Charlie Mackie
May 30th 2020

Structuring Machine Learning Projects

(*) a guide to diagnosing data-based algorithm issues

Orthoganalization tuning each hyperparameter / parameter in a Network should change the Network in it's own way

Basic Structure:

(*) Goal is to have individual means of fixing this structure with a high level of independance

Training fit to Cost function ⟷ Optimizer / Network Size
↓
Dev. fit to Cost function ⟷ Regularization
↓
Test fit to Cost function ⟷ Bigger Dev. Set
↓
Real World Performance ⟷ Change Dev. Set

Single Number Evaluation Metric

For Supervised Learning:

Precision: of positives, how many are true-positive?

Recall: of true-positives, what % are positives?

$$F1\ Score = \frac{2}{\frac{1}{R} + \frac{1}{P}}$$ (harmonic mean) combines precision and recall

Satisficing and Optimizing Metric

1. algorithm needs to perform inference in less than 20 seconds
2. Maximize the accuracy of the algorithm

typically have: 1 optimizing Metric + (N-1) satisficing (conditions)

Train / Test / Dev. Sets ⊛ Test & Dev must come from same distributions

Size: should be big enough for you to have confidence in their accuracy

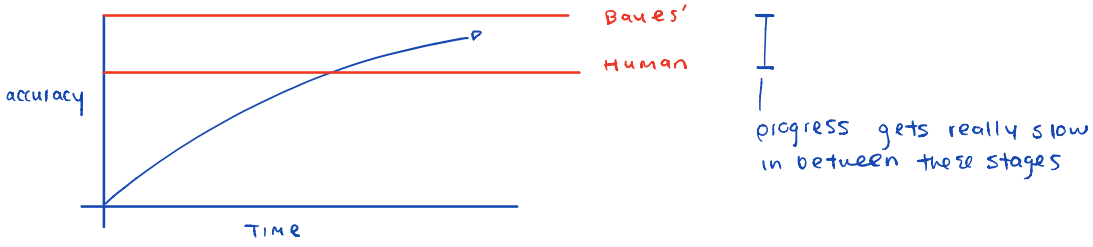
Evaluation Metric: What is the Best for your Application?

↳ example: image cannot contain red

Augment the error term: $\frac{1}{m} \sum w_i \cdot \mathcal{L}(\hat{y}, y)$

w_i: if { good = 1 (punish cost / error)
bad = 10

Human Level Performance:



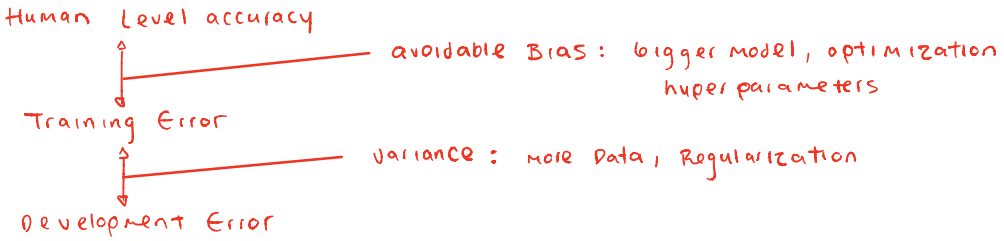
⊛ while the algo is worse than human level accuracy:

↳ get more labelled data & Analysis of Bias / Variance

Human Error is used as a Proxy for Bayes' Error → Best Possible accuracy

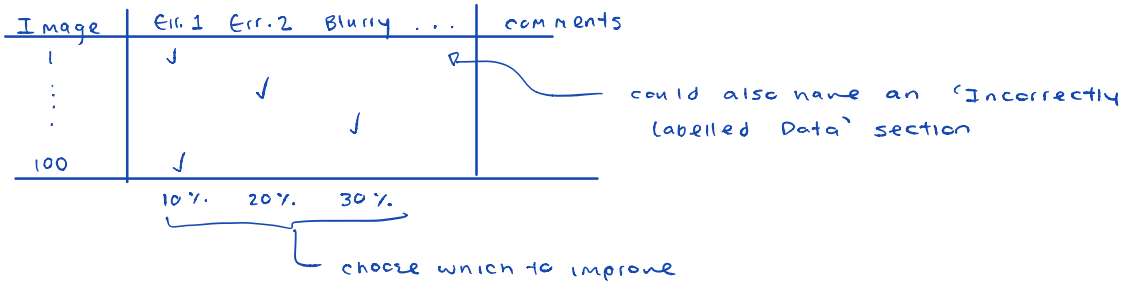
Human Error = that of the best Possible human

Machine learning Applications are very good @ tackling Structured Data but not as good @ natural Perception (humans are better)



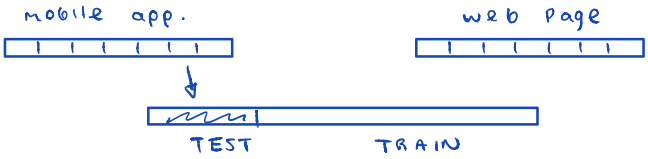
Error Analysis

↳ find the mislabelled data and identify why it was mislabelled manually



Data Distributions

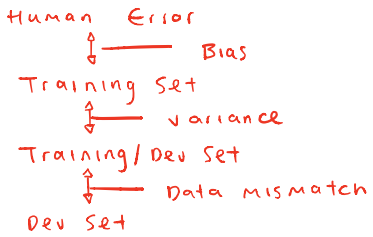
What if your train and dev/test are different



(k) you want to aim your target audience data distribution to the testing set and Optimize

(k) Issue with different distributions is inability to analyze variance/bias

↳ make a 'train-dev' set with the same distribution as Training



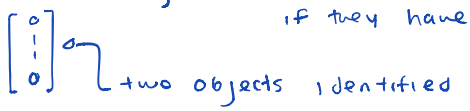
Data Mismatch : realise differences in Train – dev through Error analysis

↳ try Artificial Synthesis of Data : add error feature

Transfer Learning – pre-train a network with weights from a similar application

* useful when Data of your task is not sufficient

Multitask Learning – do many things @ the same time if they have similar low level features



End-to-End ML

Skip many individual steps including feature Engineering and maps input directly to output

* requires lots of Data, often not the best / most effecient