Practical Advice for Building Machine Learning Applications

Machine Learning

THE UNIVERSITY OF UTAH

Based on lectures and papers by Andrew Ng, Tom Mitchell and others

ML and the world

Making ML work in the world

Mostly experiential advice

Also based on what other people have said

See readings on class website

- Diagnostics of your learning algorithm
- Error analysis
- Injecting machine learning into *Your Favorite Task*
- Making machine learning matter

ML and the world

- Diagnostics of your learning algorithm
- Error analysis
- Injecting machine learning into *Your Favorite Task*
- Making machine learning matter

Debugging machine learning

Suppose you train an SVM or a logistic regression classifier for spam detection

You *obviously* follow best practices for finding hyper-parameters (such as cross-validation)

Your classifier is only 75% accurate

What can you do to improve it?

(assuming that there are no bugs in the code)

Different ways to improve your model

More training data

Features

1. Use more features
2. Use fewer features
3. Use other features

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization

Different ways to improve your model

More training data

Features

1. Use more features
2. Use fewer features
3. Use other features

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization

Tedious!

And prone to errors, dependence on luck

Let us try to make this process more methodical

First, diagnostics

Easier to fix a problem if you know where it is

Some possible problems:

1. Over-fitting (high variance)
2. Under-fitting (high bias)
3. Your learning does not converge
4. Are you measuring the right thing?

Detecting over or under fitting

Over-fitting: The training accuracy is much higher than the test accuracy

- The model explains the training set very well, but poor generalization
- Variance: Describes how much the best classifier depends on the specific choice of the training set

Under-fitting: Both accuracies are unacceptably low

- The model can not represent the concept well enough
- Bias is the true error (loss) of the *best* predictor in the hypothesis set
- Underfitting: when bias is high.

Bias variance tradeoffs

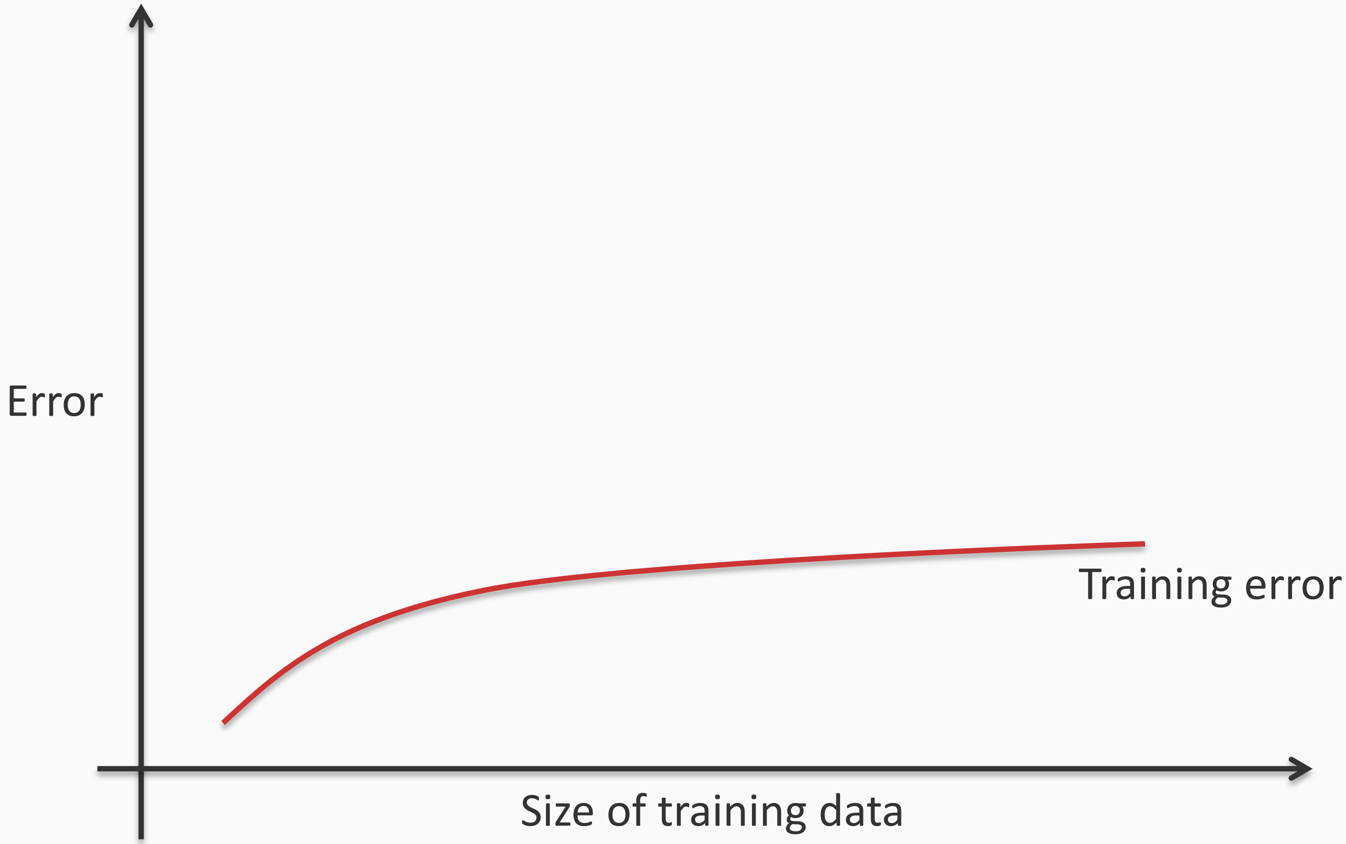
- Error = bias + variance (+ noise)
- High bias → both training and test error can be high
  - Arises when the classifier can not represent the data
- High variance → training error can be low, but test error will be high
  - Arises when the learner overfits the training set

