Machine Learning in Practice

Crash Course on Machine Learning

August 22-24, 2016

Arnaud Gotlieb and Valeriya Naumova

Simula Research Laboratory AS

Simula School of Research and Innovation (SSRI)

GOAL:

Advice on how to apply learning algorithms to different applications

Some key aspects of this lecture:

- **No math!!!** But it could be much harder material to understand and use;
- Some aspects are debatable;
- Advice might not be applicable for novel machine learning research;
- Briefly…. to give you some time to play with the labs.

Slides based on:

- ML Lecture by A. Ng, Stanford University
- Lectures and papers by P. Domingos, UC Washington
- Presentations by Scott Fortmann-Roe

ML in Practice

- Understanding domain, prior knowledge, and goals
- Data integration, selection, cleaning, pre-processing, etc.
- Learning models
- Interpreting results
- Consolidating and deploying discovered knowledge
- Loop

LEARNING

=

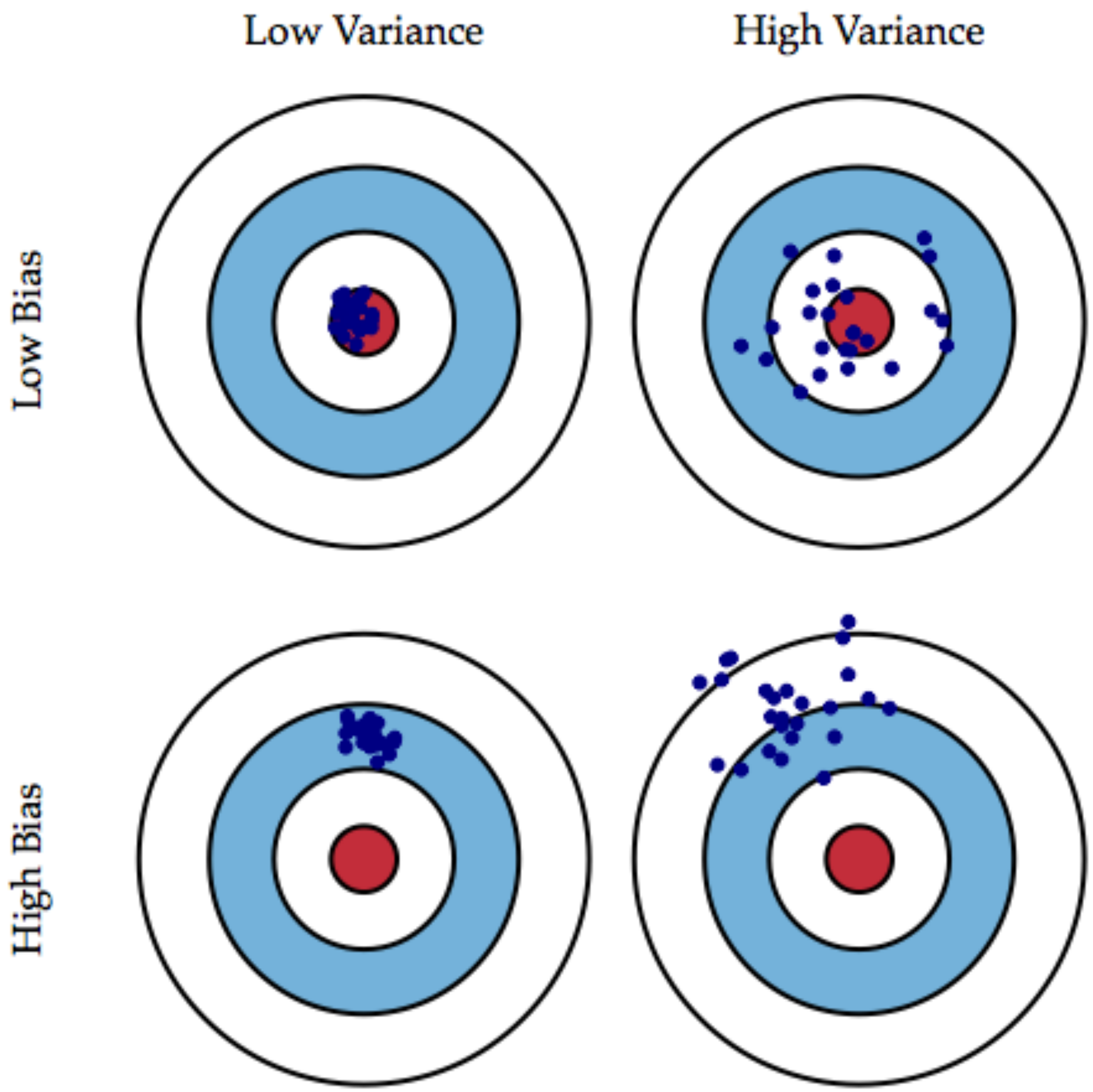
REPRESENTATION + EVALUATION + OPTIMISATION

