EECS 545: Machine Learning
Lecture 19. Advice for Applied Machine Learning

Honglak Lee

03/25/2024

UNIVERSITY OF MICHIGAN

Today's Lecture

- Advice on applying 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. Other Practical Tips in ML and Deep Learning
    4. How to get started on a machine learning problem.
        a. Avoid premature (statistical) optimization.
        b. Try to get the full pipeline working

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

$$\max_{\mathbf{w}} \sum_{n=1}^{N} \log p(y^{(n)}|\mathbf{x}^{(n)}, \mathbf{w}) - \lambda \|\mathbf{w}\|^2$$

$$\min_{\mathbf{w}} -\sum_{n=1}^{N} \log p(y^{(n)}|\mathbf{x}^{(n)}, \mathbf{w}) + \lambda \|\mathbf{w}\|^2$$

Note: these two problems are equivalent.

- What to do next?

Slide credit: Andrew Y. Ng

Fixing the learning algorithm

- Regularized logistic regression: $\max_{\mathbf{w}} \sum_{n=1}^{N} \log p(y^{(n)}|\mathbf{x}^{(n)}, \mathbf{w}) - \lambda \|\mathbf{w}\|^2$

- 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.

Slide credit: Andrew Y. Ng

Diagnostic for bias vs. variance

Better approach:

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

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

Suppose you suspect the problem is either:

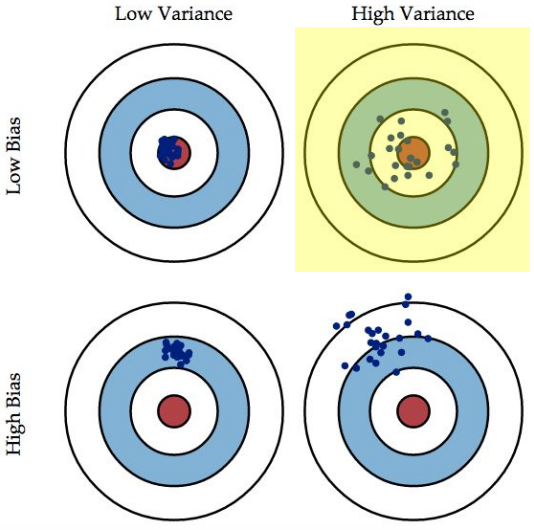
- Overfitting (high variance).
- Too few features to classify spam (high bias).

Diagnostic:

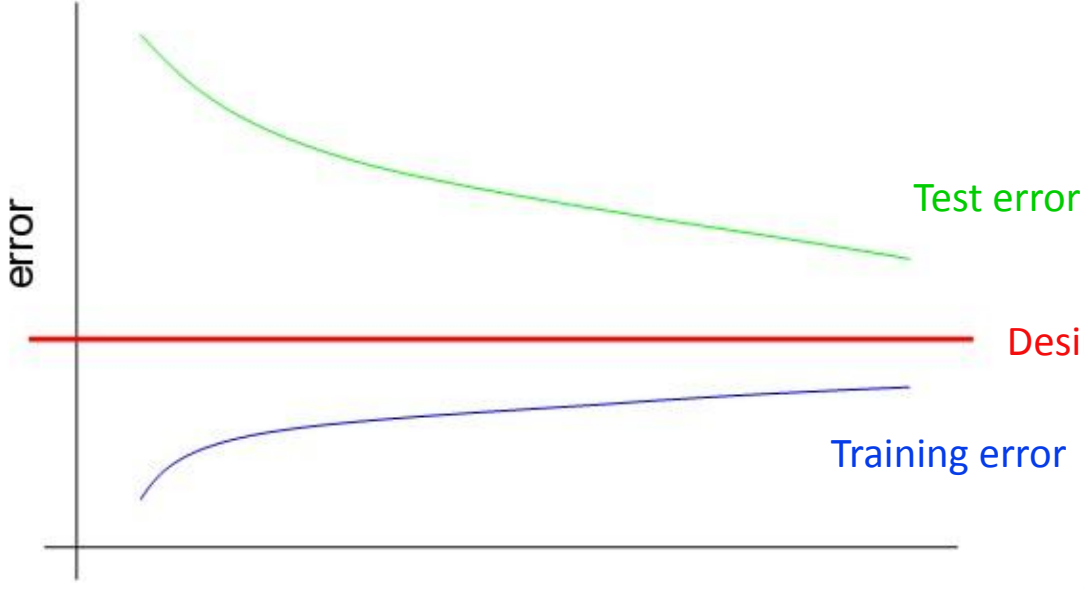
- Variance: Training error will be much lower than test error.
- Bias: Training error will also be high.

