Introduction to ML strategy

Learning Objectives:

- Explain why Machine Learning strategy is important
- Apply satisficing and optimizing metrics to set up your goal for ML projects
- Choose a correct train/dev/test split of your dataset
- Define human-level performance
- Use human-level performance to define key priorities in ML projects
- Take the correct ML Strategic decision based on observations of performances and dataset

Introduction to ML strategy

1 Why ML Strategy?
2 Orthogonalization
3 Single number evaluation metric
4 Satisficing and optimizing metrics
5 Train/dev/test distributions
6 Size of dev and test sets
7 When to change dev/test sets and metrics
8 Why human-level performance?
9 Avoidable bias
10 Understanding human-level performance
11 Surpassing human- level performance
12 Improving your model performance

FPT UNIVERSITY

Why ML Strategy?

Why ML Strategy?

FPT Education
FPT UNIVERSITY

- Choosing the wrong idea can lead to wasted time and effort with little improvement.
- Using quick and effective strategies can determine the most promising ideas, with unique lessons learned from building and shipping deep learning products.
- The strategy for machine learning is changing in the era of deep learning due to the different opportunities and challenges presented by deep learning algorithms. It aims to help learners become more effective in making their deep learning systems work.

Motivating example






Ideas:

- Collect more data
- Collect more diverse training set
- Train algorithm longer with gradient descent
- Try Adam instead of gradient descent
- Try bigger network
- Try smaller network
- Try dropout
- Add L_2 regularization
- Network architecture
- Activation functions
- # hidden units
- …

FPT UNIVERSITY

Introduction to ML strategy

Orthogonalization

Orthogonalization

- In machine learning, Orthogonalization refers to the idea of separating the concerns of a machine learning system into distinct and independent modules or components.
- Each module or component should have a clear and specific responsibility, and they should not overlap or depend on each other.
- The goal is to create a machine learning system that is modular, flexible, and easy to maintain, where changes or updates to one component do not affect the others.
- In neural networks, for example, Orthogonalization can be achieved by designing each layer to have a specific function or task, with clear inputs and outputs, and avoiding the mixing of different functionalities within the same layer.

Orthogonalization

- In practice, Orthogonalization can be achieved in several ways, depending on the specific machine learning system and its requirements. Some common techniques include:
  - - **Modular Design**: Breaking down a machine learning system into smaller, independent modules or components, with clear inputs and outputs. This can be achieved by using libraries, functions, or classes, depending on the programming language and environment.
- - **Encapsulation**: Hiding the internal details of each module or component from other modules or components, and providing a clean and well-defined interface for communication. This can be achieved by using well-defined APIs (Application Programming Interfaces), data structures, or protocols.
- - **Layered Architecture**: Designing a machine learning system as a stack of layers, where each layer performs a specific task or function, without overlapping or depending on other layers. This is commonly used in neural networks, where each layer has a specific role in processing the input data, and the output of one layer is used as the input of the next layer.

Orthogonalization

- For a supervised learning system to do well, you usually need to tune the knobs of your system to make sure that four things hold true - **chain of assumptions** in machine learning:

1. Fit training set well on cost function (near human level performance if possible).
- - If it's not achieved you could try bigger network, another optimization algorithm (like Adam)...
2. Fit dev set well on cost function.
- - If its not achieved you could try regularization, bigger training set...
3. Fit test set well on cost function.
- - If its not achieved you could try bigger dev. set...
4. Performs well in real world.
- - If its not achieved you could try change dev. set, change cost function...

Orthogonalization

FPT Education
FPT UNIVERSITY

- Each of these criteria requires a specific set of hyperparameters to be tuned, and it is important to identify these hyperparameters and ensure that they have relatively interpretable functions.

- Examples of hyperparameters that may need to be tuned include the size of the network, the optimization algorithm used, and regularization techniques. It is important to ensure that each hyperparameter has a clear and specific function that is aligned with the specific aspect of the model that needs to be improved.

Orthogonalization

- Overall, the process of orthogonalization in machine learning involves identifying the specific set of hyperparameters that need to be tuned to achieve a particular effect, and ensuring that each hyperparameter has a clear and interpretable function.
- This makes it much easier to tune the model effectively and achieve the desired performance.

Single number evaluation metric

Single number evaluation metric

- In machine learning, having a well-defined evaluation metric is crucial to compare different models or algorithms effectively.
- A single evaluation metric, such as the F1 score, can help to quickly compare different models and determine which one performs better.
- However, choosing the right evaluation metric is also essential for the specific problem at hand. For instance, if the problem involves building a cat app for different markets, tracking performance in each market separately may be necessary, while computing the average performance across all markets can be used as a single evaluation metric for comparison.