Bias variance tradeoff has been studied extensively in the context of regression

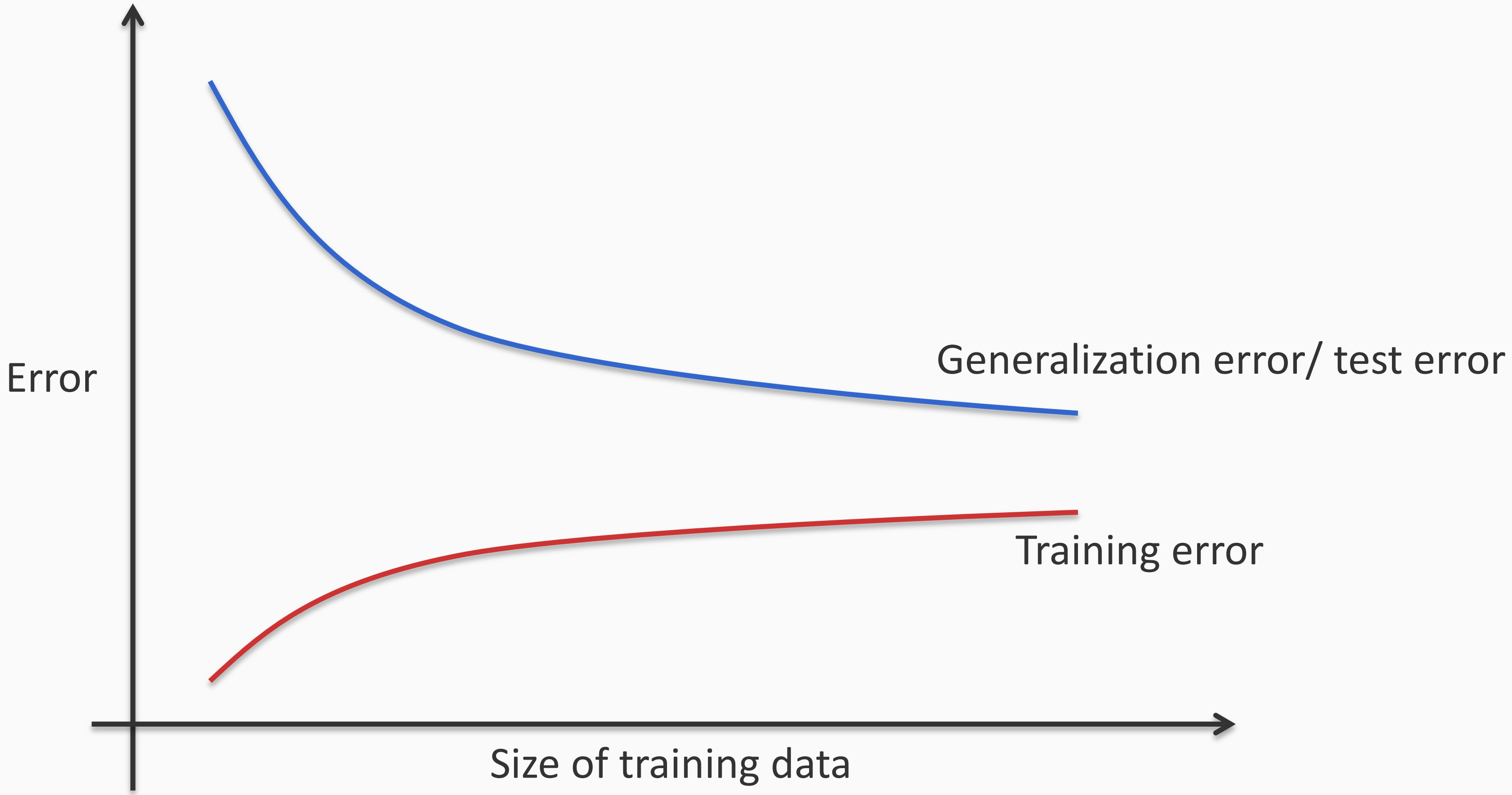
Managing bias and variance

- Ensemble methods reduce variance
  - Multiple classifiers are combined
  - Eg: Bagging, boosting
- Decision trees of a given depth
  - Increasing depth decreases bias, increases variance
- SVMs
  - Higher degree polynomial kernels decreases bias, increases variance
  - Stronger regularization increases bias, decreases variance
- Neural networks
  - Deeper models can increase variance, but decrease bias
- K nearest neighbors
  - Increasing k generally increases bias, reduces variance

