Lecture #11: Practical Advice for Applying Machine Learning*

* Slides partly based on Andrew Ng

Overview

- How to make ML work in the real-world?

- Mostly experiential advice
  - Also based on what other researchers and practitioners have said

ML and Real-world

- Diagnostics of your learning algorithm

- Error analysis

Debugging ML Algorithm

- Suppose you train an SVM or a logistic regression classifier for spam detection
- You followed the best practices for finding the hyper-parameters (e.g., cross-validation)
- Your classifier is only 65% accurate
- What can you do to improve it?

Different ways to improve your model

- **More training data**
- **Features**
  - use more
  - use fewer
  - use different ones
- **Better training**
  - run for more different iterations
  - use a different algorithm
  - use a different classifier
  - plug-and-play with regularization

Different ways to improve your model

- **More training data**
- **Features**
  - use more
  - use fewer
  - use different ones
- **Better training**
  - run for more different iterations
  - use a different algorithm
  - use a different classifier
  - plug-and-play with regularization

Tedious!

- Prone to errors, trying your luck
- How can we make this process more methodical?

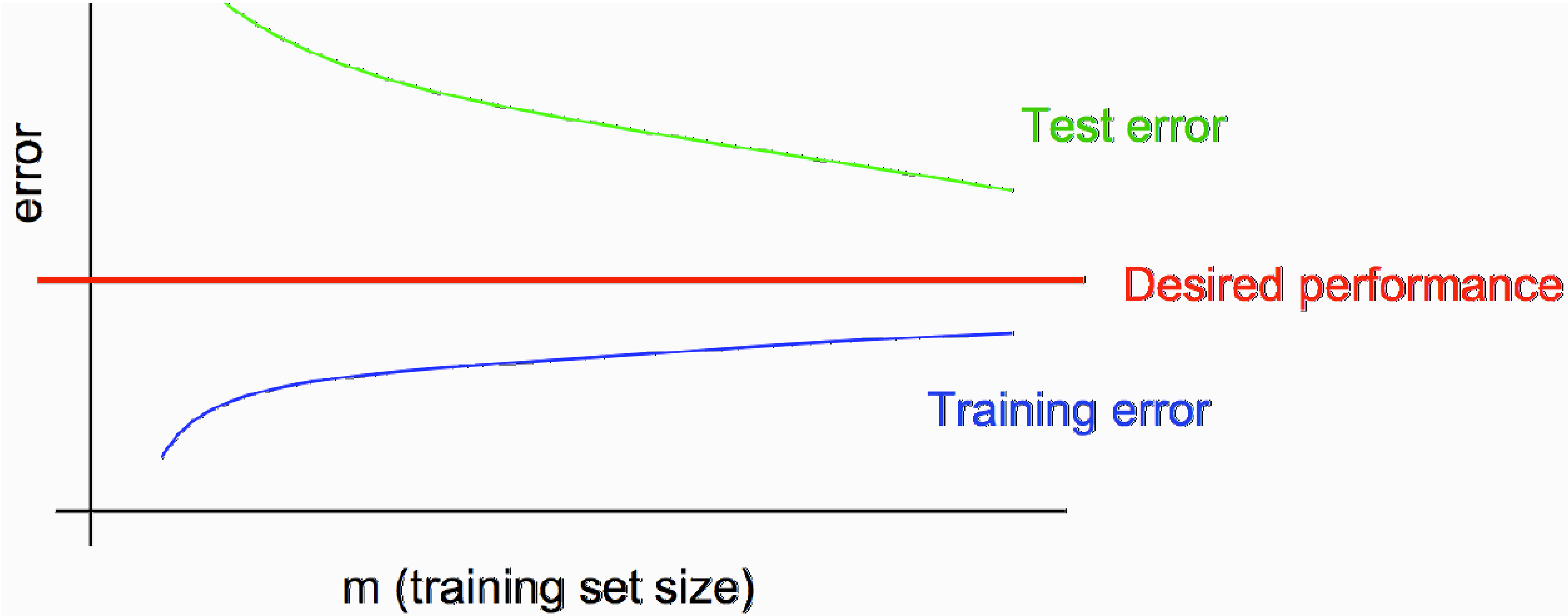
Diagnostics

- **Easier to fix a problem if you know where it is**
- **Some possible problems**
  - Over-fitting (high variance)
  - Under-fitting (high-bias)
  - Your learning does not converge
  - Your loss function is not good enough (if we want to build a classifier, we should aim for the 0-1 loss)

