INTRODUCTION TO DATA SCIENCE

JOHN P DICKERSON

Lecture #26 – 5/2/2017

CMSC320
Tuesdays & Thursdays
3:30pm – 4:45pm

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

ANNOUNCEMENTS

HW4 is due on Thursday (5/4)

- ~1/3 turned in already!
- Fill out the group assignment Google Drive document: https://docs.google.com/spreadsheets/d/1gSNYreIK9WSnfEKL0CEc-El9e_L3xDoqn7mECAJHnZA/edit?usp=sharing

Tuesday, May 9th will be a midterm & general wrap-up questions lecture day by Denis Peskov

- (Worked as a data scientist before coming to UMD.)

EXAM GRADES

N: 83

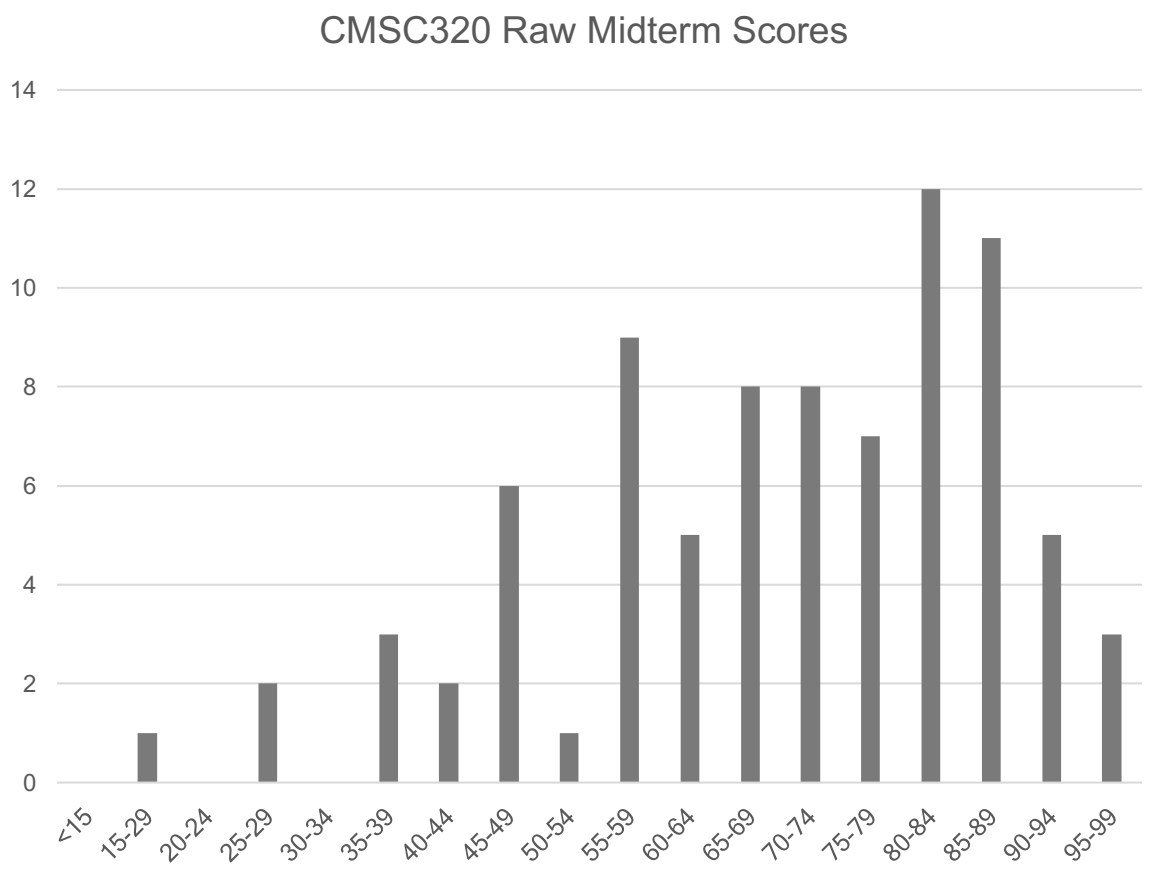
Min: 15

Max: 99

Mean: 69.67

Median: 73

Stdev: 17.8



FINAL TUTORIAL

Deliverable: URL of your own GitHub Pages site hosting an .ipynb/.html export of your final tutorial