Detecting high variance using learning curves



Detecting high variance using learning curves

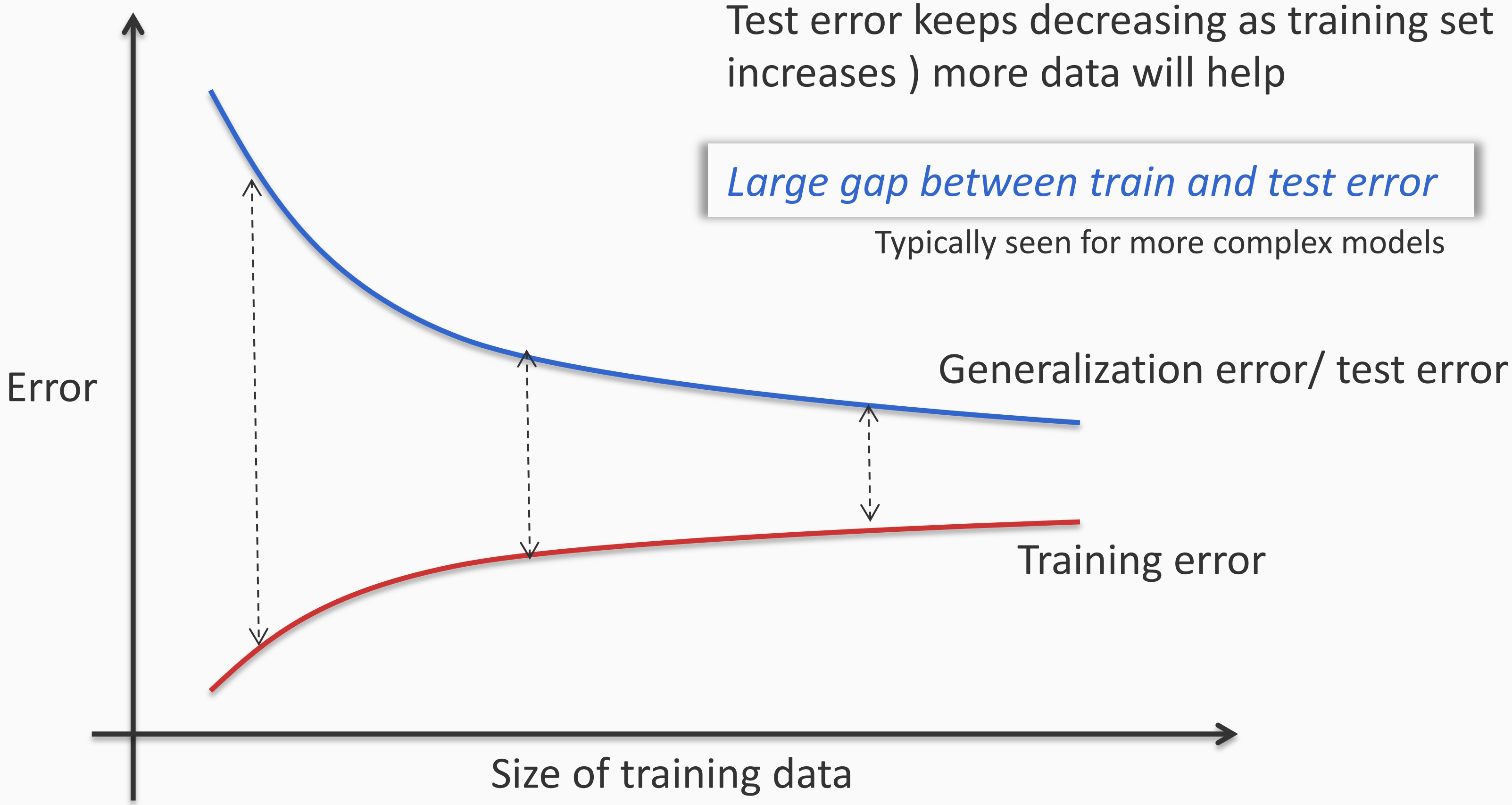


Detecting high variance using learning curves

Test error keeps decreasing as training set increases) more data will help

Large gap between train and test error

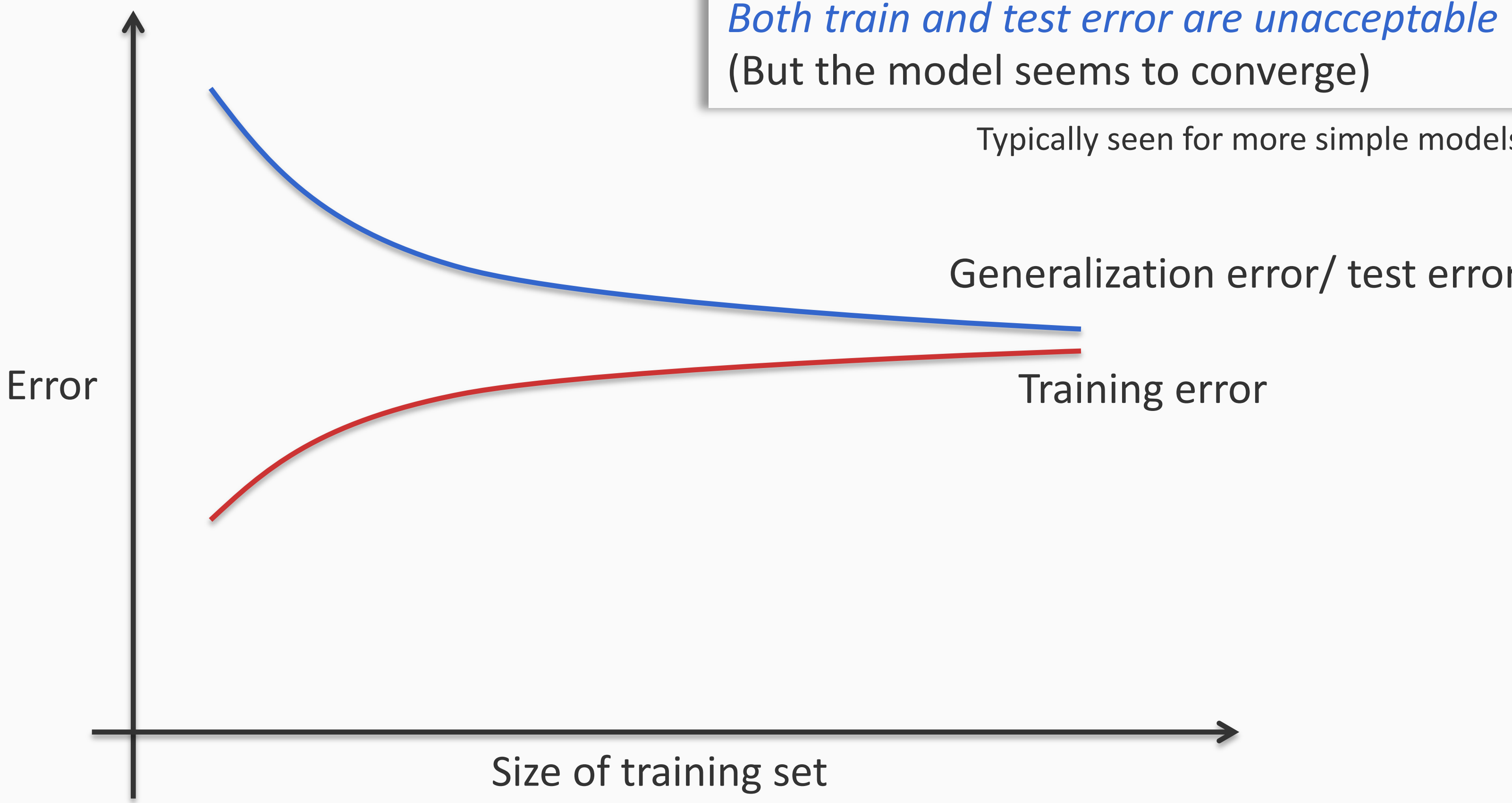
Typically seen for more complex models



Detecting high bias using learning curves

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

Typically seen for more simple models



Different ways to improve your model

More training data

Features

1. Use more features
2. Use fewer features
3. Use other features

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization

Different ways to improve your model

More training data

Helps with over-fitting

Features

1. Use more features — Helps with under-fitting
2. Use fewer features — Helps with over-fitting
3. Use other features — Could help with over-fitting and under-fitting

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization — Could help with over-fitting and under-fitting

Diagnostics

Easier to fix a problem if you know where it is

Some possible problems:

✓ Over-fitting (high variance)

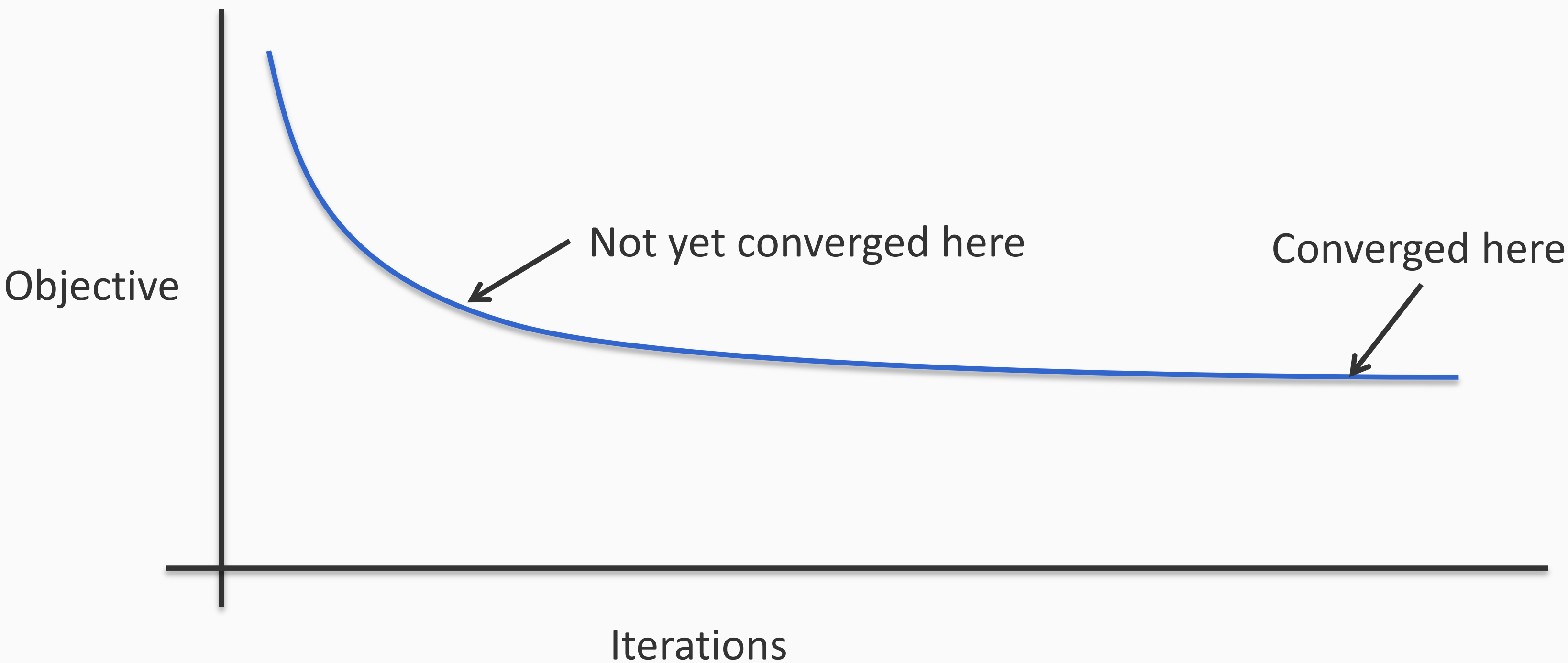
✓ Under-fitting (high bias)

3. Your learning does not converge

4. Are you measuring the right thing?

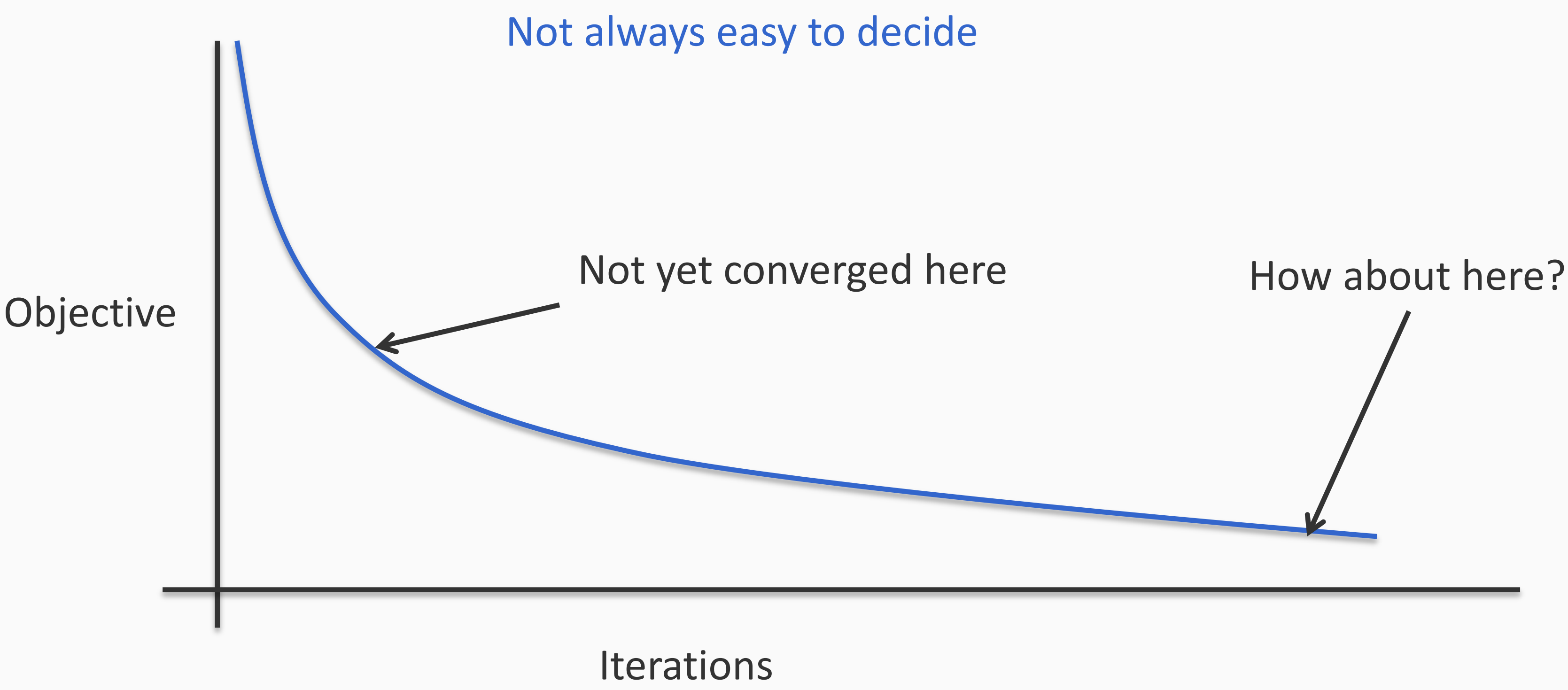
Does your learning algorithm converge?