Key aspects to remember

- **It's generalisation that counts…**
  - Set some data aside from the beginning to test your estimator at the end
  - Use cross-validation

- **Data alone is not enough**
  - Every learner must embody some knowledge or assumption beyond the data it is given in order to generalise beyond it.
  - "**No Free Lunch Theorem**"
  - one of the key criteria for choosing a representation is which kinds of knowledge are easily expressed in it.
  - **Remember:** Machine Learning is not Magic!!!

- **Overfitting has many faces**
  - Decompose the generalisation error into bias and variance
  - Use cross-validation, regularisation



High Variance = Overfitting:

- the model has too many parameters.

High Bias = Underfitting:

- the model is too rigid.

Key aspects to remember

- **Intuition fails in high dimensions**
  - Our intuitions, which come from a three- dimensional world, often do not apply in high-dimensional ones.
  - Luckily most of the real-life data has a lower-dimensional representation
- **Theoretical guarantees are not what they seem**
- **Feature engineering is the key**
  - Data pre-processing and feature extraction might be the most tedious work
- **More data beats a cleverer algorithm**
  - The issue of scalability (time, memory and training set)

Key aspects to remember

- **Learn many models not just one**
    - Model ensembles: bagging, boosting..
- **Simplicity does not imply accuracy**
- **Representable does not imply learnable**
    - Can it be represented? → Can it be learned?
- **Correlation does not imply causation**
    - Diapers - Beer Example

Getting Started on a Problem: Two Approaches

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.

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:

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

Try improving algorithms in different ways:

- Getting more training examples.
- Reduce the set of features.
- Enlarge the set of features.
- Use different features.
- Run the optimiser (gradient descent) for some more iterations.
- Choose a different optimisation algorithm.
- Use a different regularisation term or constant value.
- Try another learning algorithm (SVM).

. . . some may be fixing problems you don't have.

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.

Principled Analysis: Diagnostics

First figure out what's going on.

- Overfitting vs. Underfitting?
- Search error vs. Modelling error?
- Complex system: Find the most problematic component.

Trivial but vital:

- Visualise the data. (Plot or view frequent patterns.)
- Start with simple things.

Diagnostic for Bias vs Variance

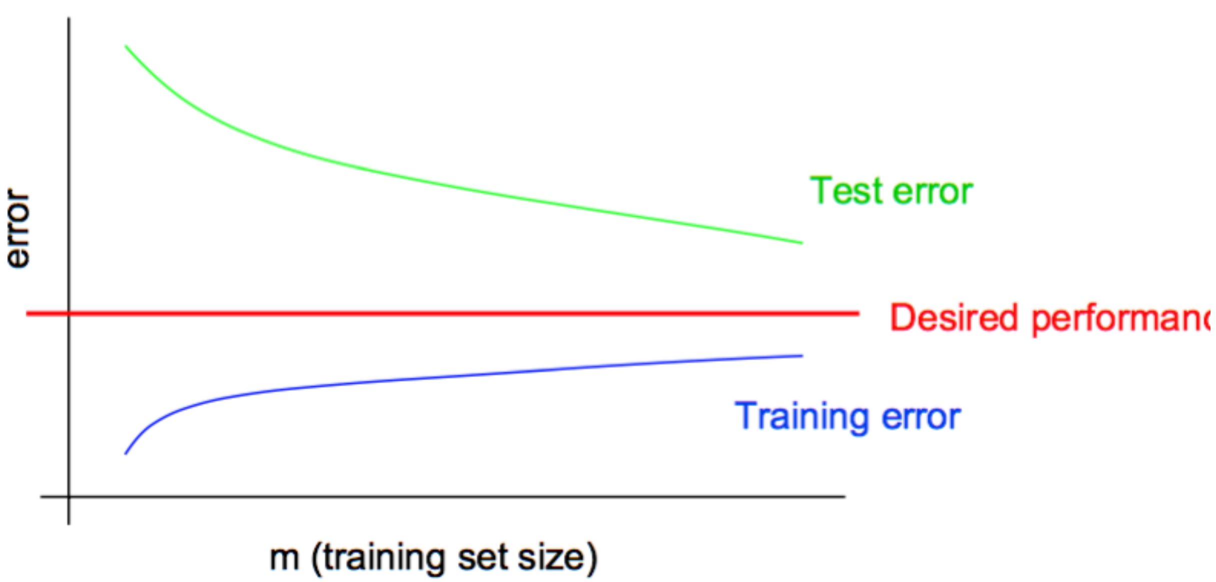
Suppose you suspect the problem is either:

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

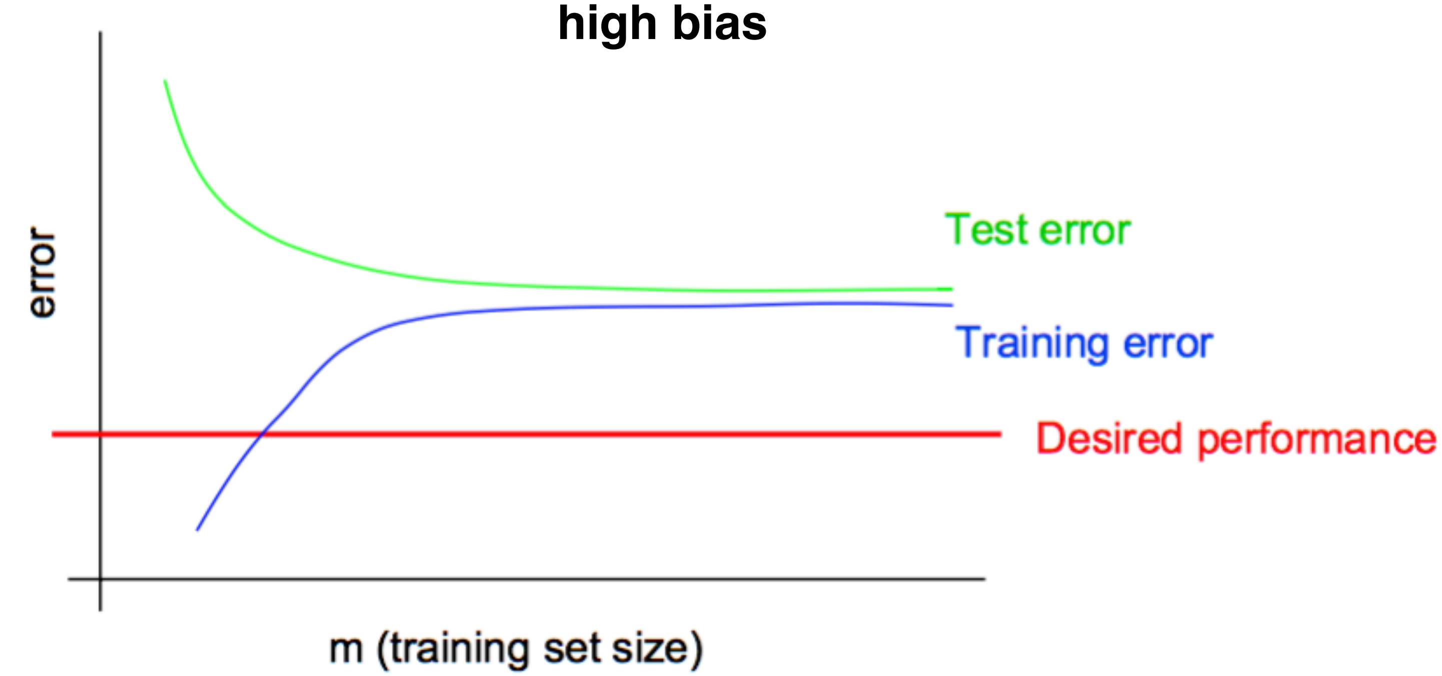
Diagnostic:

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

Typical learning curve for high variance



Typical learning curve for high bias



Diagnostics Tell You What to Try Next

Bayesian logistic regression:

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

Try improving algorithms in different ways:

- Getting more training examples. **Fixes high variance**
- Reduce the set of features. **Fixes high variance**
- Enlarge the set of features. **Fixes high bias**
- Use different features (email header). **Fixes high bias**
- Run the optimiser (gradient descent) for some more iterations.
- Choose a different optimisation algorithm.
- Use a different regularisation term or constant value.
- Try another learning algorithm (SVM).