- https://pages.github.com/ – make a GitHub account, too!
- https://github.com/blog/1995-github-jupyter-notebooks-3

The project itself:

- ~1500+ words of Markdown prose
- ~150+ lines of Python
- Should be viewable as a **static webpage** – that is, if I (or anyone else) opens the link up, everything should render and I shouldn't have to run any cells to generate output

FINAL TUTORIAL RUBRIC

The TAs and I will grade on a scale of 1-10:

Motivation: Does the tutorial make the reader believe the topic is important (a) in general and (b) with respect to data science?

Understanding: After reading the tutorial, does the reader understand the topic?

Further resources: Does the tutorial "call out" to other resources that would help the reader understand basic concepts, deep dive, related work, etc?

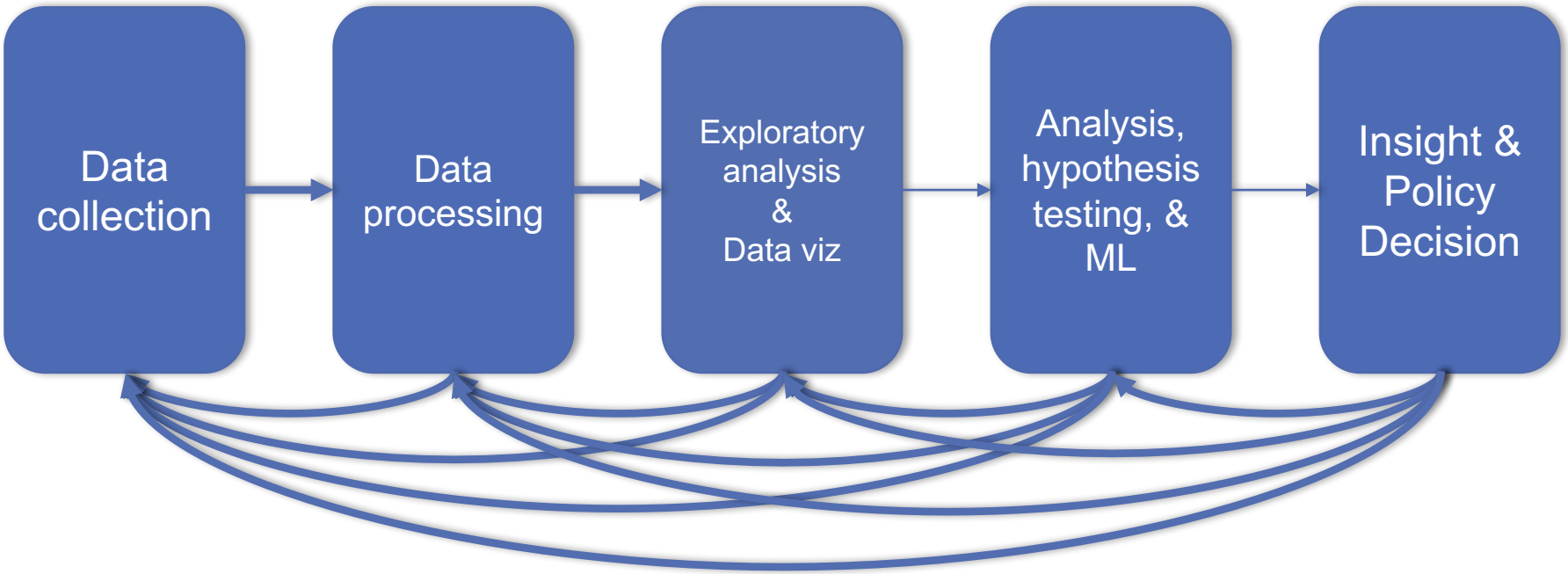
Prose: Does the prose in the Markdown portion of the .ipynb add to the reader's understanding of the tutorial?

