Improving your model performance

The two fundamental assumptions of supervised learning

There are 2 fundamental assumptions of supervised learning. The first one is to have a low avoidable bias which means that the training set fits well. The second one is to have a low or acceptable variance which means that the training set performance generalizes well to the development set and test set.

If the difference between human-level error and the training error is bigger than the difference between the training error and the development error, the focus should be on bias reduction technique which are training a bigger model, training longer or change the neural networks architecture or try various hyperparameters search.

If the difference between training error and the development error is bigger than the difference between the human-level error and the training error, the focus should be on variance reduction technique which are bigger data set, regularization or change the neural networks architecture or try various hyperparameters search.

Summary

Human-level

Avoidable bias

- Train bigger model
- Train longer, better optimization algorithms
- Neural Networks architecture/hyperparameters search

Training error

Variance

- More data
- Regularization
- Neural Networks architecture/hyperparameters search

Development error

Surpassing human-level performance

Example1: Classification task

	Classification error (%)	
	Scenario A	Scenario B
Team of humans	0.5	0.5
One human	1.0	1
Training error	0.6	0.3
Development error	0.8	0.4

Scenario A

In this case, the Bayes error is 0.5%, therefore the available bias is 0.1% et the variance is 0.2%.

Scenario B

In this case, there is not enough information to know if bias reduction or variance reduction has to be done on the algorithm. It doesn't mean that the model cannot be improve, it means that the conventional ways to know if bias reduction or variance reduction are not working in this case.

There are many problems where machine learning significantly surpasses human-level performance, especially with structured data:

- Online advertising
- Product recommendations
- Logistics (predicting transit time)
- Loan approvals

Understanding human-level performance

Human-level error gives an estimate of Bayes error.

Example 1: Medical image classification

This is an example of a medical image classification in which the input is a radiology image and the output is a diagnosis classification decision.

	Classification error (%)
Typical human	3.0
Typical doctor	1.0
Experienced doctor	0.7
Team of experienced doctors	0.5

The definition of human-level error depends on the purpose of the analysis, in this case, by definition the Bayes error is lower or equal to 0.5%.

Example 2: Error analysis

	Classification error (%)		
	Scenario A	Scenario B	Scenario C
Human (proxy for Bayes error)	1	1	0.5
	0.7	0.7	
	0.5	0.5	
Training error	5	1	0.7
Development error	6	5	0.8

Scenario A

In this case, the choice of human-level performance doesn't have an impact. The avoidable bias is between 4%-4.5% and the variance is 1%. Therefore, the focus should be on bias reduction technique.

Scenario B

In this case, the choice of human-level performance doesn't have an impact. The avoidable bias is between 0%-0.5% and the variance is 4%. Therefore, the focus should be on variance reduction technique.

Scenario C

In this case, the estimate for Bayes error has to be 0.5% since you can't go lower than the human-level performance otherwise the training set is overfitting. Also, the avoidable bias is 0.2% and the variance is 0.1%. Therefore, the focus should be on bias reduction technique.

Summary of bias/variance with human-level performance

- Human - level error – proxy for Bayes error
- If the difference between human-level error and the training error is bigger than the difference between the training error and the development error. The focus should be on bias reduction technique
- If the difference between training error and the development error is bigger than the difference between the human-level error and the training error. The focus should be on variance reduction technique

Avoidable bias

By knowing what the human-level performance is, it is possible to tell when a training set is performing well or not.

Example: Cat vs Non-Cat

	Classification error (%)	
	Scenario A	Scenario B
Humans	1	7.5
Training error	8	8
Development error	10	10

In this case, the human level error as a proxy for Bayes error since humans are good to identify images. If you want to improve the performance of the training set but you can't do better than the Bayes error otherwise the training set is overfitting. By knowing the Bayes error, it is easier to focus on whether bias or variance avoidance tactics will improve the performance of the model.

Scenario A

There is a 7% gap between the performance of the training set and the human level error. It means that the algorithm isn't fitting well with the training set since the target is around 1%. To resolve the issue, we use bias reduction technique such as training a bigger neural network or running the training set longer.

Scenario B

The training set is doing good since there is only a 0.5% difference with the human level error. The difference between the training set and the human level error is called avoidable bias. The focus here is to reduce the variance since the difference between the training error and the development error is 2%. To resolve the issue, we use variance reduction technique such as regularization or have a bigger training set.

Why human-level performance?

