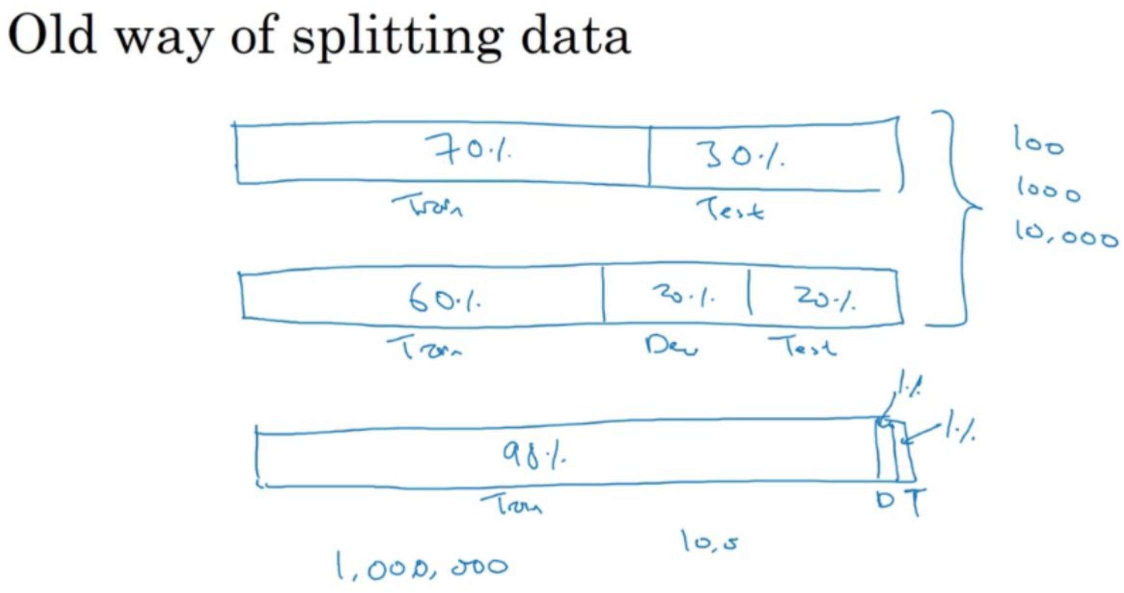
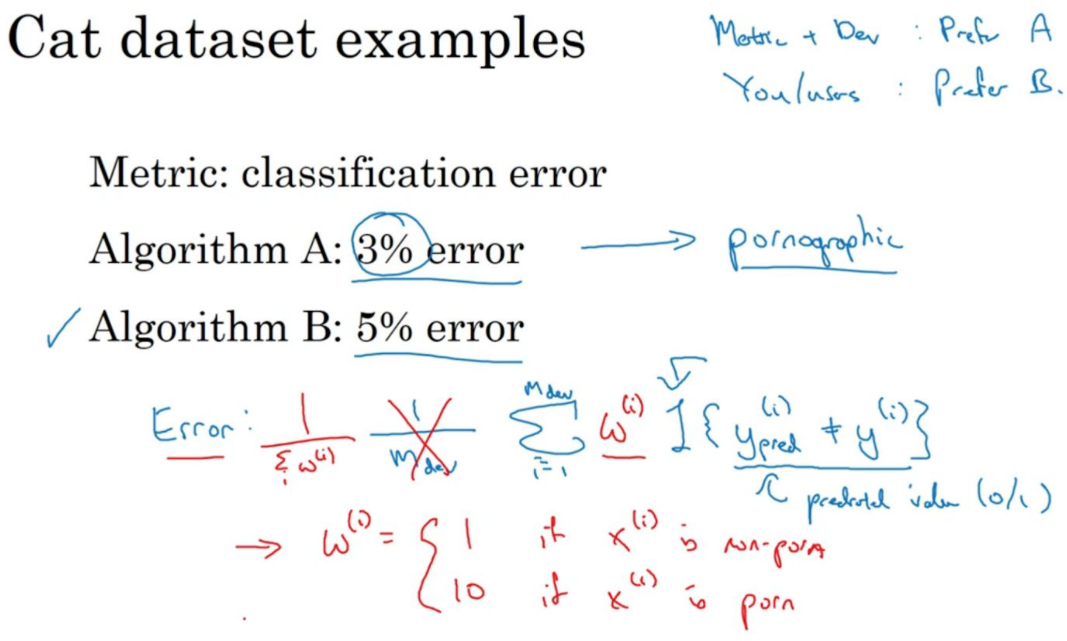
COURSE 3: STRUCTURING ML PROJECTS



.

If million examples, just go for 98-1-1 instead of such a large division which was ok for lower number of training examples

How to change metric from just error to “worse” error like pornographic images?



Cat dataset examples

Metric + Dev : Prefer A

You/users : Prefer B.

Metric: classification error

Algorithm A: 3% error → pornographic

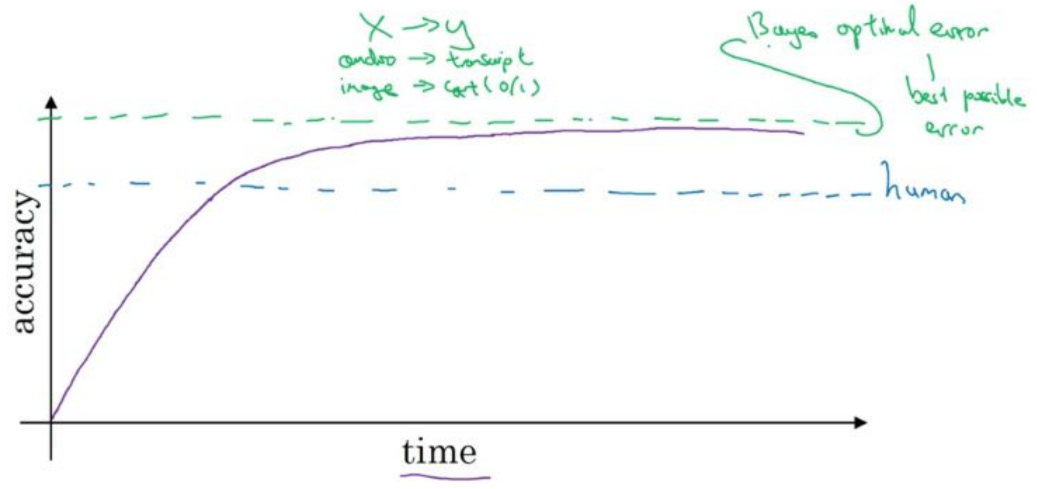
✓ Algorithm B: 5% error

$$\text{Error}: \frac{1}{\sum_i w^{(i)}} \cancel{\frac{1}{m_{dev}}} \sum_{i=1}^{m_{dev}} w^{(i)} \, \mathcal{I}\{\underbrace{y_{pred}^{(i)}}_{\text{predicted value (0/1)}} \neq y^{(i)}\}$$

