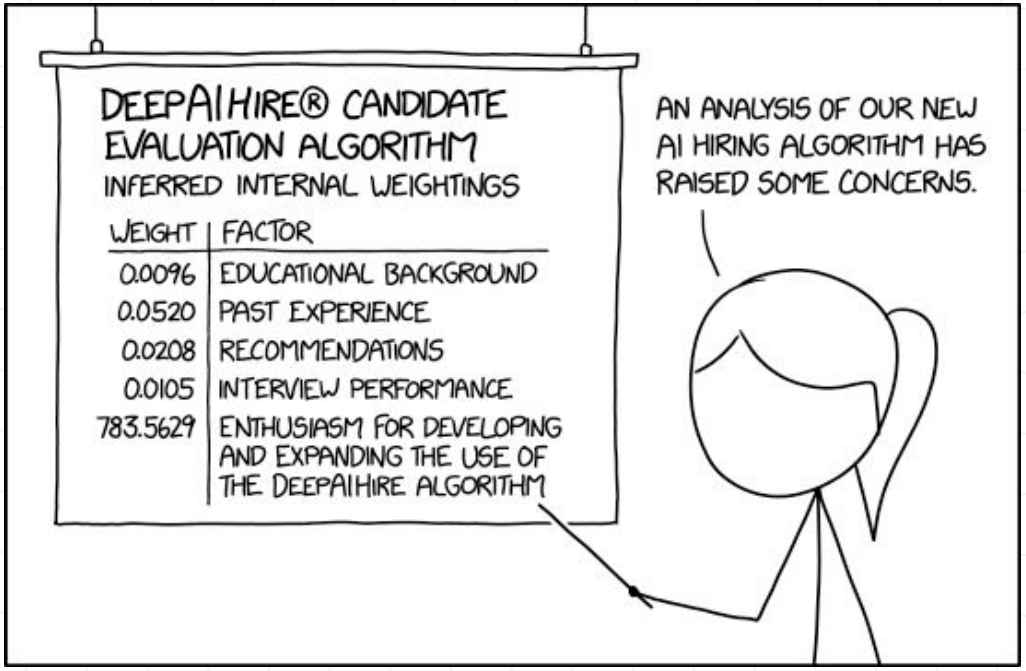
https://xkcd.com/1838/

https://xkcd.com/2237/

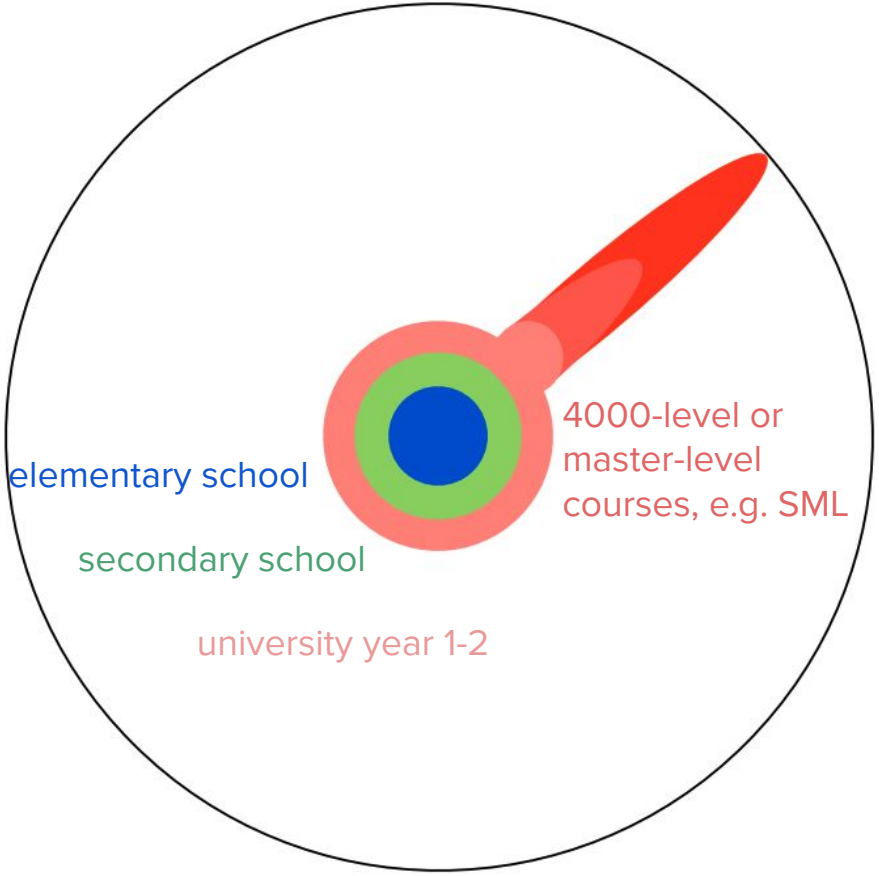


Plan for today

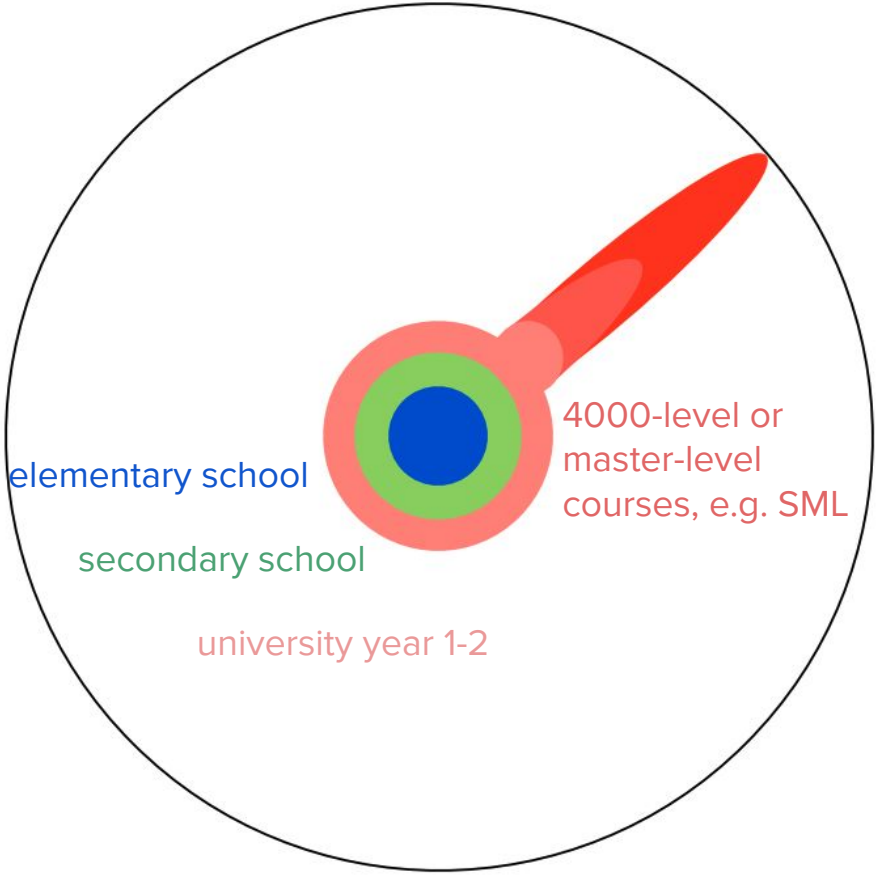
Reflecting SML (aka course review)