Slide credit: Andrew Y. Ng

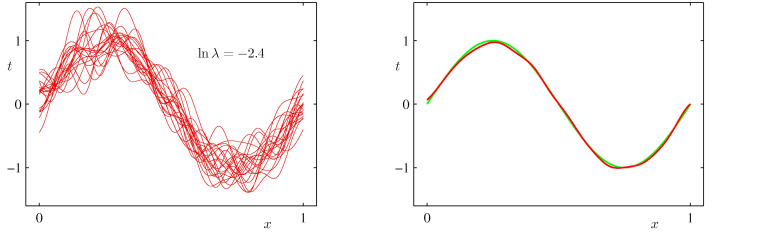
More on bias vs. variance



Typical learning curve for high variance:



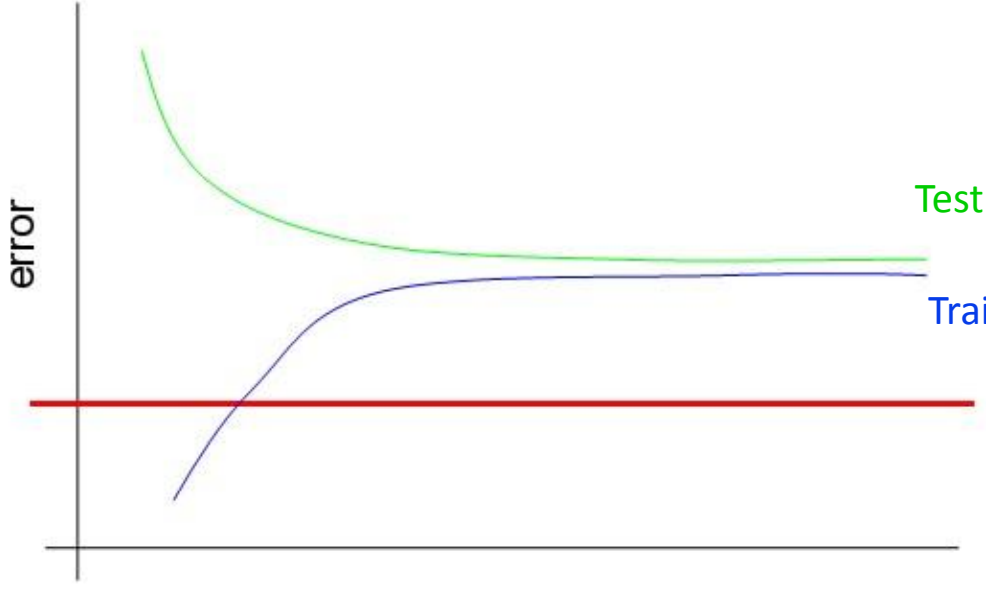
- Test error still decreasing as N increases.
- Suggests larger training set will help.
- Large gap between training and test error.



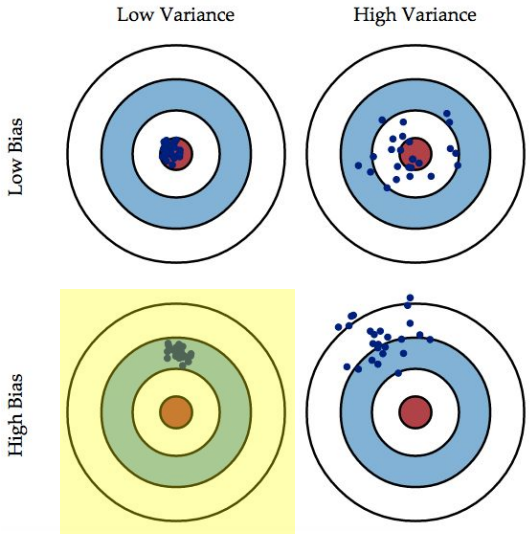
Samples of $h(\mathbf{x};\mathcal{D})$ $\mathbb{E}_{\mathcal{D}}[h(\mathbf{x};\mathcal{D})]$ vs $\mathbb{E}_{\mathcal{D}}[y|\mathbf{x}]$

