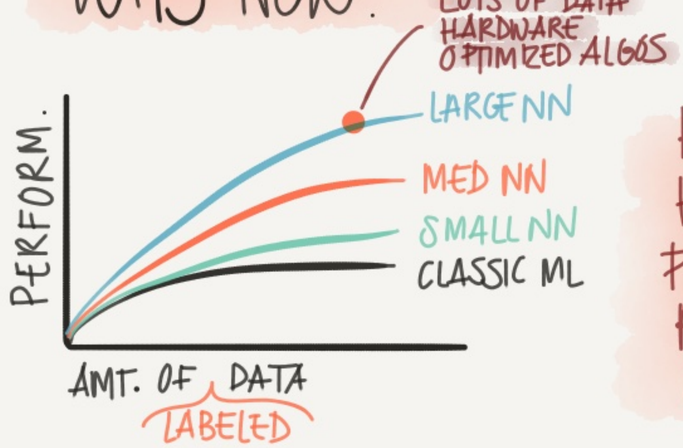
STRUCTURED

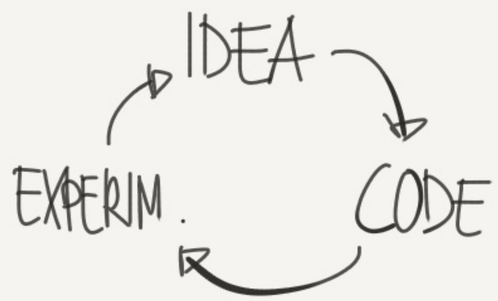
"THE QUICK BROWN FOX"

UNSTRUCTURED

HUMANS ARE GOOD AT THIS

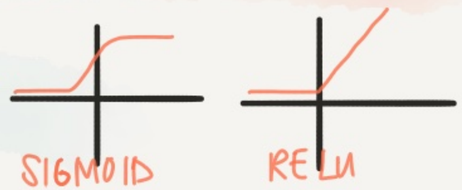
WHY NOW?





FASTER COMPUTATION IS IMPORTANT TO SPEED UP THE ITERATIVE PROCESS

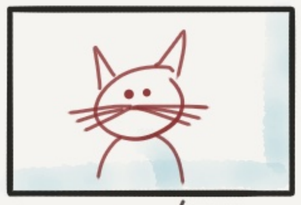
ONE OF THE BIG BREAKTHROUGHS HAS BEEN MOVING FROM SIGMOID TO RELU FOR FASTER GRADIENT DESCENT



LOGISTIC REGRESSION

AS A NEURAL NET

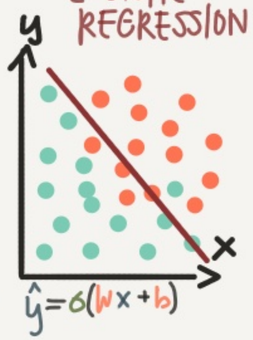
BINARY CLASSIFICATION



1 : CAT
0 : NOT CAT

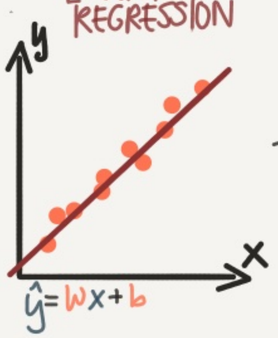
X → y

LOGISTIC REGRESSION



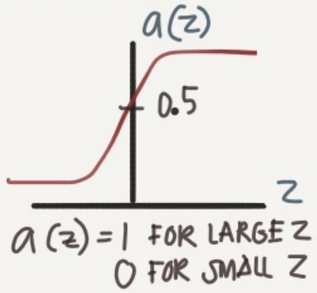
=

LINEAR REGRESSION

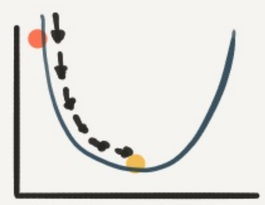


+

σ SIGMOID



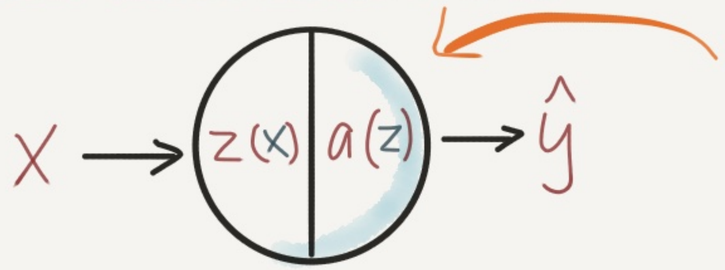
FINDING THE MINIMUM WITH GRADIENT DESCENT