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



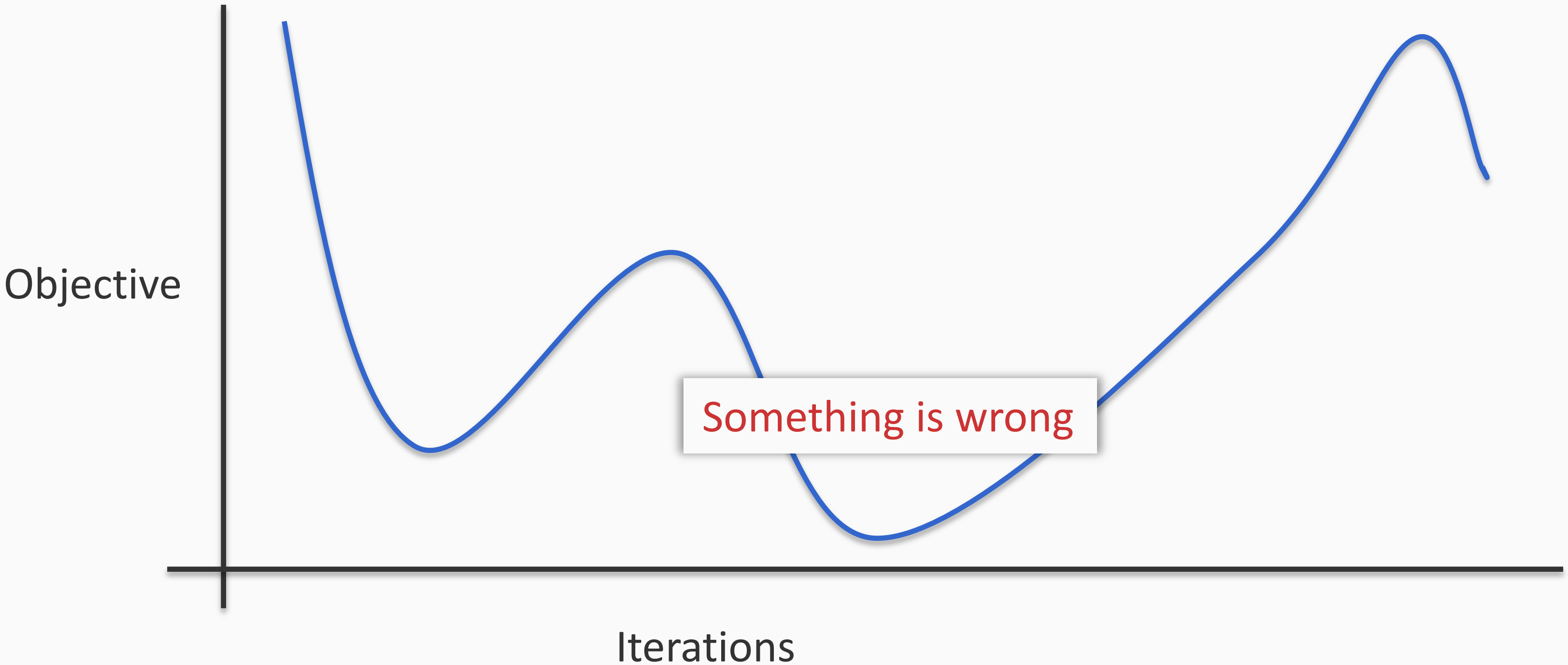
Does your learning algorithm converge?

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



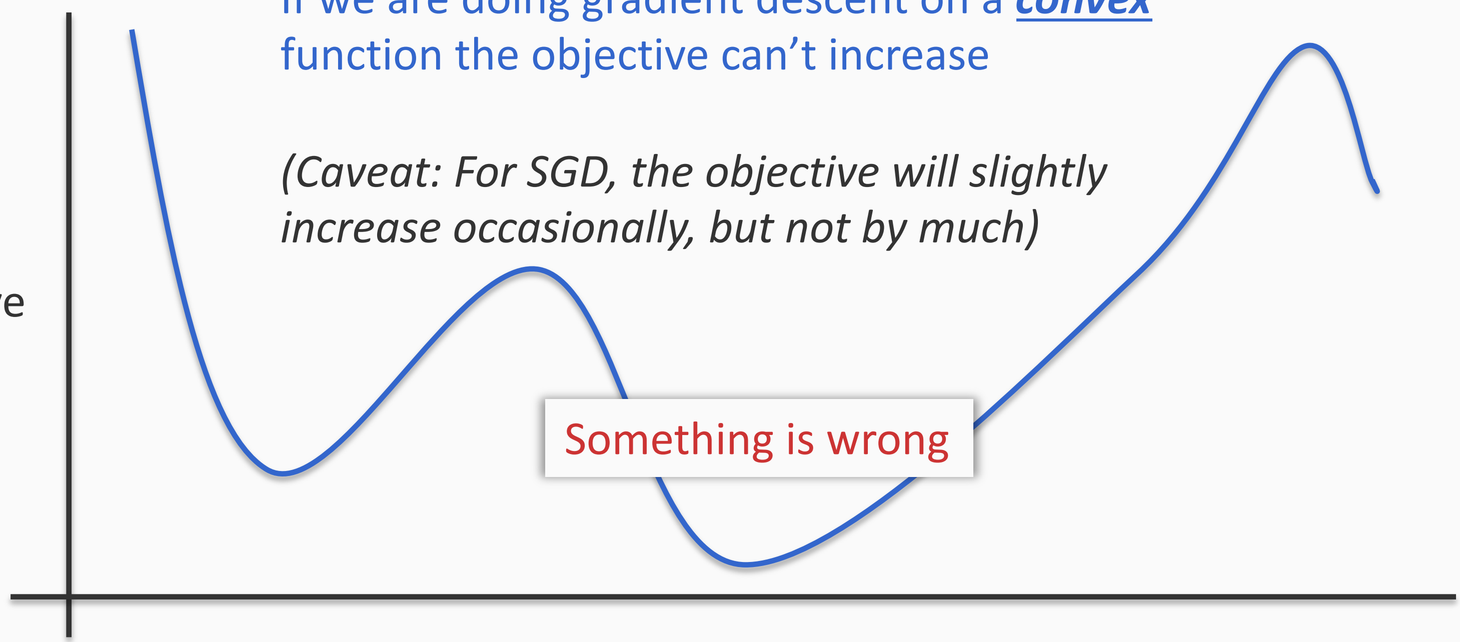
Does your learning algorithm converge?