Today, machine learning algorithms can compete with human-level performance since they are more productive and more feasible in a lot of application. Also, the workflow of designing and building a machine learning system, is much more efficient than before.

Moreover, some of the tasks that humans do are close to ''perfection'', which is why machine learning tries to mimic human-level performance.

The graph below shows the performance of humans and machine learning over time.

The

Machine learning progresses slowly when it surpasses human-level performance. One of the reason is that human-level performance can be close to Bayes optimal error, especially for natural perception problem.

Bayes optimal error is defined as the best possible error. In other words, it means that any functions mapping from x to y can't surpass a certain level of accuracy.

Also, when the performance of machine learning is worse than the performance of humans, you can improve it with different tools. They are harder to use once its surpasses human-level performance.

These tools are:

- Get labeled data from humans
- Gain insight from manual error analysis: Why did a person get this right?
- Better analysis of bias/variance.

When to change development/test sets and metrics

Example: Cat vs Non-cat

A cat classifier tries to find a great amount of cat images to show to cat loving users. The evaluation metric used is a classification error.

Algorithm	Classification error [%]
A	3%
B	5%

It seems that Algorithm A is better than Algorithm B since there is only a 3% error, however for some reason, Algorithm A is letting through a lot of the pornographic images.

Algorithm B has 5% error thus it classifies fewer images but it doesn't have pornographic images. From a company's point of view, as well as from a user acceptance point of view, Algorithm B is actually a better algorithm. The evaluation metric fails to correctly rank order preferences between algorithms. The evaluation metric or the development set or test set should be changed.

The misclassification error metric can be written as a function as follow:

$$Error: \frac{1}{m_{dev}} \sum_{i=1}^{m_{dev}} \mathcal{L}\{(\hat{y}^{(i)} \neq y^{(i)}\}$$

This function counts up the number of misclassified examples.

The problem with this evaluation metric is that it treats pornographic vs non-pornographic images equally. On way to change this evaluation metric is to add the weight term $w^{(i)}$.

$$w^{(i)} = \begin{cases} 1 & if\ x^{(i)}\ is\ non-pornographic \\ 10 & if\ x^{(i)}\ is\ pornographic \end{cases}$$

The function becomes:

$$Error: \frac{1}{\sum w^{(i)}} \sum_{i=1}^{m_{dev}} w^{(i)} \mathcal{L}\{(\hat{y}^{(i)} \neq y^{(i)}\}$$

Guideline

1. Define correctly an evaluation metric that helps better rank order classifiers
2. Optimize the evaluation metric

Size of the development and test sets

Old way of splitting data

We had smaller data set therefore we had to use a greater percentage of data to develop and test ideas and models.

Or

Modern era – Big data

Now, because a large amount of data is available, we don't have to compromised as much and can use a greater portion to train the model.

Guidelines

- Set up the size of the test set to give a high confidence in the overall performance of the system.
- Test set helps evaluate the performance of the final classifier which could be less 30% of the whole data set.
- The development set has to be big enough to evaluate different ideas.

Training, development and test distributions

Setting up the training, development and test sets have a huge impact on productivity. It is important to choose the development and test sets from the same distribution and it must be taken randomly from all the data.

Guideline

Choose a development set and test set to reflect data you expect to get in the future and consider important to do well.

Satisficing and optimizing metric

There are different metrics to evaluate the performance of a classifier, they are called evaluation matrices. They can be categorized as satisficing and optimizing matrices. It is important to note that these evaluation matrices must be evaluated on a training set, a development set or on the test set.

Example: Cat vs Non-cat

Classifier	Accuracy	Running time
A	90%	80 ms
B	92%	95 ms
C	95%	1 500 ms

In this case, accuracy and running time are the evaluation matrices. Accuracy is the optimizing metric, because you want the classifier to correctly detect a cat image as accurately as possible. The running time which is set to be under 100 ms in this example, is the satisficing metric which mean that the metric has to meet expectation set.

The general rule is:

$$N_{metric}: \begin{cases} 1 & Optimizing\ metric \\ N_{metric} - 1 & Satisficing\ metric \end{cases}$$

Single number evaluation metric

To choose a classifier, a well-defined development set and an evaluation metric speed up the iteration process.

Example : Cat vs Non- cat

y = 1, cat image detected

Predict class $\hat{y}$	Actual class y: 1	Actual class y: 0
1	True positive	False positive
0	False negative	True negative

<u>Precision</u>

Of all the images we predicted y=1, what fraction of it have cats?

