Advice on Applying ML

Instructor: Jessica Wu -- Harvey Mudd College

The instructor gratefully acknowledges Andrew Ng (Stanford), from whom these slides are adapted.

Choosing the Right Estimator

Today’s Lecture

• Advice on applying learning algorithms to different applications

• Most of today’s lecture is not very mathematical...
  – but it is also some of the hardest material in this class to understand

• Some of what I will say today...
  – is debatable
  – is not good advice for doing novel ML research

• Key ideas
  1. Diagnostics for debugging learning algorithms
  2. Error analyses and ablative analysis
  3. How to get started on a ML problem
     – Premature (statistical) optimization

Motivating Example:
Building a Spam Classifier

From: cheapsales@buystufffromme.com
To: ang@cs.stanford.edu
Subject: Buy now!
Deal of the week! Buy now!
Rolex watches - $100
Medicine (any kind) - $50
Also low cost M0rgages available.

From: Alfred Ng
To: ang@cs.stanford.edu
Subject: Christmas dates?
Hey Andrew,
Was talking to Mom about plans for Xmas. When do you get off work. Meet Dec 22?
Alf

spam (1)  non-spam (0)
Debugging Learning Algorithms

• Supervised learning
  – \( x \) = features of email
    • carefully choose 100 words indicative of spam/not spam
    • in practice, take most frequently occurring \( d \) words (10k-50k) in training set rather than manually pick 100 words
  – \( y \) = spam (1) or not spam (0)

• Learning algorithm
  – Bayesian logistic regression, implemented with gradient descent
    \[
    \max_{\theta} \sum_{i=1}^{n} \log p(y^{(i)}|x^{(i)}, \theta) - \lambda \|	heta\|^{2}
    \]
  – 20% test error, which is unacceptably high

• What should you do next?

Fixing Learning Algorithms

• Common approach: try improving algorithm in different ways...
Diagnosing Learning Algorithms

• Better approach
  – Run diagnostic to figure out problem
  – Fix problem

• Diagnostic
  – Defn: test that you can run to gain insight what is/is not working with learning algorithm, and gain guidance as to how best to improve its performance
  – Diagnostics can take time to implement, but doing so can be a very good use of your time

• Bayesian logistic regression’s test error is 20% (unacceptably high)
• Is it a bias problem or a variance problem?

Review: Bias vs Variance

\[
\begin{align*}
\text{Price} & \quad \text{Size} \\
\theta_0 + \theta_1 x & \quad \theta_0 + \theta_1 x + \theta_2 x^2 & \quad \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4
\end{align*}
\]

underfitting (high bias)  
correct fit  
overfitting (high variance)

structural error  
hypothesis space cannot model true relationship
⇒ more data does not help
⇒ need larger \( \mathcal{H} \)

balance

estimation (approximation) error  
hypothesis space can model true relationship BUT hard to identify correct model due to large \( |\mathcal{H}| \), small \( n \), or noise
⇒ reduce \( \mathcal{H} \)
⇒ add more data

Based on slide by Eric Eaton [example by Andrew Ng]
Bias vs Variance: Model Complexity

Bias vs Variance: Regularization
Learning Curves

Typical learning curve for high bias

Typical learning curve for high variance
Diagnostics Tell You What to Try Next

Spam classification through Bayesian logistic regression, implemented with gradient descent

- Try getting more training examples
- Try smaller set of features
- Try larger set of features
- Try changing features (email header vs email body features)
- Try decreasing $\lambda$
- Try increasing $\lambda$

Fixes high bias or high variance?

More on Diagnostics

- Quite often, you will need to come up with your own diagnostics to figure out what is happening in an algorithm

- Even if learning algorithm is worked well, you might also run diagnostics to make sure you understand what is going on. This is useful for
  - Understanding your application problem
    - If you are working on one important ML application for months/years, it is very valuable for you personally to get an intuitive understanding of what works and what does not 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 is an algorithm that works”, it is more interesting to say “Here is an algorithm that works because of component X, and here is my justification.”
Error Analysis

• Try to understand your sources of error -- good machine learning practice!