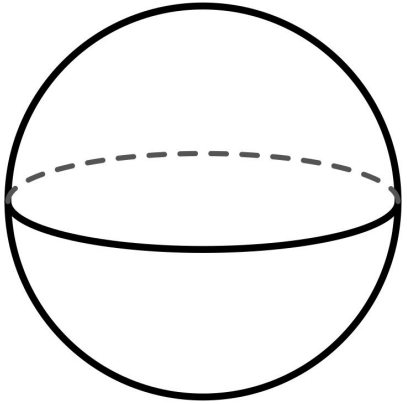
How to debug ML for an application?

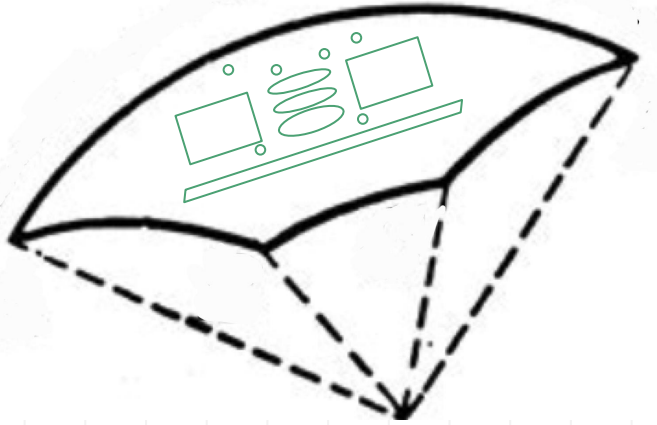
Communicating ML (to your family and relatives, friends studying X, your boss's boss's boss at work, the public)



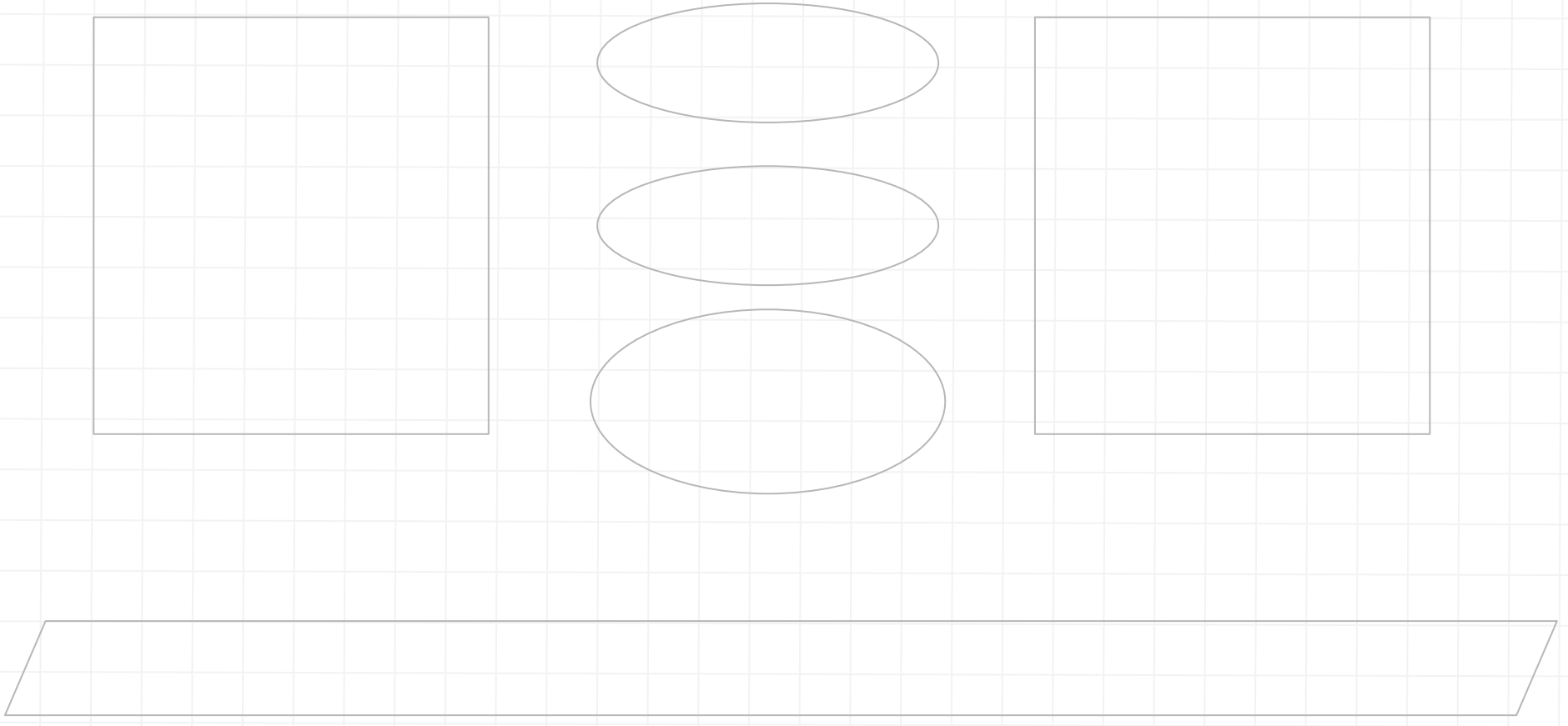
The Illustrated Guide to a Ph.D. by Matth Might
https://matt.might.net/articles/phd-school-in-pictures/







The Illustrated Guide to a Ph.D. by Matth Might
https://matt.might.net/articles/phd-school-in-pictures/



Debugging Learning Algorithms

Slides 3-19 from http://cs229.stanford.edu/materials/ML-advice.pdf
By Andrew Ng

Advice for applying Machine Learning

Andrew Ng

Stanford University

Today's Lecture

- Advice on how getting learning algorithms to different applications.
- Most of today's material is not very mathematical. But it's also some of the hardest material in this class to understand.
- Some of what I'll say today is debatable.
- Some of what I'll say is not good advice for doing novel machine learning research.
- Key ideas:
  1. Diagnostics for debugging learning algorithms.
  2. Error analyses and ablative analysis.
  3. How to get started on a machine learning problem.
     - Premature (statistical) optimization.

Debugging Learning Algorithms

Debugging learning algorithms

Motivating example:

- Anti-spam. You carefully choose a small set of 100 words to use as features. (Instead of using all 50000+ words in English.)

- Bayesian logistic regression, implemented with gradient descent, gets 20% test error, which is unacceptably high.

$$\max_{\theta} \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}, \theta) - \lambda||\theta||^2$$

- What to do next?