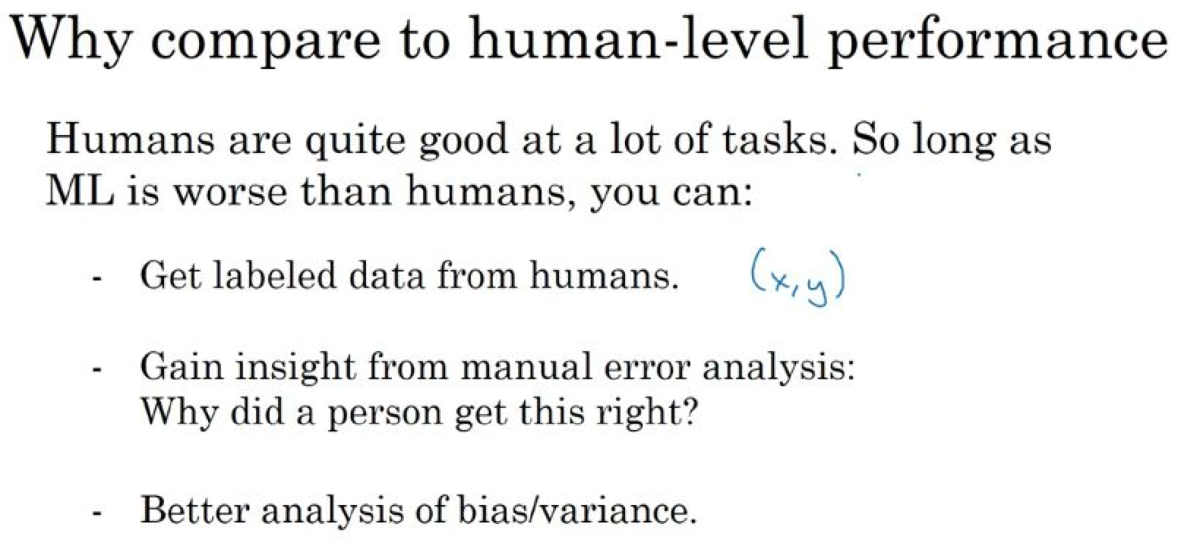
$$\rightarrow w^{(i)} = \begin{cases} 1 & \text{if } x^{(i)} \text{ is non-porn} \\ 10 & \text{if } x^{(i)} \text{ is porn} \end{cases}$$

BAYES OPTIMAL ERROR

Comparing to human-level performance



With humans you can do better error analysis, feature engineering, get labeled data, better understanding of bias and variance.



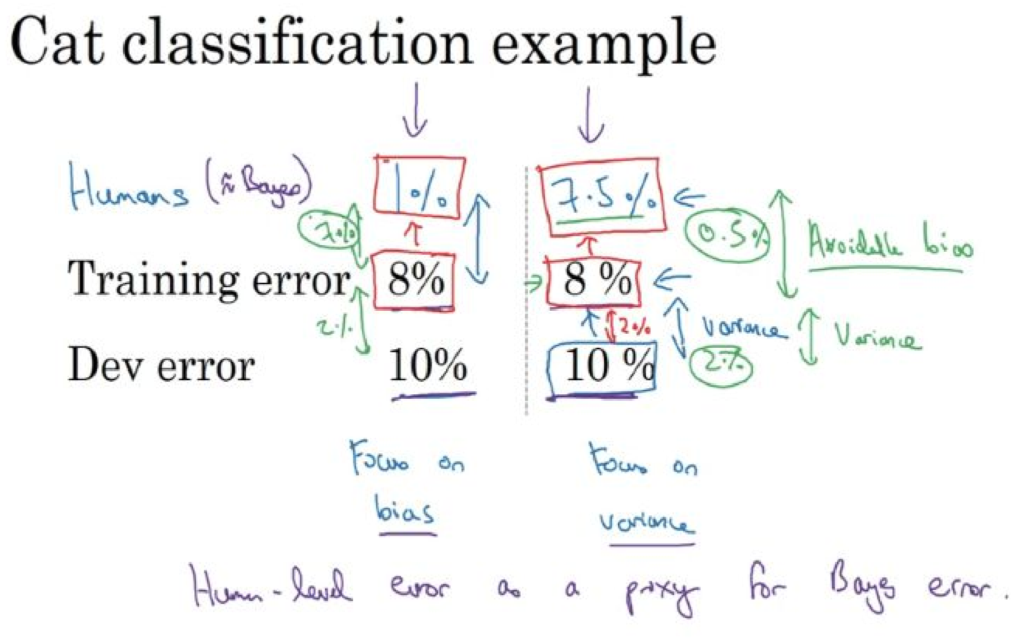
Why compare to human-level performance

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

- Get labeled data from humans. (x, y)
- Gain insight from manual error analysis: Why did a person get this right?
- Better analysis of bias/variance.

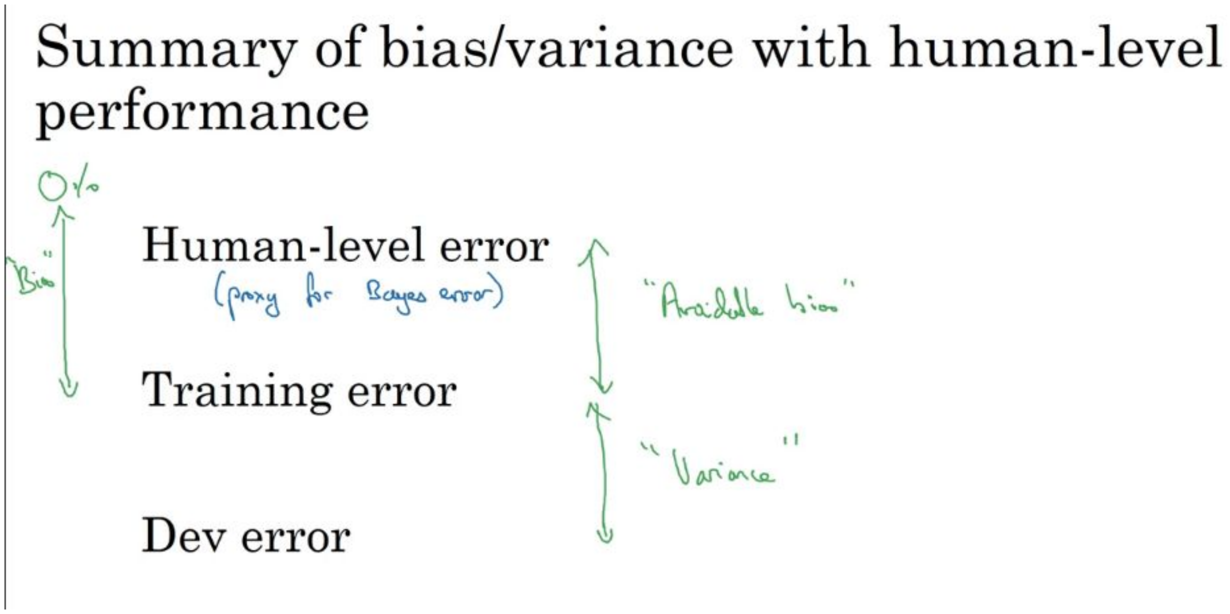
Human level error is a proxy for bayes optimal error --

Focus on bias is to go for more complex model - like deeper neural networks
Focus on variance is to go for more training data or use regularization.



Cat classification example

Humans (≈ Bayes)	1%	7.5%
Training error	8%	8%
Dev error	10%	10%



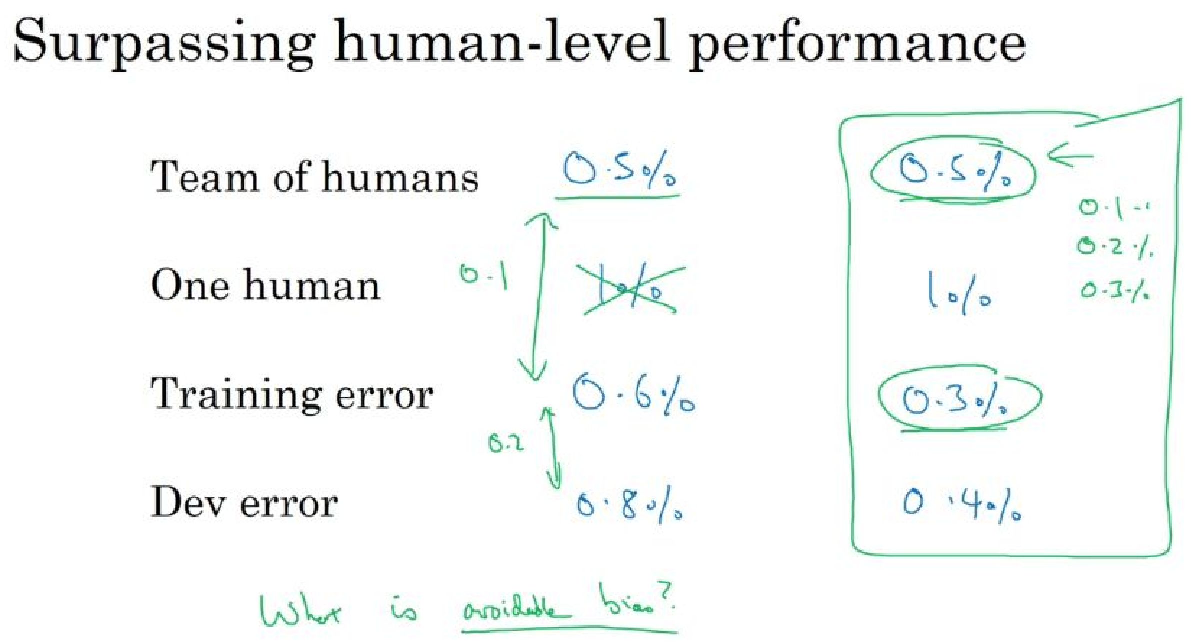
Summary of bias/variance with human-level performance

Human-level error

Training error

Dev error

Once your training and dev error beats Team of Humans --- progress is less clear. What is bias and what is variance?



