1. Orthogonalization:

   Orthogonalization in machine learning refers to the strategy of having distinct, clear-cut controls or adjustments for different aspects of a machine learning system, akin to tuning different knobs for separate functions on an old-school TV or having distinct controls for steering, acceleration, and braking in a car. This approach simplifies the process of diagnosing and addressing specific issues within the system, enabling more effective and targeted tuning of the model.

   The concept is particularly useful when considering the multitude of factors one could adjust in a machine learning model, including various hyperparameters and architectural choices. Orthogonalization allows practitioners to isolate and address issues across different stages of model development and evaluation:

   1. **Training Performance**: If the model doesn't perform well on the training set, adjustments might involve increasing model complexity or changing the optimization algorithm to ensure the model can capture the underlying pattern in the data.
   2. **Development Set Performance**: If the model performs well on the training set but not on the development (dev) set, the focus shifts to improving generalization, possibly through regularization techniques or acquiring more diverse training data to reduce overfitting.
   3. **Test Set Performance**: A discrepancy between performance on the dev set and the test set might indicate an overfitting to the dev set, suggesting the need for a larger or more representative dev set to better generalize to unseen data.
   4. **Real-world Performance**: If a model performs well according to the metrics on the test set but fails to meet real-world expectations, it might be necessary to revisit the choice of metrics (cost function) or to reassess the representativeness of the dev and test sets relative to real-world data distributions.

   Orthogonalization emphasizes the importance of having specific, targeted adjustments (knobs) for each of these areas, rather than making changes that affect multiple aspects simultaneously, which can complicate the tuning process and obscure the effects of any single adjustment. By clearly delineating the controls for each part of the model's performance, machine learning practitioners can more systematically and efficiently improve their models.

2. # Single Number Evaluation Metric

   In the discussed video, the importance of having a single real number evaluation metric for evaluating machine learning models is emphasized. Applied machine learning is described as an empirical process involving iteration through cycles of idea generation, experimentation, and refinement based on the outcomes. The example provided is a cat classifier, where two versions of the classifier, A and B, are compared using precision and recall. Precision measures the percentage of true positive predictions among all positive predictions made by the classifier, while recall measures the percentage of true positive predictions among all actual positives.

   The video highlights a common challenge: precision and recall often trade off against each other, making it difficult to determine which classifier performs better when one has higher precision and the other has higher recall. To address this, the video recommends using a single evaluation metric that combines both precision and recall, rather than evaluating them separately. The suggested metric is the F1 score, which is the harmonic mean of precision and recall, represented by the formula (2 /

(\frac{1}{P} + \frac{1}{R})). The F1 score provides a balanced measure of both precision and recall, enabling quicker and more straightforward comparison between classifiers.

Further, the video illustrates another scenario involving a cat app with users from different geographies (the US, China, India, and others). It shows how tracking performance across these geographies with multiple metrics can be cumbersome. Instead, computing an average error across all geographies offers a single real number evaluation metric that simplifies the decision-making process on which classifier to proceed with.

In summary, the video argues for the efficiency and clarity gained by using a single real number evaluation metric, like the F1 score or average error, in machine learning projects. This approach speeds up the iterative process of improving algorithms by enabling quick comparisons and decisions on which models or classifiers perform better, facilitating faster progress in machine learning endeavors.

3. # Satisficing and Optimizing Metric

The passage discusses a strategy for evaluating machine learning classifiers when multiple performance metrics are important. Specifically, it introduces the concept of "optimizing" and "satisficing" metrics as a method to balance different evaluation criteria, such as accuracy and running time, in the context of a cat classifier example.

- **Optimizing Metric**: This is the primary metric you aim to maximize or minimize. In the given example, accuracy is chosen as the optimizing metric, meaning that among the classifiers considered, the goal is to achieve the highest possible accuracy.
- **Satisficing Metric**: These are secondary metrics that need to meet a certain threshold but do not necessarily need to be optimized beyond that. In the example, running time serves as a satisficing metric, with the requirement that the classifier must classify an image in 100 milliseconds or less. As long as this condition is met, further improvements in speed are not prioritized over accuracy.