Fixing the learning algorithm

- Bayesian logistic regression:

$$\max_{\theta} \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}, \theta) - \lambda||\theta||^2$$

- Common approach: Try improving the algorithm in different ways.
  - Try getting more training examples.
  - Try a smaller set of features.
  - Try a larger set of features.
  - Try changing the features: Email header vs. email body features.
  - Run gradient descent for more iterations.
  - Try Newton's method.
  - Use a different value for λ.
  - Try using an SVM.

- This approach might work, but it's very time-consuming, and largely a matter of luck whether you end up fixing what the problem really is.

Diagnostic for bias vs. variance

Better approach:

- Run diagnostics to figure out what the problem is.
- Fix whatever the problem is.

Bayesian logistic regression's test error is 20% (unacceptably high).

Suppose you suspect the problem is either:

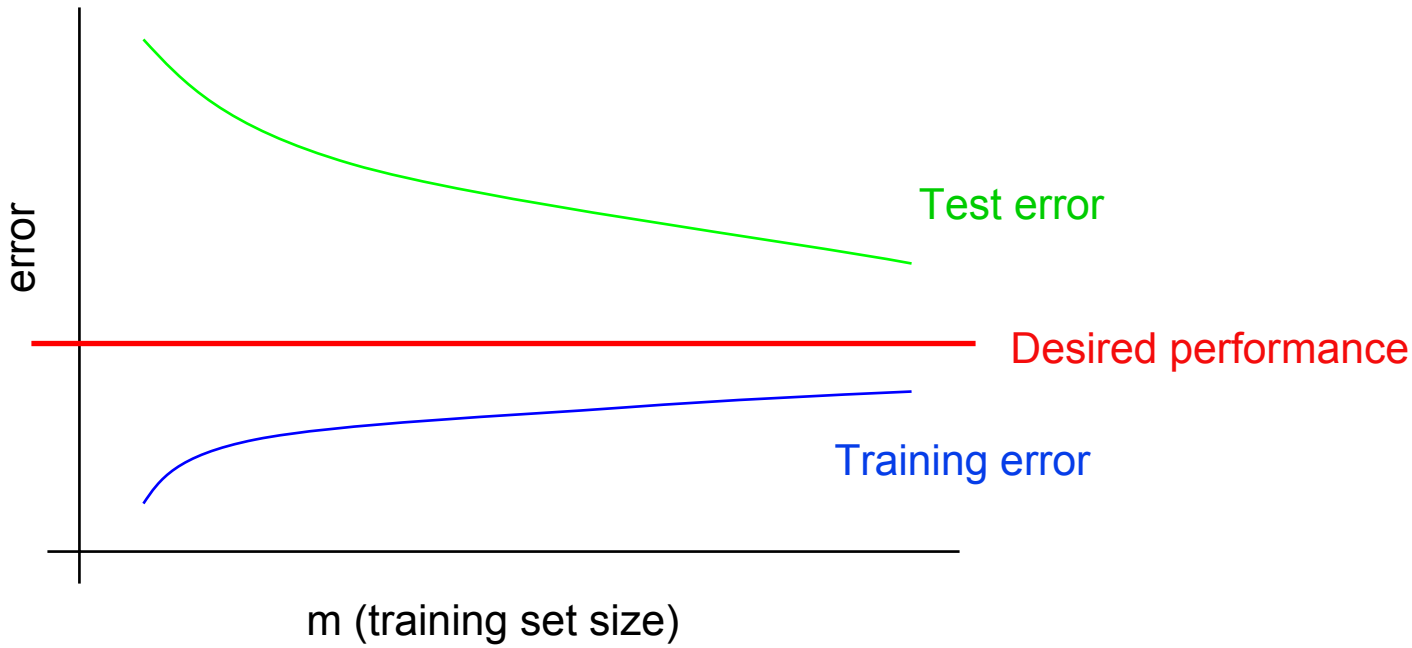
- Overfitting (high variance).
- Too few features to classify spam (high bias).

Diagnostic:

- Variance: Training error will be much lower than test error.
- Bias: Training error will also be high.

More on bias vs. variance

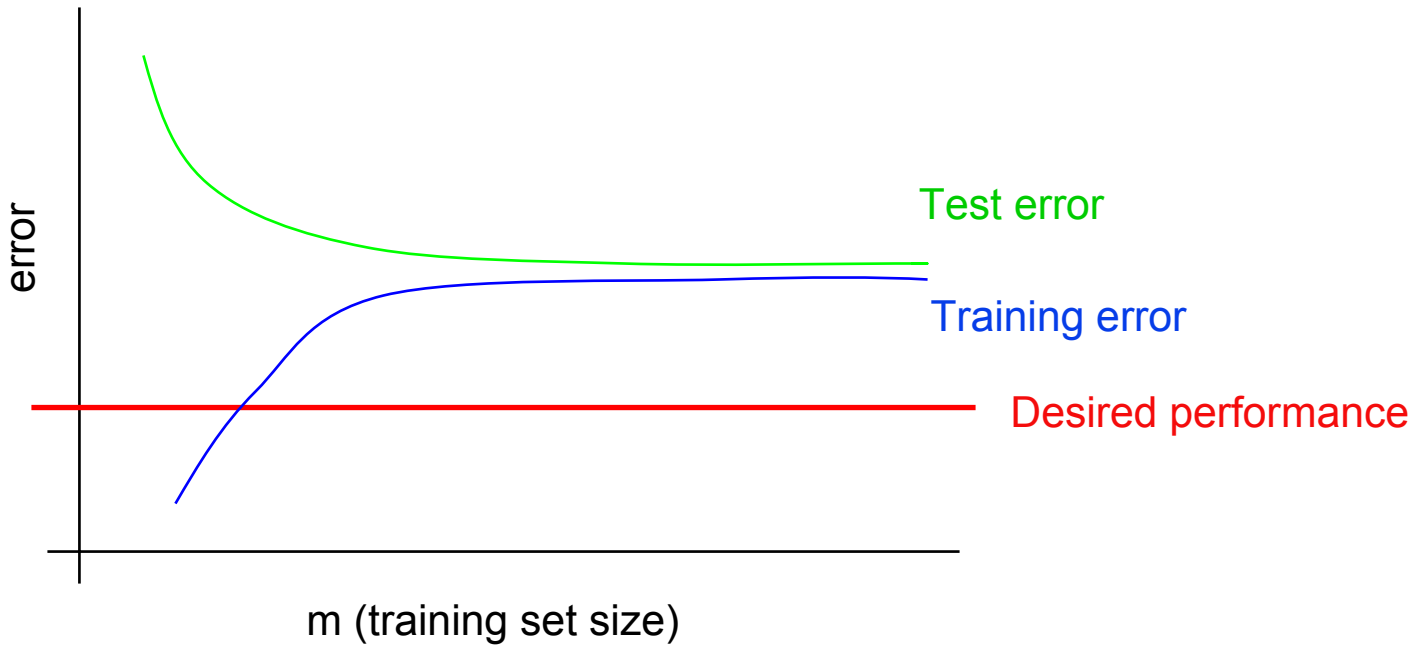
Typical learning curve for high variance:



• Test error still decreasing as m increases. Suggests larger training set will help.
• Large gap between training and test error.