Surpassing human-level performance

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

0.1

0.2

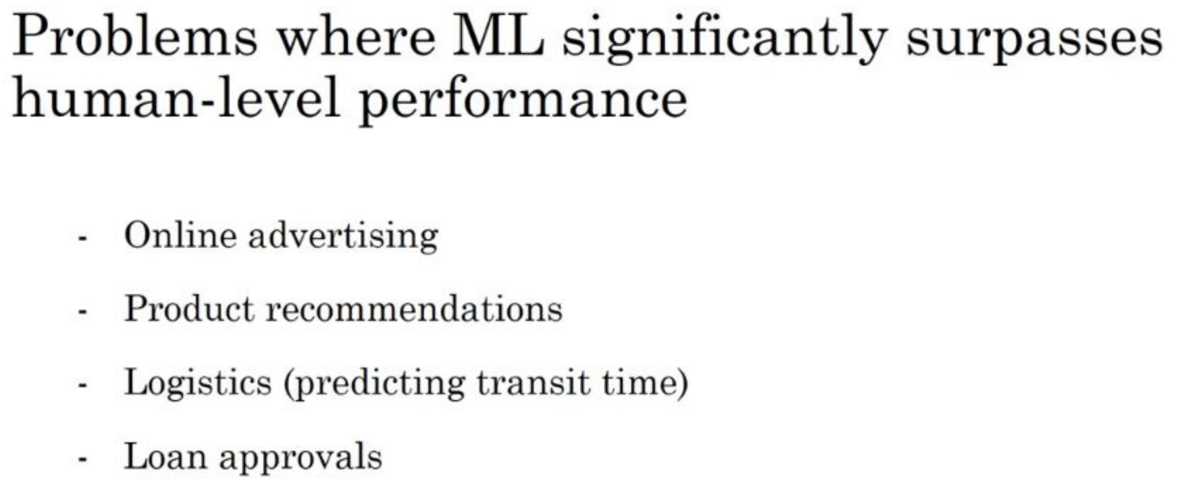
0.1%

0.2%

0.3%

What is avoidable bias?

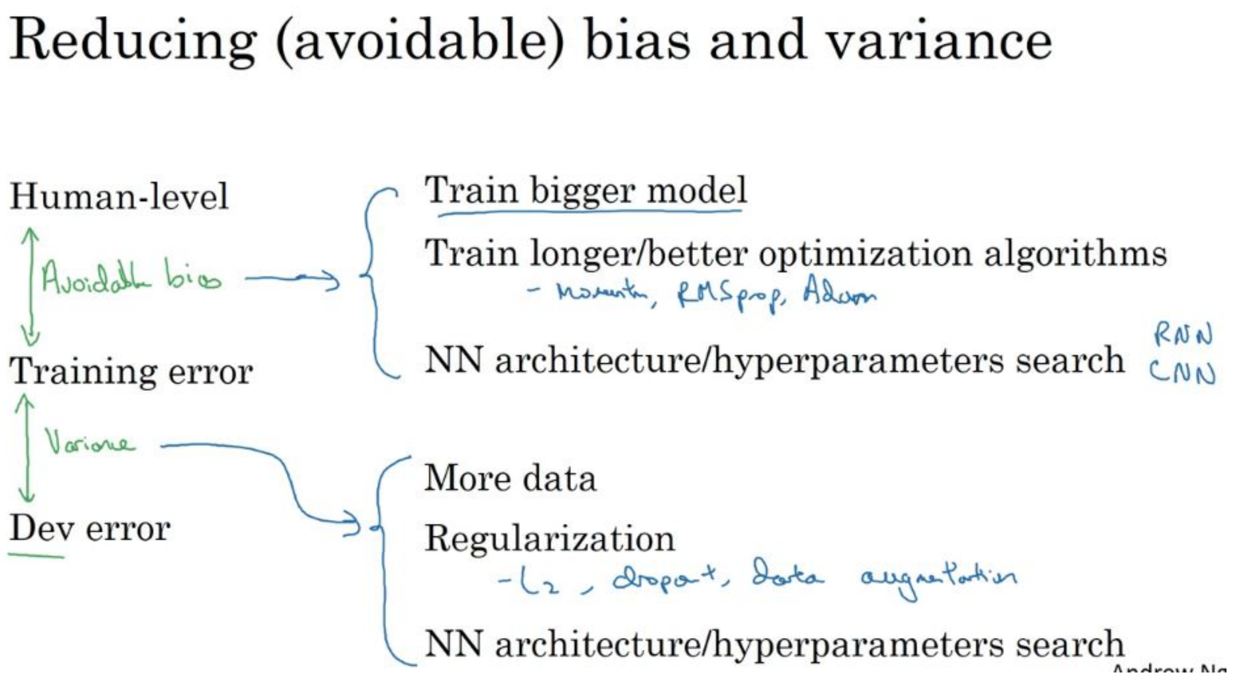
Humans are better at natural perception tasks like language. But when it comes to structured data, we can use ML to beat even humans.



Problems where ML significantly surpasses human-level performance

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

How to get better model?



Reducing (avoidable) bias and variance

Human-level

Avoidable bias →
- Train bigger model
- Train longer/better optimization algorithms
  - momentum, RMSprop, Adam
- NN architecture/hyperparameters search RNN CNN

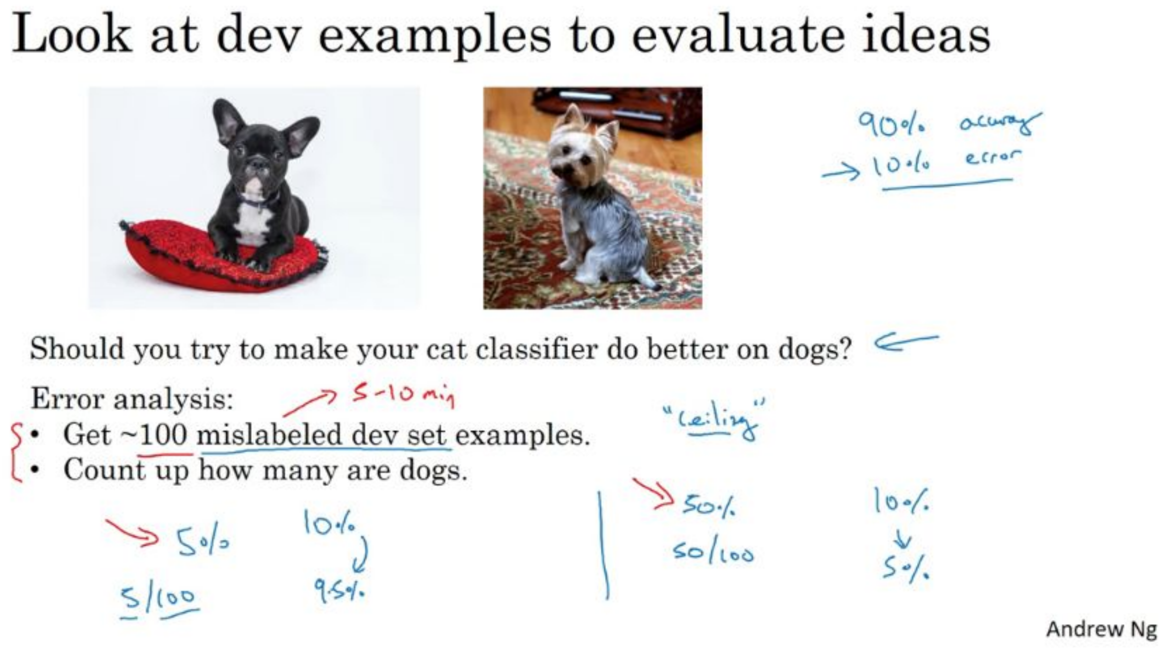
Training error

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

Dev error

ERROR ANALYSIS (WEEK 2)

1. Manually analyze the mislabelled cases and see whats the distribution like. Just 5-10 min to evaluate. Also helps get closer idea to Bayes estimate.



