Advice for applying Machine Learning

Andrew Ng

Stanford University

Today's Lecture

- Advice on getting learning algorithms to work.
- Most of today's material is not very mathematical. But it's also some of the hardest material in this class to understand.
- Art vs. science.
- 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.)

- Logistic regression with regularization (Bayesian Logistic regression), implemented with gradient ascent, 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

- Logistic regression (with regularization):

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

- 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.

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

Suppose you suspect the problem is either:

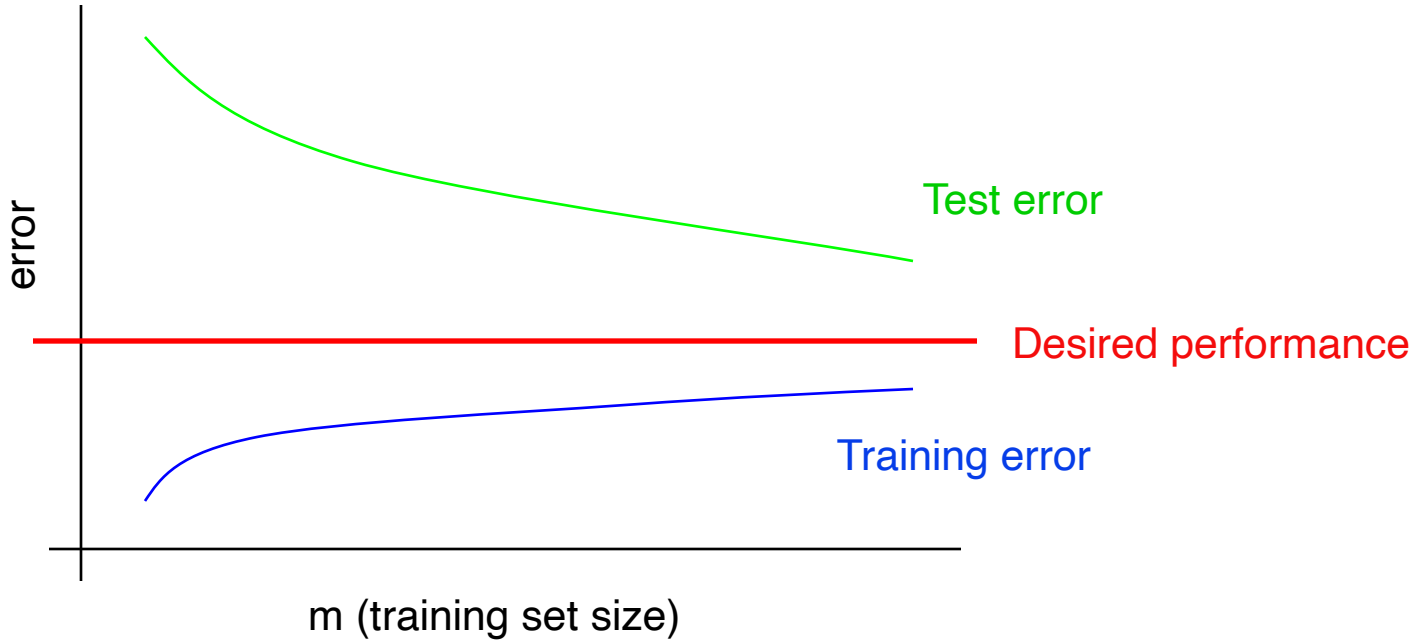
- Overfitting (high variance).
- Too few features to classify spam (high bias).

Diagnostic:

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

More on bias vs. variance

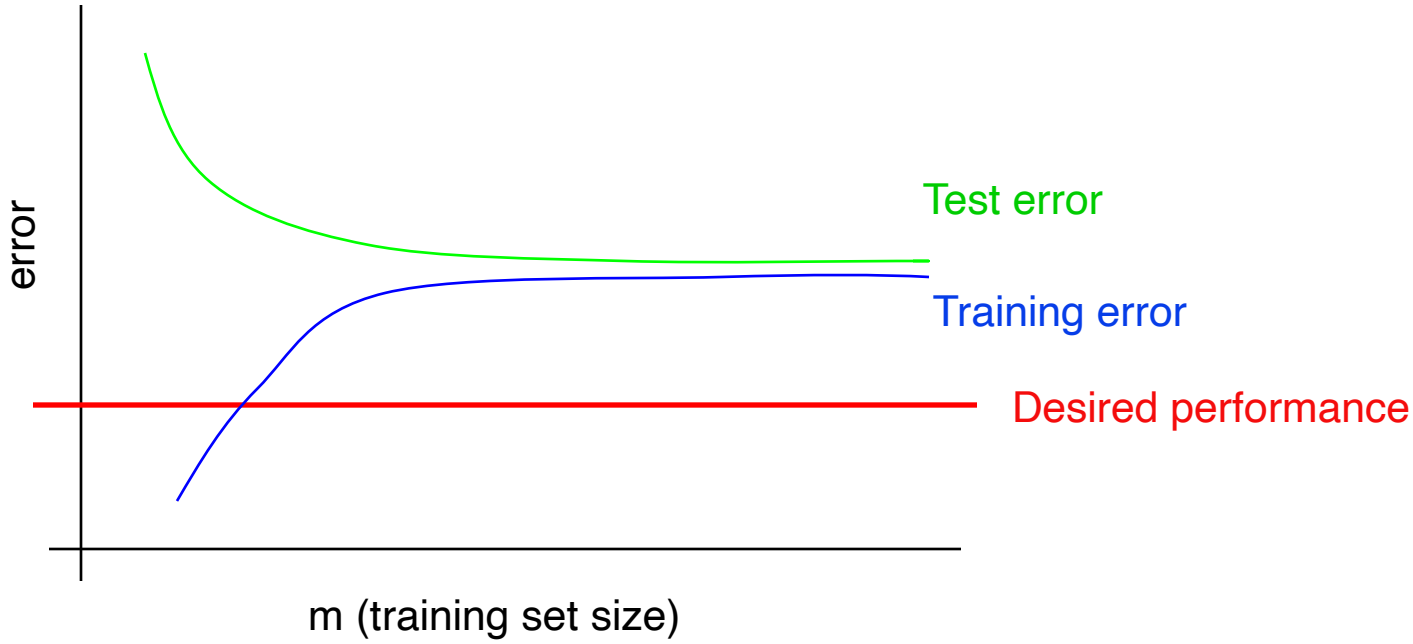
Typical learning curve for high variance:



• 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

Logistic regression, implemented with gradient ascent.

Fixes to try:

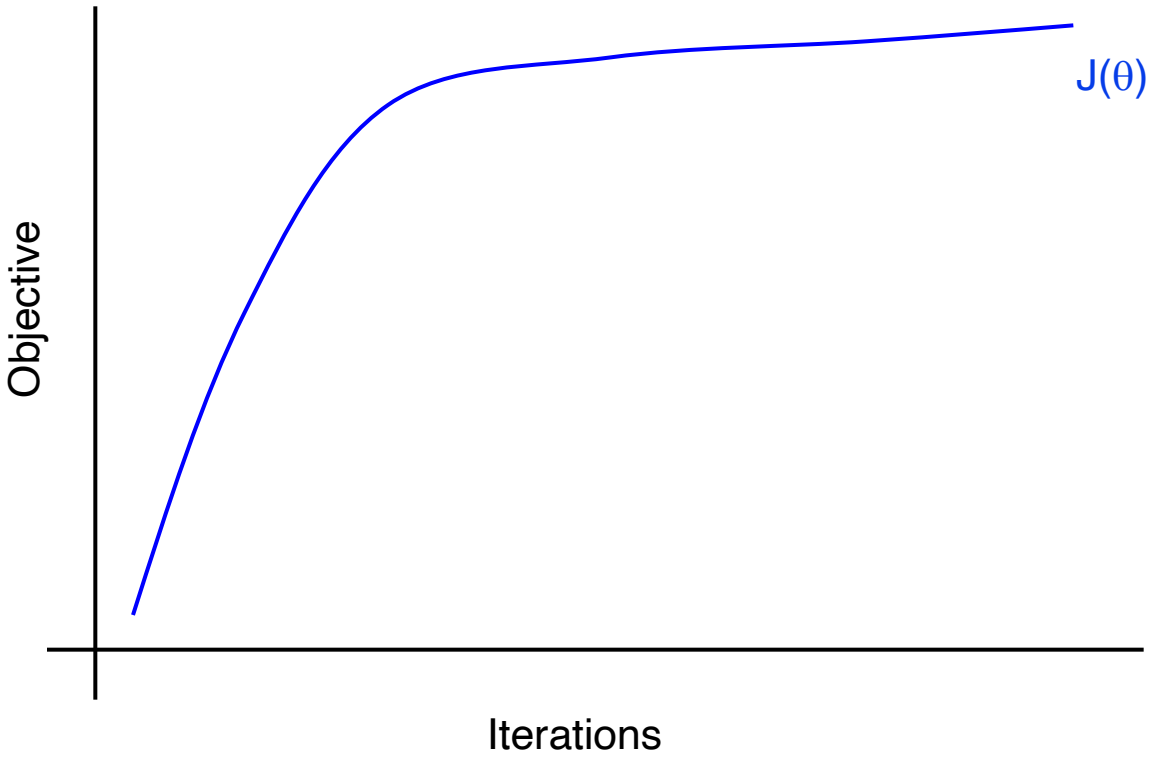
- Try getting more training examples. — Fixes high variance.
- Try a smaller set of features. — Fixes high variance.
- Try a larger set of features. — Fixes high bias.
- Try email header features. — Fixes high bias.
- Run gradient descent for more iterations.
- Try Newton's method.
- Use a different 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:
  - 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 ascent 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 ascent 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).