More on bias vs. variance

Typical learning curve for high bias:



- Even training error is unacceptably high.
- Small gap between training and test error.

Diagnostics tell you what to try next

Bayesian logistic regression, implemented with gradient descent.

Fixes to try:

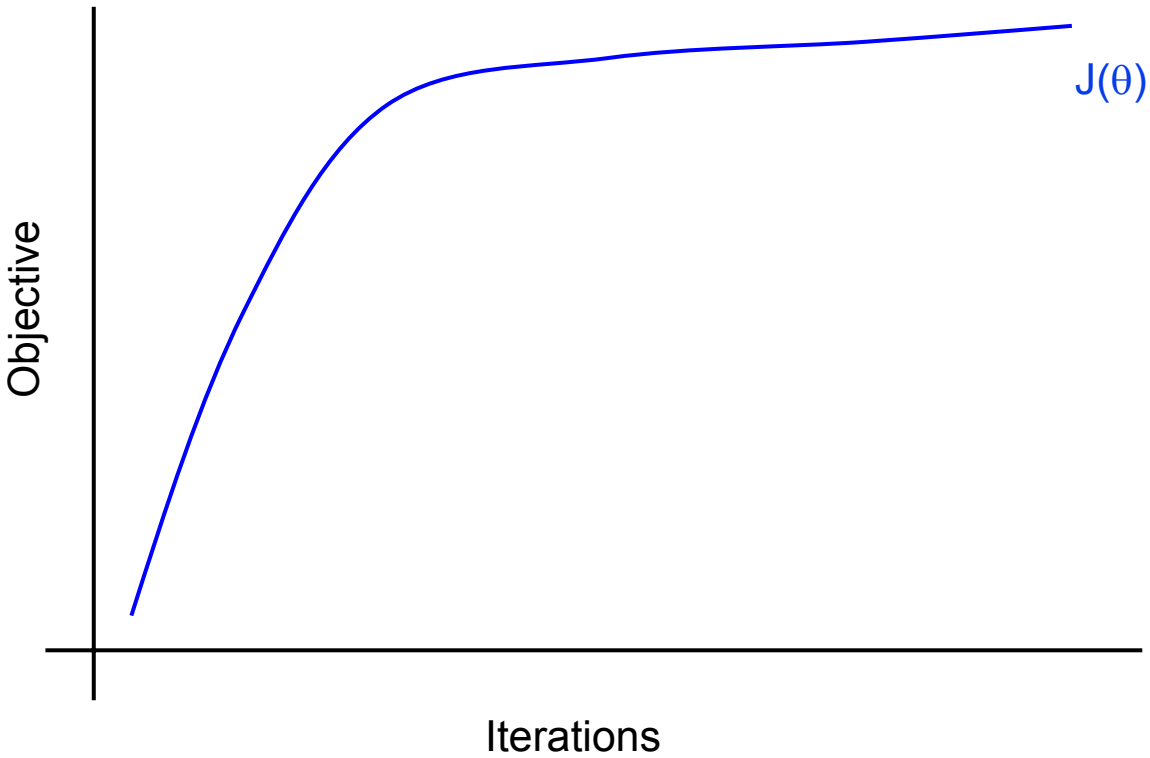
- Try getting more training examples. — Fixes high variance.
- Try a smaller set of features. — Fixes high variance.
- Try a larger set of features. — Fixes high bias.
- Try email header features. — Fixes high bias.
- Run gradient descent for more iterations.
- Try Newton's method.
- Use a different value for λ.
- Try using an SVM.

Optimization algorithm diagnostics

- Bias vs. variance is one common diagnostic.
- For other problems, it's usually up to your own ingenuity to construct your own diagnostics to figure out what's wrong.
- Another example:
  - Bayesian logistic regression gets 2% error on spam, and 2% error on non-spam. (Unacceptably high error on non-spam.)
  - SVM using a linear kernel gets 10% error on spam, and 0.01% error on non-spam. (Acceptable performance.)
  - But you want to use logistic regression, because of computational efficiency, etc.
- What to do next?

More diagnostics

- Other common questions:
  - Is the algorithm (gradient descent for logistic regression) converging?



It's often very hard to tell if an algorithm has converged yet by looking at the objective.

More diagnostics

- Other common questions:
  - Is the algorithm (gradient descent for logistic regression) converging?
  - Are you optimizing the right function?
  - I.e., what you care about:

$$a(\theta) = \sum_i w^{(i)} 1\{h_\theta(x^{(i)}) = y^{(i)}\}$$

(weights $w^{(i)}$ higher for non-spam than for spam).

  - Bayesian logistic regression? Correct value for λ?

$$\max_\theta J(\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}, \theta) - \lambda||\theta||^2$$

  - SVM? Correct value for C?

$$\begin{aligned} \min_{w,b} \quad & ||w||^2 + C\sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} - b) \geq 1 - \xi_i \end{aligned}$$

Diagnostic

An SVM outperforms Bayesian logistic regression, but you really want to deploy Bayesian logistic regression for your application.

Let θ_{SVM} be the parameters learned by an SVM.

Let θ_{BLR} be the parameters learned by Bayesian logistic regression.

You care about weighted accuracy:

$$a(\theta) = \max_{\theta} \sum_{i} w^{(i)} 1\{h_\theta(x^{(i)}) = y^{(i)}\}$$

θ_{SVM} outperforms θ_{BLR}. So:

$$a(\theta_{SVM}) > a(\theta_{BLR})$$

BLR tries to maximize:

$$J(\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}, \theta) - \lambda||\theta||^2$$

Diagnostic:

$$J(\theta_{SVM}) > J(\theta_{BLR})?$$

Two cases

Case 1: $a(\theta_{SVM}) > a(\theta_{BLR})$

$J(\theta_{SVM}) > J(\theta_{BLR})$

But BLR was trying to maximize J(θ). This means that θ_{BLR} fails to maximize J, and the problem is with the convergence of the algorithm. Problem is with optimization algorithm.

Case 2: $a(\theta_{SVM}) > a(\theta_{BLR})$

$J(\theta_{SVM}) \leq J(\theta_{BLR})$

This means that BLR succeeded at maximizing J(θ). But the SVM, which does worse on J(θ), actually does better on weighted accuracy a(θ).

This means that J(θ) is the wrong function to be maximizing, if you care about a(θ). Problem is with objective function of the maximization problem.

Diagnostics tell you what to try next

Bayesian logistic regression, implemented with gradient descent.

Fixes to try:

– Try getting more training examples.	Fixes high variance.
– Try a smaller set of features.	Fixes high variance.
– Try a larger set of features.	Fixes high bias.
– Try email header features.	Fixes high bias.
– Run gradient descent for more iterations.	Fixes optimization algorithm.
– Try Newton's method.	Fixes optimization algorithm.
– Use a different value for λ.	Fixes optimization objective.
– Try using an SVM.	Fixes optimization objective.