Look at dev examples to evaluate ideas

90% accuracy
→ 10% error

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

Error analysis: → 5-10 min
- Get ~100 mislabeled dev set examples.
- Count up how many are dogs.

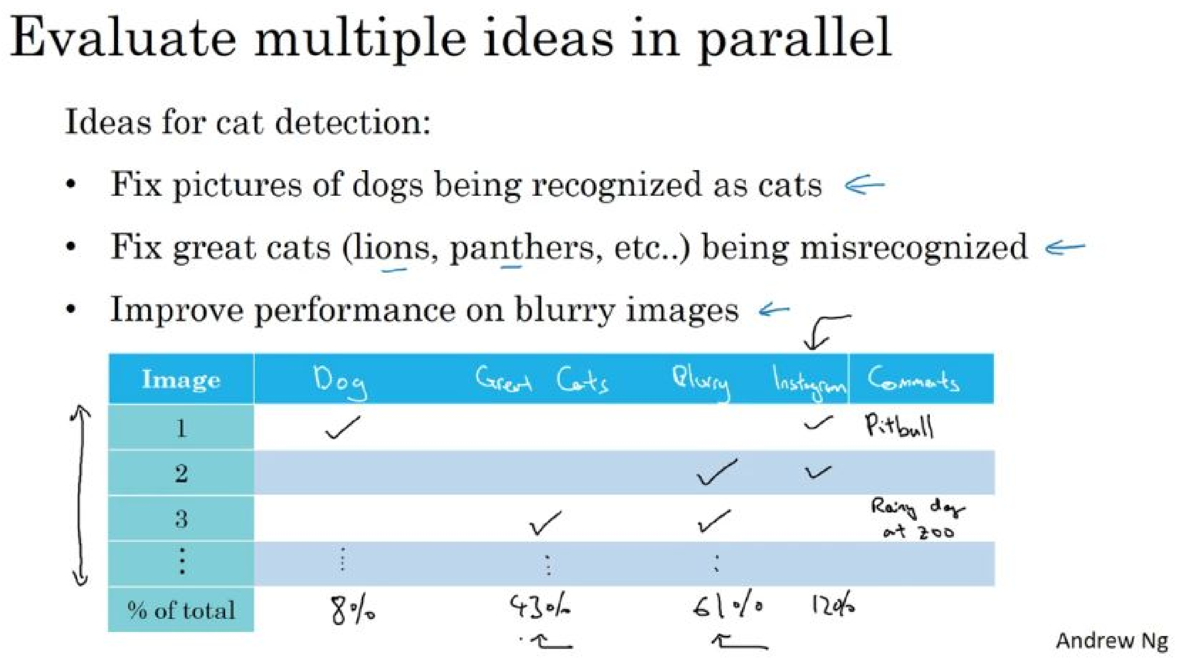
"ceiling"

→ 5% 5/100 10% → 9.5%

→ 50% 50/100 10% → 5%

Andrew Ng

To do better performance, just bigger model might not be best. You might want to see whether its blurry images that are creating problem or is it dog imageS? Maybe you need dog data. This will help you do better feature engineering also.



Evaluate multiple ideas in parallel

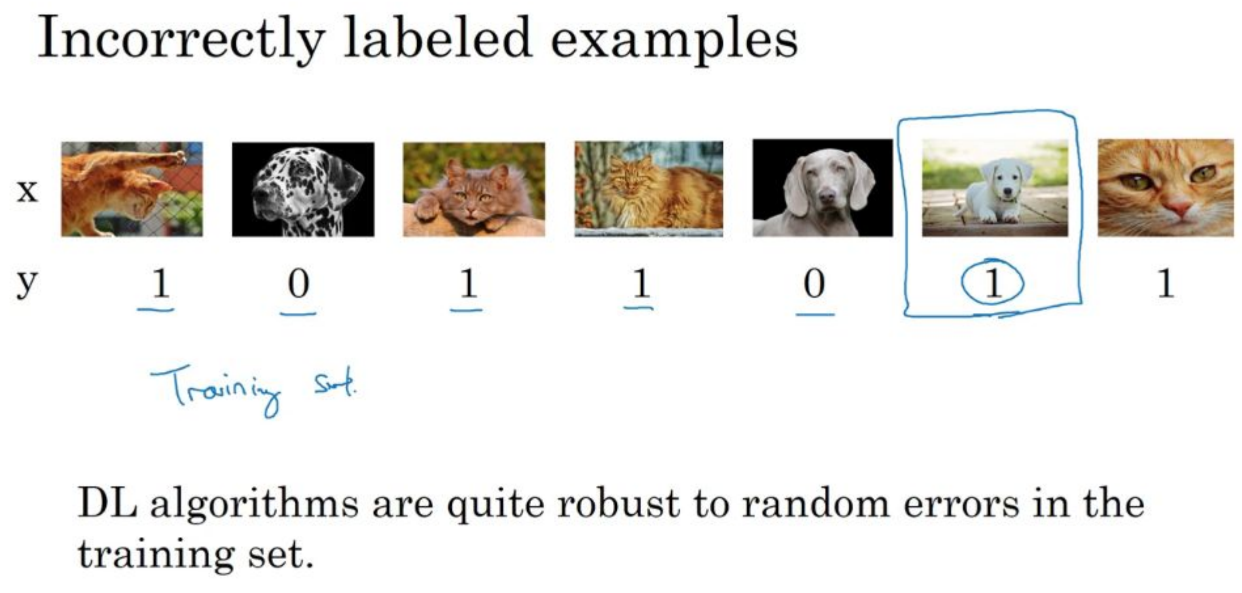
Ideas for cat detection:

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

Image	Dog	Great Cats	Blurry	Instagram	Comments
1	✓			✓	Pitbull
2			✓	✓	
3		✓	✓		Rainy day at zoo
⋮	⋮	⋮	⋮		
% of total	8%	43%	61%	12%	

Andrew Ng

Randomly mislabeled mistakes dont make much of a difference! But if most dogs are called cats (systematic errors) then its a problem!

