

Lecture 13 - Debugging ML Models and Error Analysis | Stanford CS229: Machine Learning (Autumn 2018)-Make Written Notes

Notebook: 100-Paper-Notes

Created: 01/06/2020 02:50

Updated: 01/06/2020 17:58

Author: mohammad.mohtashim76@gmail.com

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.

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:

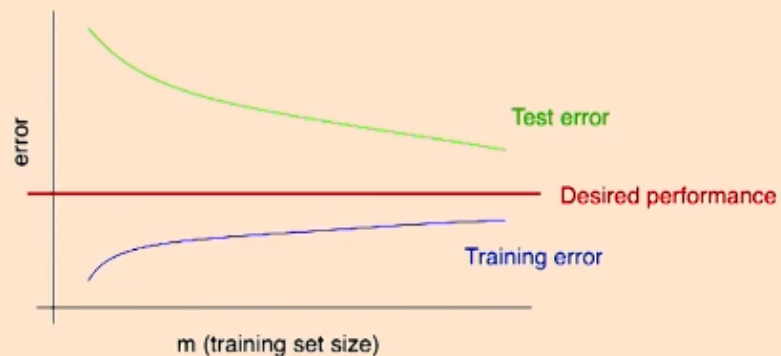
- Overfitting (high variance).
- Too few features to classify spam (high bias).

Diagnostic:

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

More on bias vs. variance

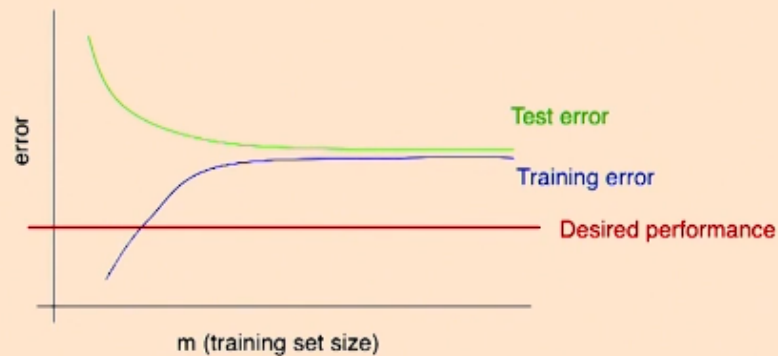
Typical learning curve for high variance:



- 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.

Stanford

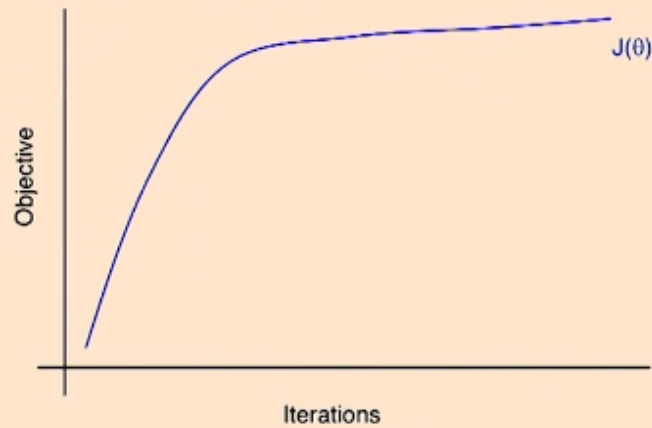
Andrew Y. Ng

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:
 - 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?



Stanford

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 ?

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

Stanford

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:

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

θ_{SVM} outperforms θ_{BLR} . So:

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

•

•

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:

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

θ_{SVM} outperforms θ_{BLR} . So:

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

BLR tries to maximize:

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

Diagnostic:

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

⊆

Stanford

Andrew Y. Ng

•

- Very Important-New Insight-Please See it Future Mohtashim

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 $J(\theta)$. 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 $J(\theta)$. But the SVM, which does worse on $J(\theta)$, actually does better on weighted accuracy $a(\theta)$.

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

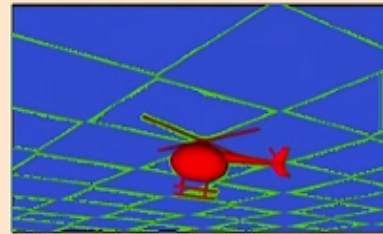
Stanford

The Stanford Autonomous Helicopter



Payload: 14 pounds
Weight: 32 pounds

Machine learning algorithm



Simulator

1. Build a simulator of helicopter.
2. Choose a cost function. Say $J(\theta) = \|x - x_{\text{desired}}\|^2$ (x = helicopter position)
3. Run reinforcement learning (RL) algorithm to fly helicopter in simulation, so as to try to minimize cost function:

$$\theta_{\text{RL}} = \arg \min_{\theta} J(\theta)$$

Suppose you do this, and the resulting controller parameters θ_{RL} gives much worse performance than your human pilot. What to do next?

