Mục lục

Chương 1

Introduction to Machine Learning Strategy

1.1 Why Machine Learing Strategy

- Đặt vấn đề: Khi xây dựng một Model, để cải thiện hiệu quả của Model đó, bạn đề đặt ra **rất nhiều** ý tưởng
- Nhưng: bạn lại không biết ý tưởng nào giúp cải thiện chất lượng Model, nên bạn chọn một hướng đi như kiểu thu thập thêm dữ liệu mà cuối cùng thì sau 1 khoảng thời gian dài bạn có nhiều dữ liệu hơn, nhưng chất lượng Model thì không.
- Và đó là lý do mà **bạn cần có một chiến lược cụ thể**.

1.2 Orthogonalization

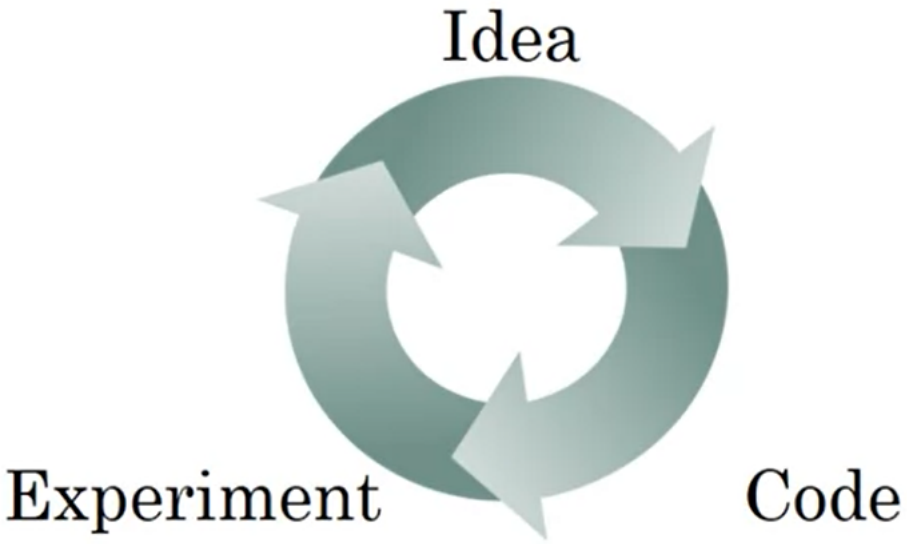
- Ý tưởng: biết phải **làm gì** để đạt được **mục đích gì**
- Ví dụ 1: nút để vặn trên TV (TV cổ lắm mới có nút vặn để điều chỉnh).
  Mỗi một nút vặn sẽ dùng để điều chỉnh một yếu tố duy nhất như là độ rộng của màn hình hay chiều dài của màn hình, hay là độ sáng.
- Ví dụ 2: Trong xe ô tô phải có vô lăng để điều chỉnh hướng đi của xe, và chân ga để tăng tốc hay phanh để giảm tốc.
  Giả sử nếu tất cả yếu tố, hướng đi của xe, tăng tốc, giảm tốc đều phụ thuộc vào vô lăng không thôi thì sẽ rất khó để điều khiển xe.
- Khi train Model cũng giống như thế, bạn nhìn vào vấn đề mà bạn đang đối mặt với ví dụ như Model của bạn hoạt động không tốt trên Test Set thì lúc đó bạn sẽ "vặn cái nút nào" để giải quyết vấn đề.
  Và mỗi cái nút mà bạn sắp vặn, nó nên chỉ ảnh hưởng đến một yếu tố của Model (vì có một số "nút" - thuật toán - ảnh hưởng đến nhiều hơn một yếu tố của Model.