Single number evaluation metric

- Its better and faster to set a single number evaluation metric for your project before you start it.
- Difference between precision and recall (in cat classification example): Suppose we run the classifier on 10 images which are 5 cats and 5 non-cats. The classifier identifies that there are 4 cats, but it identified 1 wrong cat.
- Confusion matrix:
-

	Predicted cat	**Predicted non-cat**
Actual cat	3	2
Actual non-cat	1	4

- **Precision**: percentage of true cats in the recognized result: P = 3/(3 + 1)
- **Recall**: percentage of true recognition cat of the all cat predictions: R = 3/(3 + 2)
- **Accuracy**: (3+4)/10

Single number evaluation metric

- Using a precision/recall for evaluation is good in a lot of cases, but separately they don't tell you which algorithms is better. Example:

Classifier	Precision (P)	Recall (R)
A	95%	90%
B	98%	85%

- A better thing is to combine precision and recall in one single (real) number evaluation metric. There a metric called F1 score, which combines them.
- - You can think of F1 score as an average of precision and recall F1 = 2 / ((1/P) + (1/R))

Single number evaluation metric

- It is important to distinguish between optimizing and satisfying metrics.
  - Optimizing metrics are metrics that you want to maximize, such as accuracy or F1 score.
  - Satisfying metrics are metrics that you want to meet a minimum threshold, such as a maximum false positive rate or a minimum recall.
- Having a well-defined evaluation metric is crucial for improving the efficiency of machine learning projects, but it is necessary to choose the right metric for the specific problem at hand and to understand the difference between optimizing and satisfying metrics.

FPT UNIVERSITY

Satisficing and optimizing metrics

Satisficing and optimizing metrics

- When it's difficult to combine different concerns into a single evaluation metric, it can be helpful to set up both optimizing and satisficing metrics.
- For example, if accuracy and running time are both important in a classification system, accuracy can be the optimizing metric, while running time can be a satisficing metric with a specific threshold.
- Similarly, in a wake word detection system, accuracy can be optimized while the number of false positives can be a satisficing metric with a maximum threshold, such as one per 24 hours of operation.

Another cat classification example

Classifier	Accuracy	Running time
A	90%	80ms
B	92%	95ms

Example: Cat vs Non-cat

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

In this case, accuracy and running time are the evaluation matrices. Accuracy is the optimizing metric, because you want the classifier to correctly detect a cat image as accurately as possible. The running time which is set to be under 100 ms in this example, is the satisficing metric which mean that the metric has to meet expectation set.

The general rule is:

$$N_{metric}: \begin{cases} 1 & Optimizing\ metric \\ N_{metric} - 1 & Satisficing\ metric \end{cases}$$

Satisficing and optimizing metrics

- These evaluation metrics must be evaluated on a training set, a development set, or a test set.
- Therefore, it's essential to set up training, development, and test sets properly.
- The next section will share guidelines on how to set up these sets.

FPT UNIVERSITY

Train/dev/test distributions

Train/dev/test distributions

FPT Education
FPT UNIVERSITY

- Setting up dev and test sets correctly is crucial for the efficiency of a machine learning team. The dev set is used to evaluate different models and ideas, while the test set is used to evaluate the final model. Incorrect set up of dev and test sets can lead to spending months optimizing for the wrong target, only to find out that the model performs poorly on the test set.

- For example, in a cat classifier developed for different regions, it's important to ensure that the dev and test sets come from the same distribution. Randomly chosen regions can lead to poor performance on the test set. To avoid this, it's recommended to use all available data to create dev and test sets that reflect the expected data in the future and are important to do well on.

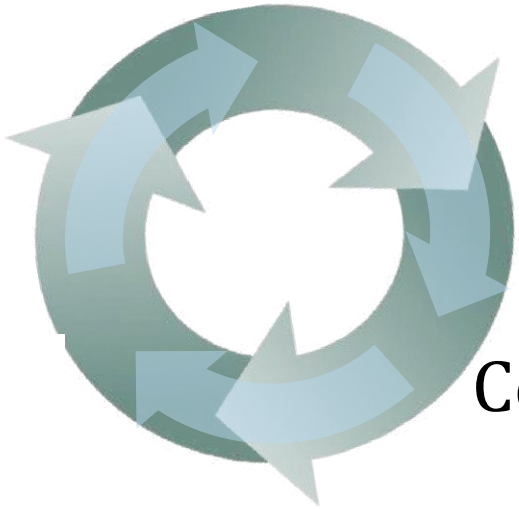
Cat classification dev/test sets

Regions:

- US
- UK
- Other Europe
- South America
- India
- China
- Other Asia
- Australia

Idea

Experiment



Code

Setting up the training, development and test sets have a huge impact on productivity. It is important to choose the development and test sets from the same distribution and it must be taken randomly from all the data.

Train/dev/test distributions

FPT Education
FPT UNIVERSITY

- An example of a machine learning team that optimized a model for several months on a dev set composed of loan approvals from medium-income zip codes, only to find that the model did not perform well on low-income zip codes. This highlights the importance of setting up the dev and test sets correctly to reflect the data expected in the future and prioritize the important aspects.