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

Helps to debug

If we are doing gradient descent on a ***convex*** function the objective can't increase

(Caveat: For SGD, the objective will slightly increase occasionally, but not by much)



Different ways to improve your model

More training data Helps with overfitting

Features

1. Use more features Helps with under-fitting
2. Use fewer features Helps with over-fitting
3. Use other features Could help with over-fitting and under-fitting

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization Could help with over-fitting and under-fitting

Different ways to improve your model

More training data

Helps with overfitting

Features

1. Use more features — Helps with under-fitting
2. Use fewer features — Helps with over-fitting
3. Use other features — Could help with over-fitting and under-fitting

Better training

1. Run for more iterations
2. Use a different algorithm — Track the objective for convergence
3. Use a different classifier
4. Play with regularization — Could help with over-fitting and under-fitting

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

4. Are you measuring the right thing?

What to measure

- **Accuracy** of prediction is the most common measurement for classifiers
  - **Mean squared error** is the most common measurement for regression
- But if your data set is unbalanced, accuracy may be misleading
  - 1000 positive examples, 1 negative example
  - A classifier that always predicts positive will get 99.9% accuracy. Has it really learned anything?
- Unbalanced labels → measure label specific precision, recall and F-score
  - **Precision** for a label: Among examples that are predicted with label, what fraction are correct
  - **Recall** for a label: Among the examples with given ground truth label, what fraction are correct
  - **F-score**: Harmonic mean of precision and recall

ML and the world

- Diagnostics of your learning algorithm
- Error analysis
- Injecting machine learning into *Your Favorite Task*
- Making machine learning matter

Machine Learning in this class

ML
code

Machine Learning in context

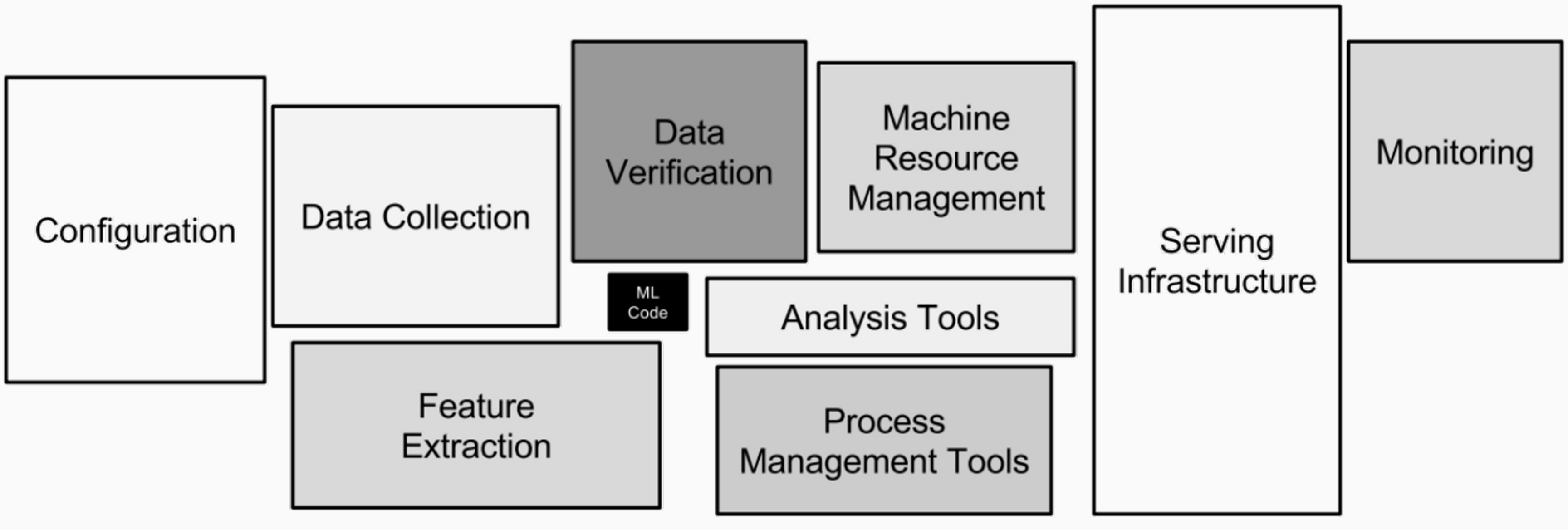


Figure from [Sculley, et al 2015]

ML and system building

Several position papers about how ML fits in the context of large software systems

Machine Learning:
The High-Interest Credit Card of Technical Debt

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov,
Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young
{dsculley,gholt,dgg,edavydov}@google.com
{toddphillips,ebner,vchaudhary,mwyoung}@google.com
Google, Inc

Hidden Technical Debt in Machine Learning Systems

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips
{dsculley,gholt,dgg,edavydov,toddphillips}@google.com
Google, Inc.

Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, Dan Dennison
{ebner,vchaudhary,mwyoung,jfcrespo,dennison}@google.com
Google, Inc.

Error Analysis

Generally machine learning plays a small (but important) role in a larger application

- Assumptions while gathering data
- Pre-processing
- Features (possibly by other ML based methods)
- Data transformations
- User interface

How much do each of these contribute to the error?

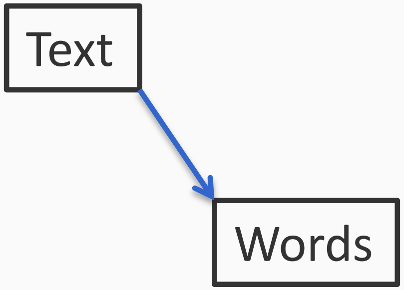
Error analysis tries to explain why a system is not performing perfectly