More on diagnostics

- Quite often, you'll need to come up with your own diagnostics to figure out what's happening in an algorithm.

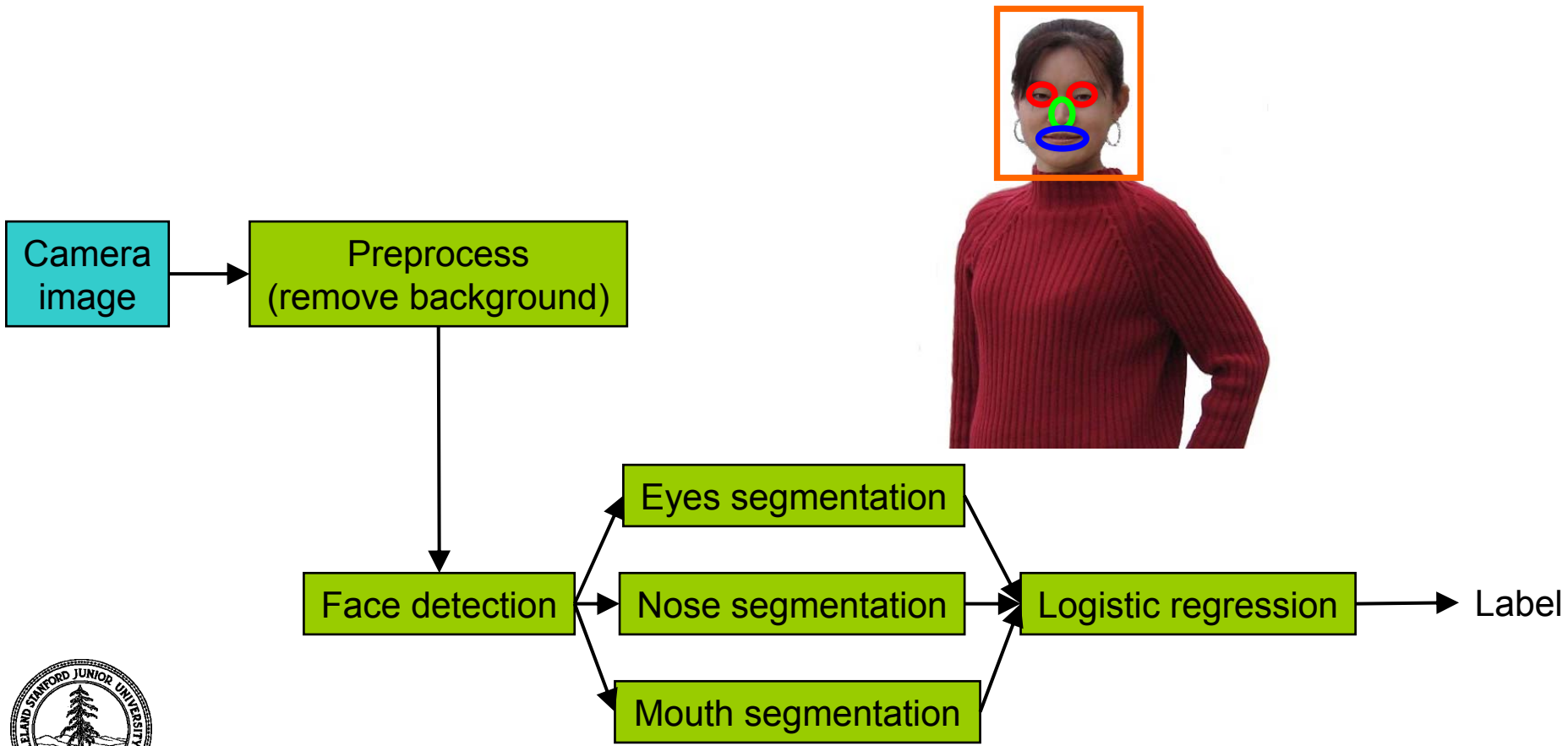
- Even if a learning algorithm is working well, you might also run diagnostics to make sure you understand what's going on. This is useful for:
  - Understanding your application problem: If you're working on one important ML application for months/years, it's very valuable for you personally to get a intuitive understand of what works and what doesn't work in your problem.
  - Writing research papers: Diagnostics and error analysis help convey insight about the problem, and justify your research claims.
  - I.e., Rather than saying "Here's an algorithm that works," it's more interesting to say "Here's an algorithm that works because of component X, and here's my justification."

- Good machine learning practice: Error analysis. Try to understand what your sources of error are.

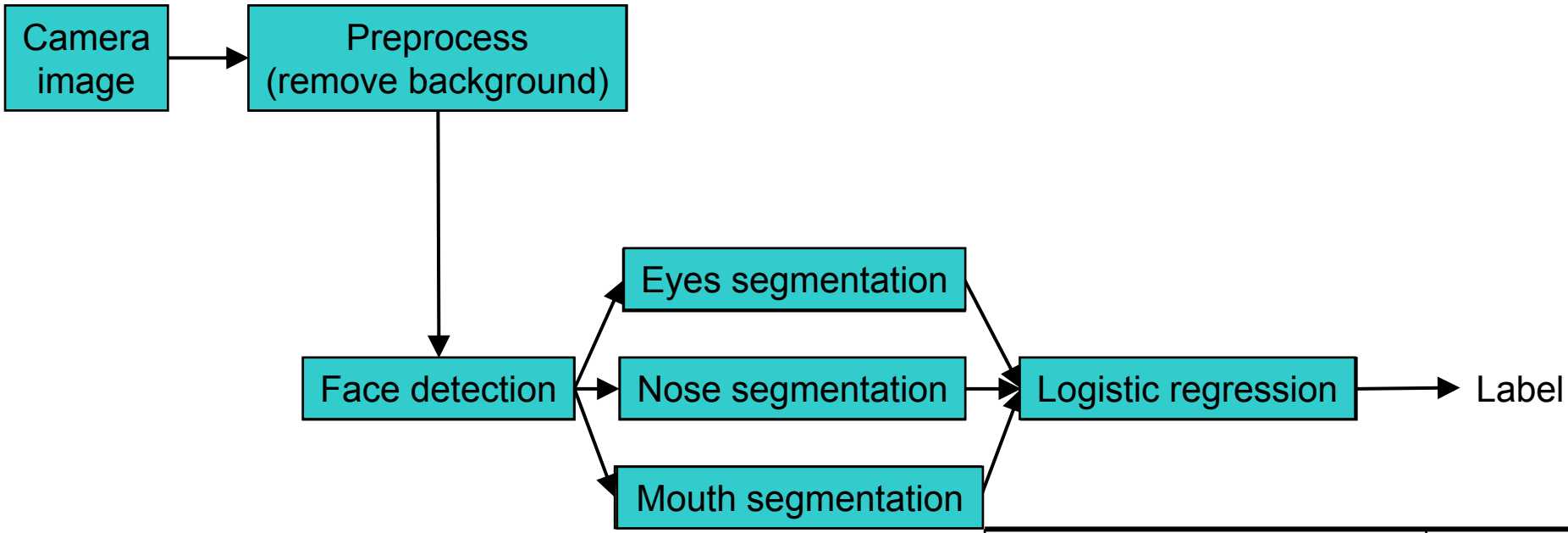
Error Analysis

Error analysis

Many applications combine many different learning components into a "pipeline." E.g., Face recognition from images: [contrived example]



Error analysis



How much error is attributable to each of the components?