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

REPEAT UNTIL YOU REACH BOTTOM (CONVERGE)

PUTTING IT ALL TOGETHER



$$z(x) = wx + b$$

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

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

REPEAT UNTIL IT CONVERGES

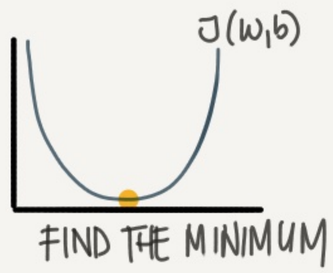
THE TASK IS TO LEARN w & b BUT HOW?

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

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

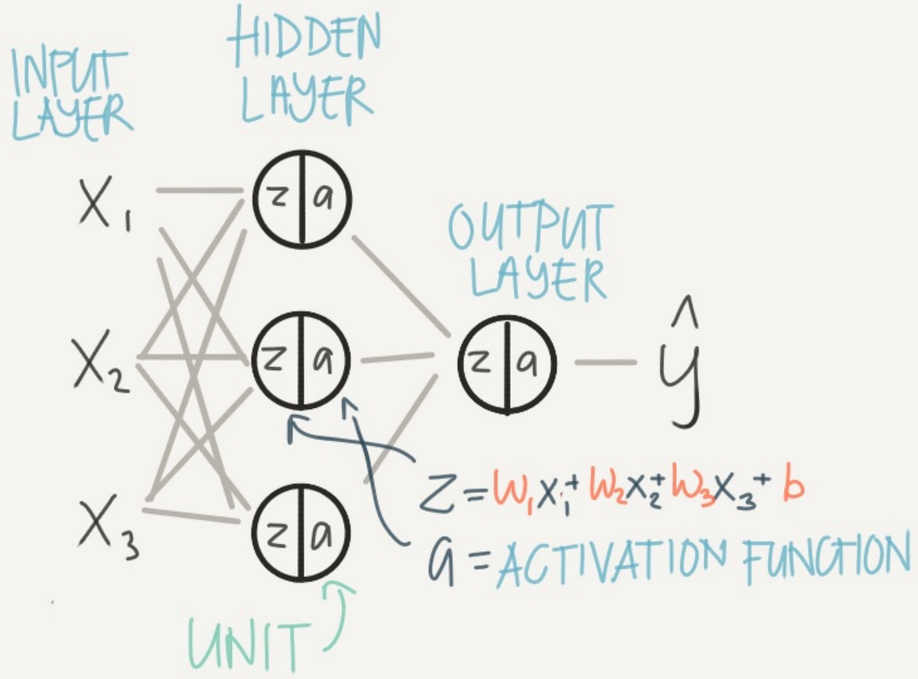
$$\text{COST} = J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

COST = LOSS FOR THE ENTIRE DATASET



SHALLOW NEURAL NETS

2 LAYER NEURAL NET



WHY ACTIVATION FUNCTIONS?

EX. WITH NO ACTIVATION – $a = z$

$$a^{[1]} = z^{[1]} = w^{[1]}x + b^{[1]} \quad \text{LAYER 1}$$
$$a^{[2]} = z^{[2]} = w^{[2]}a^{[1]} + b^{[2]} \quad \text{LAYER 2}$$