Chương 2

Setting Up your Goal

2.1 Single Number Evaluation Metric

- Tiếng Việt: đơn chỉ số đánh giá
- Quá trình xây dựng Model:



- Precision & Recall
  - Ví dụ trong nhận diện ảnh Mèo:
    - Precision là "Trong số các ví dụ được nhận diện là Mèo, thì có bao nhiêu % ví dụ đó thật sự là Méo"
    - Recall là "Có bao nhiêu ví dụ Méo được nhận diện đúng"
  - Ví dụ:

Classifier	Precision	Recall
A	95%	90%
B	98%	85%

  - Vấn đề đặt ra: Nếu dùng 2 thông số Precision và Recall thì sẽ rất khó để đánh giá thuật Classifier nào là tốt hơn (vì Classifier A có Recall cao hơn, còn B thì Precision cao hơn).
  - Để giải quyết vấn đề trên, ta có thể dùng 1 thông số mới - "Trung bình" của 2 thông số trên - để đánh giá

Classifier	Precision	Recall	F1 Score
A	95%	90%	92.4%
B	98%	85%	91.0%

    * F1 Score = "Trung bình" của Precision (P) và Recall (R)
    * F1 Score = $\frac{2}{\frac{1}{P}+\frac{1}{R}}$ (được gọi là **Harmonic Mean**)
  - Bây giờ dễ thấy rằng Classifier nào tốt hơn

- Tóm lại: Bằng cách xác định từ đầu, trước khi xây dựng Model, quyết định những đơn thông số để đánh giá thuật toán sẽ giúp việc quyết định Model nào là tốt hơn Model nhanh hơn, dễ hơn.

2.2 Satisficing and Optimizing Metric

- Ví dụ 1: Cat Classification
  - Bảng:

Classifier	Accuracy	Running Time
A	90%	80ms
B	92%	95ms
C	95%	1500ms

  - **Cách không hiệu quả:** Cost = Accuracy $- 0.5 \times$ Running Time
  - **Cách hiệu quả:** Maximize Accuracy subject to Running Time $\leq$ 100ms
    * Maximize Accuracy: tối ưu thông số Accuracy $\rightarrow$ Accuracy: **Optimizing Metric**
    * Subject to Running Time $\leq$ 100ms: tuy thuộc vào thông số Running Time nếu $\leq$ 100ms
      Running time được gọi là **Satisficing Metric** vì tao chỉ quan tâm đến nó trong một ngưỡng (threshold) nhất định, nếu lớn hơn thì bỏ qua

- Ví dụ 2: Wakewords/Trigger words
  - Trigger words: Alexa, Ok Google, Hey Siri, ... dùng trong trợ lý ảo
  - Ở bài toán này tao muốn: **Maximize Accuray** *(Độ chính xác của việc khi nói trigger word thì trợ lý ảo khởi động)* **subject to $\leq$ 1 false postive every 24 hours** *(xác suất thức dậy của trợ lý ảo dù không nói gì trong 24 giờ)*

2.3 Train/Dev/Test Distributions

- Khi chia Data thành Dev set và Test set thì phải **phân bố đều** trên cả 2

- Ví dụ: bạn có Data của 8 khu vực: US, UK, Europe, South America, India, China, Australia, Vietnam
  Và bạn chia Dev set là 4 khu vực bất kì, Test set là 4 khu vực còn lại.
  Thì như thế là không tốt, vì Dev và Test có phân bố khác nhau.
  Nó giống như kiểu, bạn luyện tập để bắn vào 1 cái bia vậy, nhưng lúc đi thi đấu thì cái bia ý lại được đặt ở vị trí khác, xa hơn, chéo hơn khiến bạn không bắn được.
  Việc phân bố Dev và Test set không đều cũng giống như thế.

- Guideline: **Choose a dev set and test set to reflect data you expect to get in the future and consider important to do well on** (Dev và Test set phải có cùng phân bố (same distribution))

2.4 Size of the Dev and Test Sets

- Ngày xưa khi không nhiều Data (100 - 10.000 training examples) thì người ta thường hay chia thành 70% Train, 30% Dev hoặc 60% Train, 20% Dev, 20% Test
- Nhưng ngày nay, với số lượng Data khủng lổ (1.000.000 training examples) thì việc chia như thế kia không còn hợp lý nữa
- Với số lượng Data lớn thì chia **98% Train, 1% Dev, 1% Test** sẽ hợp lý hơn
- Với Dev và Test set thì khoảng 10.000 training examples là ổn
- Có thể không có Test set, nhưng Train và Dev là phải có, điều này không được khuyến khích
  (Nếu ai đó nói chỉ có Train và Test thì Test của họ là Dev)