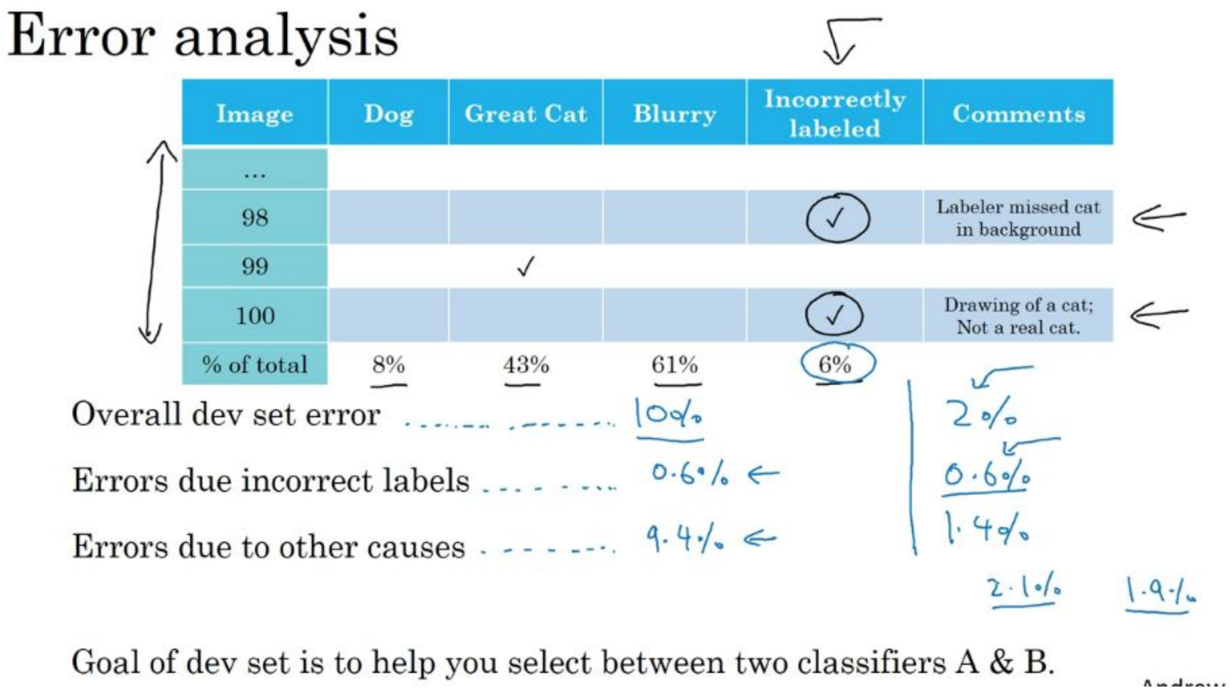


Incorrectly labeled examples

x							
y	1	0	1	1	0	1	1

Training set.

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



Error analysis

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

Overall dev set error	10%	2%
Errors due incorrect labels	0.6% ←	0.6%
Errors due to other causes	9.4% ←	1.4%

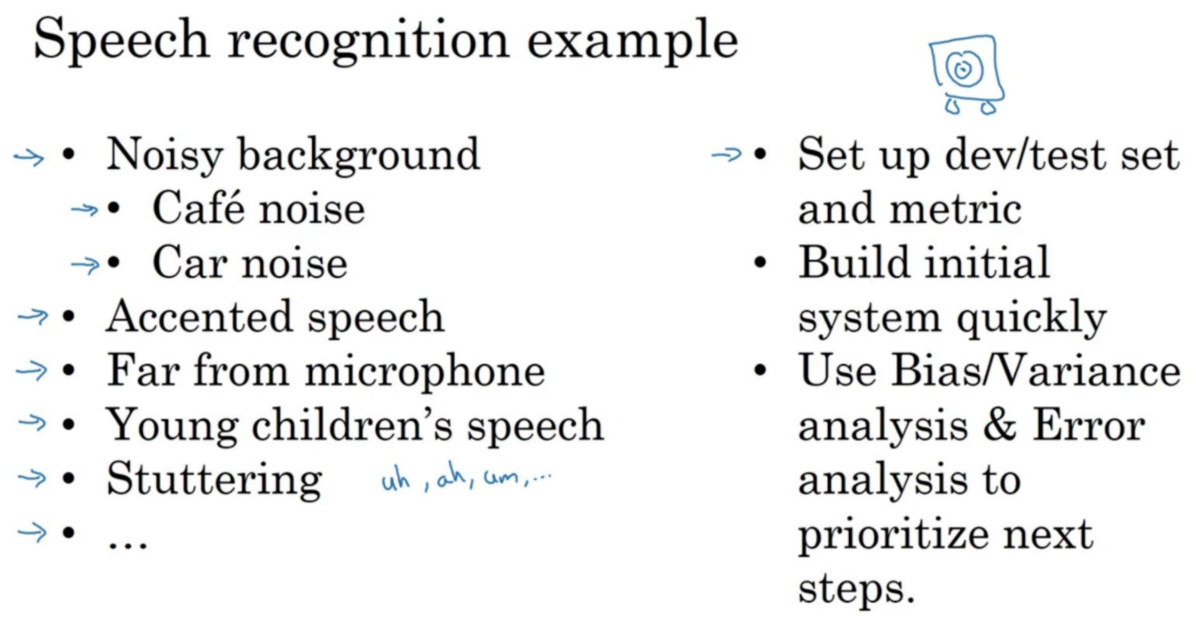
2.1% 1.9%

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

In ML Classifier, out of the 7m parts, 250k had misclassifications. That is 3.5% error.
How many of them were actual bad data --- how many mistakes by ML algo?
So maybe 2.5% are just bad labels and maybe 1% are mistakes.

Dev/ test set should be from same distributions,
Training set might be little different.

Which area to focus on?