- Setting up the dev set and evaluation metric serves as a target for the team to aim at, and the size of the dev and test sets is also crucial in the era of deep learning.

True story (details changed)

- Optimizing on dev set on loan approvals for medium income zip codes
- Same distribution
- Tested on low income zip codes

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.

FPT Education

FPT UNIVERSITY

Size of dev and test sets

Setting up your goal

Size of dev and test sets

FPT Education
FPT UNIVERSITY

- In the deep learning era, where dataset sizes are much larger than before, the guidelines for setting up dev and test sets are changing.
- The traditional 70/30 split between training and test sets or 60/20/20 split between training, dev, and test sets is no longer reasonable for very large datasets.
- Instead, a smaller percentage of the data is used for dev and test sets, with the majority going to the training set.

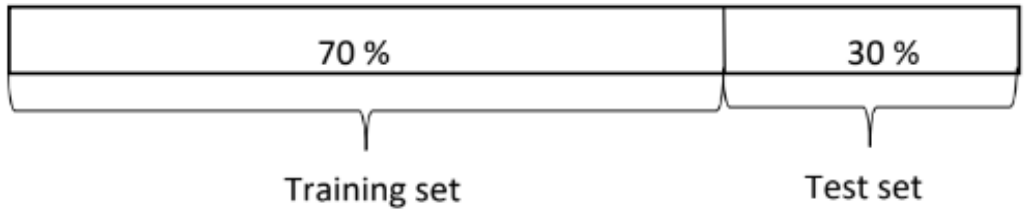
- For example, if there are a million training examples, 98% may be used for training, while 1% is used for the dev and test sets each, providing 10,000 examples for each of dev and test sets.

Old way of splitting data

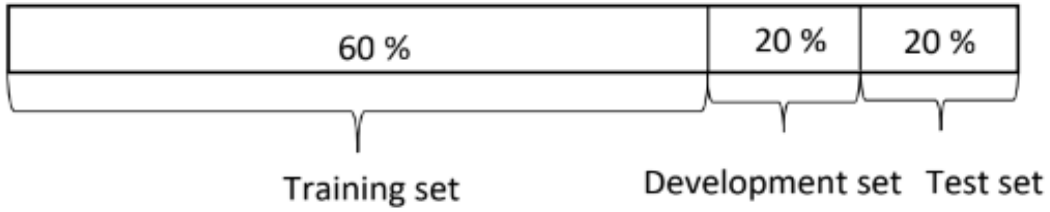
FPT Education
FPT UNIVERSITY

Old way of splitting data

We had smaller data set therefore we had to use a greater percentage of data to develop and test ideas and models.

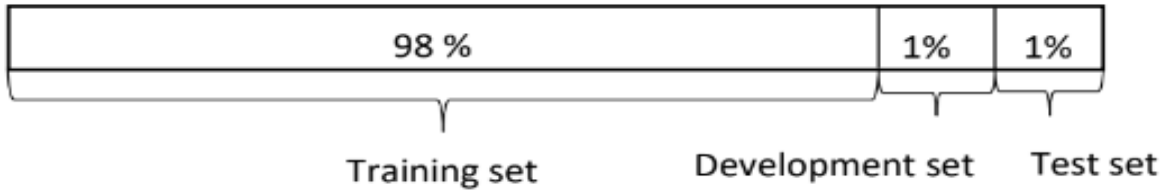


Or



Modern era – Big data

Now, because a large amount of data is available, we don't have to compromised as much and can use a greater portion to train the model.



Size of dev and test sets

- The test set should be big enough to provide high confidence in the overall performance of the final system, but the necessary size may vary depending on the application.
- A smaller test set size may be adequate in some cases, but it's not recommended to skip having a separate test set altogether as it provides an unbiased estimate of the system's performance before shipping.
- Changing the evaluation metric or dev/test sets may be necessary during a machine learning problem, but this should be done carefully to avoid bias and ensure that the new evaluation is appropriate for the problem at hand.

Size of dev set

- Set your dev set to be big enough to detect differences in algorithm/models you're trying out.

Size of test set

- Set your test set to be big enough to give high confidence in the overall performance of your system.

FPT UNIVERSITY

When to change dev/test sets and metrics

When to change dev/test sets and metrics

- A well-defined evaluation metric and dedicated datasets are crucial in machine learning for setting performance goals and facilitating iterative improvements. However, adjustments may be necessary if the initial choices fail to capture important considerations.
- For instance, in a cat classifier scenario, where Algorithm A outperforms Algorithm B in terms of classification error but allows inappropriate images, reevaluation may be needed to prioritize the correct identification of specific content, such as pornographic images.

Cat dataset examples

FPT Education
FPT UNIVERSITY

Example: Cat vs Non-cat

A cat classifier tries to find a great amount of cat images to show to cat loving users. The evaluation metric used is a classification error.