Precision (%) = $\frac{True\ positive}{Number\ of\ predicted\ positive} x\ 100$ = $\frac{True\ positive}{(True\ positive + False\ positive)}\ x\ 100$

<u>Recall</u>

Of all the images that actually have cats, what fraction of it did we correctly identifying have cats?

Recall (%) = $\frac{True\ positive}{Number\ of\ predicted\ actually\ positive} x\ 100$ = $\frac{True\ positive}{(True\ positive + False\ negative)}\ x\ 100$

Let's compare 2 classifiers A and B used to evaluate if there are cat images:

Classifier	Precision (p)	Recall (r)
A	95%	90%
B	98%	85%

In this case the evaluation metrics are precision and recall.

For classifier A, there is a 95% chance that there is a cat in the image and a 90% chance that it has correctly detected a cat. Whereas for classifier B there is a 98% chance that there is a cat in the image and a 85% chance that it has correctly detected a cat.

The problem with using precision/recall as the evaluation metric is that you are not sure which one is better since in this case, both of them have a good precision et recall. F1-score, a harmonic mean, combine both precision and recall.

$$\text{F1-Score} = \frac{2}{\frac{1}{p} + \frac{1}{r}}$$

Classifier	Precision (p)	Recall (r)	F1-Score
A	95%	90%	92.4 %
B	98%	85%	91.0%

Classifier A is a better choice. F1-Score is not the only evaluation metric that can be use, the average, for example, could also be an indicator of which classifier to use.

Orthogonalization

Orthogonalization or orthogonality is a system design property that assures that modifying an instruction or a component of an algorithm will not create or propagate side effects to other components of the system. It becomes easier to verify the algorithms independently from one another, it reduces testing and development time.

When a supervised learning system is design, these are the 4 assumptions that needs to be true and orthogonal.

1. Fit training set well in cost function
   - If it doesn't fit well, the use of a bigger neural network or switching to a better optimization algorithm might help.
2. Fit development set well on cost function
   - If it doesn't fit well, regularization or using bigger training set might help.
3. Fit test set well on cost function
   - If it doesn't fit well, the use of a bigger development set might help
4. Performs well in real world
   - If it doesn't perform well, the development test set is not set correctly or the cost function is not evaluating the right thing.

Copyright Notice

These slides are distributed under the Creative Commons License.

DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode

deeplearning.ai

Introduction to ML strategy

Why ML Strategy?

Motivating example

90%

Ideas:

- Collect more data
- Collect more diverse training set
- Train algorithm longer with gradient descent
- Try Adam instead of gradient descent
- Try bigger network
- Try smaller network
- Try dropout
- Add L_2 regularization
- Network architecture
  - Activation functions
  - # hidden units
  - …

Andrew Ng

deeplearning.ai

Introduction to ML strategy

Orthogonalization

TV tuning example

Orthogonalization

Car

→ Steering

→ {Acceleration, Braking}

→ 0.3 x angle − 0.8 speed

→ 2 x angle + 0.9 speed.

Andrew Ng

Chain of assumptions in ML

→ Fit training set well on cost function

(≈ human-level performance)

bigger network
Adam
...

width

early stopping

→ Fit dev set well on cost function

Regularization
Bigger training set

height

→ Fit test set well on cost function

Bigger dev set

→ Performs well in real world

(Happy cat pic app users.)

Change dev set or cost function

Andrew Ng

deeplearning.ai

Setting up your goal

Single number evaluation metric

Using a single number evaluation metric

Of examples recognized as cat, what % actually are cats?

what % of actual cats are correctly recognized

Classifier	Precision	Recall
A	95%	90%
B	98%	85%

F_1 score = "Average" of P and R.

$\left(\frac{2}{\frac{1}{P} + \frac{1}{R}} \right.$. "Harmonic mean")

Dev set + Single number evaluation metric

real

speed up iterating

Andrew Ng

Another example

Algorithm	US	China	India	Other
A	3%	7%	5%	9%
B	5%	6%	5%	10%
C	2%	3%	4%	5%
D	5%	8%	7%	2%
E	4%	5%	2%	4%
F	7%	11%	8%	12%

deeplearning.ai

Setting up your goal

Satisficing and optimizing metrics

Another cat classification example

optimizing satisficing

Classifier	Accuracy	Running time
A	90%	80ms
B	92%	95ms
C	95%	1,500ms

Cost = accuracy − 0.5 × running Time

maximize accuracy

subject to running Time ≤ 100 ms.

