Deep Learning

Notes : Perceptron , Activation Function , Gradient Descent , backpropagation , Loss Function , Metrics , Optimizer , Learning Rate , CNN , RNN , Structured Data Unstructured Data , Research Papers , History

Course 3 : Structuring ML Projects

Orthogonalization

Intro

- Fit training set well on cost function (≈ human-level perf) → bigger network, Adam, ...
- Fit dev set well on cost F → Regularization, Bigger train set, ...
- Fit test set well on cost F → Bigger Dev set
- Perform well in Real World → Change Dev set or Cost F

Single Number Evaluation Metric

Take one Eval metric

exp: Precision vs Recall → take F_1 Score

Satisficing and Optimizing Metric

optimizing ↓ satisficing ↓

Classifier	Accuracy	Running Time
A	90%	80 ms
B	92%	95 ms
C	95%	1,500 ms

Create new cost F = accuracy - 0,5 running time

maximize accuracy ⟶ Opti

subject to running time ≤ 1000ms ⟶ satisf

N metrics : Pick 1 optimizing

Pick N-1 satisficing

Train / Dev / Test Distributions

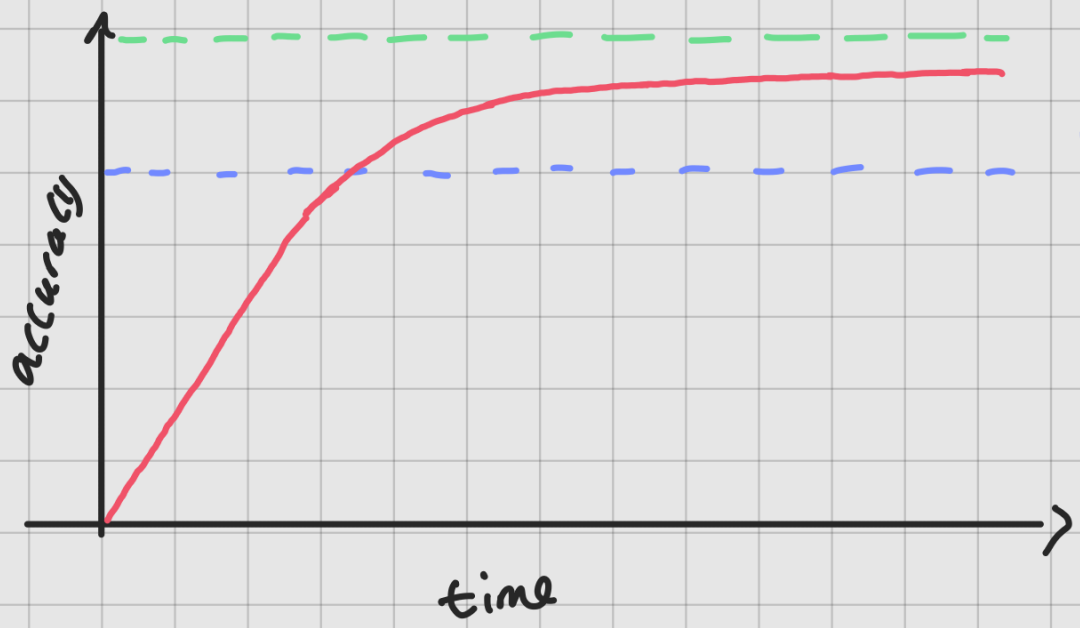
Same Distrib for Train / Dev / test set

Human-level Performance

(best possible error)

↓

Bayes optimal error



When human > ML :

- Get labeled from humans
- Gain insight from manual error analysis: why did a person get this right?
- Better analysis of bias / variance

Avoidable Bias

Human Level

↕ avoidable bias → { Bigger model / Longer / better opti (momentum, Adam...) / NN architecture / hyperparams

Training error

↕ variance → { More Data / Regularizaθ (L_2, dropout, ausmataθ ... / NN archi / hyperparams