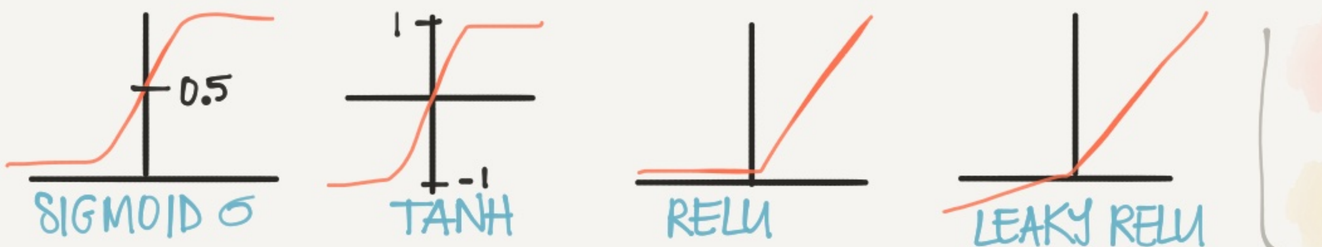
PLUG IN $a^{[1]}$

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

← LINEAR FUNCTION

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

ACTIVATION FUNCTIONS



SIGMOID σ
- BINARY CLASSIFIER - ONLY USED FOR OUTPUT LAYER
- SLOW GRAD DESCENT SINCE SLOPE IS SMALL FOR LARGE/SMALL VAL

TANH
- NORMALIZED ⇒ GRADIENT DESCENT IS FASTER

RELU
- DEFAULT CHOICE FOR ACTIVATION
- SLOPE = 1/0

LEAKY RELU
- AVOIDS UNDEF SLOPE AT 0 BUT RARELY USED IN PRACTICE

INITIALIZING w+b

WHAT IF: INIT TO 0

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

SETTING UP
YOUR ML APP

CLASSIC ML

100 - 10000 SAMPLES

DEEP LEARNING

1M SAMPLES

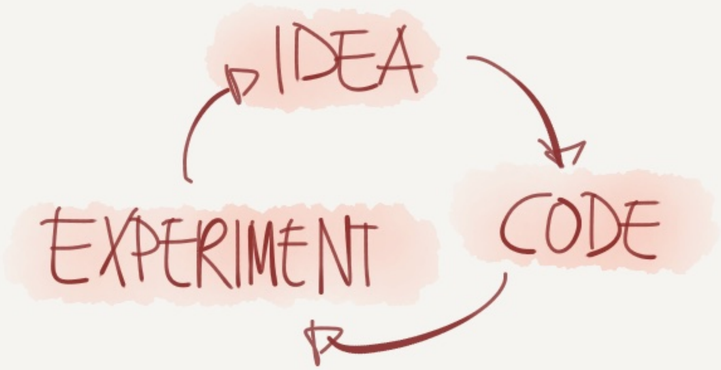
ALL FROM SAME PLACE → DISTRIBUTION

EX: TRAIN  PRO CAT PICS FROM INTERNET

DEV/TEST  BLURRY CAT PICS FROM APP

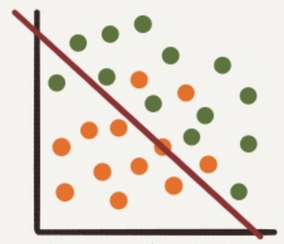
TIP

DEV & TEST SHOULD COME FROM SAME DISTRIBUTION

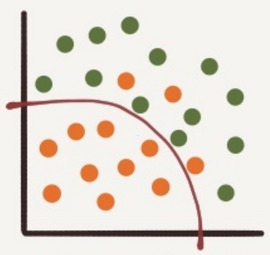


IMPROVING DEEP NEURAL NETS · COURSERA

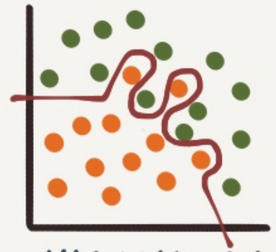
BIAS/VARIANCE



HIGH BIAS "UNDERFIT"



JUST RIGHT

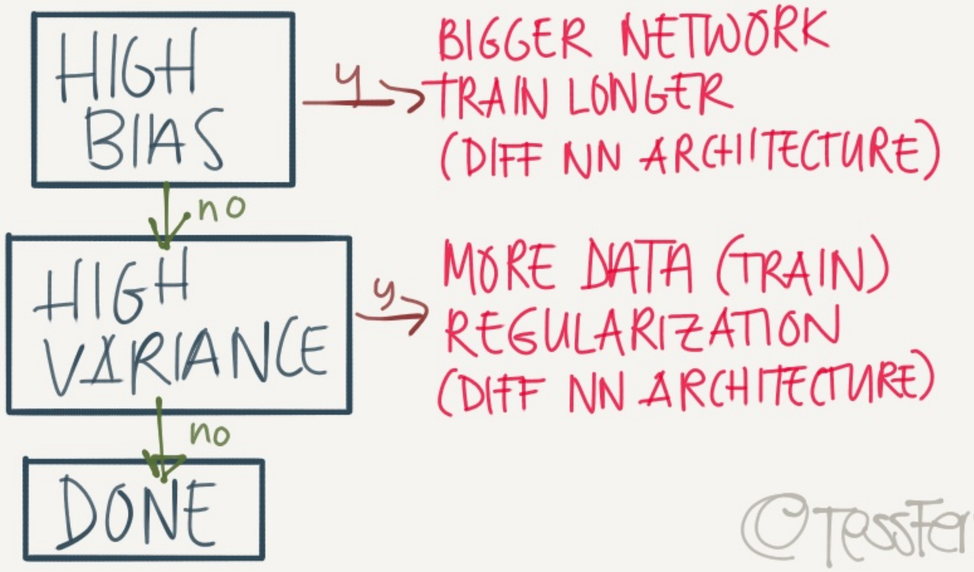


HIGH VARIANCE "OVERFIT"