N metrics: 1 optimizing

N−1 satisficing

Wakewords / Trigger words

Alexa, OK Google,

Hey Siri, nihaobaidu

你好百度

accuracy.

#false positive

maximize accuracy.

s.t. ≤ 1 false positive

every 24 hours.

Andrew Ng

deeplearning.ai

Setting up your goal

Train/dev/test distributions

Cat classification dev/test sets

development set, hold out cross validation set

Regions:

- US
- UK
- Other Europe
- South America
- India
- China
- Other Asia
- Australia

Dev

Test

Randomly shuffle into dev/test

dev set
+
metric

Andrew Ng

True story (details changed)

Optimizing on dev set on loan approvals for medium income zip codes

$x \longrightarrow y$ (repay loan?)

Tested on low income zip codes

~3 month

Andrew Ng

Guideline

Same distribution

Choose a dev set and test set to reflect data you expect to get in the future and consider important to do well on.

training

dev

metric

test

Andrew Ng

deeplearning.ai

Setting up your goal

Size of dev and test sets

Old way of splitting data

Andrew Ng

Size of dev set

A B

Set your dev set to be big enough to detect differences in algorithm/models you're trying out.

A B

97% → 97.1%

0.1%

100 : small

1%

1,000

10,000

100,000

0.01%

0.001%

Online advertising

Andrew Ng

Size of test set

- Set your test set to be big enough to give high confidence in the overall performance of your system.

10,000

100,000

Train + Dev.

No test set.

deeplearning.ai

Setting up your goal

When to change dev/test sets and metrics

Cat dataset examples

Metric + Dev : Prefer A
You/users : Prefer B.

→ Metric: classification error

Algorithm A: 3% error → pornographic

✓ Algorithm B: 5% error

Error: $\frac{1}{\sum_i w^{(i)}} \cancel{\frac{1}{m_{dev}}} \sum_{i=1}^{m_{dev}} w^{(i)} \mathbb{1}\{y_{pred}^{(i)} \neq y^{(i)}\}$

predicted value (0/1)

→ $w^{(i)} = \begin{cases} 1 & \text{if } x^{(i)} \text{ is non-porn} \\ 10 & \text{if } x^{(i)} \text{ is porn} \end{cases}$

Andrew Ng

Orthogonalization for cat pictures: anti-porn

1. So far we've only discussed how to define a metric to evaluate classifiers. ← Place target

2. Worry separately about how to do well on this metric.

↖ Aim/shoot at target

$$J = \frac{1}{\cancel{m}} \sum_{i=1}^{m} w^{(i)} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

$\sum w^{(i)}$

Andrew Ng

Another example

Algorithm A: 3% error

Algorithm B: 5% error

Dev/test

User images

If doing well on your metric + dev/test set does not correspond to doing well on your application, change your metric and/or dev/test set.

Andrew Ng

deeplearning.ai

Comparing to human-level performance

Why human-level performance?

Comparing to human-level performance

Andrew Ng

Why compare to human-level performance

Humans are quite good at a lot of tasks. So long as ML is worse than humans, you can:

- Get labeled data from humans. (x, y)
- Gain insight from manual error analysis: Why did a person get this right?
- Better analysis of bias/variance.

deeplearning.ai

Comparing to human-level performance

Avoidable bias

Bias and Variance

high bias

underfitting

"just right"

high variance

overfitting.

Andrew Ng

Bias and Variance

Cat classification

Human-level ≈ 0%

Training set error: 1% 15% 15% 0.5%

Dev set error: 11% 16% 30% 1%

high variance | high bias | high bias, high variance | low bias, low variance

Andrew Ng

Cat classification example

Humans (≈ Bayes)	1%	7.5%
Training error	8%	8%
Dev error	10%	10%

7%

2%

0.5%

Avoidable bias

2%

Variance

2%

Variance

Focus on bias

Focus on variance

Human-level error as a proxy for Bayes error.

Andrew Ng

deeplearning.ai

Comparing to human-level performance

Understanding human-level performance

Human-level error as a proxy for Bayes error

Medical image classification example:

Suppose:

(a) Typical human ..... 3 % error

(b) Typical doctor ..... 1 % error

(c) Experienced doctor ..... 0.7 % error

(d) Team of experienced doctors .. 0.5 % error

Bayes error $\leq$ 0.5%

What is "human-level" error?