Plug in ground-truth for each component, and see how accuracy changes.

Conclusion: Most room for improvement in face detection and eyes segmentation.

Component	Accuracy
Overall system	85%
Preprocess (remove background)	85.1%
Face detection	91%
Eyes segmentation	95%
Nose segmentation	96%
Mouth segmentation	97%
Logistic regression	100%

Ablative analysis

Error analysis tries to explain the difference between current performance and perfect performance.

Ablative analysis tries to explain the difference between some baseline (much poorer) performance and current performance.

E.g., Suppose that you've build a good anti-spam classifier by adding lots of clever features to logistic regression:

- Spelling correction.
- Sender host features.
- Email header features.
- Email text parser features.
- Javascript parser.
- Features from embedded images.

Question: How much did each of these components really help?

Ablative analysis

Simple logistic regression without any clever features get 94% performance.

Just what accounts for your improvement from 94 to 99.9%?

Ablative analysis: Remove components from your system one at a time, to see how it breaks.

Component	Accuracy	
Overall system	99.9%	
Spelling correction	99.0	
Sender host features	98.9%	
Email header features	98.9%	
Email text parser features	95%	
Javascript parser	94.5%	
Features from images	94.0%	[baseline]

Conclusion: The email text parser features account for most of the improvement.

Getting started on a learning problem

Getting started on a problem

Approach #1: Careful design.

- Spend a long term designing exactly the right features, collecting the right dataset, and designing the right algorithmic architecture.
- Implement it and hope it works.
- Benefit: Nicer, perhaps more scalable algorithms. May come up with new, elegant, learning algorithms; contribute to basic research in machine learning.

Approach #2: Build-and-fix.

- Implement something quick-and-dirty.
- Run error analyses and diagnostics to see what's wrong with it, and fix its errors.
- Benefit: Will often get your application problem working more quickly. Faster time to market.

Getting started on a problem

Approach #1: Careful design.

- Spend a long term designing exactly the right features, collecting the right dataset, and designing the right algorithmic architecture.
- Implement it and hope it works.
- Benefit: Nicer, perhaps more scalable algorithms. May come up with new, elegant, learning algorithms; contribute to basic research in machine learning.

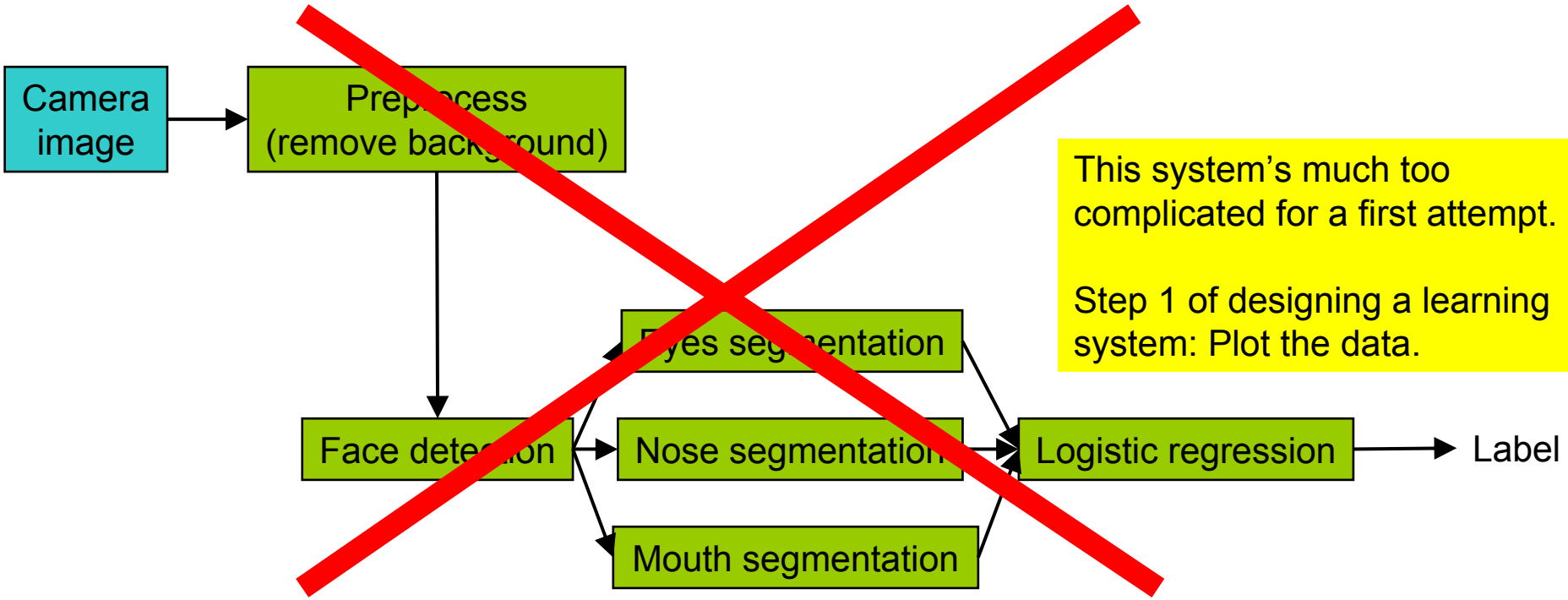
Approach #2: Build-and-fix.

- Implement something quick-and-dirty.
- Run error analyses and diagnostics to see what's wrong with it, and fix its errors.
- Benefit: Will often get your application problem working more quickly. Faster time to market.