Dev error

Carrying Out Error Analysis

10 % error → 0,6 % Cat

→ 88 % Mislabeled ← Priotize

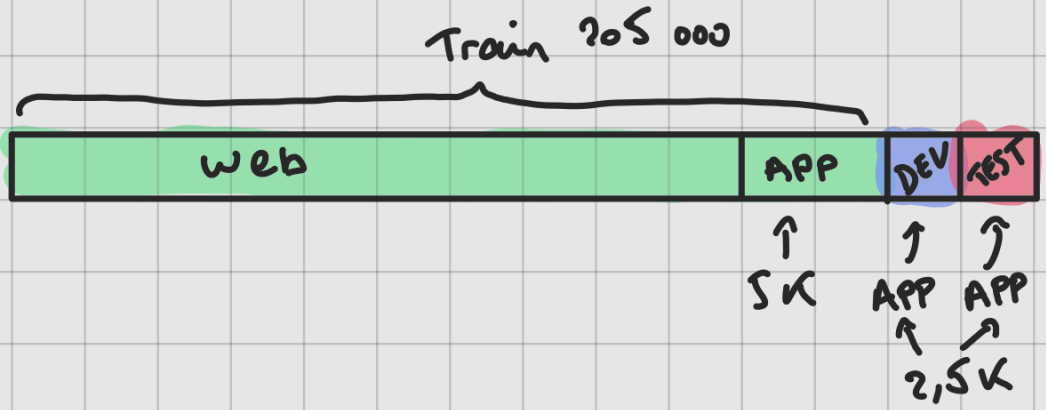
Build 1st system Quickly, then iterate

Training and Testing on Different Distribution

Target ↙

Data from web
≈ 200 000

Data from mobile
≈ 10 000



Bias and Variance with Mismatched Data distribution

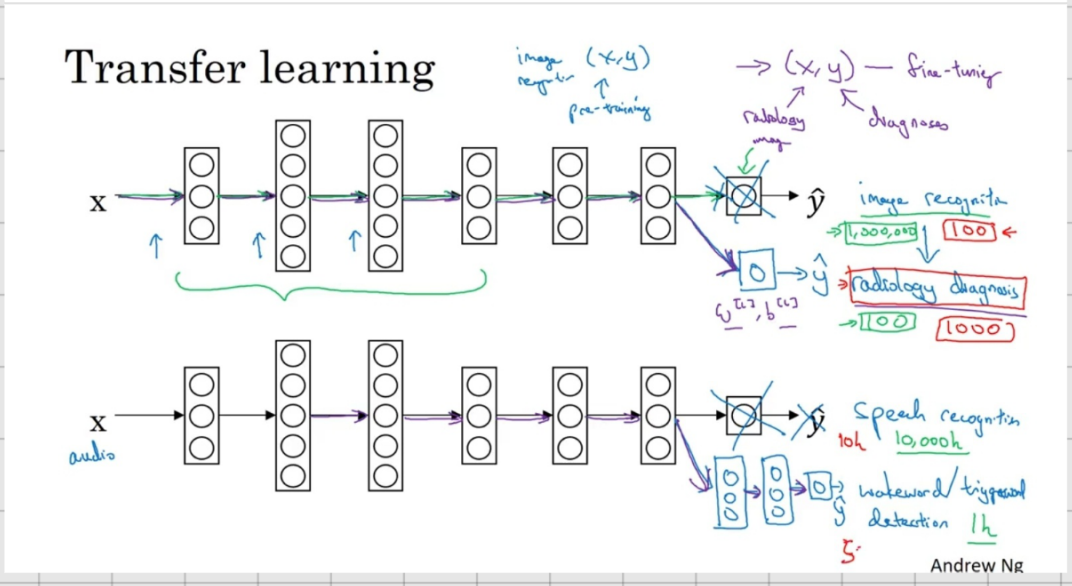
Human-level	4 %	→ Avoidable bias
Train set error	7 %	→ variance
Train-dev set error	10 %	→ data mismatch
Dev error	12 %	→ degree of overfitting to dev set
Test error	12 %	

Same Distribution

Addressing data mismatch

- Carry out manual error analysis to try to understand diff between train and dev/test sets
- Make train set more similar / or collect more data similar to dev/test sets

Transfer Learning



- Task A and B have the same input x
- A lot more data for task A than task B
- Low level features from A could be helpful to B

Multi-task learning

For autonomous cars:

$y^{(i)}$ $(4,1)$

Pedestrians	0
Cars	1
Stop signs	1
Traffic lights	0
⋮	

Loss: $\hat{y}^{(i)}$ $(4,1)$

$$\frac{1}{m}\sum_{i}^{m}\sum_{j}^{4}\mathcal{L}(\hat{y}_j^{(i)}, y_j^{(i)})$$

Unlike softmax: One image can have multiple labels

When?

- 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 tasks.

End to End Deep Learning

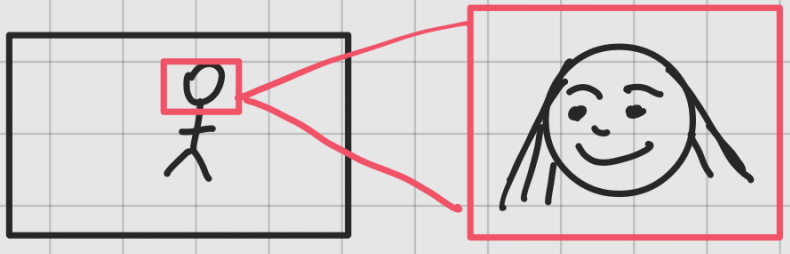
1) → audio —MFCC→ features —ML→ Phonemes → words → transcript → $\hat{y}$

2) → audio ——————————————————→ $\hat{y}$

1) → 3.000^h of Data

2) → 10000^h - 100000^h → end to end

face reco:



2 steps

→ Have enough data for each of 2 sub-tasks

Advantages End to End

- Let the data speak $x \longrightarrow y$
- Less hand-designing of components needed

Disavantages

- Need large amount of data
- Excludes potentially useful hand-designed components

Do you have enough data to learn a function of the complexity needed to map x to y?