Deep learning

Dr. Aissa Boulmerka
a.boulmerka@centre-univ-mila.dz

2023-2024

CHAPTER 7
MACHINE LEARNING STRATEGY

Why ML Strategy

- You have a lot of ideas for how to improve the accuracy of your deep learning system:
  - Collect more data.
  - Collect more diverse training set.
  - Train algorithm longer with gradient descent.
  - Try different optimization algorithm (e.g. Adam).
  - Try bigger network.
  - Try smaller network.
  - Try dropout.
  - Add L2 regularization.
  - Change network architecture (activation functions, # of hidden units, etc.)
- This course will give you some strategies to help analyze your problem to go in a direction that will help you get better results.

Orthogonalization

- Some deep learning developers know exactly what hyperparameter to tune in order to try to achieve one effect. This is a process we call orthogonalization.
- In orthogonalization, you have some controls, but each control does a specific task and doesn't affect other controls.
- For a supervised learning system to do well, you usually need to tune the knobs of your system to make sure that four things hold true:

Chain of assumptions in machine learning

Chain of assumptions in machine learning:

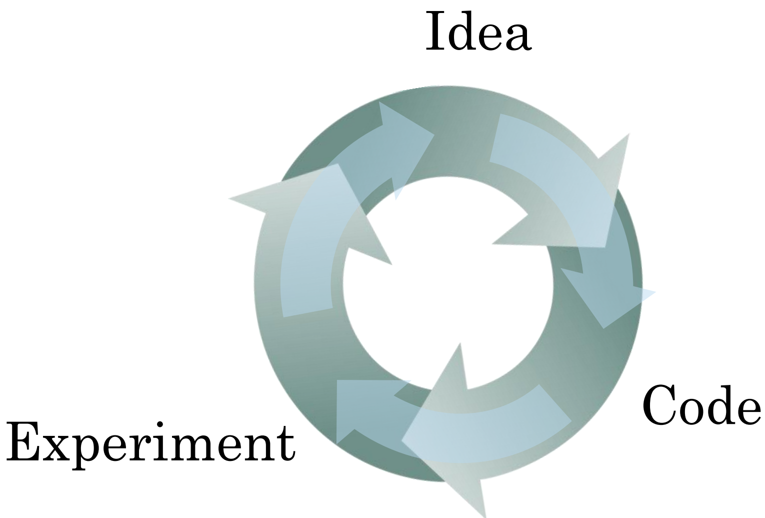
i. **Fit training set well on cost function (near human level performance if possible).**
   - If it's not achieved you could try bigger network, another optimization algorithm (like Adam)...

ii. **Fit dev set well on cost function.**
   - If its not achieved you could try regularization, bigger training set...

iii. **Fit test set well on cost function.**
   - If its not achieved you could try bigger dev. set...

iv. **Performs well in real world.**
   - If its not achieved you could try change dev. set, change cost function...

Single number evaluation metric



Classifier	Precision	Recall	F1 Score
A	95%	90%	92.4%
B	98%	85%	91.0%

Another example

Algorithm	US	China	India	Other	Average
A	3%	7%	5%	9%	6%
B	5%	6%	5%	10%	6.5%
C	2%	3%	4%	5%	3.5%
D	5%	8%	7%	2%	5.25%
E	4%	5%	2%	4%	3.75%
F	7%	11%	8%	12%	9.5%

Satisficing and optimizing metrics

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

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

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.

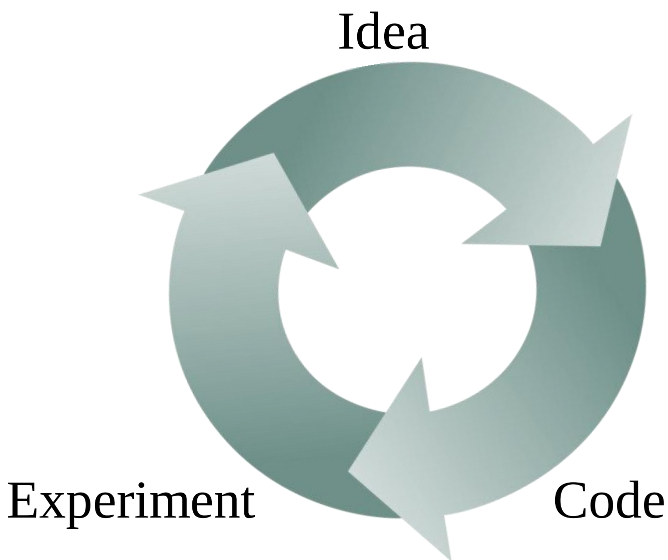
The general rule is:

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

Train/dev/test distributions

Regions:

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



Train/dev/test distributions

- Dev and test sets have to come from the same distribution.
- Choose dev set and test set to reflect data you expect to get in the future and consider important to do well on.
- Setting up the dev set, as well as the validation metric is really defining what target you want to aim at.