	ERROR			
TRAIN	1%	15%	15%	0.5%
TEST	11%	16%	30%	1%
	HIGH VARIANCE	HIGH BIAS	HIGH BIAS & VARIANCE	LOW BIAS & VARIANCE

ASSUMING HUMANS GET 0% ERROR

THE ML RECIPE



REGULARIZATION
PREVENTING OVERFITTING

L2 REGULARIZATION

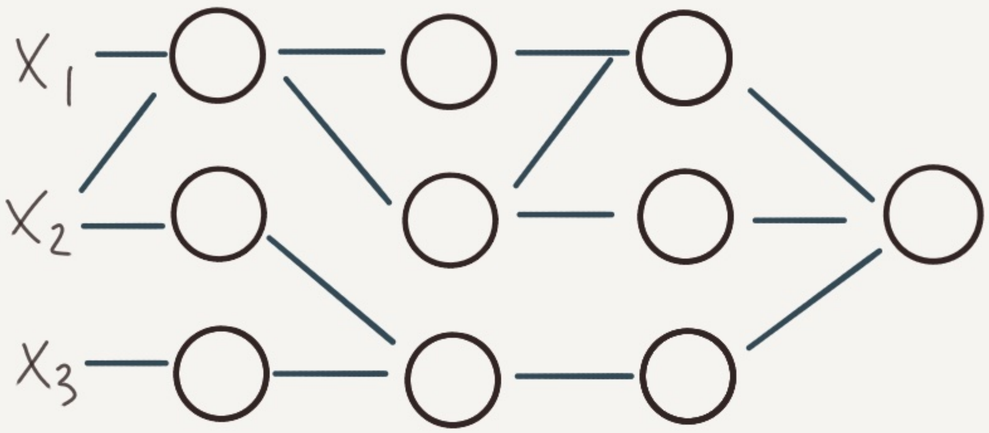
$$\text{COST}: J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y},y) + \frac{\lambda}{2m}\|w\|_2^2$$