Example: A text processing pipeline

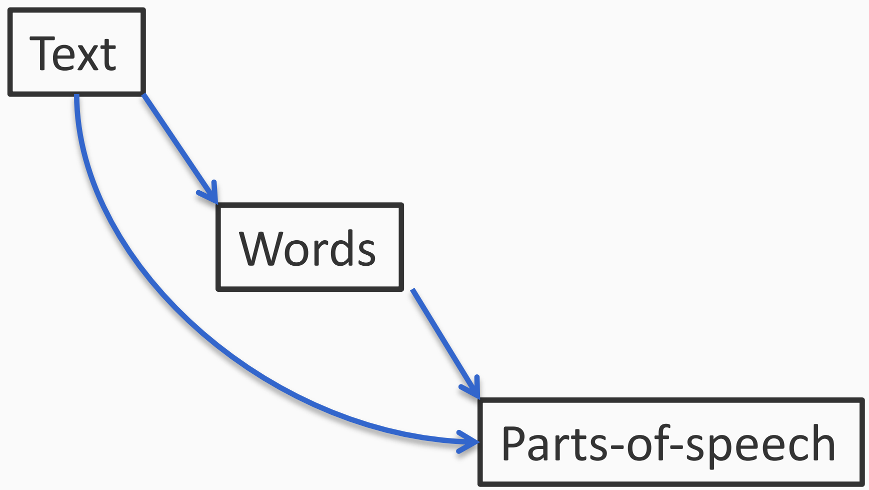
Example: A text processing pipeline

Text

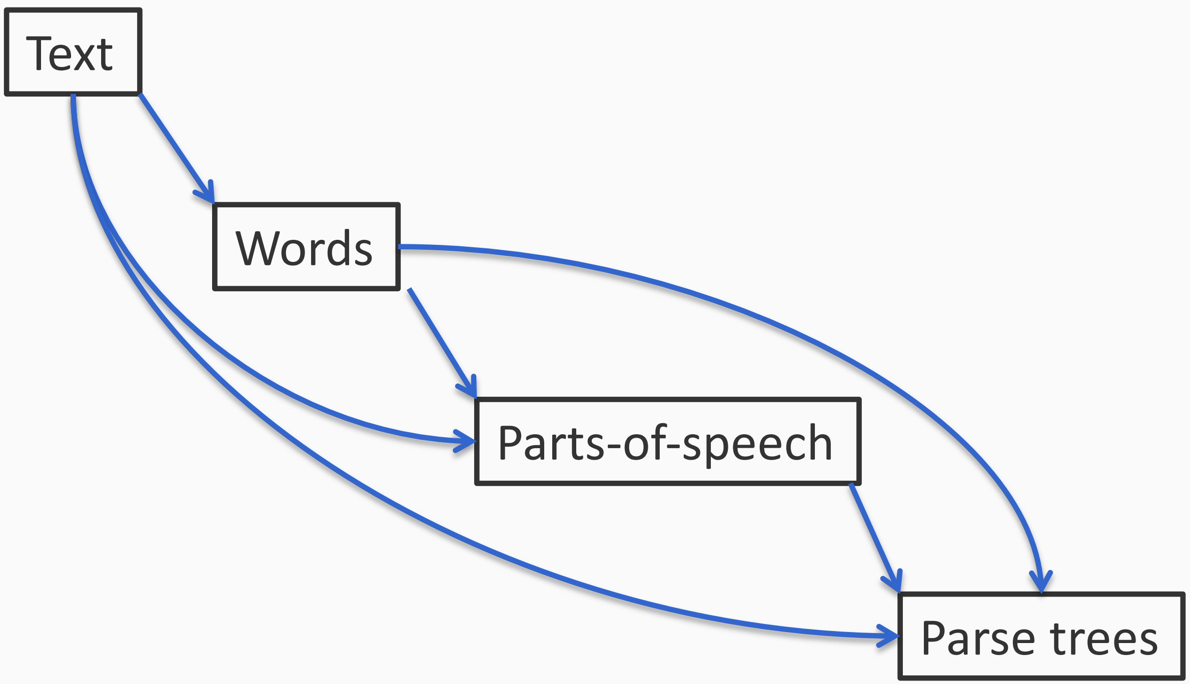
Example: A text processing pipeline



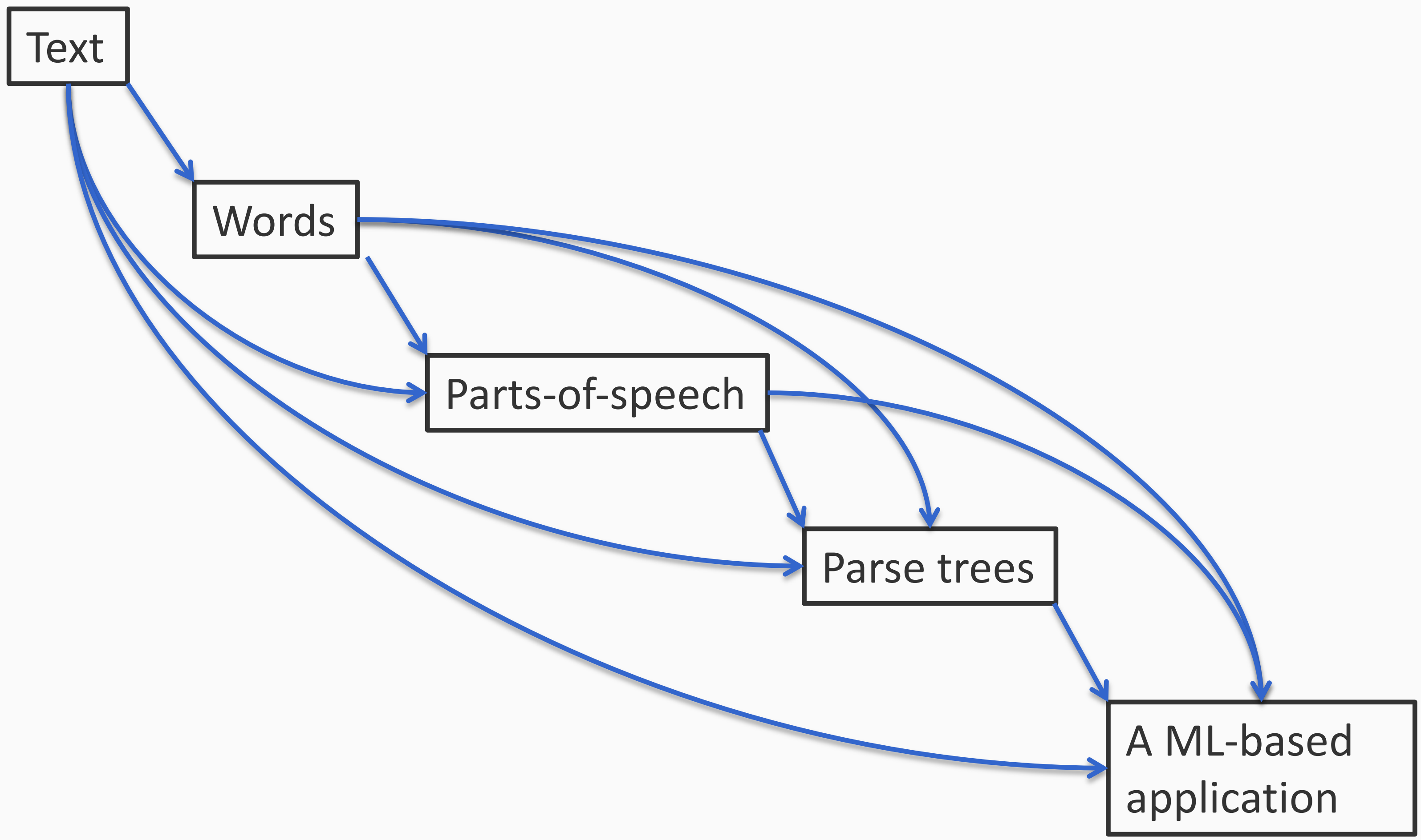
Example: A text processing pipeline



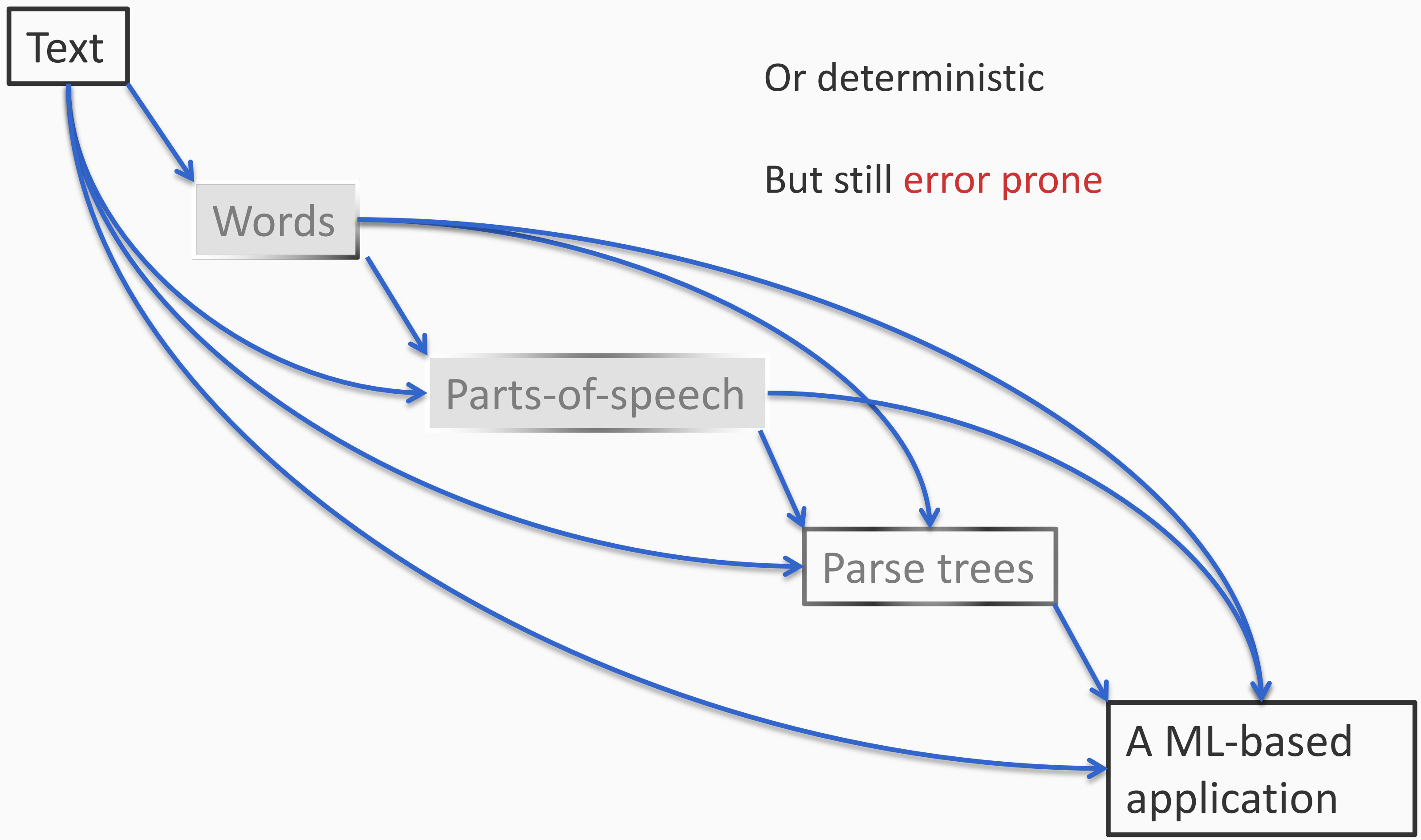
Example: A text processing pipeline



Example: A text processing pipeline



Example: A text processing pipeline



Each of these could be ML driven

Or deterministic

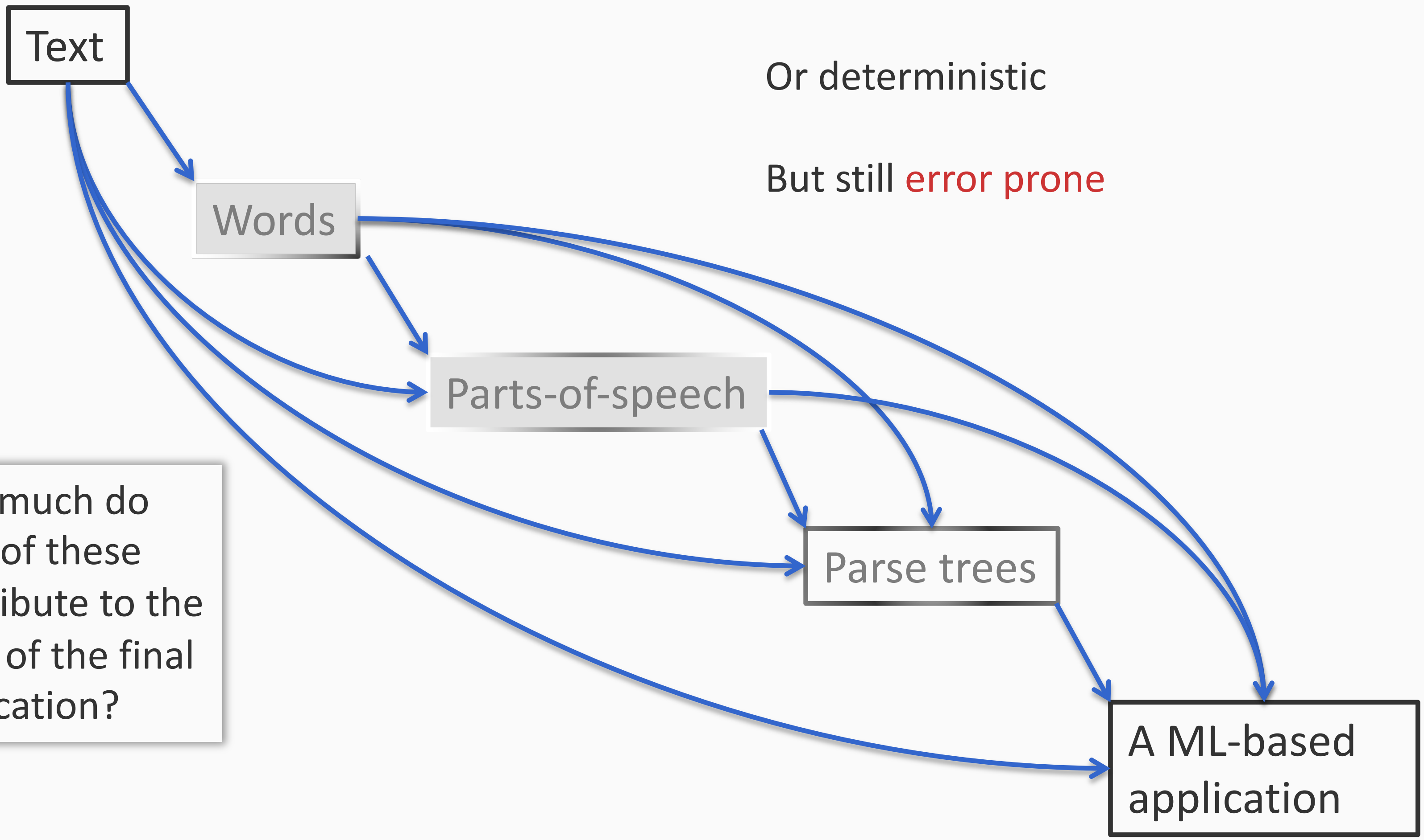
But still error prone

Example: A text processing pipeline

Each of these could be ML driven

Or deterministic

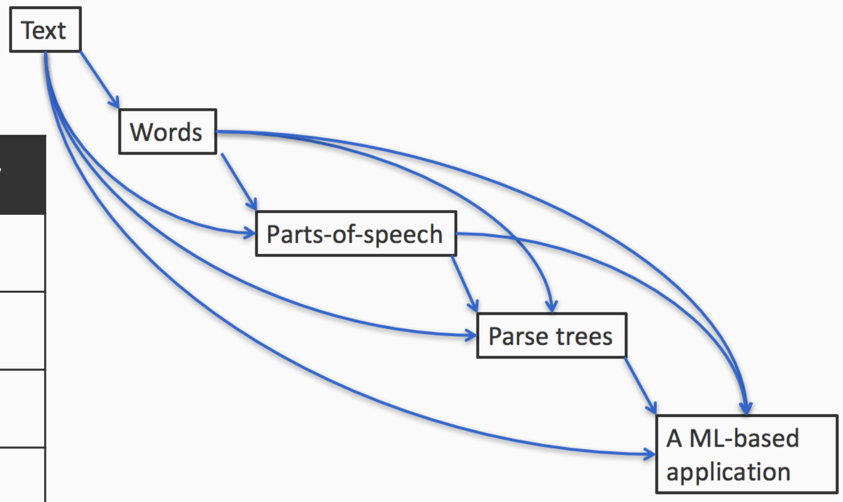
But still error prone



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 components and see how much the accuracy of the final system changes

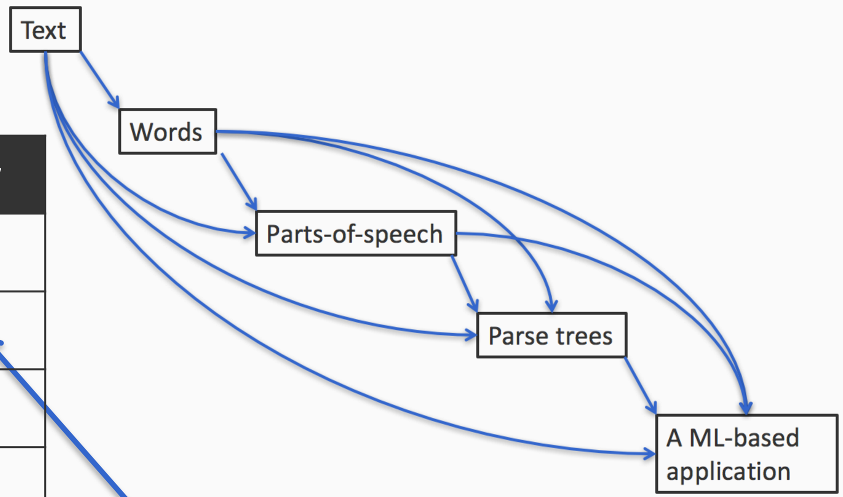


System	Accuracy
End-to-end predicted	55%
With ground truth words	60%
+ ground truth parts-of-speech	84 %
+ ground truth parse trees	89 %
+ ground truth final output	100 %

Tracking errors in a complex system

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

System	Accuracy
End-to-end predicted	55%
With ground truth words	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

Ablative study

Explaining difference between the performance between a strong model and a much weaker one (a baseline)

Usually seen with features or neural network architectures

- 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 are progressively simpler to see what gives the highest boost

It is not enough to have a classifier that works; it is useful to know why it works.

Helps interpret predictions, diagnose errors and can provide an audit trail

How many people in this picture?

