ML Strategy Week 1

Orthogonalization (每一步都有对应的解决方法，不能混着来)

- Fit training set well on Cost function (bigger network, Adam...)
  ↓
  dev (Regularization, bigger training set)
  ↓
  test set (bigger dev set)
  ↓
  perform well in the real world (Change dev set or cost function)

- thing like 'early stop', is not a good orthogonalization, because it influences many things in the same time

Setting up your goal

Single Number Evaluation Metric

Example: Rather than use precision and recall - use the F1 Score

which F_1 score = $\frac{2}{\frac{1}{P}+\frac{1}{R}}$ (Harmony mean of precision and recall)

△ a good dev set + single real number evaluation metric

Satisficing and optimizing metric

Example (optimizing: Accuracy; satisficing: Running time)

Classifer	Accuracy	Running time
A	90%	80ms
B	92%	95 ms
C	95%	1500 ms

Train/dev/test distribution

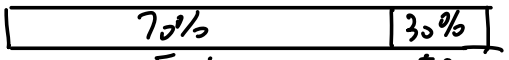
Cat classification dev/test sets
↳ development set, hold out cross-validation setup

▲ Randomly shuffle into dev/test set

▲ Choose a dev set and test set to reflect data you expect to get in the future and consider important to do well on. (Should be from same distribution!!!)

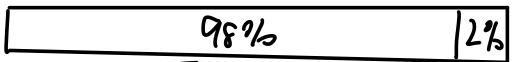
Size of the dev and test sets

old way of splitting data



10000

deep learning:



100, 001, 000, 0...

When to change Dev/Test sets and metrics?

- Can't rank correctly → change the metrics

Example:

metric: classification error

Algorithm A: 3% error → porn photo

Algorithm B: 5% error

According to current metric, Algorithm A is better one. but obviously the porn photo is intolerated. So we have to modify the metric (e.g. add some weight on porn photo)

$$\text{Error} = \frac{1}{\sum_i w^{(i)}} \sum_{i=1}^{M_{dev}} w^{(i)} I\{ y_{prep}^{(i)} \neq y^{(i)} \}$$

$$w^{(i)} = \begin{cases} 1 & \text{if } x^{(i)} \text{ is not porn} \\ 10 & \text{if } x^{(i)} \text{ is porn} \end{cases}$$

Orthogonalization

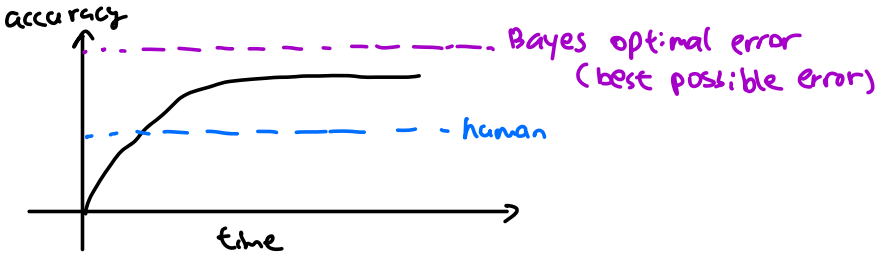
① place target

② Shoot at target

▲ If doing well on your metric + dev/test set does not correspond to doing well on your application, change your metric and/or dev/test set.

Comparing to Human-level Performance

Why human-level performance?



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

① Get labeled data from humans

② Gain insight from manual error analysis:
Why did a person get this right?

③ Better analysis of bias/variance

Avoidable bias (Before surpassing human performance)

Human (proxy for Bayes error)	1%	7.5%	↕ avoidable bias
Training error	8%	8%	
Dev error	10%	10%	
	Focus on bias	Focus on variance	

Understand human-level performance

Example:

Suppose
(a) Typical human — 3% error
(b) Typical doctor — 1% error
(c) Experienced doctor — 0.7% error
(d) Team of experienced doctor — 0.5% error

What is "human level" error? What your aim?

Surpassing human-level Performance

Not natural perception problem

Lots of data

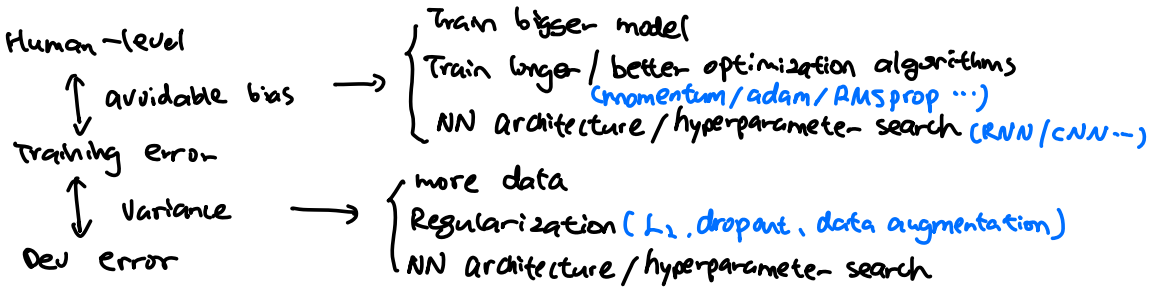
Improve your model performance

Summary

Assumption:

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

② The training set performance generalizes pretty well to the dev/test set



ML Strategy week 2

Error analysis

Carrying out error Analysis

Example.