← EUCLIDEAN NORM

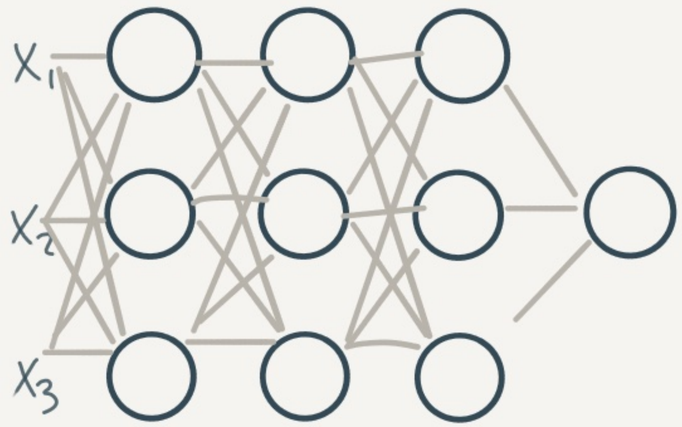
L1 REGULARIZATION

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

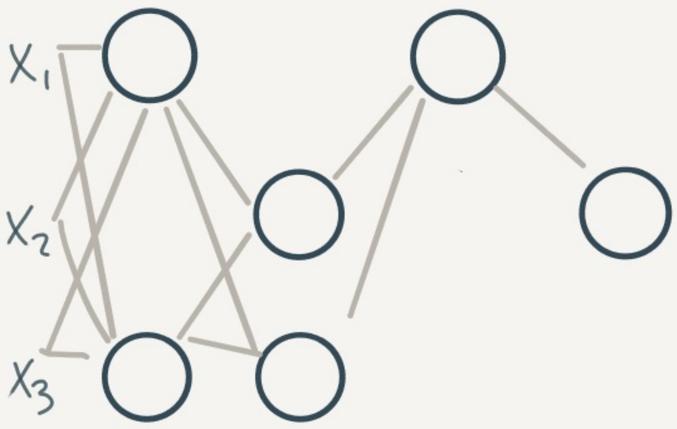
BOTH PENALIZE LARGE WEIGHTS ⇒
SOME WILL BE CLOSE TO Ø ⇒
SIMPLER NETWORKS



DROPOUT



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

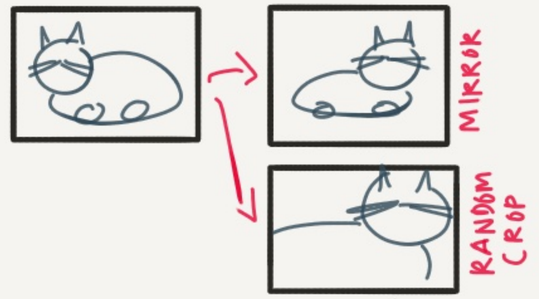


WE GET SIMPLER NWs & LESS CHANCE TO RELY ON SINGLE FEATURES

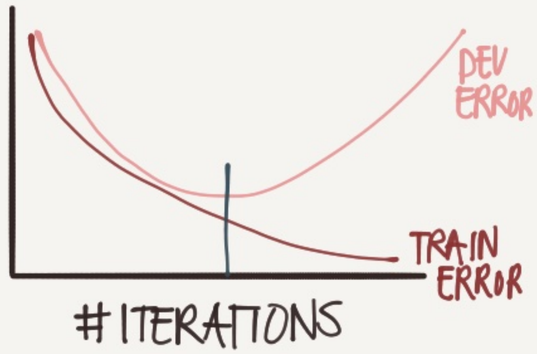
OTHER REGULARIZATION TECHNIQUES

DATA AUGMENTATION

GENERATE NEW PICS FROM EXISTING



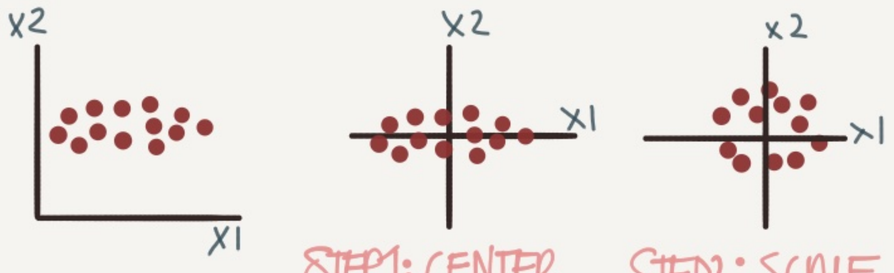
EARLY STOPPING



PROBLEM: AFFECTS BOTH BIAS & VARIANCE

OPTIMIZING TRAINING

NORMALIZING INPUTS



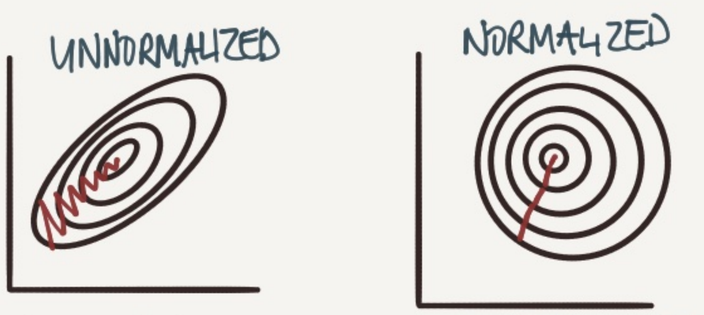
STEP1: CENTER AROUND 0,0

STEP2: SCALE SO VARIANCE IS SAME CX $-1 \rightarrow 1$

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

DEALING WITH VANISHING/EXPLODING GRADIENTS

EX: DEEP NW (L LAYERS)

$\hat{y} = W^{[L-1]} W^{[L-2]} \cdots W^{[1]} X + b$

IF $W = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \Rightarrow 0.5^{L-1} \Rightarrow$ VANISHING

OR $W = \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix} \Rightarrow 1.5^{L-1} \Rightarrow$ EXPLODING

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)

$n^{[l-1]}$ = #inputs

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

SETS THE VARIANCE

GRADIENT CHECKING

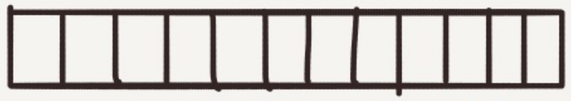
IF YOUR COST DOES NOT DECREASE ON EACH ITER YOU MAY HAVE A BACKPROP BUG.