How many people in this picture?

Three heads

Three hands

Four legs

How many people in this picture?

Three heads

Three hands

Four legs

And yet five people!

How many people in this picture?

Three heads

Three hands

Four legs

And yet five people!

Learned systems are not used in isolation, but used in conjunction with each other

And in the context of a larger application

ML and the world

- Diagnostics of your learning algorithm
- Error analysis
- Injecting machine learning into *Your Favorite Task*
- Making machine learning matter

Classifying fish

Say you want to build a classifier that identifies whether a real physical fish is salmon or tuna

How do you go about this?

Classifying fish

Say you want to build a classifier that identifies whether a real physical fish is salmon or tuna

How do you go about this?

The slow approach

1. Carefully identify features, get the best data, the software architecture, maybe design a new learning algorithm
2. Implement it and hope it works

Advantage: Perhaps a better approach, maybe even a new learning algorithm. Research.

Classifying fish

Say you want to build a classifier that identifies whether a real physical fish is salmon or tuna

How do you go about this?

The slow approach

1. Carefully identify features, get the best data, the software architecture, maybe design a new learning algorithm
2. Implement it and hope it works

Advantage: Perhaps a better approach, maybe even a new learning algorithm. Research.

The hacker's approach

1. First implement something
2. Use diagnostics to iteratively make it better

Advantage: Faster release, will have a solution for your problem quicker

Classifying fish

Say you want to build a classifier that identifies whether a real physical fish is salmon or tuna

How do you go about this?

The slow approach

1. Carefully identify features, get the best dat arc de algorithm
2. Implement it and hope it works

Advantage: Perhaps a better approach, maybe even a new learning algorithm. Research.

The hacker's approach

1. First implement something to e it better

Advantage: Faster release, will have a solution for your problem quicker

Be wary of premature optimization

Be equally wary of prematurely committing to a bad path

What to watch out for

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

- Beware of contamination: Ensure that your training data is not contaminated with the test set
  - Learning = generalization to new examples
  - Do not see your test set either. You may inadvertently contaminate the model
  - Beware of contaminating your features with the label!
  - (Be suspicious of perfect predictors)

What to watch out for

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

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

- A theoretical guarantee may only be theoretical
  - May make invalid assumptions (eg: if the data is separable)
  - May only be legitimate with infinite data (eg: estimating probabilities)
  - Experiments on real data are equally important

Big data is not enough

But more data is always better

- Cleaner data is even better

Remember that learning is impossible without some bias that simplifies/guides the search over hypotheses

- Otherwise, no generalization

Learning requires knowledge to guide the learner

- *Machine learning is not a magic wand*

What knowledge?

- Which model is the right one for this task?
  – Linear models, decision trees, deep neural networks, etc
- Which learning algorithm?
  – Does the data violate any crucial assumptions that were used to define the learning algorithm or the model?
  – Does that matter?
- Feature engineering is crucial
- 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

ML and the world

- Diagnostics of your learning algorithm
- Error analysis
- Injecting machine learning into *Your Favorite Task*
- Making machine learning matter

Making machine learning matter

Machine Learning that Matters [ICML 2012]

Kiri L. Wagstaff KIRI.L.WAGSTAFF@JPL.NASA.GOV
Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109 USA

Challenges to the greater ML community

1. A law passed or legal decision made that relies on the result of an ML analysis
2. $100M saved through improved decision making provided by an ML system
3. A conflict between nations averted through high quality translation provided by an ML system
4. A 50% reduction in cybersecurity break-ins through ML defenses
5. A human life saved through a diagnosis or intervention recommended by an ML system
6. Improvement of 10% in one country's Human Development Index attributable to an ML system

Data-driven decision making: Increasingly prevalent

Algorithms are no longer just about showing proof-of-concept learning

Some broader concerns about algorithmic decision-making emerge:

- Do classifiers exhibit biases? What about datasets?
- How do we ensure that such systems are transparent in their decision-making?
- When can you believe your model? Should you leave decision making to it?
- What if statistical models are used in ethically dubious ways?

These refer to auxiliary criteria that need not directly tied to the loss that we minimize

Biased classifiers

All these are real examples

What if classifiers are used to decide…

- … how long someone should be sentenced for a crime?
- … or whether someone's loan application should be approved?
- … or whether someone should be fired?

- How can we ensure that the classifiers do not exhibit illegal or unethical behavior?
  - i.e. the classifiers are fair?
- Can we guarantee that a machine learning based system does not do harm?
  - In terms of its representation of groups of people, or in how it makes decisions about them
- How do we design algorithms and evaluation frameworks which avoid discrimination?
- What if the data itself (either knowingly or unknowingly) is unfair?

The right to explanations

Imagine you have a job and a classifier decides to fire you.

- Maybe because it made an error on this instance (ie. you!)

Questions:

- How do we develop statistical methods that not only make a prediction, but are also transparent in their decision making process?

- How do we develop algorithms that can explain their decisions?

Are current legal systems robust?

Perhaps there is room to rewrite/update laws to account for machine learning

- What if evidence is faked by a sampling from a generative model?
- Who is to blame if a ML-based automatic car is involved in an accident?
- What if an ML-based autonomous weapon decides to kill someone without any human intervention?

Fairness, Accountability and Transparency

- We need to ensure that
  - Our algorithms are ***Fair***
  - Our algorithms can be held ***Accountable***
  - Our algorithms exhibit ***Transparency***

- These are all difficult to formalize
  - But still important questions

- And it is important to keep these concerns when we…
  - Define a task
  - Collect data
  - Define evaluations
  - Design features, models

  … really at every step along the way

https://facctconference.org

A retrospective look at the course

Learning = generalization

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Tom Mitchell (1999)

We saw different “models”

Or: what kind of a function should a learner learn

– Linear classifiers

– Decision trees

– Non-linear classifiers, feature transformations, neural networks

– Ensembles of classifiers

Different learning protocols

- Supervised learning
  - A *teacher* supplies a collection of examples with labels
  - The *learner* has to learn to label new examples using this data

- We did not see
  - Unsupervised learning
    - No *teacher*, *learner* has only unlabeled examples
    - Data mining
  - Semi-supervised learning
    - *Learner* has access to both labeled and unlabeled examples

Learning algorithms

- Online algorithms: Learner can access only one labeled at a time
  - Perceptron
- Batch algorithms: Learner can access to the entire dataset
  - Support vector machines, logistic regression
  - Decision trees, random forests
  - Nearest neighbors
  - Boosting
  - Neural networks

Representing data

What is the best way to represent data for a particular task?

- Features
- Dimensionality reduction (we didn't cover this, but do look at the material if you are interested)

The theory of machine learning

Mathematically defining learning

- Online learning
- Probably Approximately Correct (PAC) Learning
- Bayesian learning

Representation, optimization, evaluation

The three components of learning algorithms.

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		

Table from [Domingos, 2012]

Machine learning is *too* easy!

- Remarkably diverse collection of ideas
- Yet, in practice many of these approaches work roughly equally well
  - Eg: SVM vs logistic regression vs averaged perceptron

What we did not see

Machine learning is a large and growing area of scientific study

We did not cover

- Kernel methods
- Specific neural network architectures
- Unsupervised learning, clustering
- Hidden Markov models
- Multiclass support vector machines
- Topic models
- Structured models
- ….

But we saw the foundations of *how to think about machine learning*

What we did not see

Machine learning is a large and growing area of scientific study

We did not cover

- Kernel methods
- Sp
- U
- H
- M
- To
- St
- ….

to

e

Several classes that can follow (or are related to) this course:

- Data Mining
- Artificial Intelligence (various sub-topics)
- Theory of Machine Learning
- Various applications (NLP, vision, robotics, …)
- Data visualization
- Specializations: Deep learning, Structured Prediction, etc

This course

Focus on the underlying concepts and algorithmic ideas in the field of machine learning

Not about

- Using a specific machine learning tool
- Any single learning paradigm

What we saw

1. A broad theoretical and practical understanding of machine learning paradigms and algorithms

2. Ability to implement learning algorithms

3. Identify where machine learning can be applied and make the most appropriate decisions (about algorithms, models, supervision, etc)