"A well running ML system is a rewritten poorly running ML system." - Chris Re, Stanford

Premature statistical optimization

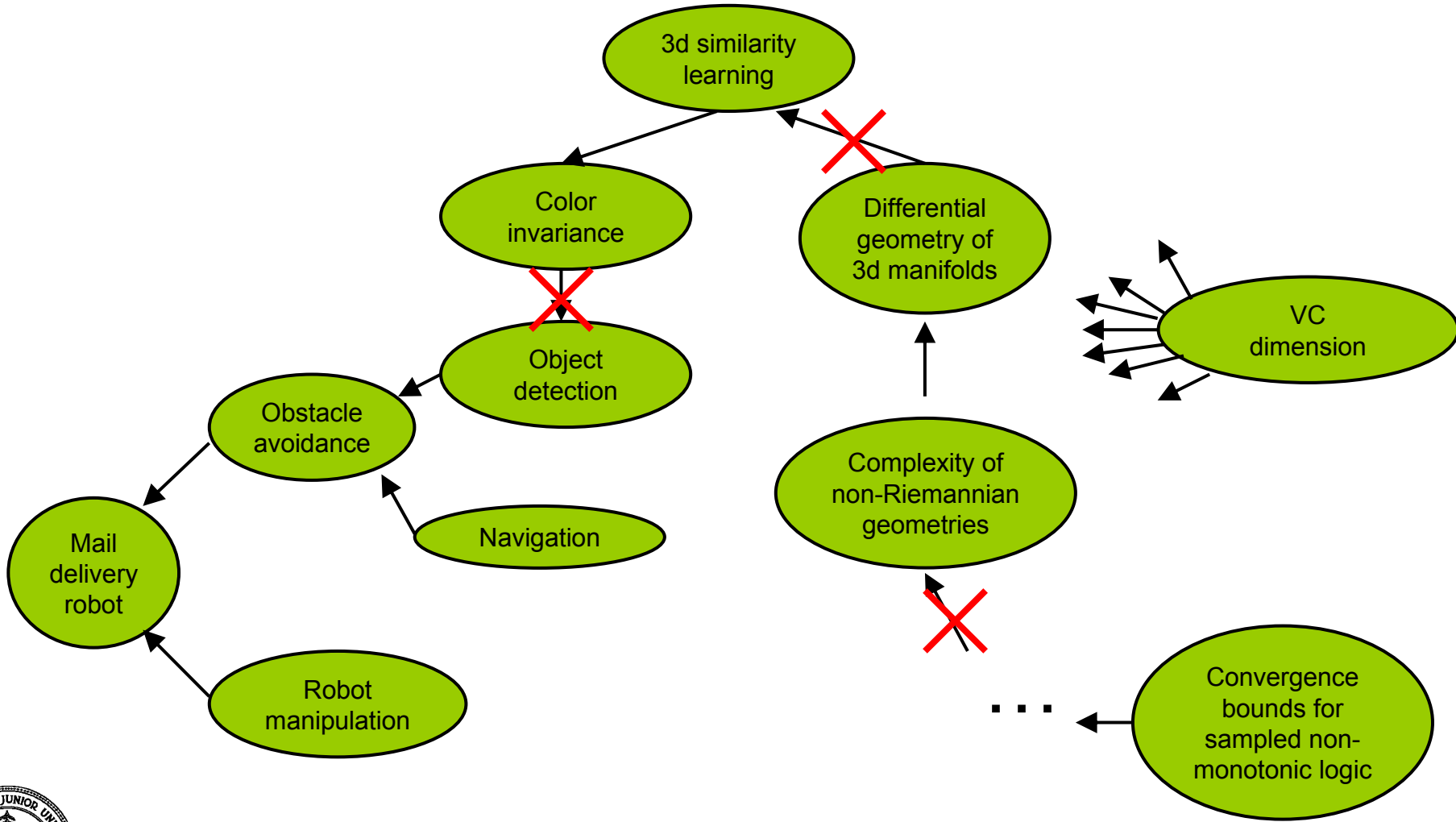
Very often, it's not clear what parts of a system are easy or difficult to build, and which parts you need to spend lots of time focusing on. E.g.,



The only way to find out what needs work is to implement something quickly, and find out what parts break.

[But this may be bad advice if your goal is to come up with new machine learning algorithms.]

The danger of over-theorizing



[Based on Papadimitriou, 1995]

Summary

- Time spent coming up with diagnostics for learning algorithms is time well-spent.
- It's often up to your own ingenuity to come up with right diagnostics.
- Error analyses and ablative analyses also give insight into the problem.
- Two approaches to applying learning algorithms:
  - Design very carefully, then implement.
    - Risk of premature (statistical) optimization.
  - Build a quick-and-dirty prototype, diagnose, and fix.

Also recommended: ML advice by Christopher Ré
http://cs229.stanford.edu/notes2020spring/ML_Advice_lecture.pdf

7 Steps of ML Systems.

- Step 1: Acquire Data
- Step 2: Look at your data* -- after every step.
- Step 3: Create train/dev/test splits
- Step 4: Create/Refine a specification
- Step 5: Build model (simplest that works!)
- Step 6: Measurement
- Step 7: Repeat.

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.

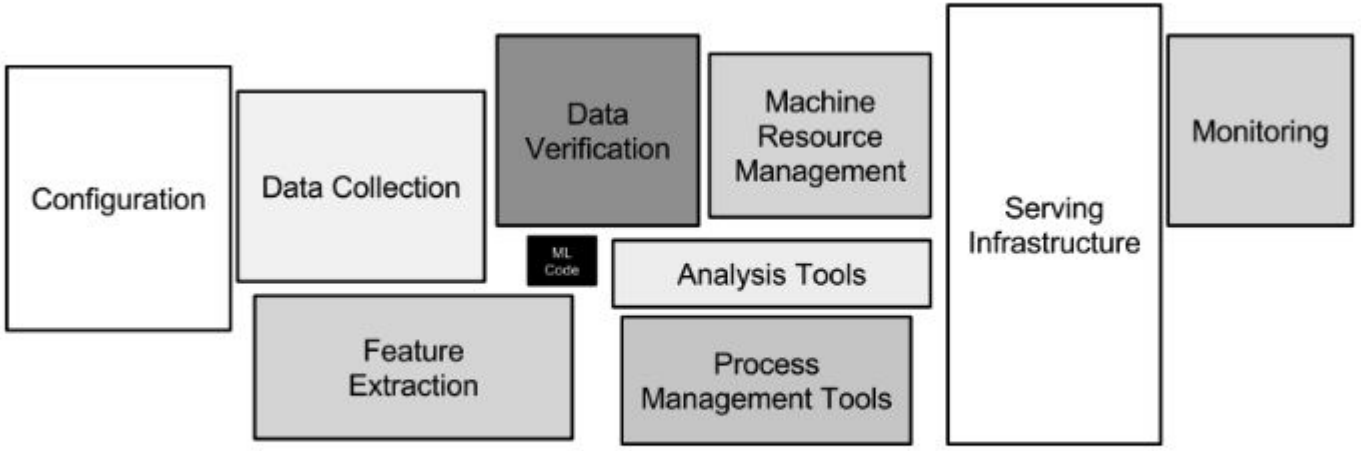


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

BBC Sign in | Home | News | Sport | Reel | Worklife | Travel

NEWS

Home | Coronavirus | Video | World | Asia | UK | Business | Tech | Science | Stories | Entertainment & Arts

Tech

Stephen Hawking warns artificial intelligence could end mankind

By Rory Cellan-Jones
Technology correspondent

2 December 2014 | Comments

Borrowed from Michael Wooldridge, AAAI 2021 talk.

The choice *to communicate about AI/ML or not* depend on **your worldview**.

In public communication, “AI” is often conflated with “machine learning” (ML), a subfield of artificial intelligence.