Andrew Ng

Error analysis example

Human (proxy for Bayes error)

Avoidable bias

Training error

Variance

Dev error

Andrew Ng

Summary of bias/variance with human-level performance

0%

"Bias"

Human-level error

(proxy for Bayes error)

"Avoidable bias"

Training error

"Variance"

Dev error

Andrew Ng

deeplearning.ai

Comparing to human-level performance

Surpassing human-level performance

Surpassing human-level performance

Team of humans	0.5%
One human	~~1%~~
Training error	0.6%
Dev error	0.8%

0.1

0.2

0.5%

0.1%

0.2%

0.3%

1%

0.3%

0.4%

What is avoidable bias?

Andrew Ng

Problems where ML significantly surpasses human-level performance

- Online advertising
- Product recommendations
- Logistics (predicting transit time)
- Loan approvals

Structural data

Not natural perception

lots of data

- Speech recognition
- Some image recognition
- Medical
  - ECG, Skin cancer, ...

Andrew Ng

deeplearning.ai

Comparing to human-level performance

Improving your model performance

The two fundamental assumptions of supervised learning

1. You can fit the training set pretty well.

   ~ Avoidable bias

2. The training set performance generalizes pretty well to the dev/test set.

   ~ Variance

Reducing (avoidable) bias and variance

Human-level

Avoidable bias

Training error

Variance

Dev error

Train bigger model

Train longer/better optimization algorithms

- momentum, RMSprop, Adam

NN architecture/hyperparameters search

RNN

CNN

More data

Regularization

- L_2, dropout, data augmentation

NN architecture/hyperparameters search

Andrew Ng

Whether to use end-to-end deep learning

Before applying end-to-end deep learning, you need to ask yourself the following question: Do you have enough data to learn a function of the complexity needed to map x and y?

Pro:

- Let the data speak
  - By having a pure machine learning approach, the neural network will learn from x to y. It will be able to find which statistics are in the data, rather than being forced to reflect human preconceptions.

- Less hand-designing of components needed
  - It simplifies the design work flow.

Cons:

- Large amount of labeled data
  - It cannot be used for every problem as it needs a lot of labeled data.

- Excludes potentially useful hand-designed component
  - Data and any hand-design's components or features are the 2 main sources of knowledge for a learning algorithm. If the data set is small than a hand-design system is a way to give manual knowledge into the algorithm.

What is end-to-end deep learning

End-to-end deep learning is the simplification of a processing or learning systems into one neural network.

Example - Speech recognition model

The traditional way - small data set

The hybrid way - medium data set

The End-to-End deep learning way – large data set

End-to-end deep learning cannot be used for every problem since it needs a lot of labeled data. It is used mainly in audio transcripts, image captures, image synthesis, machine translation, steering in self-driving cars, etc.

Multi-task learning

Multi-task learning refers to having one neural network do simultaneously several tasks.

When to use multi-task learning

- Training on a set of tasks that could benefit from having shared lower-level features
- Usually: Amount of data you have for each task is quite similar
- Can train a big enough neural network to do well on all tasks

Example: Simplified autonomous vehicle

The vehicle has to detect simultaneously several things: pedestrians, cars, road signs, traffic lights, cyclists, etc. We could have trained four separate neural networks, instead of train one to do four tasks. However, in this case, the performance of the system is better when one neural network is trained to do four tasks than training four separate neural networks since some of the earlier features in the neural network could be shared between the different types of objects.

The input $x^{(i)}$ is the image with multiple labels

The output $y^{(i)}$ has 4 labels which are represents:

$$y^{(i)} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \begin{array}{l} \text{Pedestrians} \\ \text{Cars} \\ \text{Road signs - Stop} \\ \text{Traffic lights} \end{array}$$

$$Y = \begin{bmatrix} | & | & | & | \\ y^{(1)} & y^{(2)} & y^{(3)} & y^{(4)} \\ | & | & | & | \end{bmatrix} \qquad \begin{array}{l} Y = (4, m) \\ Y = (4,1) \end{array}$$

Neural Network architecture

To train this neural network, loss function is defined as follow:

$$-\frac{1}{m}\sum_{i=1}^{m}\boxed{\sum_{j=1}^{4}}\left(y_j^{(i)}\log\left(\hat{y}_j^{(i)}\right) + \left(1 - y_j^{(i)}\right)\log\left(1 - \hat{y}_j^{(i)}\right)\right)$$

Also, the cost can be compute such as it is not influenced by the fact that some entries are not labeled. Example:

$$Y = \begin{bmatrix} 1 & 0 & ? & ? \\ 0 & 1 & ? & 0 \\ 0 & 1 & ? & 1 \\ ? & 0 & 1 & 0 \end{bmatrix}$$

Transfer Learning

Transfer learning refers to using the neural network knowledge for another application.