Algorithm	Classification error [%]
A	3%
B	5%

It seems that Algorithm A is better than Algorithm B since there is only a 3% error, however for some reason, Algorithm A is letting through a lot of the pornographic images.

Algorithm B has 5% error thus it classifies fewer images but it doesn't have pornographic images. From a company's point of view, as well as from a user acceptance point of view, Algorithm B is actually a better algorithm. The evaluation metric fails to correctly rank order preferences between algorithms. The evaluation metric or the development set or test set should be changed.

The misclassification error metric can be written as a function as follow:

$$Error: \frac{1}{m_{dev}} \sum_{i=1}^{m_{dev}} \mathcal{L}\{(\hat{y}^{(i)} \neq y^{(i)}\}$$

This function counts up the number of misclassified examples.

The problem with this evaluation metric is that it treats pornographic vs non-pornographic images equally. On way to change this evaluation metric is to add the weight term $w^{(i)}$.

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

Orthogonalization for cat pictures: anti-porn

- So far we've only discussed how to define a metric to evaluate classifiers.
- Worry separately about how to do well on this metric.

$$Error: \frac{1}{\sum w^{(i)}} \sum_{i=1}^{m_{dev}} w^{(i)} \mathcal{L}\{(\hat{y}^{(i)} \neq y^{(i)}\}$$

When to change dev/test sets and metrics

FPT Education
FPT UNIVERSITY

- Another example is that an algorithm may perform worse when deployed on lower quality images uploaded by users, even though the evaluation was done on high-quality, well-framed images. In this case, the dev/test set may need to be changed to better reflect the type of data the algorithm needs to perform well on.

- It's crucial to set up an evaluation metric and dev/test set quickly, even if they're not perfect initially, to drive the team's iteration speed. If necessary, they can be changed later to better capture what the algorithm needs to do well on. The overall goal is to have a well-defined target that the team can iterate towards to improve the algorithm's performance efficiently.

Another example

- Algorithm A: 3% error
- Algorithm B: 5% error
- Dev/test



User images



- 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.

FPT UNIVERSITY

Why human-level performance?

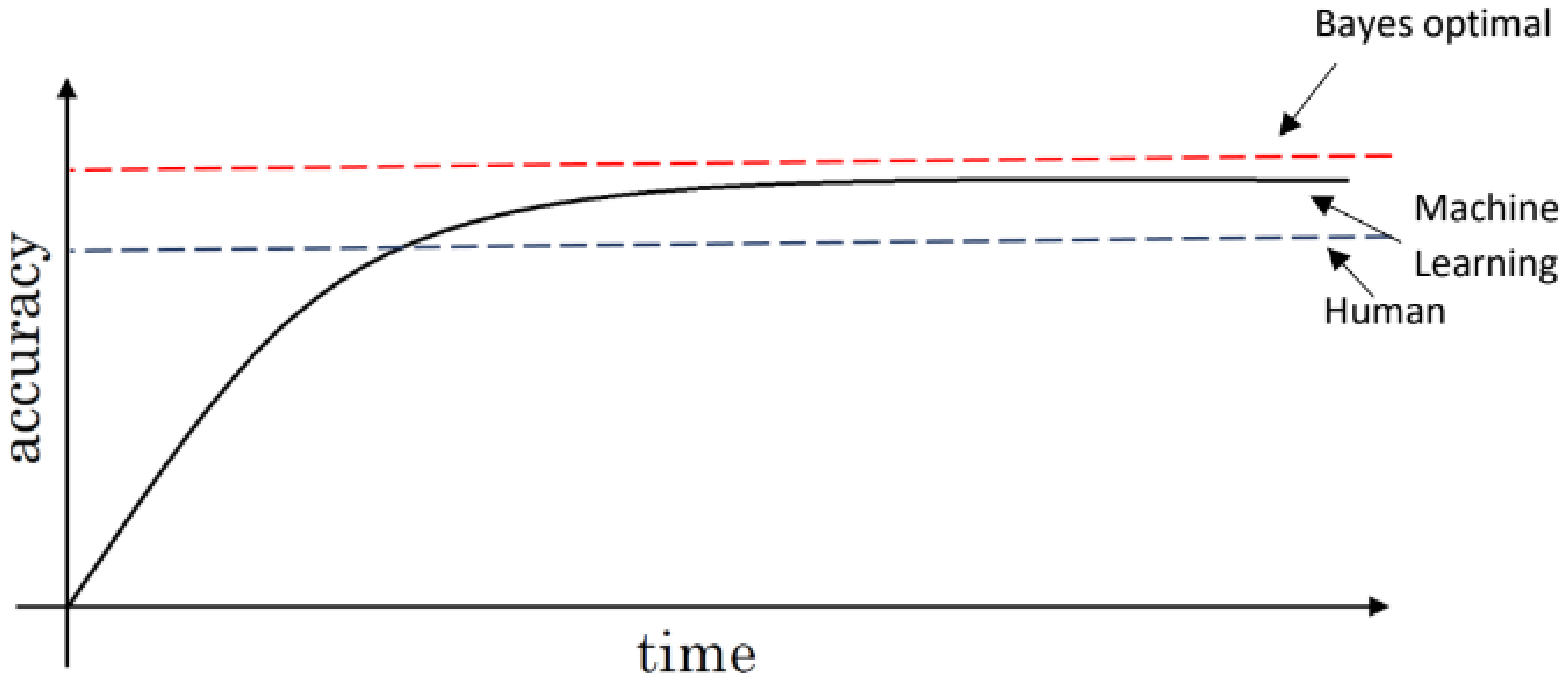
Why human-level performance?