Detecting Over or Under fitting

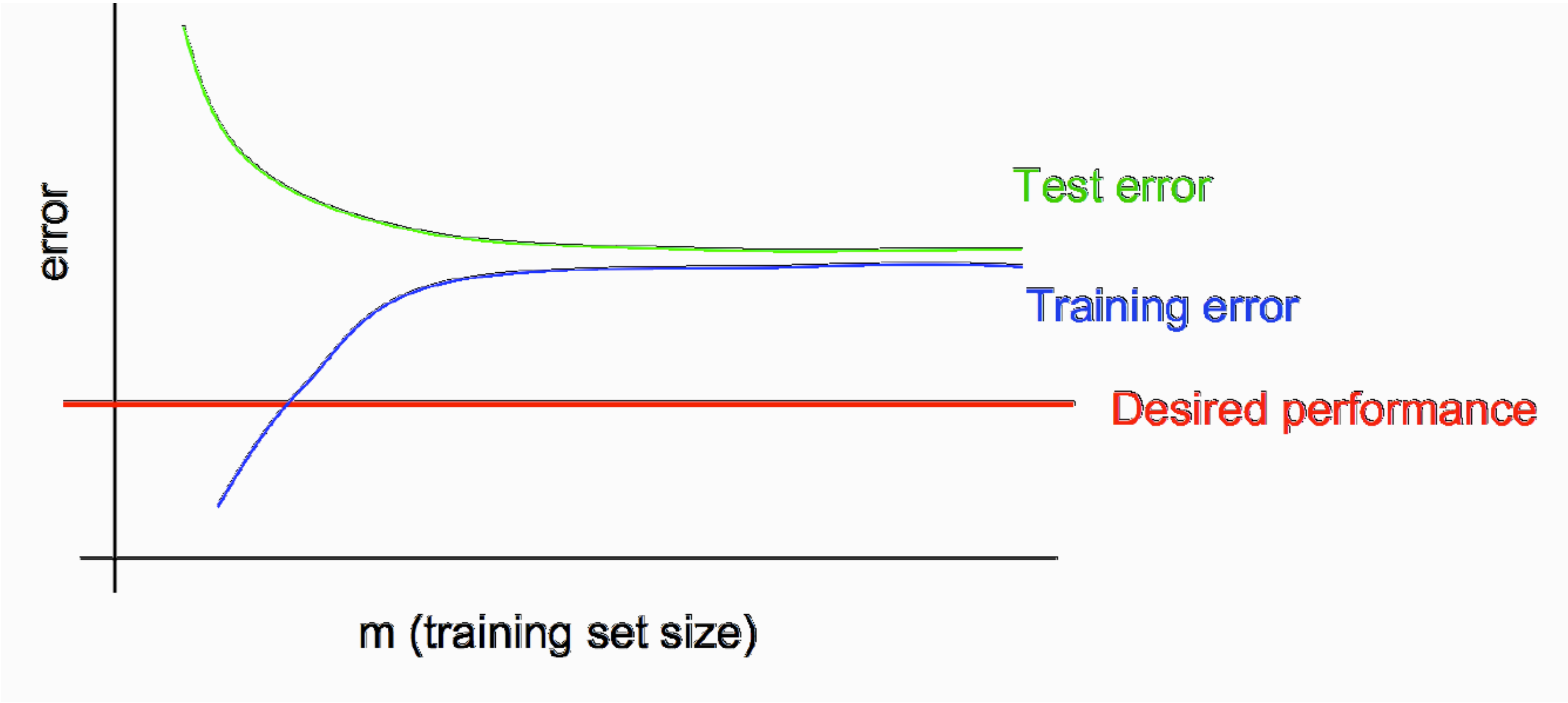
- **Over-fitting**
  - The training accuracy is much higher than the testing accuracy
  - The model explains the training set very well, but poor generalization
- **Under-fitting**
  - Both training and testing accuracies are very low
  - The model cannot represent the concept well enough

Detecting high variance using learning curves



- Test error keeps decreasing as training set increases => more data will help
- Large gap between train and test error