Size of the dev and test sets

- An old way of splitting the data was 70% training, 30% test or 60% training, 20% dev, 20% test.
- The old way was valid for a number of examples ~ <100000
- In the modern deep learning if you have a million or more examples a reasonable split would be 98% training, 1% dev, 1% test.

- Set your dev set to be big enough to detect differences in algorithm/models you’re trying out.
- Set your test set to be big enough to give high confidence in the overall performance of your system.

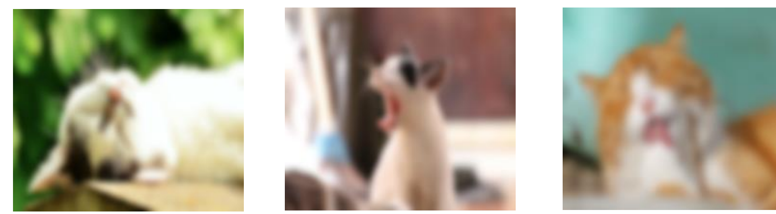
example

Algorithm A: 3% error

Algorithm B: 5% error

Dev/test

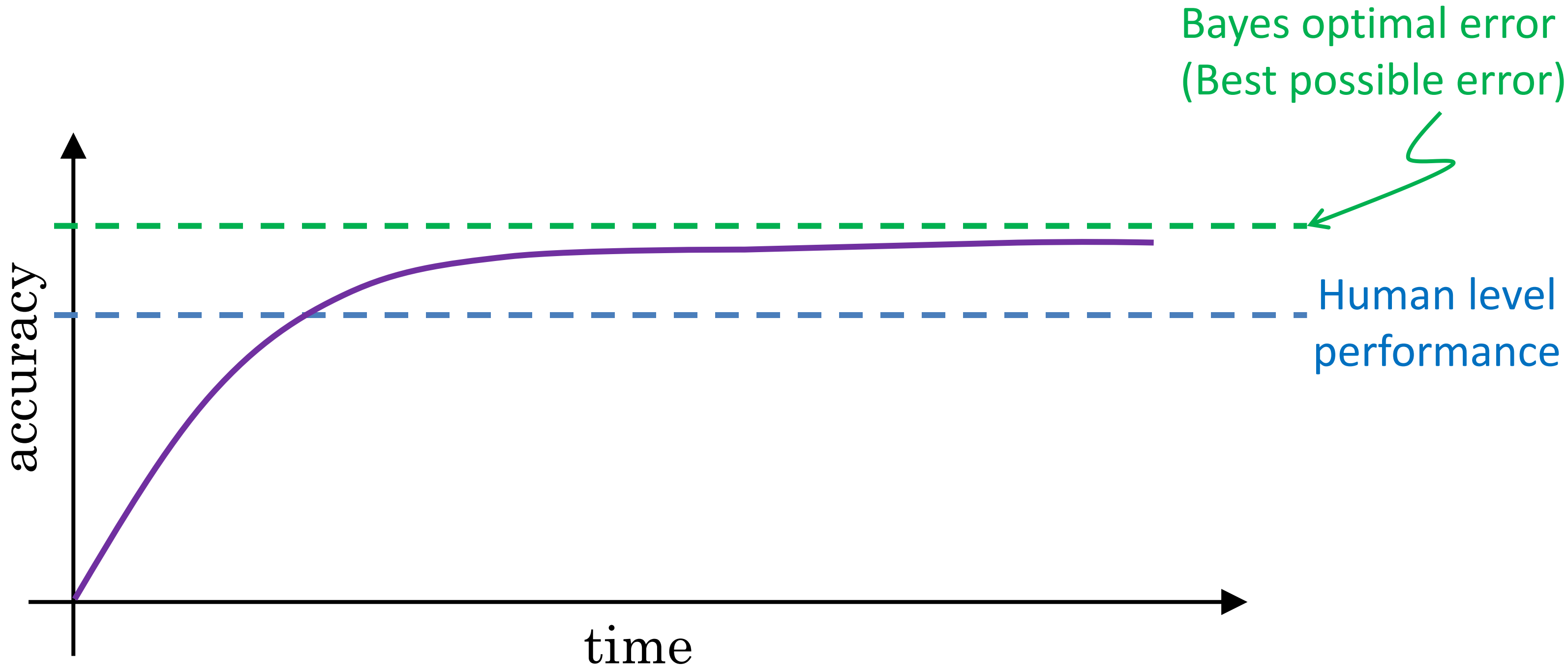
User images



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.