There has been real progress in ML, driven by scientific advances (e.g. the “deep” in “deep learning”), larger datasets, cheap and easily available compute power.

These advances are narrow, and does not deliver “grand dream” of AI.

Don’t be distracted by media hype, AI advances are real and exciting, but they aren’t magic.

Long term speculative concerns shouldn’t distract us from near-term worries (bias, brittleness, employment, climate …)

GOOGLE \ TECH \ ARTIFICIAL INTELLIGENCE

Google 'fixed' its racist algorithm by removing gorillas from its image-labeling tech

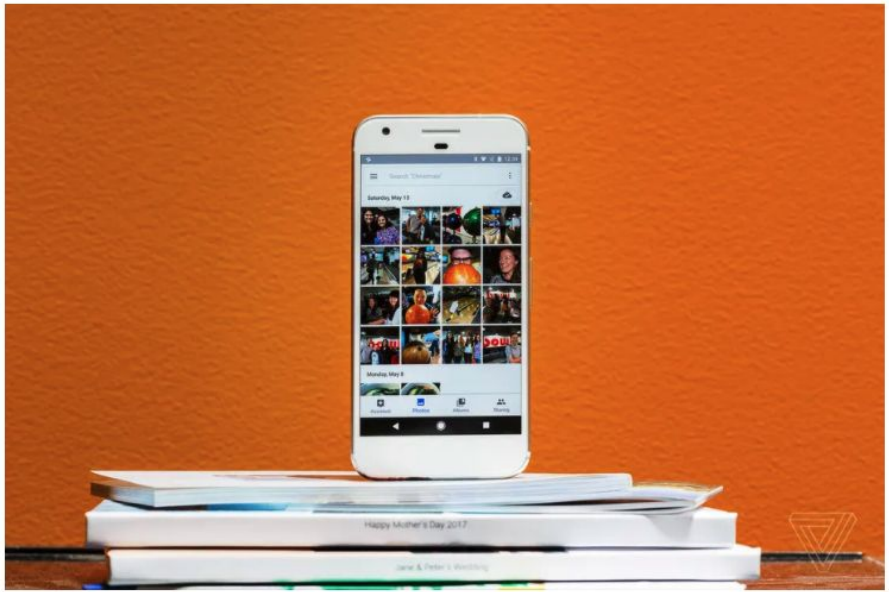
51

Nearly three years after the company was called out, it hasn't gone beyond a quick workaround

By James Vincent | Jan 12, 2018, 10:35am EST

https://www.theverge.com/2018/1/12/16882408/google-racist-gorillas-photo-recognition-algorithm-ai

SHARE



The AI algorithms in Google Photos sort images by a number of categories. | Photo by Vjeran Pavic / The Verge

Back in 2015, software engineer Jacky Alciné pointed out that the image recognition

REUTERS World Business Markets Breakingviews Video More

10 MINUTES AGO | 30 MINUTES AGO

RETAIL

OCTOBER 11, 2018 / 10:04 AM / UPDATED 3 YEARS AGO

Amazon scraps secret AI recruiting tool that showed bias against women

By Jeffrey Dastin

SAN FRANCISCO (Reuters) - Amazon.com Inc's AMZN.O machine-learning specialists uncovered a big problem: their new recruiting engine did not like women.

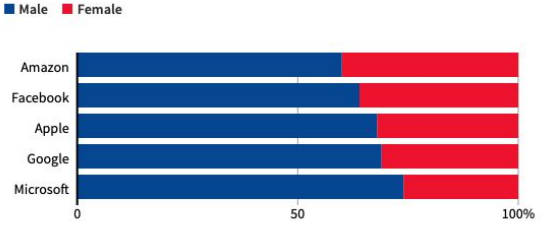
The team had been building computer programs since 2014 to review job applicants' resumes with the aim of mechanizing the search for top talent, five people familiar with the effort told Reuters.

https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G

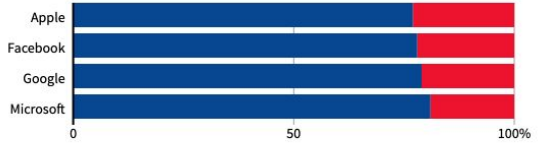
Dominated by men

Top U.S. tech companies have yet to close the gender gap in hiring, a disparity most pronounced among technical staff such as software developers where men far outnumber women. Amazon's experimental recruiting engine followed the same pattern, learning to penalize resumes including the word "women's" until the company discovered the problem.

GLOBAL HEADCOUNT



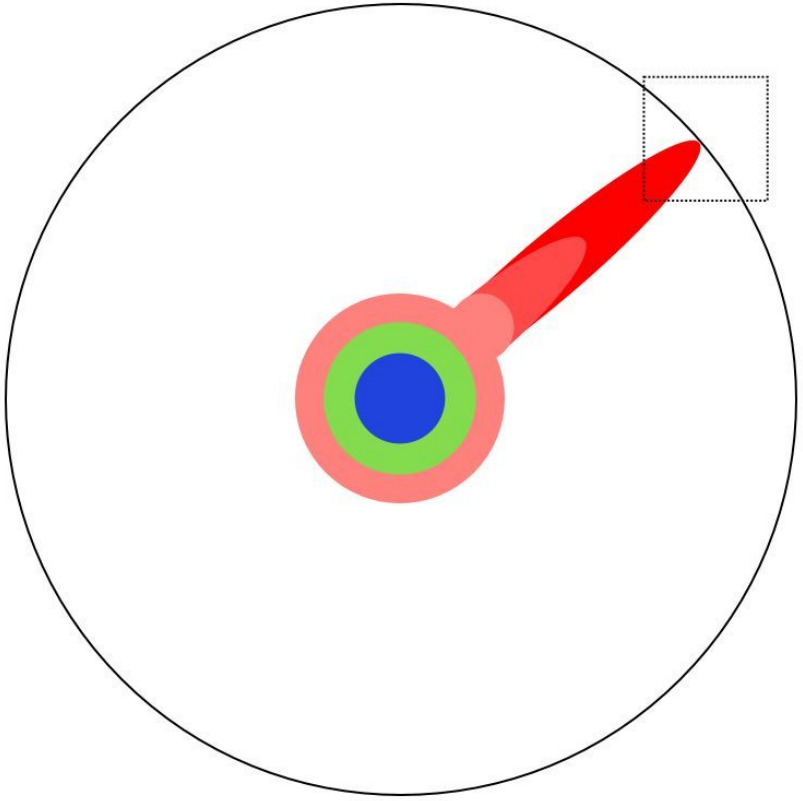
EMPLOYEES IN TECHNICAL ROLES



Note: Amazon does not disclose the gender breakdown of its technical workforce.
Source: Latest data available from the companies, since 2017.
By Han Huang | REUTERS GRAPHICS

Lessons from history abound: crash test dummies, mice models, clinical studies that under-represent minorities.

Keep pushing



The Illustrated Guide to a Ph.D. by Matth Might
https://matt.might.net/articles/phd-school-in-pictures/