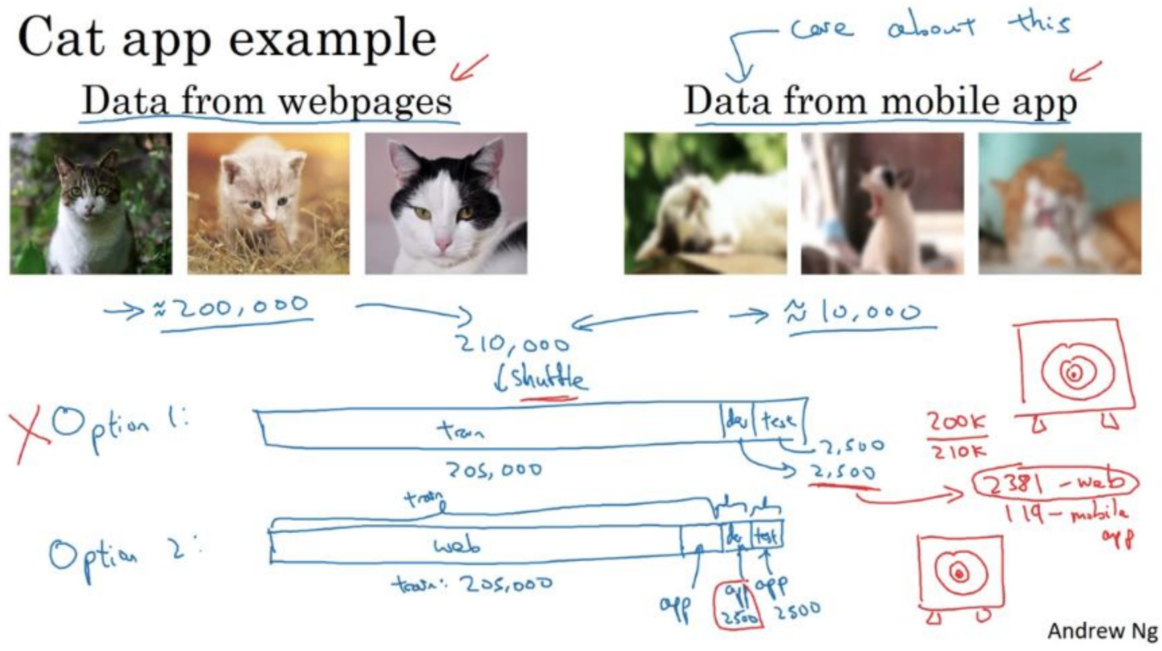


Speech recognition example

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

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

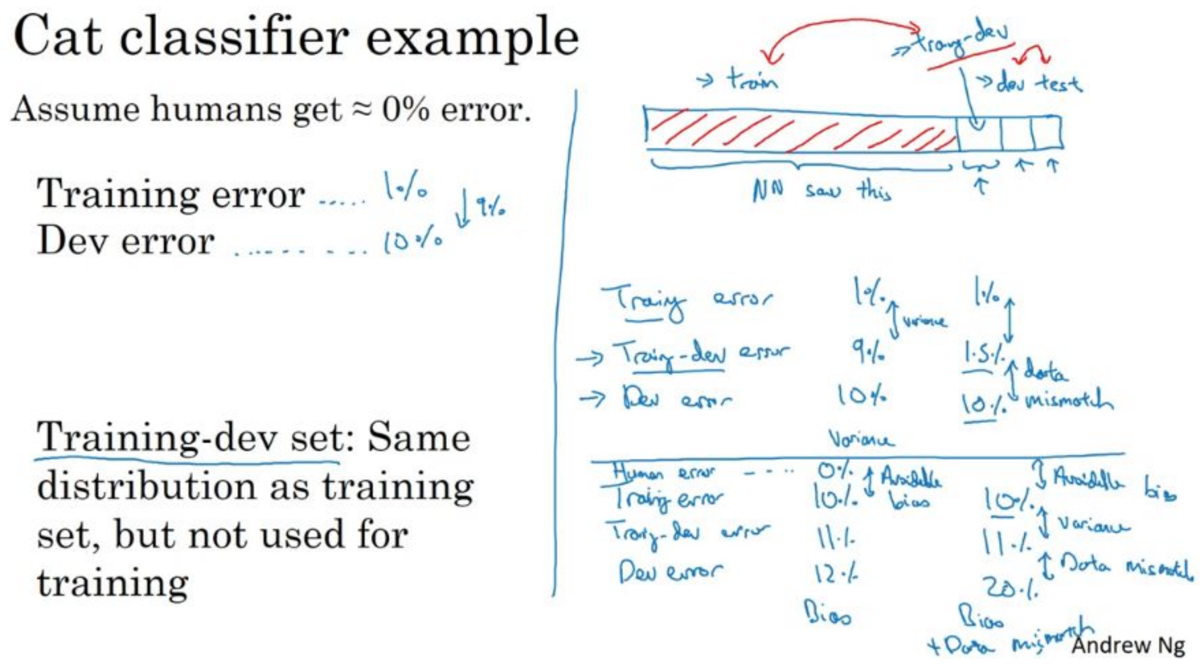
You want blurry images in dev and test set distributions. Instead of randomly shuffling them fully.



So basically keep dev and test set from distributions that ACTUALLY matter

Training error 1%
Dev error 10% ----- but since distribution is different, how do you know if its bias or variance problem? As in dev set is just much harder as its blurry images how do we know for sure what are the metrics?

So you need to again break training into training-real and training-dev sets.



Cat classifier example

Assume humans get ≈ 0% error.

Training error 1%
Dev error 10%

↓ 9%

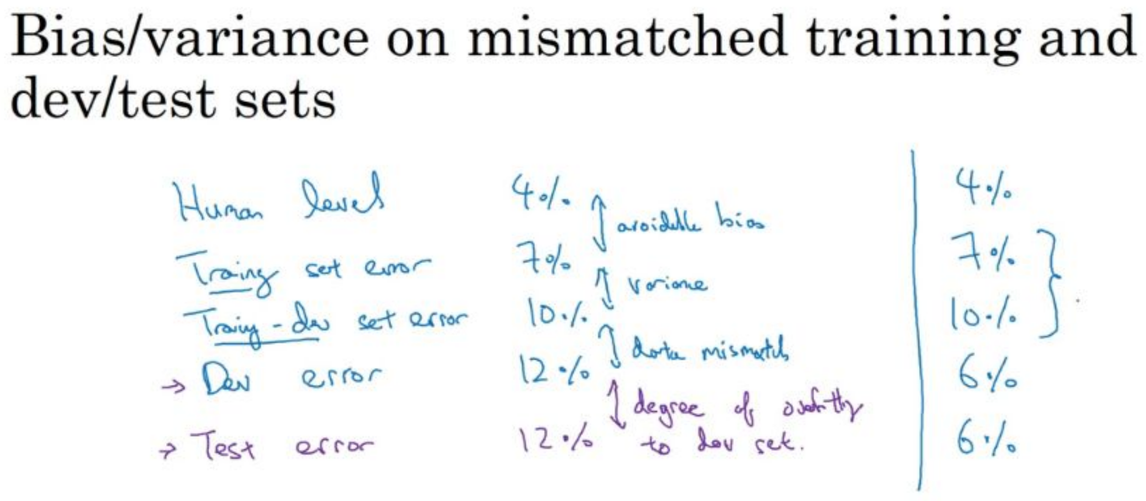
Training-dev set: Same distribution as training set, but not used for training

→ train
→ traing-dev
→ dev test
NN saw this

Traing error	1%	variance	1%	
→ Traing-dev error	9%		1.5%	data mismatch
→ Dev error	10%		10%	
	Variance			

Human error	0%	Avoidable bias		Avoidable bias
Traing error	10%		10%	
Traing-dev error	11%		11%	variance
Dev error	12%		20%	Data mismatch
	Bias		Bias + Data mismatch	

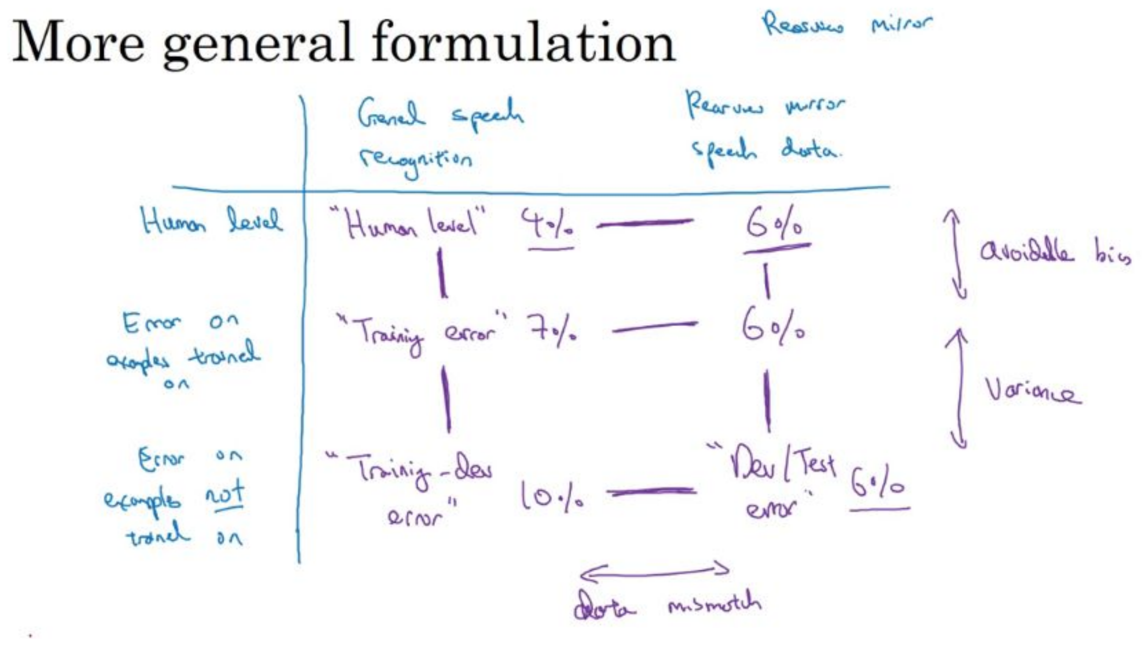
Andrew Ng



Bias/variance on mismatched training and dev/test sets

Human level	4%	avoidable bias	4%
Traing set error	7%	variance	7%
Traing-dev set error	10%	data mismatch	10%
→ Dev error	12%	degree of overfitting to dev set.	6%
→ Test error	12%		6%

Sometimes dev set is actually easier than training set then it can go down.