HUMAN-LEVEL PERFORMANCE

Comparing to human-level performance



Why compare to human-level performance

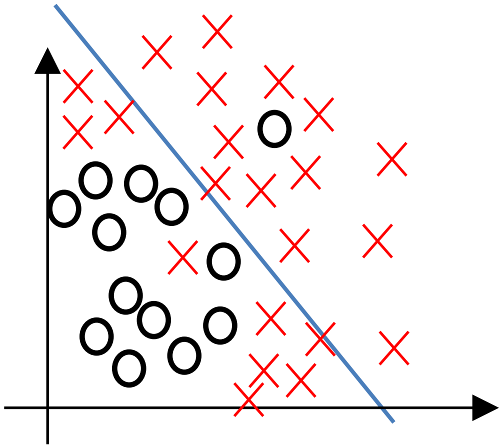
- We compare to human-level performance because of two main reasons:
  1. Because of advances in deep learning, machine learning algorithms are suddenly working much better and so it has become much more feasible in a lot of application areas for machine learning algorithms to actually become competitive with human-level performance.
  2. The workflow of designing and building a machine learning system is much more efficient when you're trying to do something that humans can also do.

- After an algorithm reaches the human level performance the progress and accuracy slow down.

Bayes optimal error

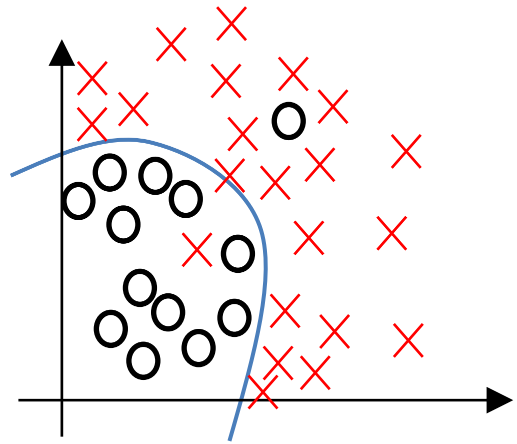
- You won't surpass an error that's called "Bayes optimal error".
- There isn't much error range between human-level error and Bayes optimal error.
- Humans are quite good at a lot of tasks. So as long as Machine learning is worse than humans, you can:
  - Get labeled data from humans.
  - Gain insight from manual error analysis: why did a person get it right?
  - Better analysis of bias/variance.

Bias and Variance



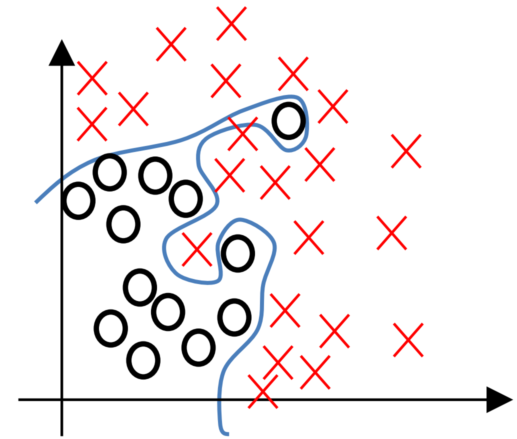
high bias

Underfitting



"just right"

Appropriate

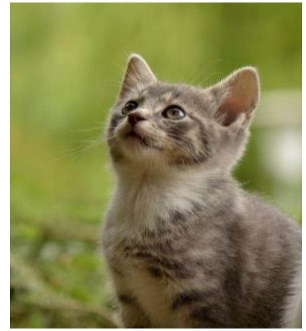


high variance

Overfitting

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

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

What is "human-level" error?

Error analysis example

Human (Proxy for Bayes error)	1% 0.7% 0.5%	1% 0.7% 0.5%
Avoidable bias		
Training error	5%	1%
Variance		
Dev error	6%	5%

Problems where ML significantly surpasses human-level performance

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

Some characteristics of these problems:

- Structural data
- Not natural perception
- Lots of data

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

Avoidable bias:

- Train bigger model
- Train longer/better optimization algorithms (Momentum, RMSprop, Adam,…)
- NN architecture/hyperparameters search (CNN, RNN,…)

Variance:

- More data
- Regularization (L2, dropout, data augmentation,…)
- NN architecture/hyperparameters search

References

- Andrew Ng. Deep learning. Coursera.
- Geoffrey Hinton. Neural Networks for Machine Learning.
- Kevin P. Murphy. Probabilistic Machine Learning An Introduction. MIT Press, 2022.
- MIT Deep Learning 6.S191 (http://introtodeeplearning.com/)