When to use transfer learning

- Task A and B have the same input x
- A lot more data for Task A than Task B
- Low level features from Task A could be helpful for Task B

Example 1: Cat recognition - radiology diagnosis

The following neural network is trained for cat recognition, but we want to adapt it for radiology diagnosis. The neural network will learn about the structure and the nature of images. This initial phase of training on image recognition is called pre-training, since it will pre-initialize the weights of the neural network. Updating all the weights afterwards is called fine-tuning.

For cat recognition

Input x: image

Output y – 1: cat, 0: no cat

Radiology diagnosis

Input x: Radiology images – CT Scan, X-rays

Output y :Radiology diagnosis – 1: tumor malign, 0: tumor benign

Guideline

- Delete last layer of neural network
- Delete weights feeding into the last output layer of the neural network
- Create a new set of randomly initialized weights for the last layer only
- New data set (x, y)

Addressing data mismatch

This is a general guideline to address data mismatch:

- Perform manual error analysis to understand the error differences between training, development/test sets. Development should never be done on test set to avoid overfitting.

- Make training data or collect data similar to development and test sets. To make the training data more similar to your development set, you can use is artificial data synthesis. However, it is possible that if you might be accidentally simulating data only from a tiny subset of the space of all possible examples.

Bias and variance with mismatched data distributions

Example: Cat classifier with mismatch data distribution

When the training set is from a different distribution than the development and test sets, the method to analyze bias and variance changes.

	Classification error (%)					
	Scenario A	Scenario B	Scenario C	Scenario D	Scenario E	Scenario F
Human (proxy for Bayes error)	0	0	0	0	0	4
Training error	1	1	1	10	10	7
Training-development error	-	9	1.5	11	11	10
Development error	10	10	10	12	20	6
Test error	-	-	-	-	-	6

Scenario A

If the development data comes from the same distribution as the training set, then there is a large variance problem and the algorithm is not generalizing well from the training set.

However, since the training data and the development data come from a different distribution, this conclusion cannot be drawn. There isn't necessarily a variance problem. The problem might be that the development set contains images that are more difficult to classify accurately.

When the training set, development and test sets distributions are different, two things change at the same time. First of all, the algorithm trained in the training set but not in the development set. Second of all, the distribution of data in the development set is different.

It's difficult to know which of these two changes what produces this 9% increase in error between the training set and the development set. To resolve this issue, we define a new subset called training-development set. This new subset has the same distribution as the training set, but it is not used for training the neural network.

Scenario B

The error between the training set and the training- development set is 8%. In this case, since the training set and training-development set come from the same distribution, the only difference between them is the neural network sorted the data in the training and not in the training development. The neural network is not generalizing well to data from the same distribution that it hadn't seen before

Therefore, we have really a variance problem.

Scenario C

In this case, we have a mismatch data problem since the 2 data sets come from different distribution.

Scenario D

In this case, the avoidable bias is high since the difference between Bayes error and training error is 10 %.

Scenario E

In this case, there are 2 problems. The first one is that the avoidable bias is high since the difference between Bayes error and training error is 10 % and the second one is a data mismatched problem.

Scenario F

Development should never be done on the test set. However, the difference between the development set and the test set gives the degree of overfitting to the development set.

General formulation

Bayes error

↕ Avoidable Bias

Training set error

↕ Variance

Development - Training set error

↕ Data mismatch

Development set error

↕ Degree of overfitting to the development set

Test set error

Training and testing on different distributions

Example: Cat vs Non-cat

In this example, we want to create a mobile application that will classify and recognize pictures of cats taken and uploaded by users.

There are two sources of data used to develop the mobile app. The first data distribution is small, 10 000 pictures uploaded from the mobile application. Since they are from amateur users, the pictures are not professionally shot, not well framed and blurrier. The second source is from the web, you downloaded 200 000 pictures where cat's pictures are professionally framed and in high resolution.

The problem is that you have a different distribution:

1- small data set from pictures uploaded by users. This distribution is important for the mobile app.
2- bigger data set from the web.

The guideline used is that you have to choose a development set and test set to reflect data you expect to get in the future and consider important to do well.