2.5 When to Change Dev/Test Sets and Metrics?

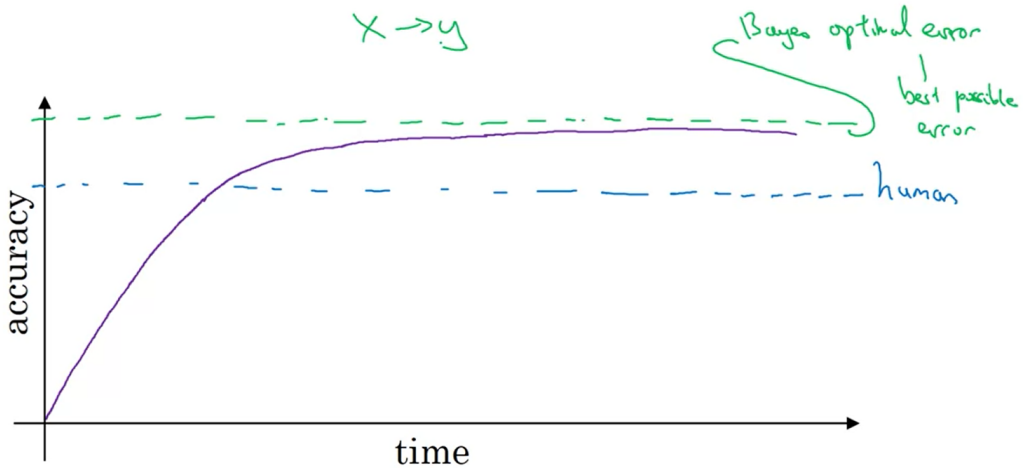
- Ví dụ 1: bạn chọn Metric cho 1 model Cat classification như sau:
  Metric: Classification Error
  Classifier A: 3% Error
  Classifier B: 5% Error
  - Nhìn vào dễ thấy Classifier A làm việc tốt hơn
  - Nhưng mà Classifier A lại gắn mác cho pornographic là Cat, trong khi Classifier B thì không
  - Tức là ở góc nhìn của Người dùng, thì họ sẽ thấy Classifier B hoạt động tốt hơn
- Ví dụ 2: Vẫn như ví dụ 1, nhưng mà Metric + Dev/Test set của bạn lại được chạy trên những hình ảnh Mèo chất lượng cao. Nhưng người dùng lại sử dụng toàn những hình ảnh mờ, lạ, chất lượng xấu.
  Lúc này, Classifier B lại hoạt động tốt hơn A trên những hình ảnh xấu đó.
- Những ví dụ trên là lúc mà bạn phải đổi hướng, chọn Metric + Dev/Set mới

Chương 3

Comparing to Human-level Performance

3.1 Why Human-level Performance?

- Nhận xét về đồ thị sau:



  - Đây là đồ thị thể hiện độ chính xác của thuật toán theo thời gian.
  - Ta có thể thấy độ chính xác của thuật toán chạm đến độ chính xác của con người trong khoảng thời gian rất ngắn.
  - Thuật toán có thể vượt qua độ chính xác của con người, nhưng rồi sẽ chạm đến một cái giới hạn gọi là **Bayes Optimal Error**.
  - Khoảng cách giữa Bayes Error (cách gọi tắt của Bayes Optimal Error) với độ chính xác của con người là rất ngắn vì con người khá giỏi trong việc nhận diện các vấn đề của AI nói chung.