Background: 90% accuracy 10% error.

Judge dog as cat

Question: Should you try to make your cat classifer do better on dogs?

Error analysis:

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

Evaluate multiple ideas in parallel:

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	Comments
1	✓			Pitbull
2			✓	
3		✓	✓	Rainy day at zoo
⋮	⋮		⋮	
% of total	8% X	43% ✓	61% ✓	

A

This list will help you find which features is the most important to handle.

Cleaning up incorrectly labeled data

- DL algorithms are quite robust to random errors in the training set.
- In dev/test set, you can try to use the list above, record the % of labeled wrong. Question: how many % worth fixing?

Answer:
- overall dev error
- Error due to incorrect label
- Error due to other causes

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

Example:

Overall error A	3%	overall error B	3.1%
labeled error A	0.1%	labeled error B	0.7%

In this case, the dev set is not reliable and we have to fix the labeled error.

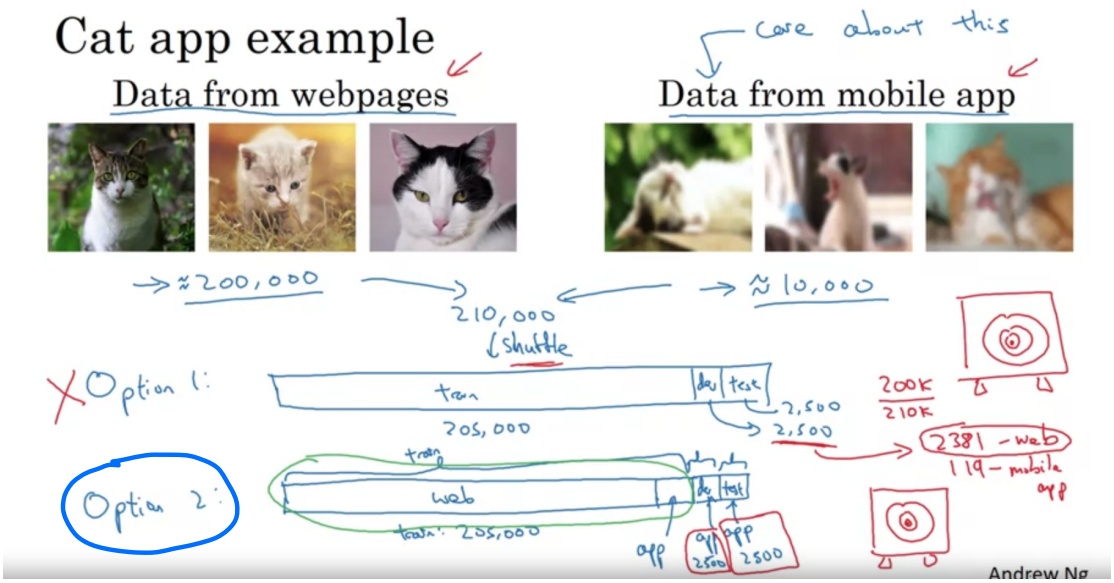
- Guideline to correct incorrect dev/test set examples
  - Apply same process to your dev and test sets to make sure they continue to come from the same distribution
  - Consider examining examples your algorithm got right as well as ones it got wrong
  - Train and dev/test data may now come from slightly different contributions.