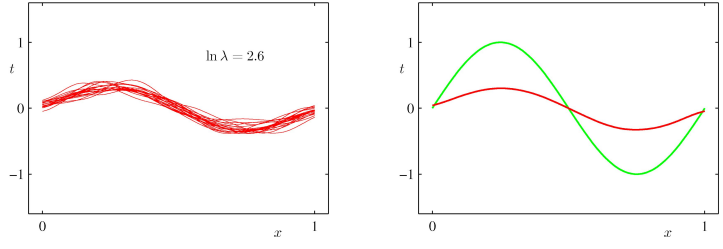
More on bias vs. variance

Typical learning curve for high bias:



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

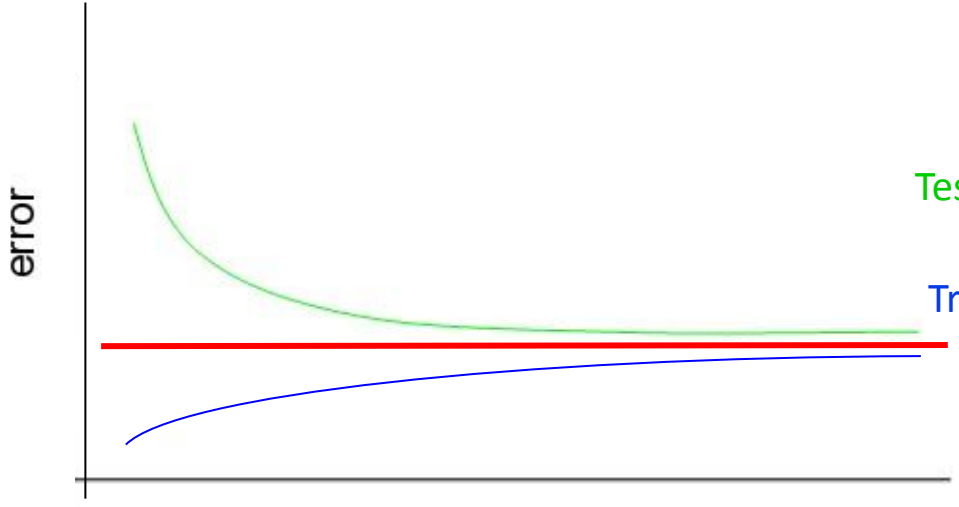


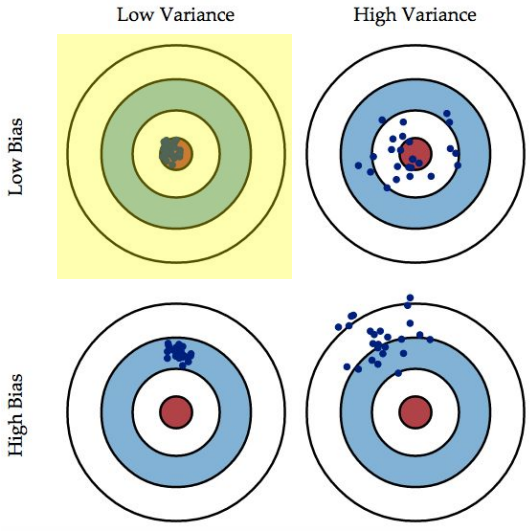


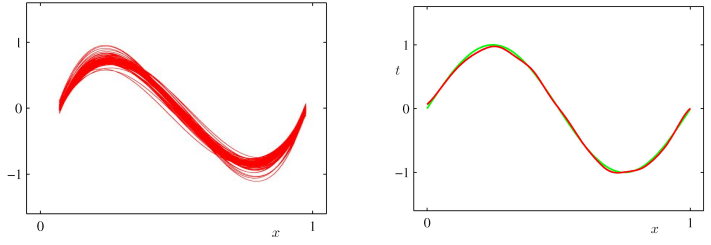
Samples of $h(\mathbf{x};\mathcal{D})$ $\mathbb{E}_{\mathcal{D}}[h(\mathbf{x};\mathcal{D})]$ vs $\mathbb{E}_{\mathcal{D}}[y|\mathbf{x}]$

More on bias vs. variance

Typical learning curve for low bias and low variance:
(ideal scenario)







Samples of $h(\mathbf{x};\mathcal{D})$ $\mathbb{E}_{\mathcal{D}}[h(\mathbf{x};\mathcal{D})]$ vs $\mathbb{E}_{\mathcal{D}}[y|\mathbf{x}]$

- Small gap between training and test error.