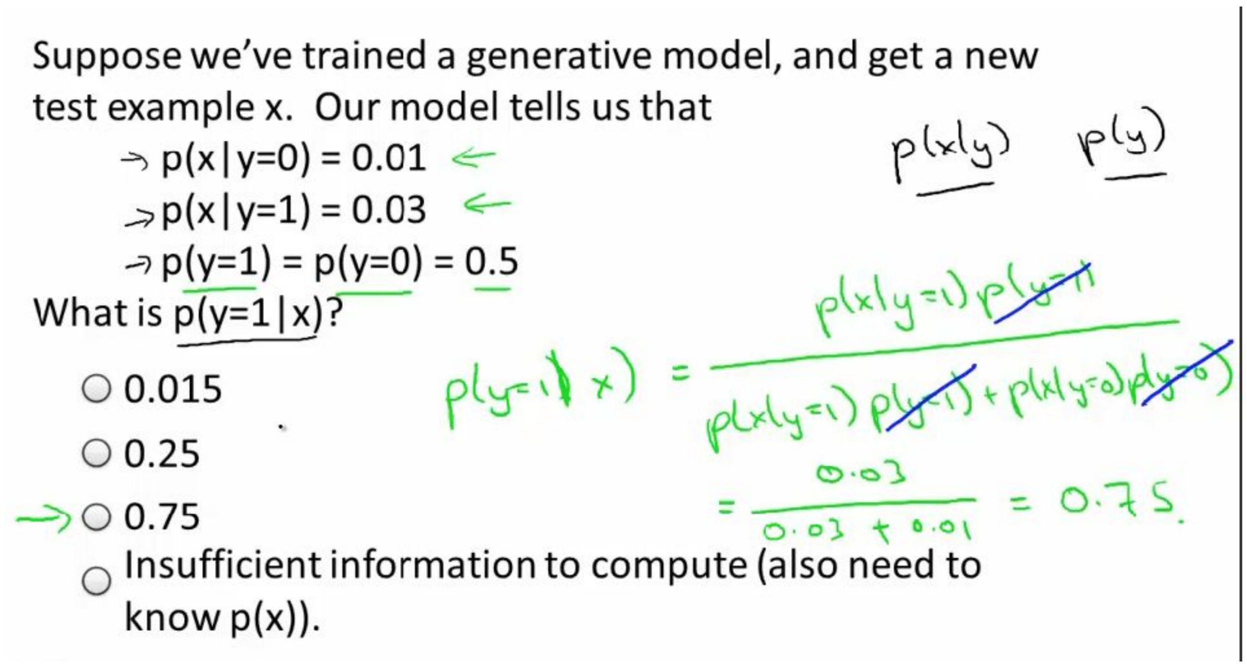


More general formulation

Rearview Mirror

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

data mismatch



Suppose we've trained a generative model, and get a new test example x. Our model tells us that

- $p(x|y=0) = 0.01$
- $p(x|y=1) = 0.03$
- $p(y=1) = p(y=0) = 0.5$

What is $p(y=1|x)$?

$\underline{p(x|y)}$ $\underline{p(y)}$

- 0.015
- 0.25
- [✓ 0.75]
- Insufficient information to compute (also need to know p(x)).

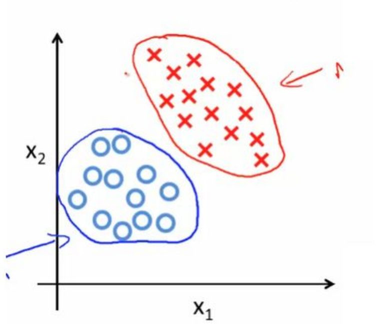
$$p(y=1|x) = \frac{p(x|y=1)\cancel{p(y=1)}}{p(x|y=1)\cancel{p(y=1)} + p(x|y=0)\cancel{p(y=0)}}$$

$$= \frac{0.03}{0.03 + 0.01} = 0.75.$$

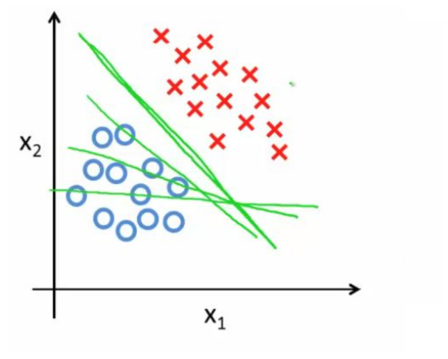
NB vs Logistic Regression

Started trying out different algorithms including Naive Bayes, Logistic Regression, Linear SVM etc.
NB is a generative classifier, it uses Bayes rule to calculate P(y|x) from P(x|y) and P(y) or "probability of features given a category multiplied by the prior"

Logistic Regression is a discriminative classifier, it directly predicts P(y|x) "probability of a category given certain words or features"
Discriminative classifiers overall perform better