Detecting high bias using learning curves



- Both train and test error are unacceptable
- But the model seems to converge

Different ways to improve your model

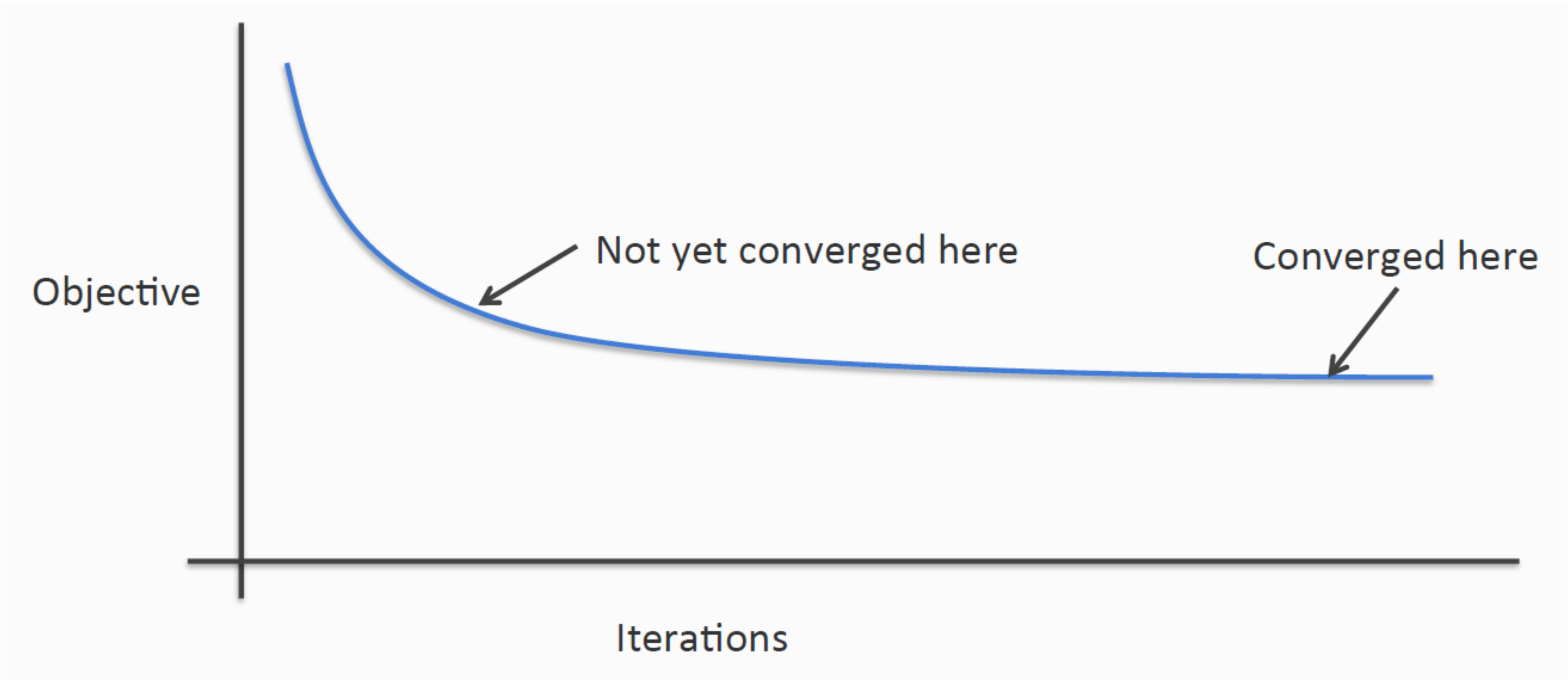
- **More training data** : Helps with over-fitting
- **Features**
  - use more : Helps with under-fitting
  - use fewer : Helps with over-fitting
  - use different ones : Could help both
- **Better training**
  - run for more different iterations
  - use a different algorithm
  - use a different classifier
  - plug-and-play with regularization : could help both

Diagnostics

- **Easier to fix a problem if you know where it is**
- **Some possible problems**
  - Over-fitting (high variance)
  - Under-fitting (high-bias)
  - Your learning does not converge
  - Your loss function is not good enough (if we want to build a classifier, we should aim for the 0-1 loss)