  . . . some may be fixing problems you don't have.

Optimisation 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?

Search vs Modelling Error

Search Error:

- the optimiser fails to find the best parameters
- . .. a problem with the optimiser.

Modelling Error:

- the best parameters do not lead to the best performance.
- . . . a problem with the objective function.

Consider:

- Will more iterations help? Is the algorithm (gradient descent for logistic regression) converging?
- When can two learners help to diagnose the problem?
- Are you optimising the right function?
- Correct value of the regularisation parameter?

Diagnostics tell you what to try next

Bayesian logistic regression:

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

Try improving algorithms in different ways:

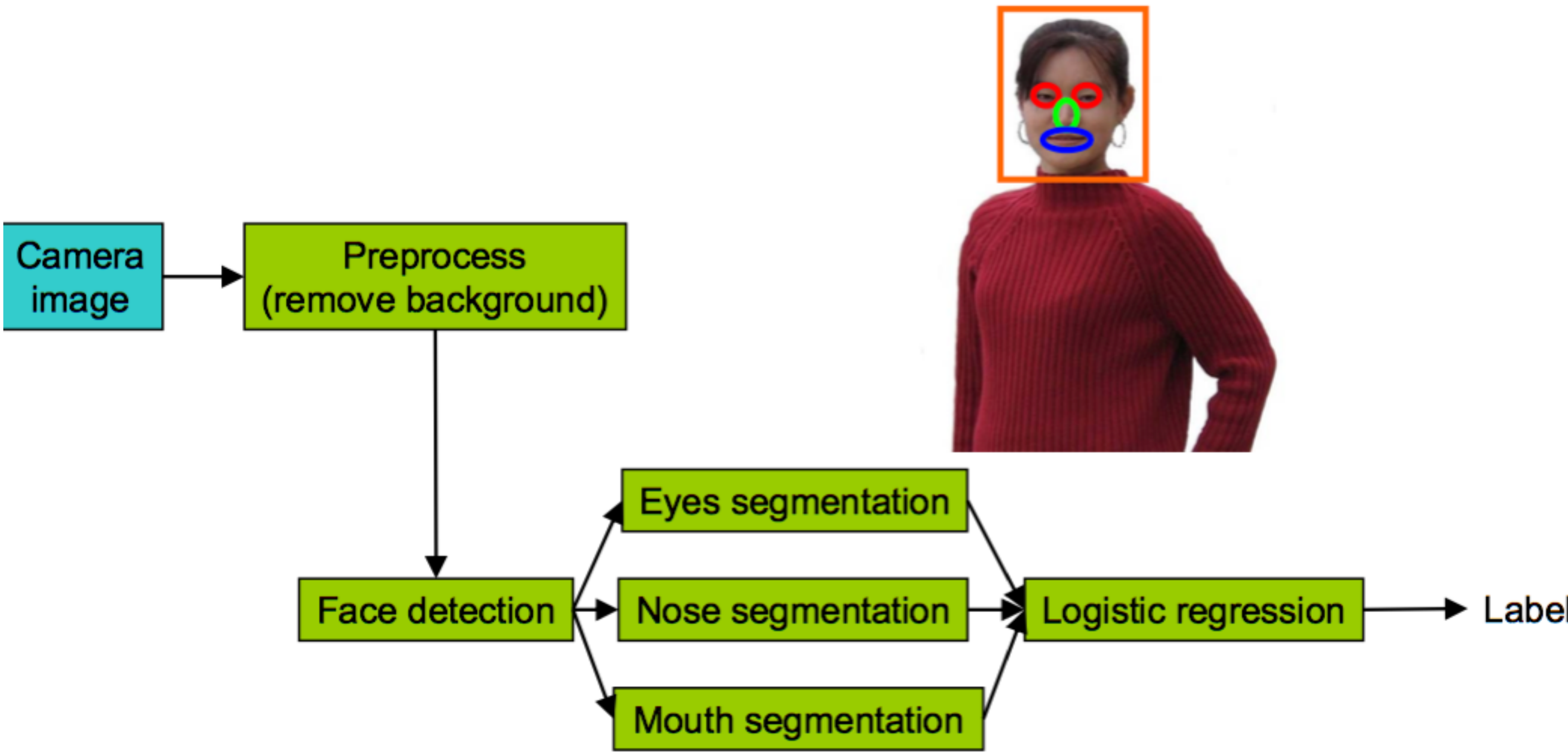
- Getting more training examples. **Fixes high variance**
- Reduce the set of features. **Fixes high variance**
- Enlarge the set of features. **Fixes high bias**
- Use different features (email header). **Fixes high bias**
- Run the optimiser (gradient descent) for some more iterations. **Fixes opt algorithm**
- Choose a different optimisation algorithm. **Fixes opt algorithm**
- Use a different regularisation term or constant value. **Fixes opt objective**
- Try another learning algorithm (SVM). **Fixes opt objective**