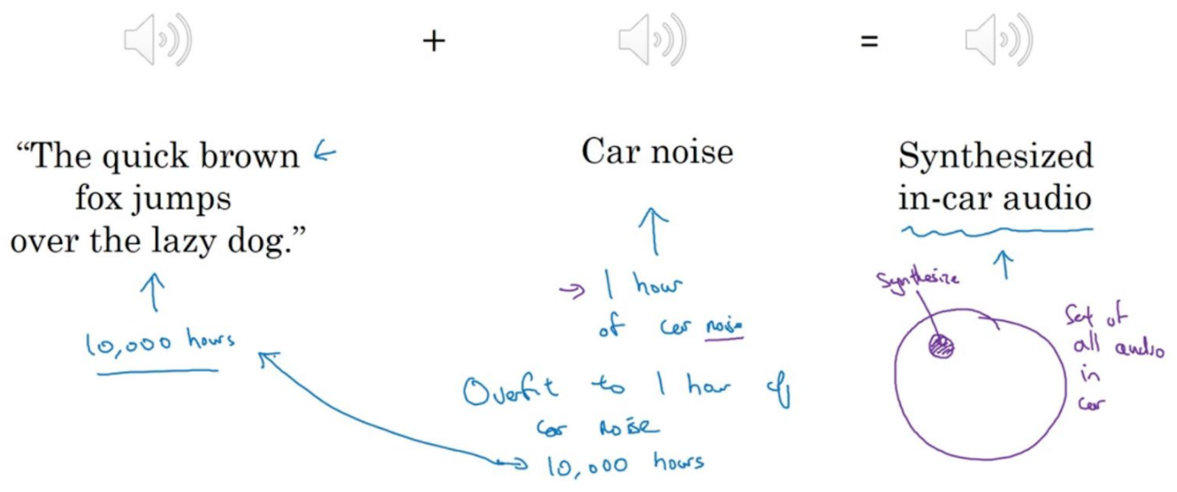


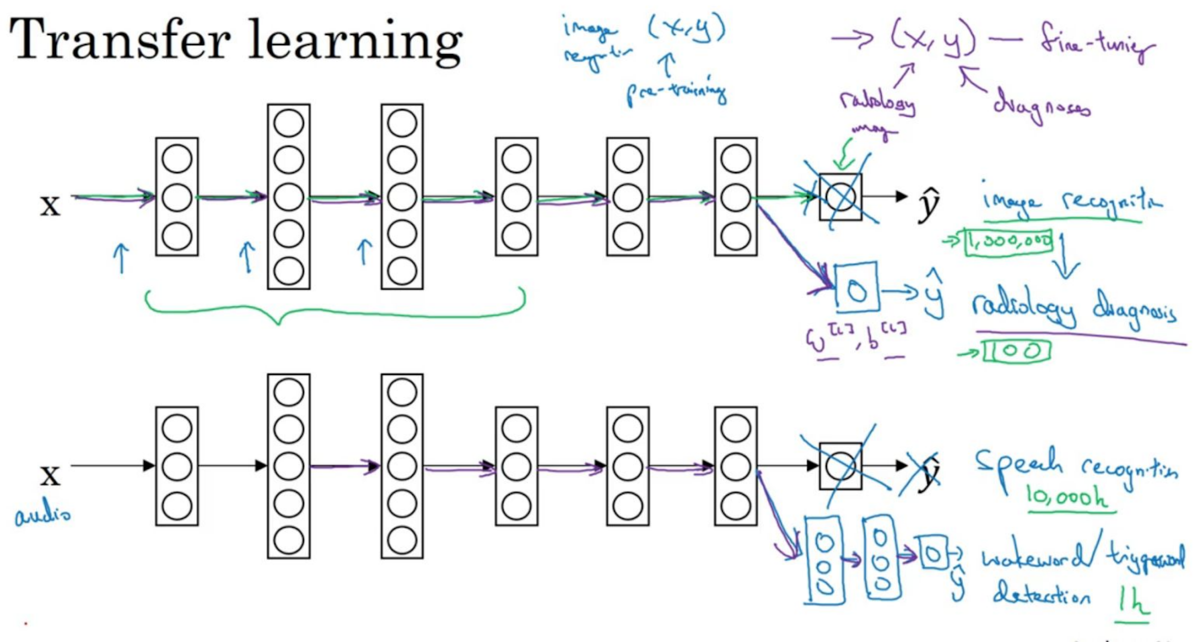
Generative / NB



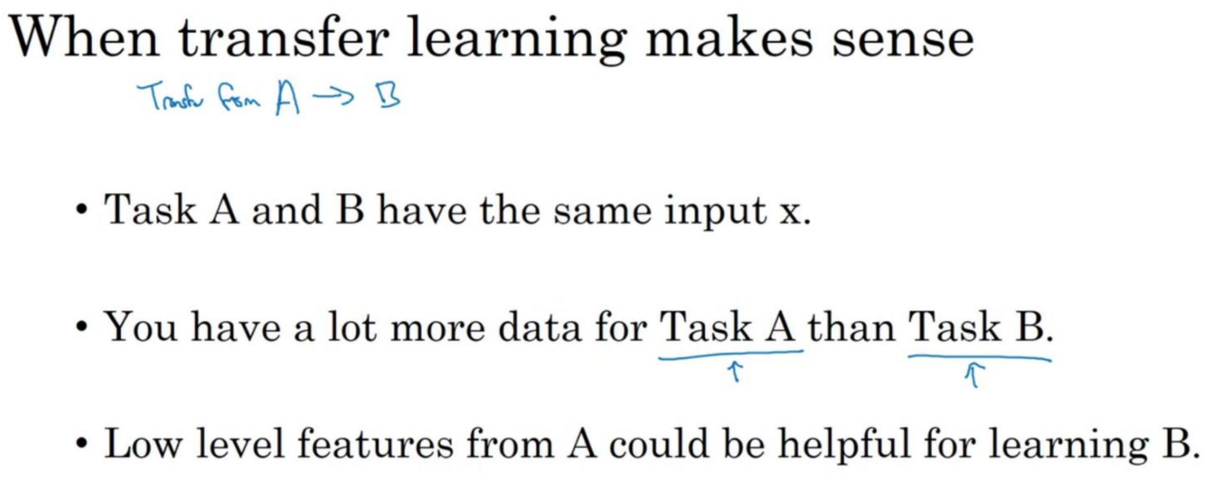
Discriminative/ Logistic Reg

Artificial data synthesis





So for transfer learning you can re-train all network with new data set or only the last layer. If it is all the layers then its **pre-tuning** to initialize all the weights (instead of random) you'll need large data set for this. And then **fine tuning** later with actual data set. Or you can only train last layer's weights and biases.



When transfer learning makes sense

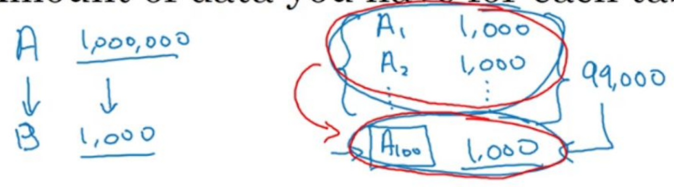
Transfer from $A \rightarrow B$

- Task A and B have the same input x.
- You have a lot more data for $\underset{\uparrow}{\underline{\text{Task A}}}$ than $\underset{\uparrow}{\underline{\text{Task B}}}$.
- Low level features from A could be helpful for learning B.

Multi-Task Learning: More than one label at output. Does an image have a stop sign, traffic light, pedestrian and car. Instead of 4 neural networks for each label separately, but one NN is better as you can share same low level features.

When multi-task learning makes sense

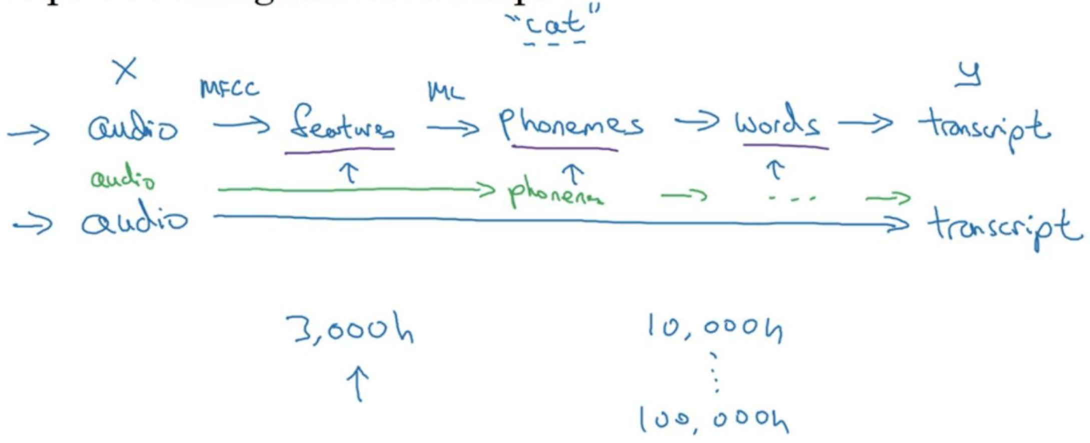
- Training on a set of tasks that could benefit from having shared lower-level features.
- Usually: Amount of data you have for each task is quite similar.



- Can train a big enough neural network to do well on all the tasks.

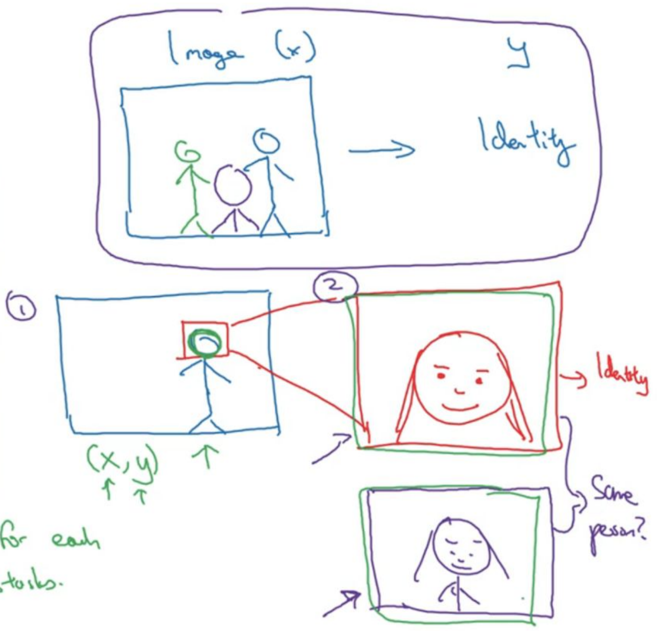
What is end-to-end learning?

Speech recognition example



Face recognition

[Image courtesy of Baidu]



Have data for each of 2 subtasks.

Applying end-to-end deep learning

Key question: Do you have sufficient data to learn a function of the complexity needed to map x to y?

$X \rightarrow y$

$$\begin{tikzcd}
\text{image, radar, lidar} \arrow[r, "DL"] \arrow[dr, "DL"'] & \text{cars} \arrow[r, "\text{motion planning}"] & \text{route} \arrow[r, "\text{control}"'] & \text{steering} \\
& \text{pedestrians} \arrow[ur] & &
\end{tikzcd}$$

- Use DL to learn individual components
- Carefully choose X → Y depending what tasks you can get data for.

$\text{image} \longrightarrow \text{steering}$

\