  - 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 logistic regression, but you really want to deploy logistic regression for your application.

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

Let θ_{BLR} be the parameters learned by logistic regression. (BLR = 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:
$$a(\theta_{\text{SVM}}) > a(\theta_{\text{BLR}})$$
$$J(\theta_{\text{SVM}}) > J(\theta_{\text{BLR}})$$

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:
$$a(\theta_{\text{SVM}}) > a(\theta_{\text{BLR}})$$
$$J(\theta_{\text{SVM}}) \leq J(\theta_{\text{BLR}})$$

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.

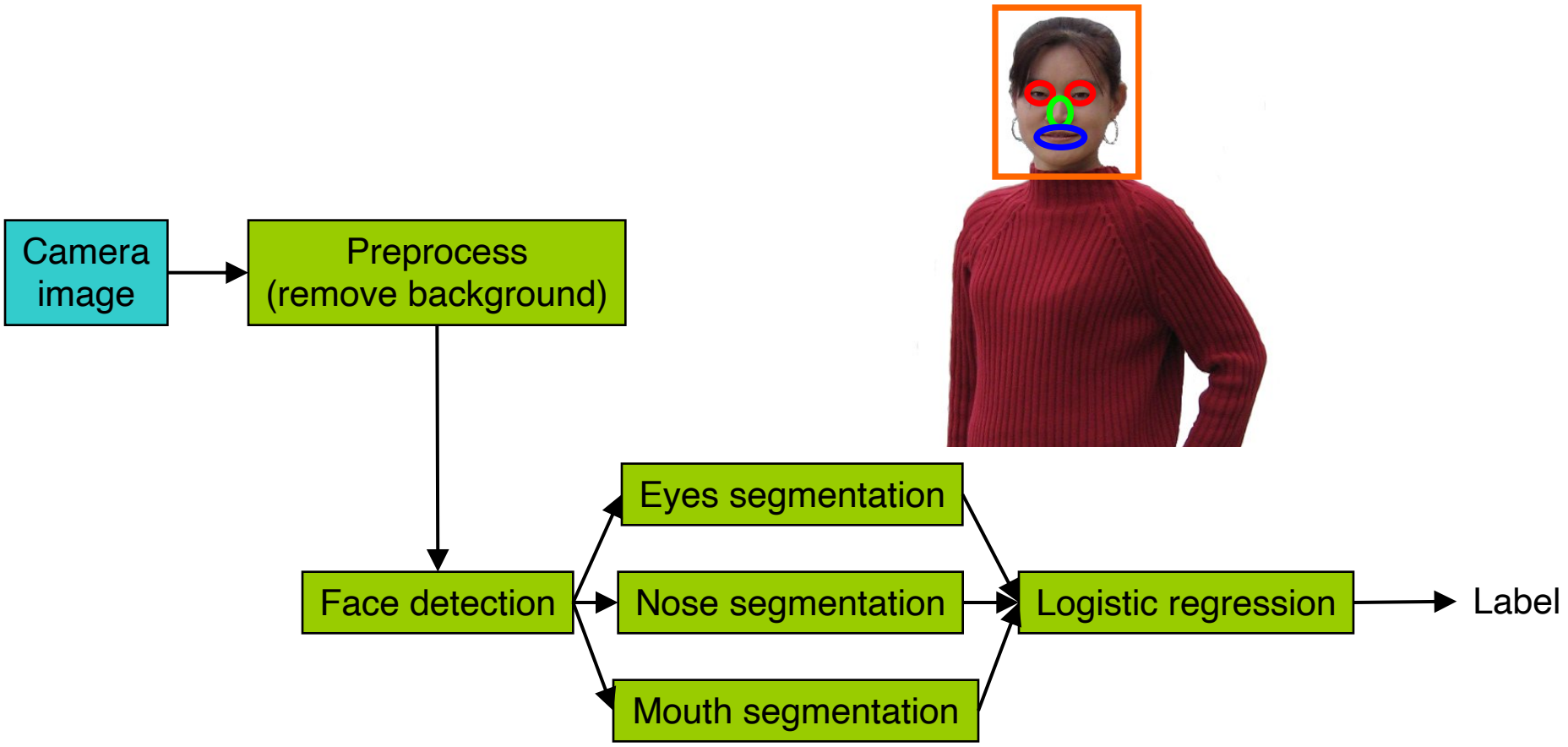
Fixes to try:

– 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.

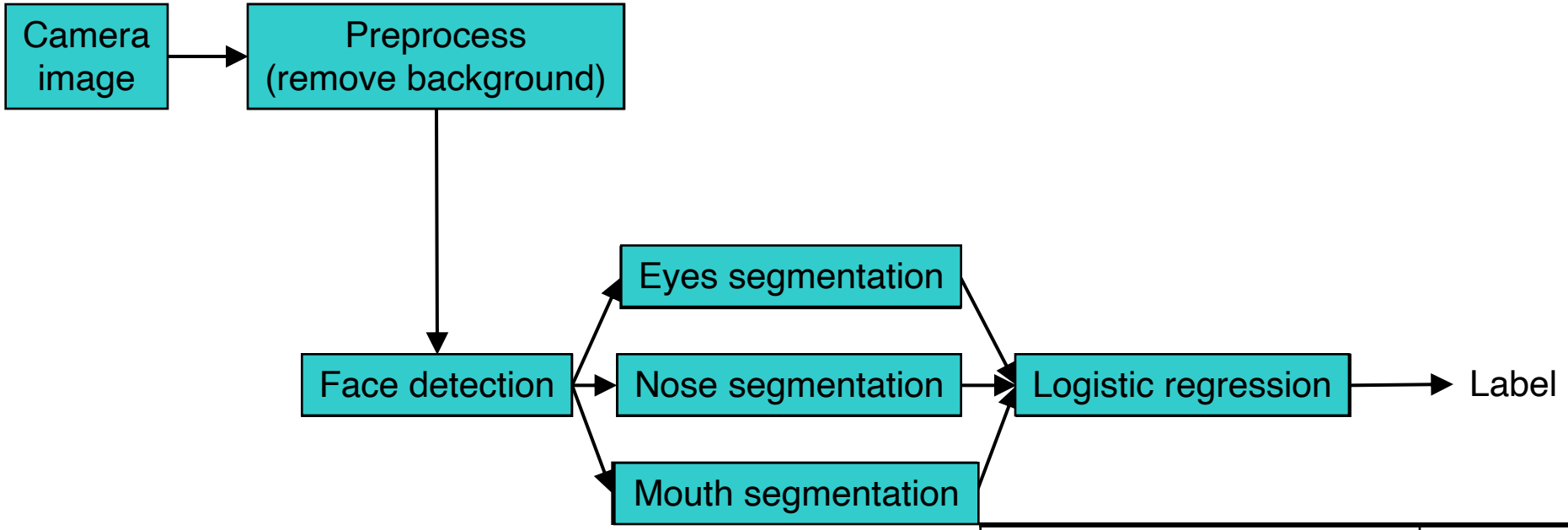
Error Analysis

Error analysis

Many applications combine many different learning components into a "pipeline." E.g., Face recognition from images: [artificial 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	
Preprocess (remove background)	
Face detection	
Eyes segmentation	
Nose segmentation	
Mouth segmentation	
Logistic regression	