- Lý do so sánh độ chính xác của thuật toán với con người: chừng nào độ chính xác của thuật toán vẫn chưa chạm được đến độ chính xác của con người, thì tức là ta vẫn có thể cải thiện thuật toán qua những cách sau:

  - Để con người gán label cho data
  - Tìm hiểu lỗi bằng cách phân tích thủ công
  - Phân tích tốt hơn về Bias/Variance

3.2 Avoidable Bias

- Cho bảng sau:

Human ($\approx$ Bayes)	1%	7.5%
Training Error	8%	8%
Dev Error	10%	10%

- Trong ví dụ này, giả sử như là nhận diện ảnh, thì chúng ta có thể chọn Human-level Error làm Bayes Error vì con người có khả năng nhận diện ảnh rất tốt. Đối với Computer Vision (CV) thì cách chọn Bayes Error này khá hợp lý.
- Avoidable Bias = Training Error − Humun-level Error
- Variance = Dev Error − Training Error
- Ở cột 1: Vì Avoidable Bias (7%) cao hơn Variance (2%) nên sẽ tốt hơn nếu bạn tập trung vào việc làm giảm Bias trước
- Ở cột 2: Vì Avoidable Bias (0.5%) nhỏ hơn Variance (2%) nên sẽ tốt hơn nếu bạn tập trung vào việc làm giảm Variance trước

3.3 Understanding Human-level Performance

- Thế nào là Human-level Performance?
  - Trong vấn đề chuẩn đoán bệnh qua ảnh:
    - 1 người thường: 3% Error
    - 1 bác sĩ: 1% error
    - 1 bác sĩ nhiều kinh nghiệm: 0.7%
    - 1 nhóm bác sĩ nhiều kinh nghiệm: 0.5%
  - Vậy thì đâu mới là giá trị hợp lý cho Human-level Performance?
- Ví dụ:

Human ($\approx$ Bayes)	0.5-1%	0.5-1%	0.5%
Train Error	5%	1%	0.7%
Dev Error	6%	5%	0.8%
	Bias	Variance	

  - Ở cột 1 ta chọn giá trị cho Human-level Error trong khoảng từ 0.5-1% đều có thể thấy rằng Avoidable Bias lớn hơn Variance, nên ở trường hợp này thì ta nên tập trung vào xử lý Bias
  - Ở cột 2 thì rõ ràng thấy được Variance lớn hơn Avoidable Bias nên tao tập trung vào xử lý Variance
  - Còn ở cột 3, thì nếu như ta chọn Human-level Error là 0.7% thay cho 0.5% trong ảnh thì sẽ thấy Variance lớn hơn Avoidable Bias, nhưng với 0.5% thì ta thấy điều ngược lại
    $\rightarrow$ Tức là khi mà thuật toán đã trở nên tốt hơn, gần đạt tới Human-level Performance thì việc chọn Human-level Error nào sẽ ảnh hưởng nhiều hơn khi mà thuật toán còn làm việc chưa tốt
- Tóm lại việc chọn Human-level Error nào sẽ có ảnh hưởng đến việc quyết định xử lý Bias hay Variance

3.4 Surpassing Human-level Performance

- 1 vấn đề nảy sinh khi thuật toán vượt qua Human-level Performance là lúc đó bạn sẽ không thể biết được là nên chọn giá trị Bayes Error như thế nào để tiếp tục khiến thuật toán của bạn tốt hơn.
  → Đó là lý do tại sao mà sau khi vượt Human-level Performance, các thuật toán rất khó để phát triển hơn nữa.

- Đến bây giờ đã xuất hiện rất nhiều những lĩnh vực, mà AI vượt qua con người, như trong nhận diện ảnh, hay nhận diện giọng nói.

3.5 Improving your Model Performance

- Avoidable Bias:
  - Train bigger model
  - Train longer/better optimization algorithms
  - NN architecture/hyperparameters search (như CNN, RNN)
- Variance:
  - More data
  - Regularization (L2, dropout, ...)
  - NN architecture/hyperparameters search (như CNN, RNN)