Classifier B is identified as the best choice because it offers the highest accuracy while keeping the running time below the 100-millisecond threshold. This approach allows for a more nuanced evaluation of classifiers that accounts for multiple factors, enabling the selection of a classifier that best meets all criteria without the need to artificially combine metrics into a single composite score.

The passage further illustrates this approach with a wake word detection system example, where the optimizing metric is the accuracy of detecting the trigger word, and the satisficing metric is the number of false positives per day, which is set not to exceed one.

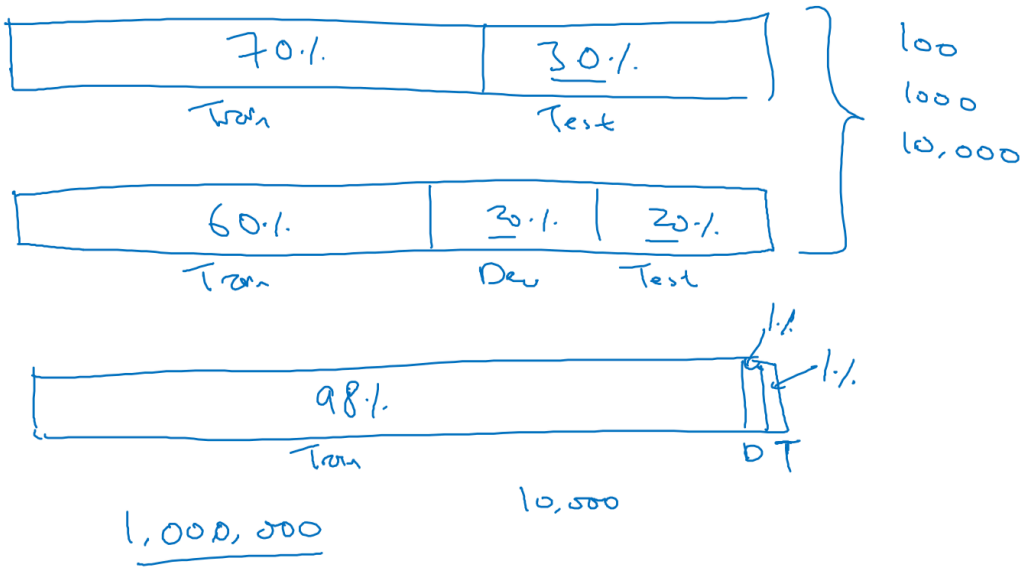
This methodology provides a structured way to make decisions when faced with multiple performance metrics. By setting clear priorities and acceptable thresholds, it allows for the efficient selection of the "best" option among multiple classifiers or models based on the specific requirements of the project.

4. # Train/Dev/Test Distributions

They should come from the same distribution

5. # Size of the Dev and Test Sets

Old way of splitting data



The video discusses the evolution of best practices for dividing datasets into training, development (dev), and test sets in the context of deep learning. Traditional machine learning guidelines often recommended a 70/30 split between training and test sets, or a 60/20/20 split if including a dev set. These proportions were suitable when datasets were relatively small, such as hundreds or thousands of examples.

However, with the advent of deep learning, data sizes have increased significantly, making it common to work with much larger datasets, such as millions of examples. In such cases, the video suggests that it might be more appropriate to allocate a larger percentage of the data to the training set, with 98% for training and just 1% each for dev and test sets. This adjustment is justified by deep learning's substantial demand for data and the realization that even 1% of a large dataset can provide tens of thousands of examples, which is often sufficient for effective evaluation and testing.

The purpose of the test set, as emphasized in the video, is to give a high confidence in the overall performance of the system after development is complete. The size of the test set should be large enough to achieve this goal, but not necessarily as large as 30% of the total data, especially in the context of very large datasets. The exact size depends on the specific requirements of the application and the desired confidence level in the system's performance.

The video also touches on scenarios where a test set might not be necessary, especially in cases where the development set (dev set) is large enough to prevent overfitting. While not generally recommended, forgoing a test set can be considered if the dev set is significantly large and the primary focus is on tuning the model based on this set.