- More machine learning teams have been comparing their systems to human-level performance because of two primary factors.
  - Firstly, with advancements in deep learning, machine learning algorithms can now compete with human-level performance in many application areas.
  - Secondly, designing and building a machine learning system is more efficient when replicating tasks that humans can do because humans can provide labeled data for the algorithm to learn from and manual error analysis to gain insight into why the algorithm got it wrong.

Why human-level performance?

- Machine learning progress often plateaus beyond human-level performance, nearing the Bayes optimal error.
- Surpassing human-level performance may limit further improvements, making certain strategies challenging to apply.
- Nevertheless, comparing algorithms to human performance serves as a benchmark, guiding efforts to mitigate bias and variance.

Comparing to human-level performance



Why compare to human-level performance

FPT Education
FPT UNIVERSITY

- Humans are quite good at a lot of tasks. So long as ML is worse than humans, you can:
  - Get labeled data from humans.
  - Gain insight from manual error analysis: Why did a person get this right?
  - Better analysis of bias/variance.

FPT Education

FPT UNIVERSITY

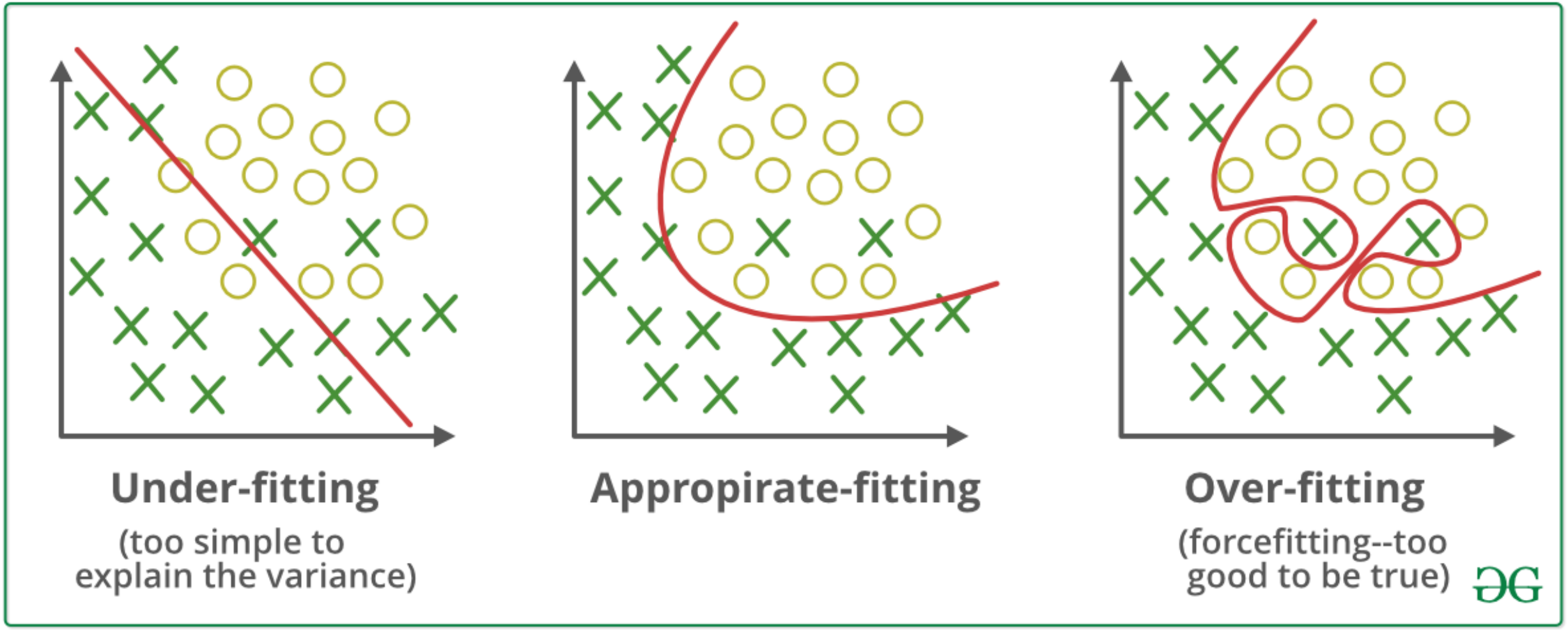
Comparing to human- level performance

Avoidable bias

Avoidable bias

- Human-level performance serves as a crucial benchmark for evaluating machine learning algorithms, guiding their design, and focusing efforts on bias and variance reduction. If algorithms perform close to human levels, they are considered effective; otherwise, improvements are needed.
- Advances in deep learning have made comparisons with human performance more common, leveraging human expertise for well-defined and structured tasks in algorithm design.
- This comparison informs the direction of efforts, emphasizing variance reduction when human performance is near Bayes optimal and bias reduction when significantly worse.