Diagnostics tell you what to try next

Regularized 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.

Slide credit: Andrew Y. Ng

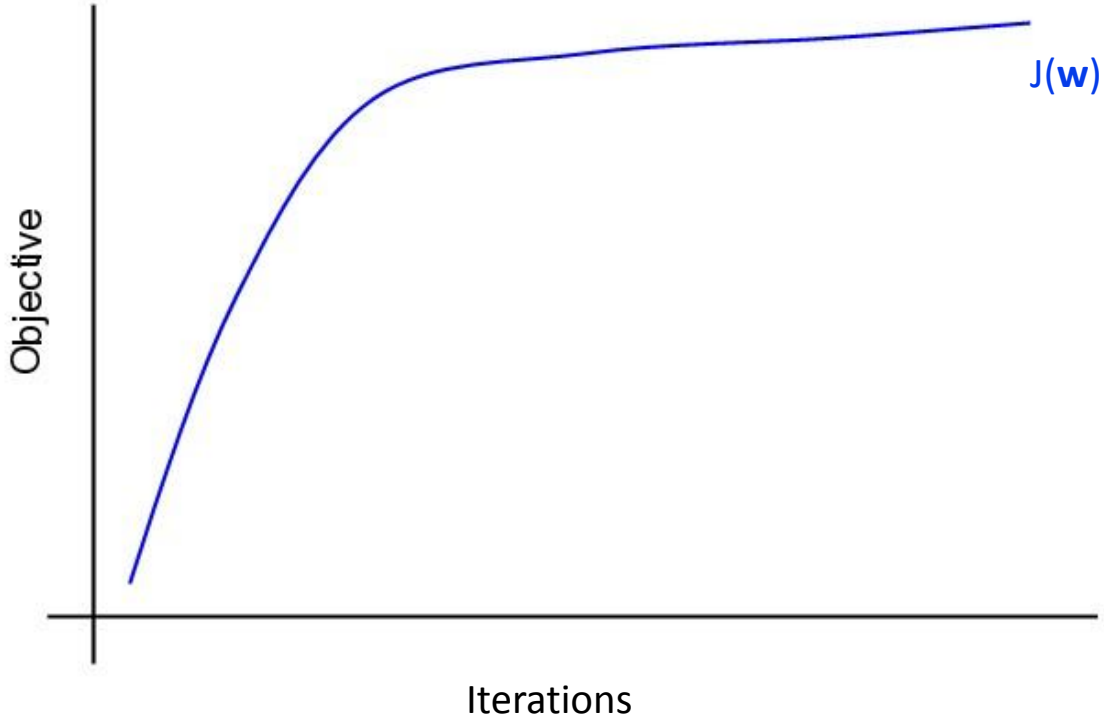
Optimization algorithm diagnostics

- 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:
  - Regularized 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?

Slide credit: Andrew Y. Ng

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.

Slide credit: Andrew Y. Ng

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: $a(\mathbf{w}) = \sum_{n=1}^{N} c^{(n)} \mathbb{1}\{h(\mathbf{x}^{(n)}; \mathbf{w}) = y^{(n)}\}$

    (weights c(n) higher for non-spam than for spam)
  - Regularized logistic regression? Correct value for λ?

$$\max_{\mathbf{w}} \sum_{n=1}^{N} \log p(y^{(n)}|\mathbf{x}^{(n)}, \mathbf{w}) - \lambda \|\mathbf{w}\|^2$$

  - SVM? Correct value for C?

$$\min_{\mathbf{w},b} \quad \|\mathbf{w}\|^2 + C \sum_{n=1}^{N} \xi^{(n)}$$

$$\text{s.t.} \quad y^{(n)}(\mathbf{w}^\top \mathbf{x}^{(n)} - b) \geq 1 - \xi^{(n)}$$

Slide credit: Andrew Y. Ng

Diagnostic

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

Let $\mathbf{w}_{\text{SVM}}$ be the parameters learned by an SVM. Let $\mathbf{w}_{\text{LR}}$ be the parameters learned by regularized logistic regression.