In summary, the video advises a shift from the traditional fixed percentage splits to a more flexible approach that allocates more data to training and adjusts the sizes of dev and test sets based on the size of the dataset and the specific needs of the project. This reflects the changing dynamics in machine learning and deep learning, where larger datasets have become the norm, necessitating adjustments in how data is partitioned to effectively train, evaluate, and test models.

6. # When to Change Dev/Test Sets and Metrics?

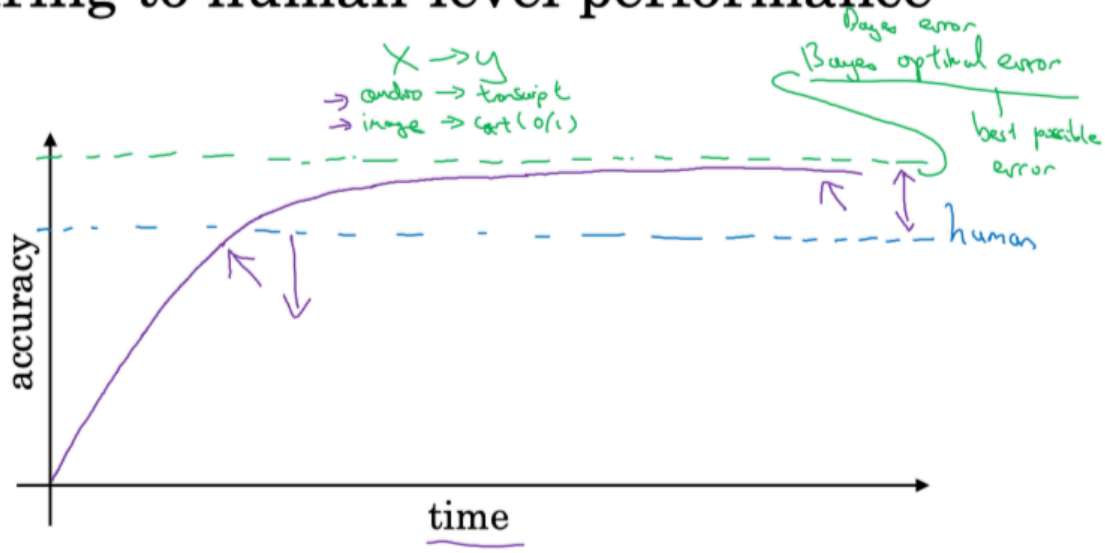
The text discusses the importance of having a well-defined development (dev) set and evaluation metric in machine learning projects to guide the team's efforts towards a specific goal, akin to setting a target for them to aim at. However, it highlights that sometimes the chosen target (evaluation metric) may not accurately reflect the project's needs, necessitating a reevaluation and adjustment of the target. The example provided illustrates a scenario where two algorithms, A and B, are developed to classify cat images, with A initially appearing superior based on a lower classification error. However, upon realizing that Algorithm A also misclassifies pornographic images as cat images, which is unacceptable, it becomes clear that despite its lower error rate, it is not the better algorithm for the project's real-world application. This discrepancy indicates that the evaluation metric (classification error in this case) fails to adequately prioritize the project's actual needs, suggesting a need for modifying the evaluation metric to more heavily penalize certain types of misclassifications, such as mislabeling pornographic content.

The text proposes adjusting the evaluation metric by introducing a weight term to give greater penalty to specific misclassifications, emphasizing the need for the metric to reflect the practical requirements of the project accurately. It also discusses the concept of orthogonalization in machine learning, suggesting that defining the evaluation metric and optimizing the model to perform well according to that metric should be considered as separate, distinct steps. Additionally, it touches upon the potential need to adjust the dev and test sets to better reflect the real-world conditions the algorithm will operate under, ensuring the evaluation process accurately predicts real-world performance.