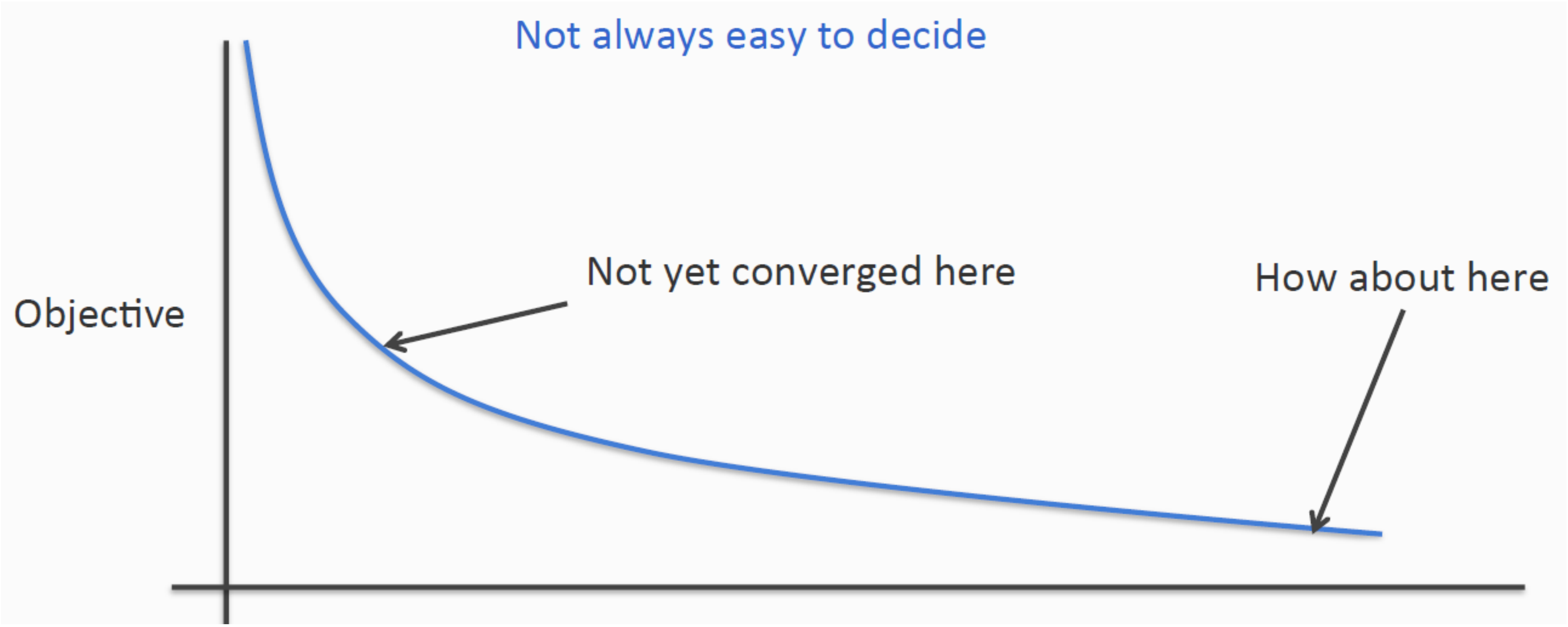
Does your learning algorithm converge?