Bias and Variance



Bias and Variance


Cat classification

Cat classification

0.5%

1%

Training set error: 1% Dev set error:

Cat classification example

	Classification error (%)	
	Scenario A	Scenario B
Humans	1	7.5
Training error	8	8
Development error	10	10

Training error 8% | 8 %

Dev error 10% | 10 %

In this case, the human level error as a proxy for Bayes error since humans are good to identify images. If you want to improve the performance of the training set but you can't do better than the Bayes error otherwise the training set is overfitting. By knowing the Bayes error, it is easier to focus on whether bias or variance avoidance tactics will improve the performance of the model.

Scenario A

There is a 7% gap between the performance of the training set and the human level error. It means that the algorithm isn't fitting well with the training set since the target is around 1%. To resolve the issue, we use bias reduction technique such as training a bigger neural network or running the training set longer.

Scenario B

The training set is doing good since there is only a 0.5% difference with the human level error. The difference between the training set and the human level error is called avoidable bias. The focus here is to reduce the variance since the difference between the training error and the development error is 2%. To resolve the issue, we use variance reduction technique such as regularization or have a bigger training set.

FPT UNIVERSITY

Understanding human-level performance

Understanding human-level performance

- The definition of "human-level performance" is important to ensure precision in machine learning research.
- One way to define it is by using it as an estimate for Bayes error. By doing so, we can measure bias and variance in machine learning models.
- The difference between the estimate of Bayes error and the training error tells us how much avoidable bias is a problem, while the difference between training error and dev error tells us how much variance is a problem.

Human-level error as a proxy for Bayes error

- Medical image classification example:
- Suppose:
  - Typical human …….. 3 % error
  - Typical doctor …….... 1 % error
  - Experienced doctor ……... 0.7 % error
  - Team of experienced doctors .. 0.5 % error

- What is “human-level” error?

Error analysis example

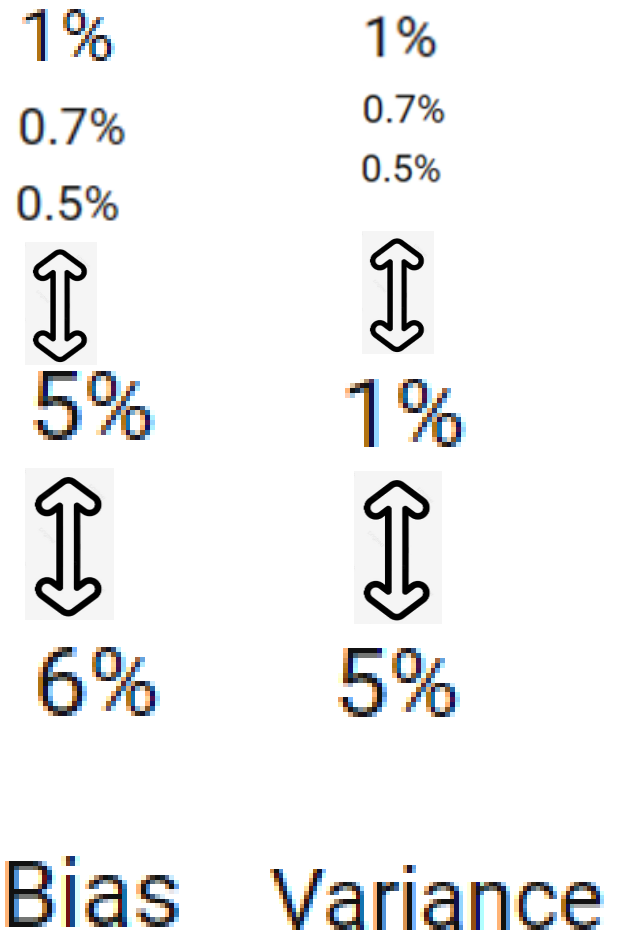
Human-level error gives an estimate of Bayes error.