You care about the weighted accuracy: $a(\mathbf{w}) = \max_{\mathbf{w}} \sum_{n=1}^{N} c^{(n)} \mathbb{1}\{h(\mathbf{x}^{(n)}; \mathbf{w}) = y^{(n)}\}$

$\mathbf{w}_{\text{SVM}}$ outperforms $\mathbf{w}_{\text{LR}}$. So: $a(\mathbf{w}_{\text{SVM}}) > a(\mathbf{w}_{\text{LR}})$

LR tries to maximize: $J(\mathbf{w}) = \sum_{n=1}^{N} \log p(y^{(n)}|\mathbf{x}^{(n)}; \mathbf{w}) - \lambda \|\mathbf{w}\|^2$

Diagnostic: $J(\mathbf{w}_{\text{SVM}}) > J(\mathbf{w}_{\text{LR}})$?

Slide credit: Andrew Y. Ng

Two cases

Case 1: $a(\mathbf{w}_{\text{SVM}}) > a(\mathbf{w}_{\text{LR}})$
$J(\mathbf{w}_{\text{SVM}}) > J(\mathbf{w}_{\text{LR}})$

But LR was trying to maximize J(**w**). This means that w_{LR} fails to maximize J, and the problem is with the convergence of the algorithm. Problem is with the optimization algorithm.

Case 2: $a(\mathbf{w}_{\text{SVM}}) > a(\mathbf{w}_{\text{LR}})$
$J(\mathbf{w}_{\text{SVM}}) \leq J(\mathbf{w}_{\text{LR}})$

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

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

Slide credit: Andrew Y. Ng

Diagnostics tell you what to try next

Regularized 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.

Slide credit: Andrew Y. Ng

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.

Slide credit: Andrew Y. Ng

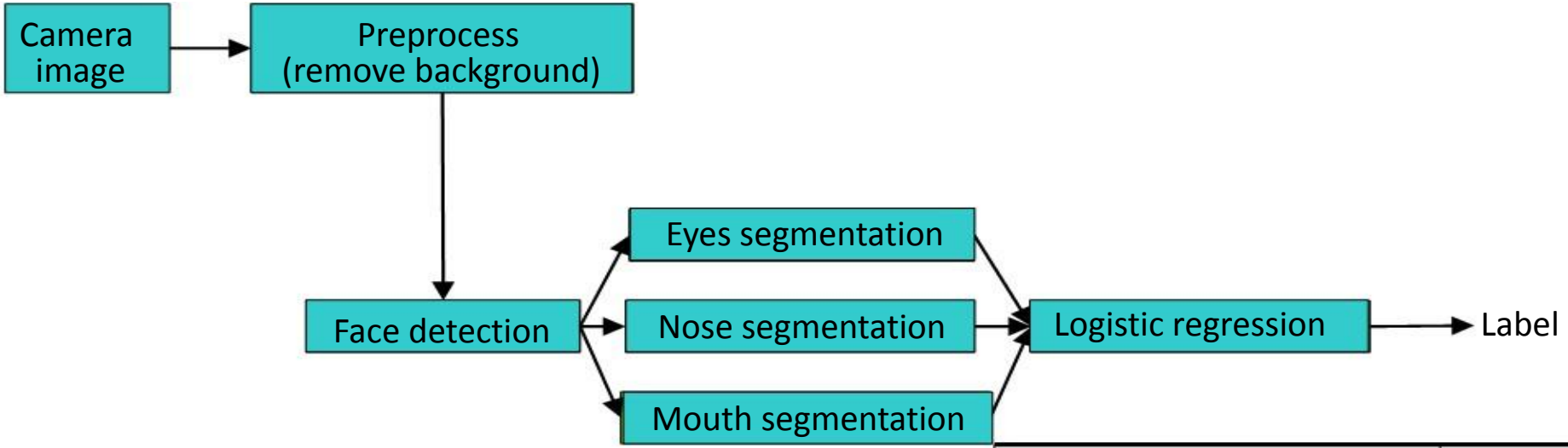
Error Analysis

Error analysis

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

Slide credit: Andrew Y. Ng

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?

Slide credit: Andrew Y. Ng

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.

Slide credit: Andrew Y. Ng

Other Practical Tips in ML and Deep Learning

Practical Tips in ML applications

- Start with simple models and evaluate multiple methods
  - linear methods vs deep learning methods
  - more complex models are only useful when they perform better than simpler models ("sanity check")
- Choose appropriate evaluation metrics
  - squared error, classification accuracy, precision/recall, ROC, computational complexity for training and inference, etc.
- Tune hyperparameters and use cross-validation
- Visualize results
- Understand the limitations of your model
  - no model is perfect!