The data is split as follow:

The advantage of this way of splitting up is that the target is well defined.

The disadvantage is that the training distribution is different from the development and test set distributions. However, this way of splitting the data has a better performance in long term.

Build system quickly, then iterate

Depending on the area of application, the guideline below will help you prioritize when you build your system.

Guideline

1. Set up development/ test set and metrics
   - Set up a target
2. Build an initial system quickly
   - Train training set quickly: Fit the parameters
   - Development set: Tune the parameters
   - Test set: Assess the performance
3. Use Bias/Variance analysis & Error analysis to prioritize next steps

Copyright Notice

These slides are distributed under the Creative Commons License.

DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode

deeplearning.ai

Error Analysis

Carrying out error analysis

Look at dev examples to evaluate ideas

90% accuracy
→ 10% error

Should you try to make your cat classifier do better on dogs?

Error analysis: → 5-10 min

- Get ~100 mislabeled dev set examples.
- Count up how many are dogs.

"ceiling"

→ 5%
5/100

10%
↓
9.5%

→ 50%
50/100

10%
↓
5%

Andrew Ng

Evaluate multiple ideas in parallel

Ideas for cat detection:

- Fix pictures of dogs being recognized as cats
- Fix great cats (lions, panthers, etc..) being misrecognized
- Improve performance on blurry images

Image	Dog	Great Cats	Blurry	Instagram	Comments
1	✓			✓	Pitbull
2			✓	✓	
3		✓	✓		Rainy day at zoo
⋮	⋮	⋮	⋮		
% of total	8%	43%	61%	12%	

Andrew Ng

deeplearning.ai

Error Analysis

Cleaning up Incorrectly labeled data

Incorrectly labeled examples

x							
y	1	0	1	1	0	1	1

Training set.

DL algorithms are quite robust to random errors in the training set.

Systematic errors

Andrew Ng

Error analysis

Image	Dog	Great Cat	Blurry	Incorrectly labeled	Comments
...					
98				✓	Labeler missed cat in background
99		✓			
100				✓	Drawing of a cat; Not a real cat.
% of total	8%	43%	61%	6%	

Overall dev set error 10% 2%

Errors due incorrect labels 0.6% 0.6%

Errors due to other causes 9.4% 1.4%

2.1% 1.9%

Goal of dev set is to help you select between two classifiers A & B.

Andrew Ng

Correcting incorrect dev/test set examples

- Apply same process to your dev and test sets to make sure they continue to come from the same distribution
- Consider examining examples your algorithm got 98.1% right as well as ones it got wrong. 2%
- Train and dev/test data may now come from slightly different distributions.

Andrew Ng

deeplearning.ai

Error Analysis

Build your first system quickly, then iterate

Speech recognition example

- Noisy background
  - Café noise
  - Car noise
- Accent
- Far fro
- Young
- Stutter
- …

Guideline:

Build your first system quickly, then iterate

- Set up dev/test set and metric
- Build initial system quickly
- Use Bias/Variance analysis & Error analysis to prioritize next steps.

Andrew Ng

deeplearning.ai

Mismatched training and dev/test data

Training and testing on different distributions

Cat app example

care about this

Data from webpages

→ ≈200,000

Data from mobile app

→ ≈10,000

210,000

(shuffle

Option 1:

205,000

200K / 210K

2381 - web

119 - mobile app

Option 2:

train: 205,000

Andrew Ng

Speech recognition example

Speech activated rearview mirror

Training

- Purchased data x, y
- Smart speaker control
- Voice keyboard
- …

500,000 utterances

Dev/test

Speech activated rearview mirror

→ 20,000

10K 5K 5K

Andrew Ng

deeplearning.ai

Mismatched training and dev/test data

Bias and Variance with mismatched data distributions

Cat classifier example

Assume humans get ≈ 0% error.