Avoidable bias

Training error

variance

Dev error

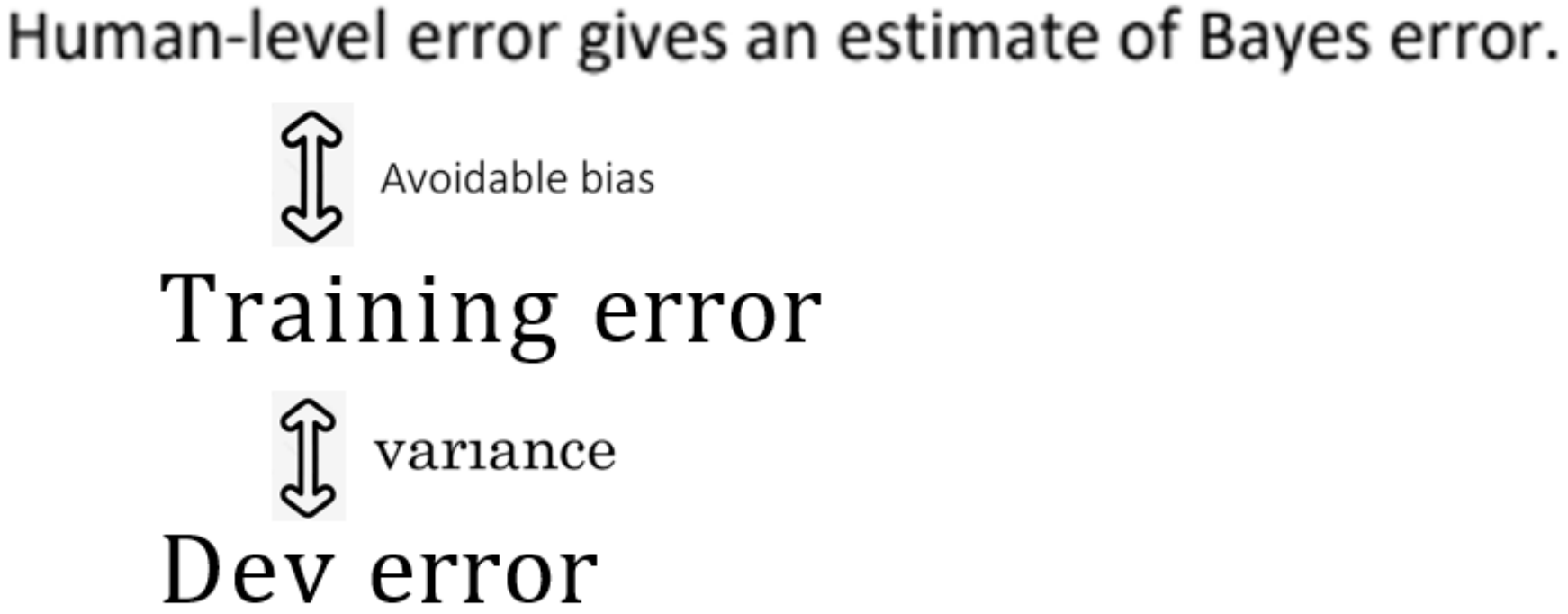


Understanding human-level performance

FPT Education
FPT UNIVERSITY

- As we approach human-level performance, it becomes harder to tease out the bias and variance effects, and progress on a machine learning project becomes more challenging. However, as machine learning algorithms continue to improve, it's possible to surpass human-level performance, which opens up new opportunities for applications and innovation.
- Overall, having a precise definition of human-level performance can help drive progress in machine learning projects by providing a benchmark for comparison and a way to estimate Bayes error and analyse bias and variance.