Practical Tips in Deep Learning

- Preprocess the data (e.g., data cleaning, normalizing, etc.)
  - In neural nets, data scale matter; proper normalization (especially for input) will be needed
    (e.g. image pixel: [0 ~ 255] v.s. normalized to [-1, 1])
- Do data augmentation whenever possible
- Select the right loss function
- Regularize the model (e.g., dropout, weight decay, etc.)
- Tune hyperparameters
- Consider using pre-trained models (for images, text, etc.)
  - they may perform quite well right out of the box with small amounts of training!

Practical tips for Deep Learning (cont'd)

- Overfitting on a small batch
  - The loss should be 0 (or as small as possible)!
  - If it cannot, it means that the model is either too complex or not complex enough, or there is some bug, to overfit even on a small batch (few examples).
- Gradient clipping can help preventing model divergence
  - Watch (the norm of) per-layer gradients
- Make sure that your gradient calculation is correct
  - e.g., your backprop should return approximately the same values as "numerical gradients"
    - (we did it in our HWs!)

$$\frac{df}{dx}(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

  - Good news: Most of deep learning libraries would do this automatically!

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.

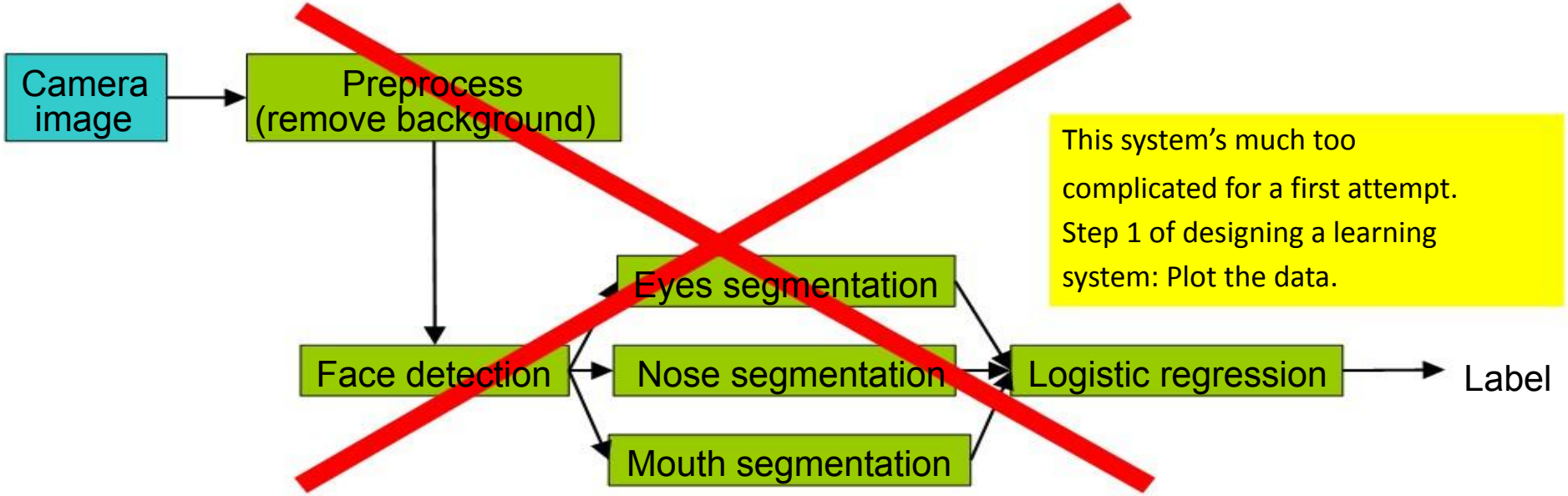
Approach #2: Build-and-fix.

- 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.