• Simple approach
  – Manually examine the examples (in cross validation set) that your algorithm made errors on
  – See if you spot any systematic trend in what type of examples it is making errors on

• Example
  – 500 example in CV set
  – Algorithm misclassifies 100 emails
  – Manually examine 100 errors, and categorize them based on
    • what type of email it is
      – pharma [12], replica/fake [4], steal passwords [53], other [31]
    • what cues (features) you think would have helped the algorithm classify them correctly
      – misspellings [5], unusual punctuation [32], unusual email routing [16], …

Error Analysis

Many applications combine 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 (output of) each component and see how error changes
- **Conclusion:**

<table>
<thead>
<tr>
<th>component</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>overall system</td>
<td>85.0%</td>
</tr>
<tr>
<td>preprocess (remove bkgd)</td>
<td>85.1%</td>
</tr>
<tr>
<td>face detection</td>
<td>91.0%</td>
</tr>
<tr>
<td>eyes segmentation</td>
<td>95.0%</td>
</tr>
<tr>
<td>nose segmentation</td>
<td>96.0%</td>
</tr>
<tr>
<td>mouth segmentation</td>
<td>97.0%</td>
</tr>
<tr>
<td>logistic regression</td>
<td>100%</td>
</tr>
</tbody>
</table>

Ablative Analysis

- **Error analysis** tries to explain difference between current performance and **perfect performance**
- **Ablative analysis** tries to explain difference between current performance and **some baseline (much poorer) performance**

- Example: Suppose you have built 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
- How much did each of these components really help?
### Ablative Analysis

- Simple logistic regression without any clever features gets 94% performance
  - Just what accounts for improvement from 94% to 99.9%?
- Ablative analysis: remove components from your system one at a time to see how it breaks (remove component X, obtain accuracy Y)

<table>
<thead>
<tr>
<th>component</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>overall system</td>
<td>99.9%</td>
</tr>
<tr>
<td>spelling correction</td>
<td>99.0%</td>
</tr>
<tr>
<td>sender host features</td>
<td>98.9%</td>
</tr>
<tr>
<td>email header features</td>
<td>98.9%</td>
</tr>
<tr>
<td>email text parser features</td>
<td>95.0%</td>
</tr>
<tr>
<td>Javascript parser</td>
<td>94.5%</td>
</tr>
<tr>
<td>features from images</td>
<td>94.0%</td>
</tr>
</tbody>
</table>

- Conclusion:

### Getting Started on a Learning Problem

**Approach #1: Careful Design**
- Spend a long time designing exactly the right features, collecting the right dataset, and designing the right algorithmic architecture
- Implement it, hope it works

**Benefits**
- Nicer, perhaps more scalable algorithms
- May come up with new, elegant learning algorithms and 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 is wrong with it, and fix errors

**Benefits**
- Will often get your application problem working more quickly, so faster time-to-market
Premature Statistical Optimization

- Very often, it is not clear what parts of 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

Data for Machine Learning

Task: Classify between confusible words.
{to, two, too}
{then, than}
For breakfast I ate _____ eggs.

“It’s not who has the best algorithm that wins.
It’s who has the most data.”
-- Banko and Brill 2001
Large Data Rationale

• Assume feature $x$ has sufficient information to predict $y$ accurately
  – Example: For breakfast I ate {to, two, too} eggs
  – Counterexample: Predict housing price from only size (feet$^2$) and no other features

• Useful test: Given input $x$, can human expert confidently predict $y$?

• Why large data?
  – Use learning algorithm with many parameters (e.g. logistic regression/linear regression with many features)
    • low bias algorithms, so $J_{\text{train}}(\theta)$ will be small
  – Use very large training set (unlikely to overfit)
    • low variance, so $J_{\text{train}}(\theta) \approx J_{\text{test}}(\theta)$
    • combined with above, $J_{\text{test}}(\theta)$ will be small

Summary

• Time spent coming up with diagnostics for learning algorithms is time well-spent
  – It is often up to your own ingenuity to come up with right diagnostics

• Error analyses and ablative analyses also give insight into 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