GRADIENT CHECKING APPROXIMATES THE GRADIENTS SO YOU CAN VERIFY CALC.

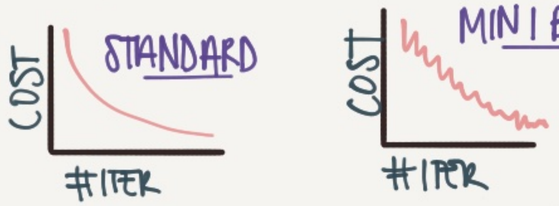
NOTE ONLY USE WHEN DEBUGGING SINCE IT'S SLOW

OPTIMIZATION ALGORITHMS

MINI-BATCH GRAD. DESCENT

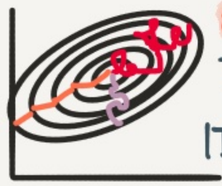


SPLIT YOUR DATA INTO MINI-BATCHES & DO GRAD DESCENT AFTER EACH BATCH
THIS WAY YOU CAN PROGRESS AFTER JUST A SHORT WHILE



CHOOSING THE MINIBATCH SIZE

SIZE = m → BATCH GRAD DESC.
SIZE = 1 → STOCHASTIC GRAD DESC

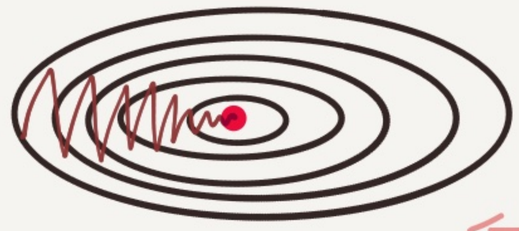


BATCH — TOO LONG PER ITERATION
MIDDLE
STOCHASTIC — LOOSE ALMOST ALL SPEED FROM VECTORIZATION

- 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

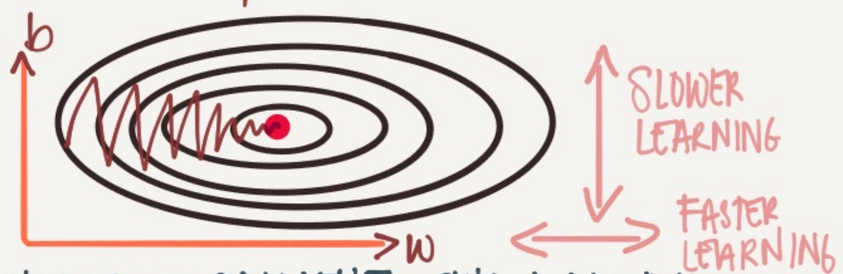
GRADIENT DESCENT W. MOMENTUM



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.

$$S_{dw} = \beta S_{dw} + (1-\beta)\, dw^2$$
$$S_{db} = \beta S_{db} + (1-\beta)\, db^2$$
$$w = w - \alpha \frac{dw}{\sqrt{S_{dw}}} \qquad b = b - \alpha \frac{db}{\sqrt{S_{db}}}$$

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

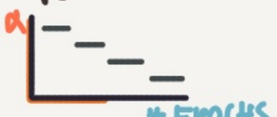
OPTION 1: $\alpha = \frac{1}{1 + \text{DECAYRATE} \cdot \text{EPOCH}} \alpha_0$

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

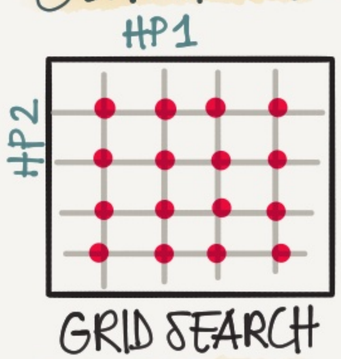
HYPERPARAM TUNING

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)