- If learning is framed as an optimization problem, track the objective



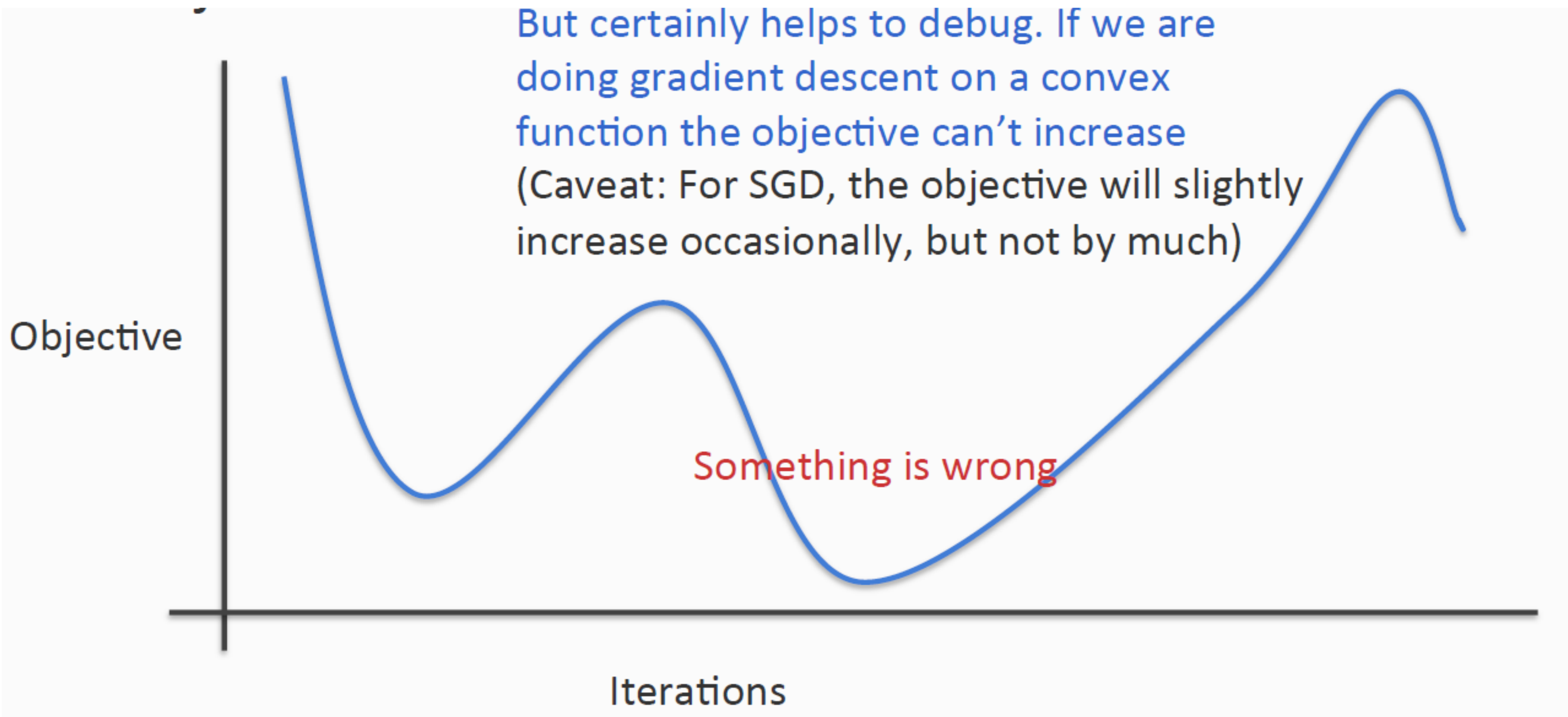
Does your learning algorithm converge?

- If learning is framed as an optimization problem, track the objective



Does your learning algorithm converge?

- If learning is framed as an optimization problem, track the objective



Diagnostics

- **Easier to fix a problem if you know where it is**
- **Some possible problems**
  - Over-fitting (high variance)
  - Under-fitting (high-bias)
  - Your learning does not converge
  - Your loss function is not good enough (if we want to build a classifier, we should aim for the 0-1 loss)