Build your first system quickly, then iterate

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

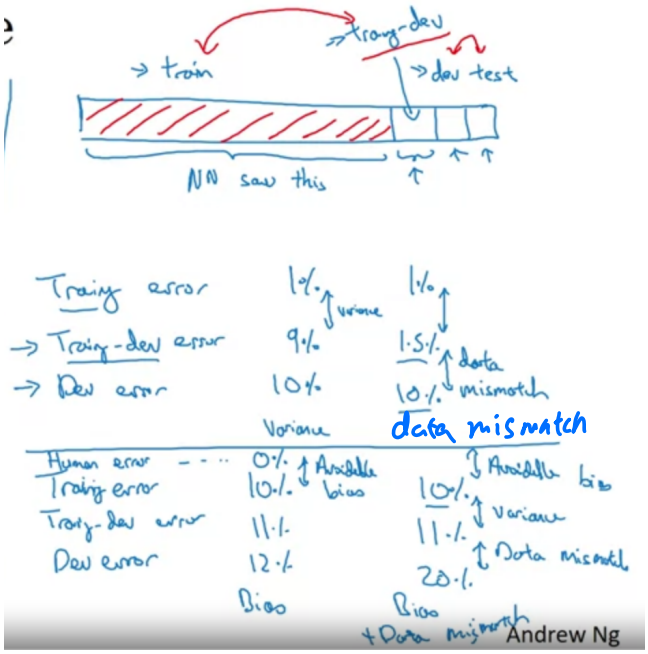
Mismatched Training and Dev/Test Set

Training and Testing on Different distributions



Bias and Variance with Mismatched Data Distributions

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

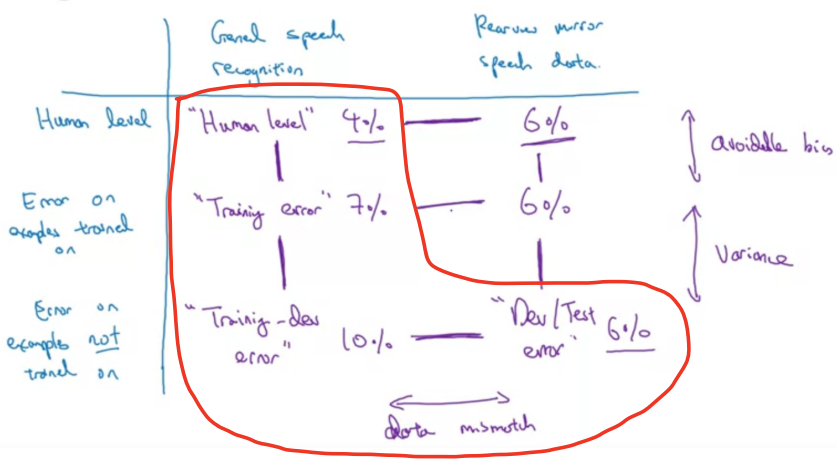


Summary:

	Human level	4%
same distribution	Training set error	7%
same distribution	Train-dev set error	10%
	Dev error	12%
	Test error	12%

↕ avoidable bias
↕ variance
↕ data mismatch
↕ degree of overfitting to the dev set

More general formulation



Address data mismatch

- Carry out manual error analysis to try to understand difference between training and dev/test sets
- Make training data more similar; or collect more data similar to dev/test sets.

how?

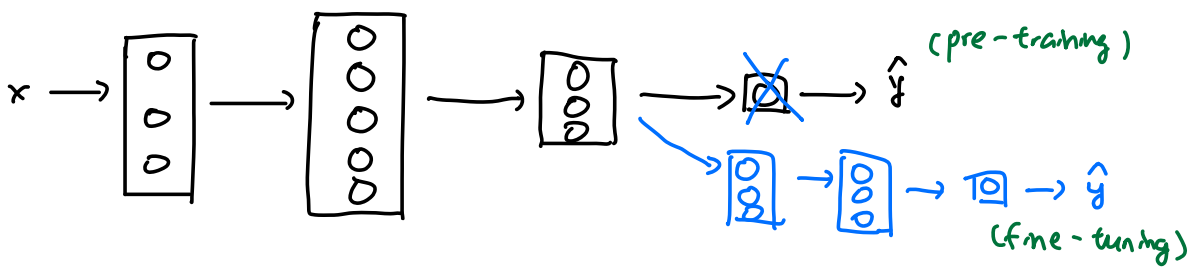
Artificial data synthesis

Learning from Multiple Tasks

Transfer Learning

When transfer learning makes sense?

- Task A and B have the same input x (same type)
- You have a lot more data for Task A than Task B
- Low level features from A could be helpful for learning B



Multi-task learning

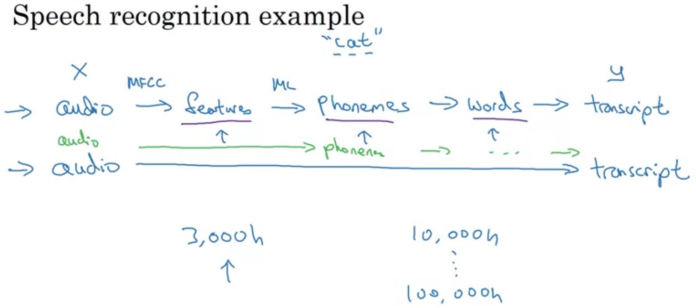
When multi-task learning makes sense

- 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

End-to-end Deep learning

What is end-to-end learning?

Speech recognition example



Whether to use End-to-end deep learning

Pros and cons of end-to-end deep learning

Pros:

- Let the data speak
- less hand-designing of components need

Cons:

- May need large amount of data
- excludes potentially useful hand-designed components

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