TESTING VALUES

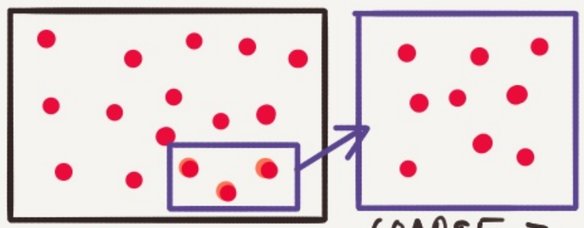
CLASSIC ML



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



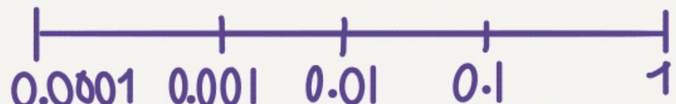
RANDOM SEARCH + COARSE → DENSE

USE AN APPROPRIATE SCALE

HIDDEN UNITS

UNIFORMLY RANDOM

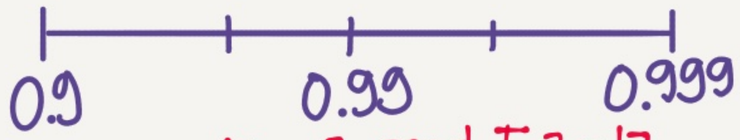
α LEARNING RATE



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

$\alpha = 10^r$

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

SPAWN LOTS OF MODELS W DIFF HP

GOOD IF YOU HAVE LOTS OF SPARE COMP POWER

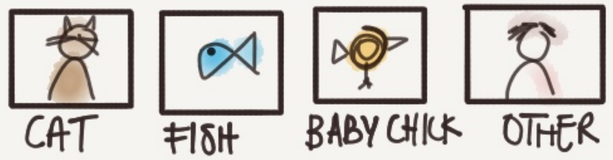
MISC. EXTRAS

BATCH NORMALIZATION

NORMALIZE LAYER OUTPUT

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

MULTICLASS CLASSIFIC.

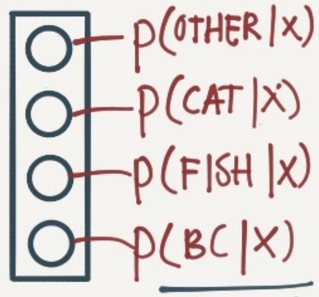


C = # CLASSES = 4

SOFTMAX ACTIVATION

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

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



SUM: 1

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

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

★ DEFINE OPTIMIZING VS SATISFICING METRICS

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

MAXIMIZE ACC.
GIVEN TIME ≤ 100ms

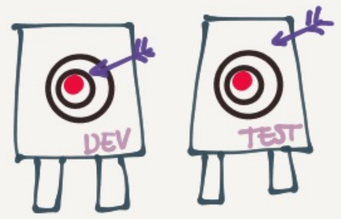
ACCURACY = OPTIMIZING
RUNTIME = SATISFICING

SELECTING YOUR DEV/TEST SETS

DATA

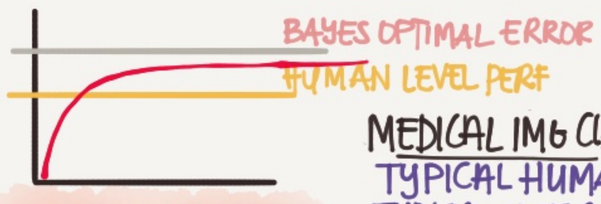
US
UK
~~EUROPE~~
S.AM
INDIA
CHINA
AUST.

OPTION 1:
DEV = UK, US, EUR
TEST = REST



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

HUMAN LEVEL PERF



WHY DOES ACC SLOW DOWN WHEN WE SURPASS 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)

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

CAT CLASSIFICATION

BLURRY ↓

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

HUMAN
| AVOIDABLE BIAS
TRAIN
| VARIANCE
DEV

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

	A	B	
HUMAN	0.5	0.5	← AVOIDABLE BIAS
TRAIN ERR	0.6	0.3	← VARIANCE
DEV ERR	0.8	0.4	
AVOID. BIAS	0.1	?	↳ DON'T KNOW IF WE OVERFIT OR IF WE'RE CLOSE TO BAYES

OPTIONS TO PROCEED ARE UNCLEAR

STRUCTURING ML PROJECTS • COURSERA

ERROR ANALYSIS

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

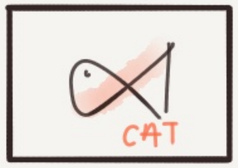
1. PICK 100 MIS-LABLED
2. COUNT ERROR REASONS

	DOG	BLURRY	INSTA FILTER	BIG CAT	...
1	1		1		
2				1	
3		1			
...					
100			1		
	5	...			

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.

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