  . . . some may be fixing problems you don't have.

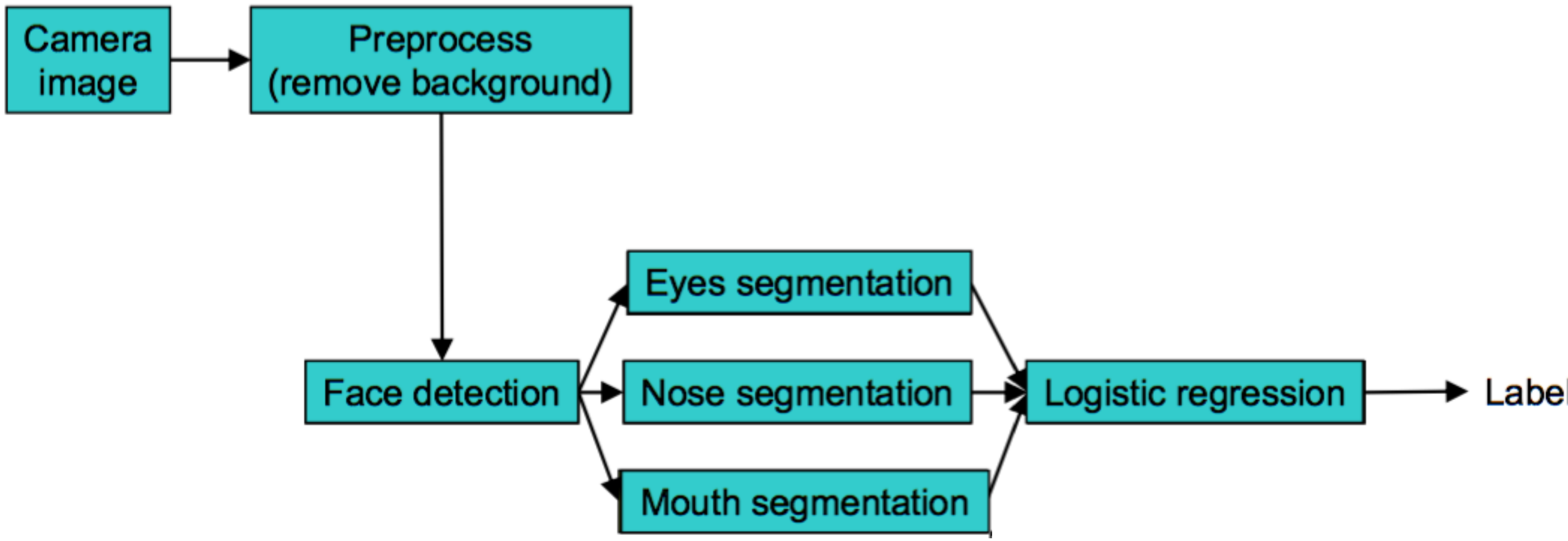
Error Analysis

Machine learning "pipeline" consists of many learning algorithms.

Example: face recognition from images.



Error Analysis



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

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.

Complex Systems

Error Analysis:

- Compares the best possible vs. current accuracy.
- Provide more and more golden truth data as part of the input.
- Find the component where the jump in accuracy is the highest.

Ablative Analysis:

- Compares some baseline vs. current accuracy.
- Switch off more and more components.
- Find the component where the loss in accuracy is the highest.