The overarching message is that while setting up an initial evaluation metric and dev set is crucial for guiding development efforts efficiently, it is equally important to remain flexible and willing to revise these parameters as better understanding of the project's needs develops. This approach ensures that the team can iteratively improve the algorithm's performance in a way that is genuinely aligned with the project's goals.

7. # Why Human-level Performance?

Comparing to human-level performance



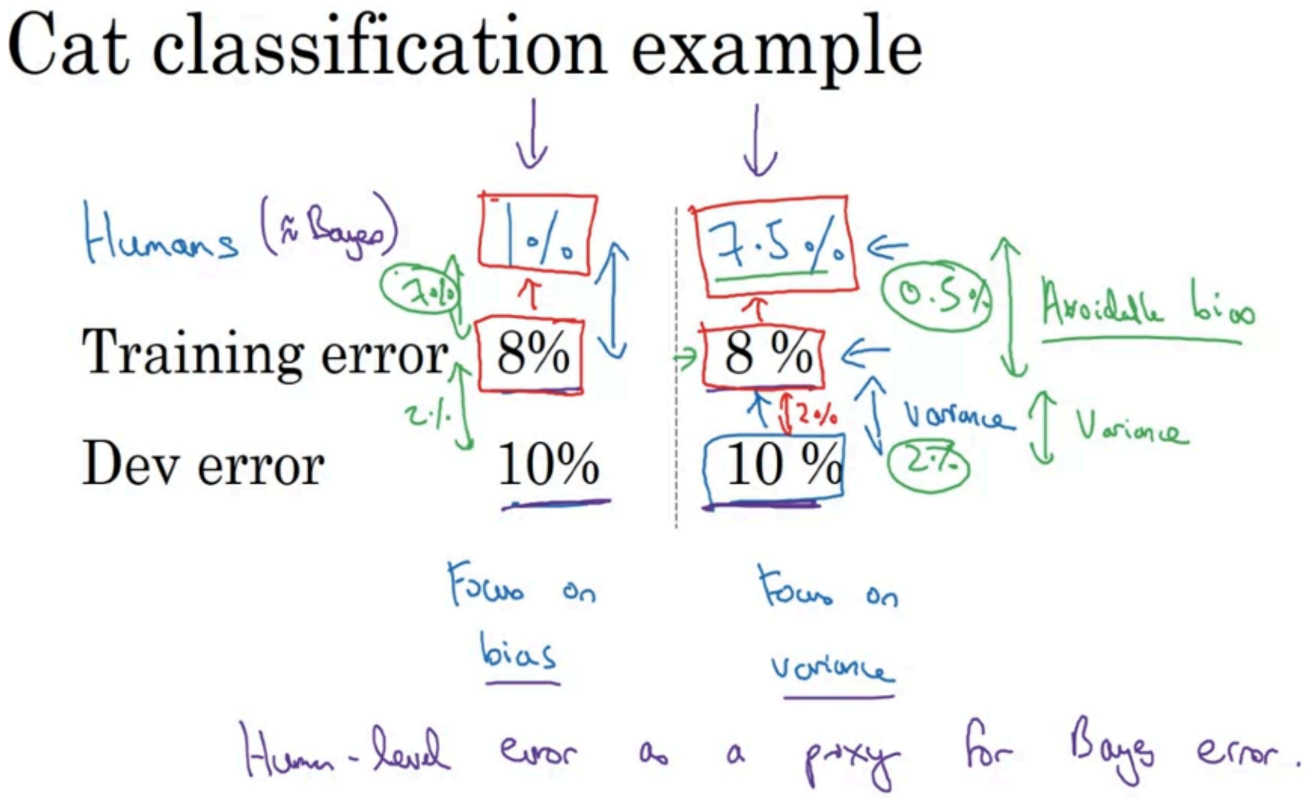
Andrew Ng

Why compare to human-level performance

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.

8. Avoidable Bias:



The conversation discusses the importance of comparing a learning algorithm's performance to human-level performance to guide decisions on improving the algorithm. When evaluating a learning algorithm, such as one for cat classification, understanding human-level error is crucial. In scenarios where human-level error is very low (e.g., 1%), and the algorithm shows significantly higher training (8%) and development (dev) error (10%), the focus should be on reducing bias. This suggests the algorithm isn't fitting the training set well enough, and strategies like training a larger neural network or extending training time are recommended.