Slide credit: Andrew Y. Ng

Premature 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.,

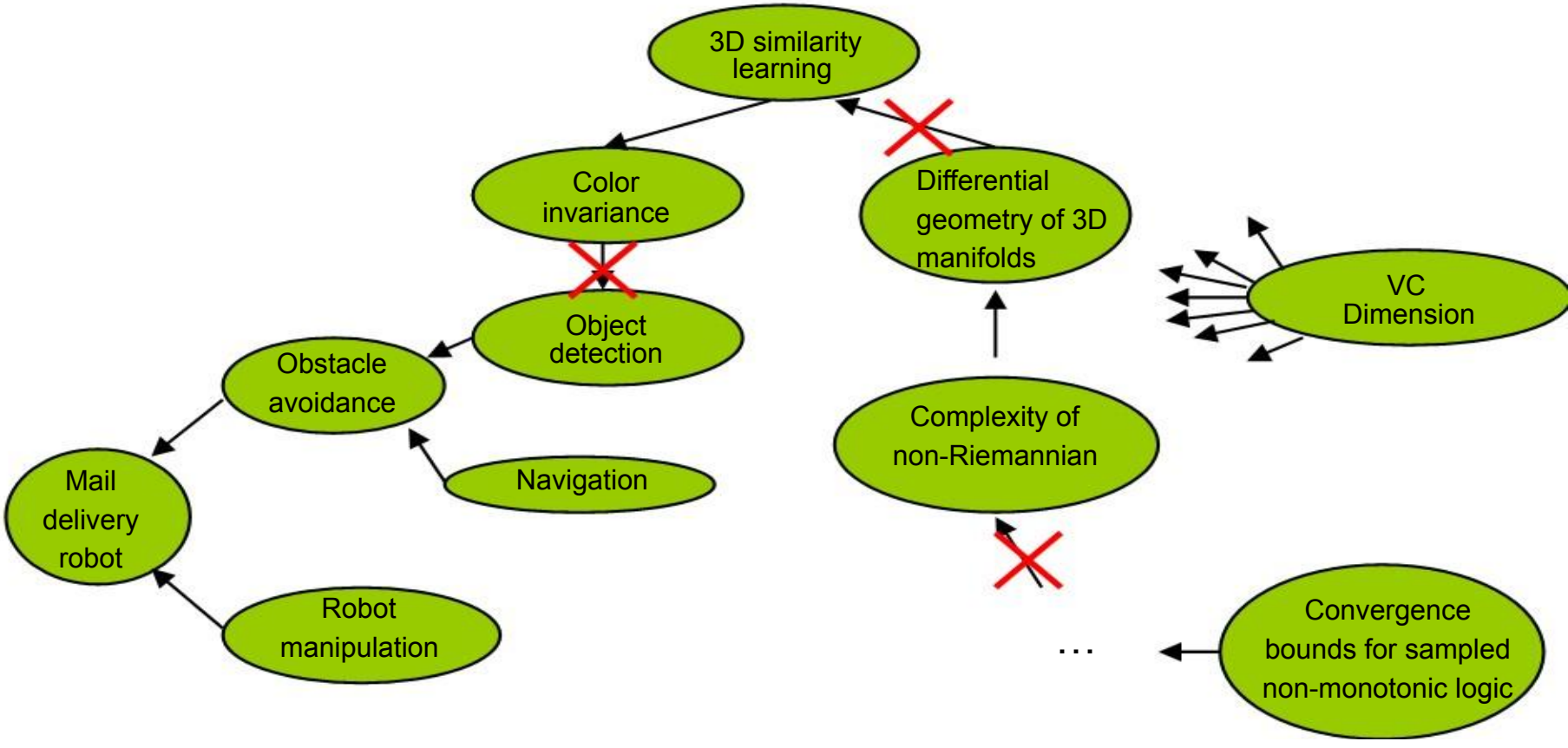


The only way to find out what needs work is to implement something quickly, and find out what parts break.

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

Slide credit: Andrew Y. Ng

The danger of over-theorizing



[Based on Papadimitriou, 1995]

Slide credit: Andrew Y. Ng

Summary

Summary

- Time spent coming up with diagnostics for ML 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.
- Other Practical Tips in ML and Deep Learning
- 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.

Any feedback (about lecture, slide, homework, project, etc.)?

(via **anonymous** google form: https://forms.gle/99jeftYTaozJvCEF8)

Change Log of lecture slides:
https://docs.google.com/document/d/e/2PACX-1vRKx40eOJKACqrKWraio0AmlFS1_xBMINuWcc-jzpfo-ySj_gBuqTVdfHy8v4HDmqDJ3b3TvAW1FVuH/pub