Improve simulator?
Modify cost function J ?
Modify RL algorithm?

Stanford

Andrew Y. Ng

Debugging an RL algorithm

The controller given by θ_{RL} performs poorly.

Suppose that:

1. The helicopter simulator is accurate.
2. The RL algorithm correctly controls the helicopter (in simulation) so as to minimize $J(\theta)$.
3. Minimizing $J(\theta)$ corresponds to correct autonomous flight.

Then: The learned parameters θ_{RL} should fly well on the actual helicopter.

Diagnostics:

1. If θ_{RL} flies well in simulation, but not in real life, then the problem is in the simulator. Otherwise:
2. Let θ_{human} be the human control policy. If $J(\theta_{\text{human}}) < J(\theta_{\text{RL}})$, then the problem is in the reinforcement learning algorithm. (Failing to minimize the cost function J .)
3. If $J(\theta_{\text{human}}) \geq J(\theta_{\text{RL}})$, then the problem is in the cost function. (Maximizing it doesn't correspond to good autonomous flight.)

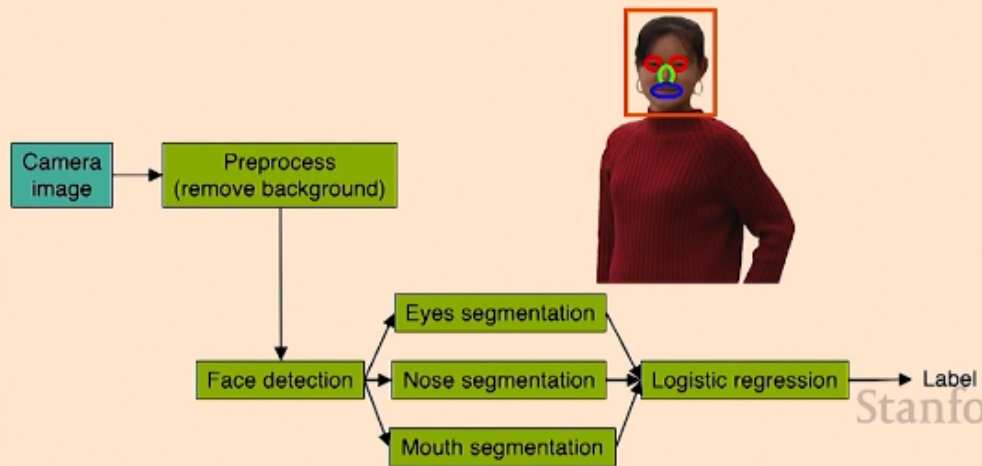
Stanford

Andrew Y. Ng

- Remember two things:
 - o Bias/Variance Trade-off
 - o Analyse whether the problem is in algorithm or objective that you are trying to achieve

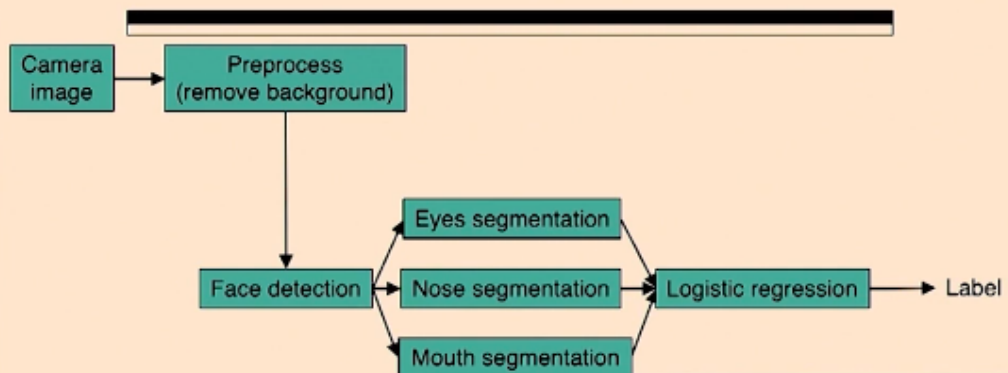
Error analysis

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



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

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.

Stanford

-
-
-