Conversely, if human-level error is higher (e.g., 7.5%), possibly due to challenging conditions like blurry images, and the algorithm's training and dev errors are similar to the first scenario, the approach changes. Here, the algorithm performs close to human level, indicating it's fitting the training set reasonably well. The focus shifts towards reducing variance, for instance, by applying regularization techniques to narrow the gap between training and dev errors.

This discussion introduces the concept of "Bayes error" as a theoretical minimum error rate, with human-level error serving as a practical proxy. Depending on the gap between human-level (or Bayes) error and the algorithm's performance, efforts can be directed towards reducing "avoidable bias" (the difference between human-level error and training error) or "variance" (the difference between training and dev errors). The choice between focusing on bias reduction or variance reduction is guided by the potential for improvement relative to human-level performance, emphasizing a strategic approach to enhancing learning algorithms.

9. Understanding Human-level Performance:

Human-level error as a proxy for Bayes error

Medical image classification example:

Suppose:

(a) Typical human ..... 3 % error

(b) Typical doctor ..... 1 % error

(c) Experienced doctor ..... 0.7 % error

(d) Team of experienced doctors .. 0.5 % error

Bayes error $\leq$ 0.5%

What is "human-level" error?

In the provided transcript, the concept of human-level performance is discussed in the context of machine learning, specifically in relation to medical image classification. The discussion centers on defining human-level error as a means to estimate Bayes error—the theoretical minimum error rate that any classifier could achieve on a given task. This concept is crucial for understanding the limits of machine learning models and guiding the development of these models to achieve better performance.

Human-level error rates are presented for a medical image classification task, with varying error rates depending on the expertise of the human evaluating the images:

- An untrained human achieves a 3% error rate.
- A typical doctor achieves a 1% error rate.
- An experienced doctor achieves a 0.7% error rate.
- A team of experienced doctors, working together and discussing the images, achieves a 0.5% error rate.

In this context, human-level error is proposed to be defined as the lowest error rate achievable by humans working on the task—0.5% in this example—because this rate serves as the best current estimate for Bayes error. This definition is particularly useful for analyzing bias and variance within a machine learning model. Bias refers to the difference between the expected (or average) prediction of the model and the correct value that we are trying to predict. Variance refers to the variability of model prediction for a given data point. Understanding these concepts helps identify whether a model's errors are due to a lack of complexity (high bias) or due to being too sensitive to the training data (high variance).

The choice of human-level performance as a benchmark is crucial for guiding the development of machine learning models. It helps in:

- Estimating Bayes error, which is essential for understanding the theoretical limits of model performance on a given task.
- Analyzing the model's bias and variance, which in turn, guides the selection of strategies for model improvement (e.g., focusing on reducing bias through training a bigger network or reducing variance through techniques like regularization or acquiring more training data).

The transcript also notes that the definition of human-level performance might differ based on the context, such as research publication versus practical deployment of a system. For example, surpassing a typical doctor's performance might be a significant achievement for deploying a system in a practical medical setting.

Furthermore, the closer a machine learning model gets to human-level performance, the more challenging it becomes to make further improvements. This is because distinguishing between bias and variance, and thus understanding where to focus development efforts, becomes harder as performance improves. This complexity underscores the importance of accurately estimating Bayes error and understanding human-level performance in the progression of machine learning projects.

Overall, the discussion highlights the nuanced nature of measuring and striving for human-level performance in machine learning, emphasizing its role in estimating theoretical performance limits (Bayes error) and guiding the reduction of bias and variance in models.