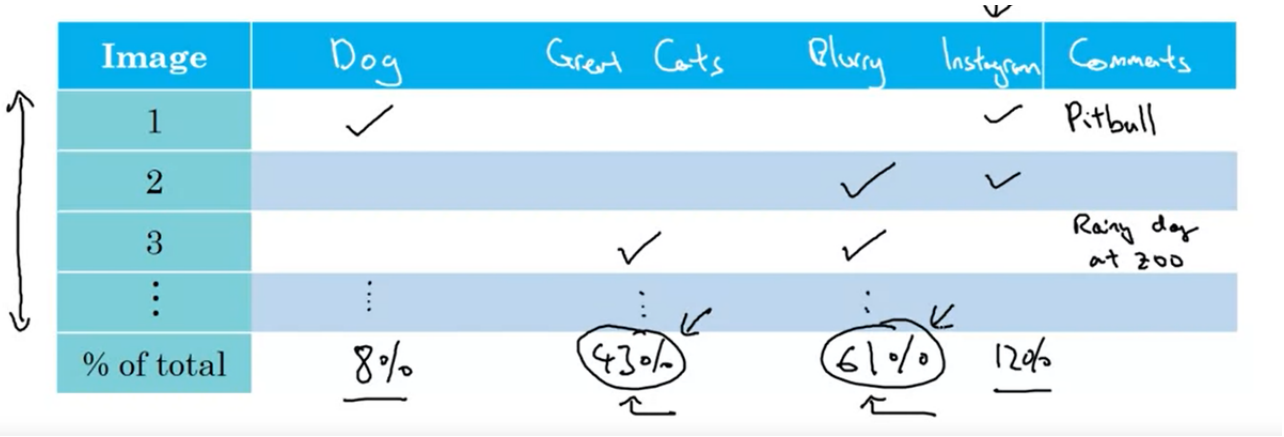
Chương 4

Error Analysis

4.1 Carrying Out Error Analysis

- Các bước phân tích lỗi:

  1. Tìm 1 tập các ví dụ bị gắn nhầm Label trong Dev set hoặc Test set
  2. Tìm các ví dụ **false positives** và **false negatives** trong đó
  3. Đếm số lượng sai trong các mục lỗi khác nhau
  4. Trong quá trình bạn có thể gặp mục lỗi mới
  5. Từ đó quyết định xử lý lỗi nào trước hoặc đi theo 1 hướng mới



4.2 Cleaning Up Incorrectly Labeled Data

- Câu hỏi đầu tiên được đặt ra là **Nó có đáng không?**
  Vì có thể việc gán sai nhãn cho một vài ví dụ một cách **ngẫu nhiên** không làm ảnh hưởng quá nhiều đến tổng thể
  Nếu là lỗi một cách **hệ thống** thì chắc chắn cần phải xem xét

- Bằng cách thêm 1 cột **"Incorrectly labeled"** vào bảng trên có thể giúp bạn quyết định là việc chỉnh lại nhãn cho đúng có đáng hay không

- Một số lưu ý:

  – Áp dụng cùng 1 quá trình lên cả Dev và Test set để đảm bảo cả 2 có cùng phân bố (distribution)

- Xem xét cả những ví dụ đúng cùng với ví dụ sai
- Tran và Dev/Set có thể trở nên hơi khác về phân bố (distribution)

4.3 Build your First System Quickly, then Iterate

- Thay vì xây dựng 1 hệ thống phức tạp từ đầu, hãy xây dựng 1 hệ thống đơn giản nhanh chóng rồi từ đó, dựa vào phân tích Bias/Variance và Error để quyết định hướng đi sẽ nhanh hơn trong việc tạo ra 1 hệ thống hoạt động

Chương 5

Mismatched Traing and Dev/Test Set

5.1 Training and Testing on Different Distributions

- Nếu bạn 500.000 data kiểu A và 20.000 data kiểu B mà bạn lại muốn hệ thống của mình hoạt động tốt trên B thì bạn nên chia data kiểu B thành 10.000, 5.000, 5.000 rồi chia Train/Dev/Test như sau:

500.000 A + 10.000 B	5.000 B	5.000 B

- Không nên trộn hết vào rồi random chúng

5.2 Bias and Variance with Mismatched Data Distributions

-

5.3 Addressing Data Mismatch

Chương 6

Learning from Multiple Tasks

6.1 Transfer Learning

6.2 Multi-task Learing

Chương 7

End-to-end Deep Learning

7.1 What is End-to-end Deep Learning?

7.2 Whether to use End-to-end Deep Learning