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	
Sender host features	
Email header features	
Email text parser features	
Javascript parser	
Features from images	

[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.

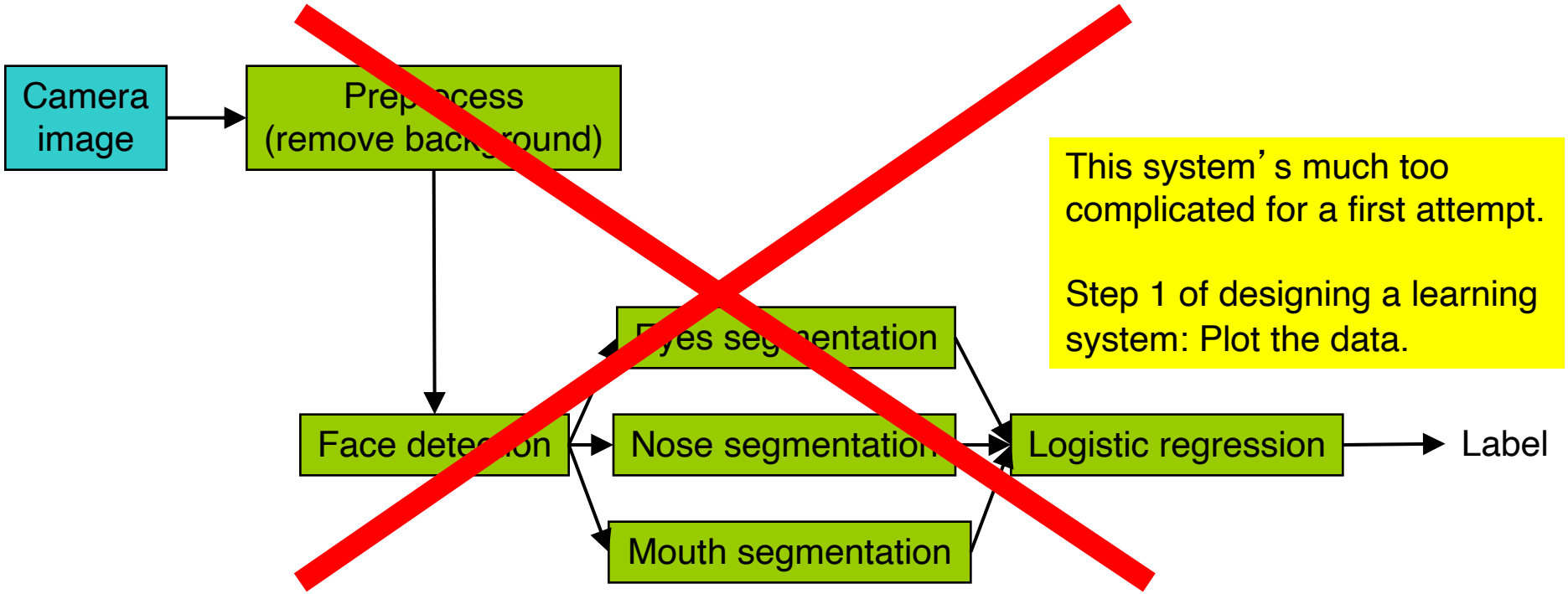
- Spend a long term designing exactly the right features, collecting the right dataset, and designing the right algorithmic architecture.
- Implement it and hope it works.
- Benefit: Nicer, perhaps more scalable algorithms. May come up with new, elegant, learning algorithms; contribute to basic research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

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.]

Summary

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.

Machine Learning Yearning

See also: http://mlyearning.org