10. Surpass Human-level performance:

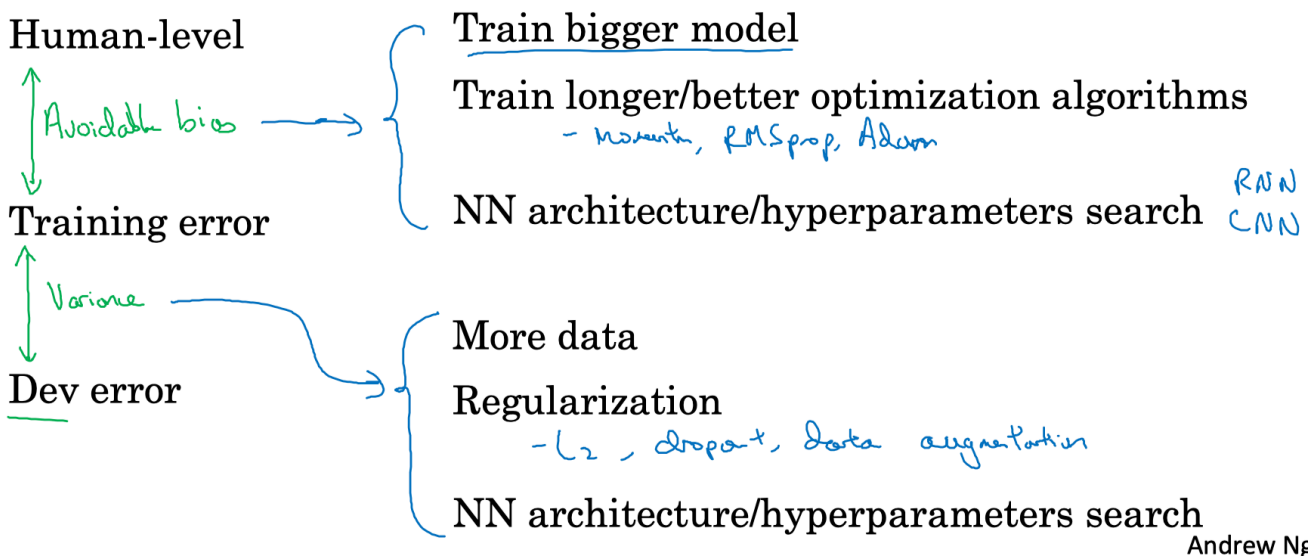
    Surpassing human-level performance in machine learning tasks is a significant milestone that brings both excitement and challenges. The difficulty in making progress increases as performance approaches or surpasses human levels due to a few key factors:

    1. **Determining Avoidable Bias and Variance**: When an algorithm's performance nears human-level, it becomes challenging to assess whether to focus on reducing avoidable bias or variance. For example, if a team of humans achieves a 0.5% error rate and an algorithm achieves 0.4% dev error, it's unclear whether the focus should be on reducing bias or variance, complicating progress.
    2. **Lack of Clear Improvement Directions**: Surpassing human-level performance makes it harder to rely on human intuition for further improvements. Traditional methods of identifying enhancements may not be as effective, requiring new strategies for progress.
    3. **Availability of Large Data Sets**: Success in surpassing human levels often comes in areas with access to vast amounts of structured data, allowing algorithms to identify patterns beyond human capability. This includes fields like online advertising, product recommendations, logistics, and loan approval processes, where machine learning significantly outperforms human judgment.
    4. **Natural Perception Tasks**: While machine learning has excelled in structured data tasks, surpassing human performance in natural perception tasks (like computer vision, speech recognition, and natural language processing) is more challenging due to the inherent proficiency of humans in these areas.
    5. **Medical and Specific Tasks**: In certain domains, such as medical diagnostics and radiology, machine learning has begun to surpass human performance, benefiting from large data sets and recent advances in deep learning.

    In conclusion, surpassing human-level performance in machine learning is complex and requires navigating issues of bias, variance, and the diminishing utility of human intuition for further improvements. Success often depends on the availability of large data sets and the specific domain of application, with structured data tasks currently offering more opportunities for machine learning to outperform humans. Despite these challenges, the ongoing advances in deep learning continue to push the boundaries, enabling machines to achieve and exceed human-level performance in increasingly diverse tasks.

11. Improve your model performance:

Reducing (avoidable) bias and variance



Andrew Ng

a