ML and Real-world

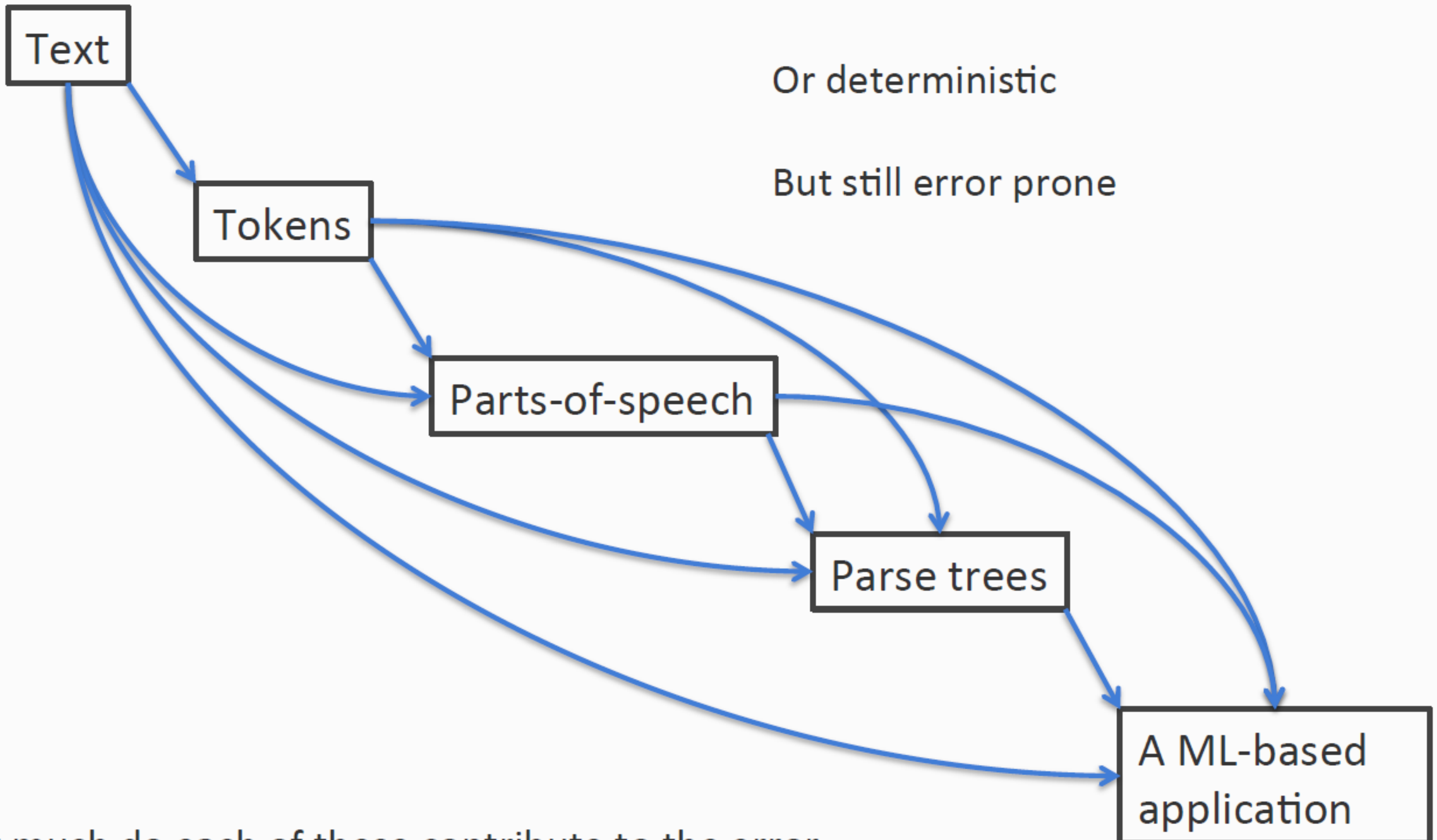
- Diagnostics of your learning algorithm

- Error analysis

Error Analysis

- Generally machine learning plays a small role in a larger application
  - Pre-processing
  - Feature extraction
  - Data transformations
  - …
- How much do each of these contribute to the error?
- Error analysis tries to explain why a system is not performing perfectly

Example: A typical NLP pipeline



Each of these could be ML driven

Or deterministic

But still error prone

How much do each of these contribute to the error of the final application?

Tracking errors in a complex system

- Plug-in the ground truth for the intermediate component and see how much the accuracy of the final system changes

System	Accuracy
End-to-end predicted	55%
With ground truth tokens	60%
+ ground truth parts-of-speech	84 %
+ ground truth parse trees	89 %
+ ground truth final output	100 %

Error in the part-of-speech component hurts the most

Ablation Study

- Explaining difference between the performance of a strong model and a much weaker one (baseline)
- Usually seen with features
- Suppose we have a collection of features and our system does well, but we don’t know which features are giving us the performance
- Evaluate simpler systems that progressively use fewer and fewer features to see which features give the highest boost

A new real-world application

- Do you have the right evaluation metric?
  - Does your loss function reflect it?

- Be aware of bias vs. variance trade-off (or over-fitting vs. under-fitting)

- Be aware that intuitions do not work in high dimensions
  - No proof by picture
  - Curse of dimensionality

A new real-world application

- A theoretical guarantee may only be theoretical
  - May make invalid assumptions (e.g., data is separable)
  - May only be legitimate with infinite data (e.g., estimating probabilities)
  - Experiments on real data are equally important

Big data is not enough

- Remember that learning is impossible without some bias that simplifies the search
  - Otherwise, no generalization
- Learning requires knowledge to guide the learner
  - Machine learning is not a magic wand
- But more data is always better
  - cleaner data is even better

What knowledge?

- Which model is the right one for this task?
  - Linear models, decision trees, kernels etc.
- Which learning algorithm?
- Feature engineering is important
- Implicitly, these are all claims about the nature of the problem

Miscellaneous advice

- Learn simpler models first
  - If nothing, at least they form a baseline that you can improve upon
- Ensembles seem to work better
- Think about whether your problem is learnable at all
  - Learning = generalization