Summary of bias/variance with human-level performance



FPT UNIVERSITY

Surpassing human- level performance

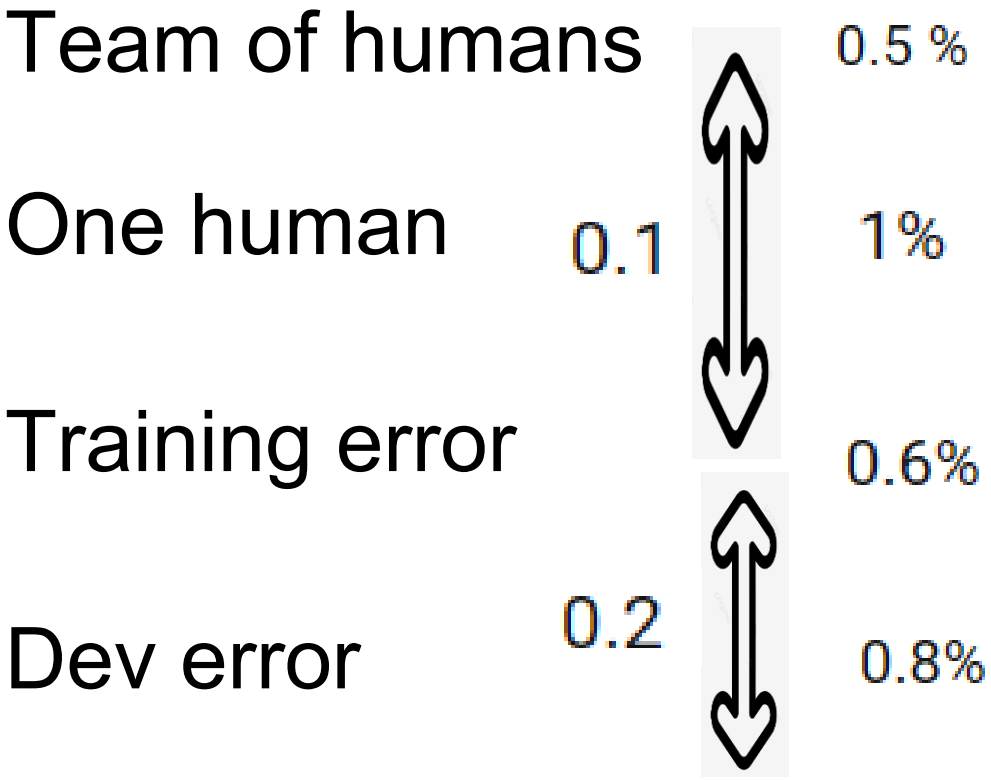
Surpassing human- level performance

- Surpassing human-level performance in machine learning is a notable achievement, yet it complicates further progress due to challenges in discerning whether to prioritize bias or variance reduction.
- Human intuition becomes less reliable as algorithms approach or exceed human-level capabilities.
- While machine learning excels in structured data tasks with extensive data, it faces difficulties surpassing humans in natural perception tasks like speech recognition, computer vision, and natural language processing.

Surpassing human- level performance

- In recent years, there have been some medical tasks like diagnosing skin cancer or reading ECGs, where computers are getting good at surpassing human-level performance.
- Overall, surpassing human-level performance in machine learning is not easy, but with enough data and advances in deep learning, it is possible.

Surpassing human-level performance



Problems where ML significantly surpasses human-level performance

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

FPT UNIVERSITY

Comparing to human- level performance

Improving your model performance

Improving your model performance

- To improve the performance of a learning algorithm, it's important to consider two things: fitting the training set well and generalizing well to the dev or test set. These two issues can be addressed by different techniques, which can be thought of as two separate sets of knobs to adjust.
- To reduce avoidable bias, some tactics include training a bigger model, using a better optimization algorithm, finding a better neural network architecture, and trying out other models.
- To reduce variance, some techniques to try include getting more training data, using regularization, data augmentation, or trying various neural network architecture and hyperparameters.

Improving your model performance

- Striking the right balance between avoiding bias and variance is a complex task that requires a lot of experimentation and iteration.
- Systematically applying the concepts of estimating avoidable bias and variance can help make this process more efficient and strategic.

The two fundamental assumptions of supervised learning

- You can fit the training set pretty well.
- The training set performance generalizes pretty well to the dev/test set.

Reducing (avoidable) bias and variance

FPT Education
FPT UNIVERSITY

Human-level	Train bigger model
	Train longer/better optimization algorithms
Training error	NN architecture/hyperparameters search
Dev error	More data
	Regularization
	NN architecture/hyperparameters search

Summarization

- Strategic ML involves efficient goal achievement, resource allocation, and informed decisions.
- Orthogonalization advises breaking down complex problems into distinct components.
- Single-number metrics simplify evaluation, ensuring clear model comparisons.
- Satisficing and optimizing metrics achieve a balanced approach to goal fulfillment.
- Consistent data distribution in train, dev, and test sets ensures fair evaluation. Sufficiently large dev sets provide meaningful feedback, and representative test sets gauge generalization.
- Changes to data or metrics may be needed based on problem shifts.
- Human-level performance serves as a benchmark for model potential and improvement areas.

Summarization

- Addressing avoidable bias improves algorithms, while variance is mitigated by more data.
- Analyzing gaps between model and human performance guides realistic goal-setting.
- Surpassing human-level performance indicates addressing previously challenging tasks.
- Strategies for model improvement include understanding errors, exploring architectures, regularization, and acquiring more data.

Questions

FPT Education
FPT UNIVERSITY

1. Does having three evaluation metrics slow down algorithm choice and team iteration?
2. Which model would you choose: 98% accuracy, 9s runtime, 9MB memory?
3. Is accuracy optimizing and runtime/memory satisficing metrics?
4. Should you avoid adding citizens' data to training if it differs from dev/test data?
5. Why object to adding citizens' data to the test set?
6. Does 4% train error and 4.5% dev error suggest training a bigger network to reduce train error?
7. Define human-level performance as a proxy for Bayes error given these accuracy levels?
8. Can learning algorithm performance exceed human-level but not Bayes error?
9. Given 0.1% human-level, 2% train error, 2.1% dev error, what are two promising improvement options?
10. Given 0.1% human-level, 2% train, 2.1% dev, 7% test error, what are the two best conclusions?
11. With 0.1% human-level, 0.05% train/dev error, what conclusions can be made?
12. If your system has higher accuracy but more false negatives than a competitor's, and the client prefers fewer false negatives, what should you do?
13. If system performance degrades due to a new bird species, what action should be taken?
14. Agree with statements about training a cat detector with a large dataset and long training time?