Training error 1%

Dev error 10%

9%

Training-dev set: Same distribution as training set, but not used for training

Training error	1%	1%
→ Training-dev error	9%	1.5%
→ Dev error	10%	10%
	Variance	Data mismatch

Human error	0%	Avoidable bias
Training error	10%	Avoidable bias
Training-dev error	11%	Variance
Dev error	12%	Data mismatch
	Bias	Bias + Data mismatch

Andrew Ng

Bias/variance on mismatched training and dev/test sets

Human level	4%	avoidable bias	4%
Training set error	7%	variance	7%
Training-dev set error	10%	data mismatch	10%
→ Dev error	12%	degree of overfitting to dev set.	6%
→ Test error	12%		6%

Andrew Ng

More general formulation

Rearview mirror

	General speech recognition	Rearview mirror speech data.
Human level	"Human level" 4%	6%
Error on examples trained on	"Training error" 7%	6%
Error on examples not trained on	"Training-dev error" 10%	"Dev/Test error" 6%

Avoidable bias

Variance

data mismatch

Andrew Ng

deeplearning.ai

Mismatched training and dev/test data

Addressing data mismatch

Addressing data mismatch

- Carry out manual error analysis to try to understand difference between training and dev/test sets

E.g. noisy – car noise street numbers

- Make training data more similar; or collect more data similar to dev/test sets

E.g. Simulate noisy in-car data

Artificial data synthesis

+ =

"The quick brown fox jumps over the lazy dog."

Car noise

Synthesized in-car audio

10,000 hours

1 hour of car noise

Overfit to 1 hour of car noise

10,000 hours

Andrew Ng

Artificial data synthesis

Car recognition:

≈ 20 cars

Andrew Ng

deeplearning.ai

Learning from multiple tasks

Transfer learning

Transfer learning

Andrew Ng

When transfer learning makes sense

Transfer from A → B

- Task A and B have the same input x.
- You have a lot more data for Task A than Task B.
- Low level features from A could be helpful for learning B.

deeplearning.ai

Learning from multiple tasks

Multi-task learning

Simplified autonomous driving example

$x^{(i)}$

$y^{(i)}$ $(4,1)$

Pedestrians	0
Cars	1
Stop signs	1
Traffic lights	0
⋮	⋮

$$Y = \begin{bmatrix} | & | & | & & | \\ y^{(1)} & y^{(2)} & y^{(3)} & \cdots & y^{(m)} \\ | & | & | & & | \end{bmatrix}$$

$(4, m)$

Andrew Ng

Neural network architecture

Loss: $\underset{(4,1)}{\hat{y}^{(i)}}$ $\rightarrow \frac{1}{m}\sum_{i=1}^{m}\boxed{\sum_{j=1}^{4}} \mathcal{L}(\hat{y}_j^{(i)}, y_j^{(i)})$

Sum only over value of j with 0/1 label.

→ Usual logistic loss

$-y_j^{(i)} \log \hat{y}_j^{(i)} - (1-y_j^{(i)}) \log (1-\hat{y}_j^{(i)})$

Unlike softmax regression:

One image can have multiple labels

Multi-task learning ←

$$Y = \begin{bmatrix} 1 & \boxed{1} & & \boxed{0} & & ? & \\ 0 & \boxed{1} & \cdots & \boxed{1} & \cdots & 1 & \cdots \\ \boxed{?} & ? & & \boxed{1} & & ? & \\ ? & ? & & \boxed{0} & & ? & \\ \vdots & \vdots & & & & \vdots & \end{bmatrix} \leftarrow$$

Andrew Ng

When multi-task learning makes sense

- Training on a set of tasks that could benefit from having shared lower-level features.
- Usually: Amount of data you have for each task is quite similar.

- Can train a big enough neural network to do well on all the tasks.

Andrew Ng

deeplearning.ai

End-to-end deep learning

What is end-to-end deep learning

What is end-to-end learning?

Speech recognition example

Andrew Ng

Face recognition

[Image courtesy of Baidu]

Have data for each of 2 subtasks.

Andrew Ng

More examples

Machine translation

(x, y)

English French

English → text analysis → ... → French

English → French

Estimating child's age:

Image → bones → age

Image → age

deeplearning.ai

End-to-end deep learning

Whether to use end-to-end learning

Pros and cons of end-to-end deep learning

Pros:

- Let the data speak

$x \longrightarrow y$

→ "phonemes"
c a t

- Less hand-designing of components needed

$x \longrightarrow y$

Cons:

- May need large amount of data

input end

output end

$x \longrightarrow y$

(x, y)

- Excludes potentially useful hand-designed components

Data.

Hand-design.

Andrew Ng

Applying end-to-end deep learning

Key question: Do you have sufficient data to learn a function of the complexity needed to map x to y?

$X \to y$

→ age

image, radar, lidar → (DL) cars, (DL) pedestrians → (motion planning) route → (control) steering

- Use DL to learn individual components
- Carefully choose $X \to Y$ depending what tasks you can get data for.

→ image → steering

Andrew Ng