Code: Does the code help solidify understanding, is it well documented, and does it include helpful examples?

Subjective Evaluation: If somebody linked to this tutorial from Hacker News, would people actually read the whole thing?

Thanks to: Zico Kolter

TODAY'S LECTURE



DATA SCIENCE
IN INDUSTRY

WHAT IS A DATA SCIENTIST?

Many types of "data scientists" in industry …

- **Business analysts, renamed**
  - "… someone who analyzes an organization or business domain (real or hypothetical) and documents its business or processes or systems, assessing the business model or its integration with technology." – Wikipedia
- **Statisticians**
- **Machine learning engineer**
- **Backend tools developer**

Thanks to: Zico Kolter

KEY DIFFERENCES

Classical statistics vs machine learning approaches

- (Two are nearly mixed in most job calls you will see.)

Developing data science tools vs. doing data analysis

Working on a core business product vs more nebulous "identification of value" for the firm

FINDING A JOB

Make a personal website.

- Free hosting options: GitHub Pages, Google Sites
- Pay for your own URL (but not the hosting).
- Make a clean website, and make sure it renders on mobile:
  - Bootstrap: https://getbootstrap.com/
  - Foundation: http://foundation.zurb.com/

Highlight relevant coursework, open source projects, tangible work experience, etc

Highlight tools that you know (not just programming languages, but also frameworks like TensorFlow and general tech skills)

“REQUIREMENTS”

Data science job postings – and, honestly, CS postings in general – often have completely nonsense requirements

1. The group is filtering out some noise from the applicant pool
2. Somebody wrote the posting and went buzzword crazy

In most cases (unless the position is a team lead, pure R&D, or a very senior role) you can work around requirements:

- A good, simple website with good, clean projects can work wonders here …
- Reach out and speak directly with team members
- Alumni network, internship network, online forums

SURVEY

Has anyone worked in a data scientist position here?

Interview questions?

INTERVIEWING

We saw that there is no standard for being a "data scientist" – and there is also no standard interview style …

… but, generally, you'll be asked about the five "chunks" we covered in this class, plus core CS stuff:

- Software engineering questions
- Data collection and management questions (SQL, APIs, scraping, newer DB stuff like NoSQL, Graph DBs, etc)
- General "how would you approach …" EDA questions
- Machine learning questions ("general" best practices, but you should be able to describe DTs, RFs, SVM, basic neural nets, KNN, OLS, boosting, PCA, feature selection, clustering)
- Basic "best practices" for statistics, e.g., hypothesis testing

Take-home data analysis project (YMMV)

GRADUATE SCHOOL, ACADEMIA, R&D, …

Data science isn't really an academic discipline by itself, but it comes up everywhere within and without CS

- **Modern science is built on a "CS and Statistics stack" …**

Academic work in the area:

- **Outside of CS, using techniques from this class to help fundamental research in that field**
- **Within CS, fundamental research in:**
  - Machine learning
  - Statistics (non-pure theory)
  - Databases and data management
  - Incentives, game theory, mechanism design
- **Within CS, trying to automate data science (e.g., Google Cloud's Predictive Analytics, "Automatic Statistician," …)**

UNDERGRADUATE RESEARCH AT UMD

Outside of CS:

- Is there an application area you really like?
- Most researchers will be thrilled to talk about how data science/ML can be applied to their domain
- You will have to learn about their domain ☺ …

Inside of CS:

- Many of us will want you to have taken or be taking at least CMSC422, or CMSC424, or both
- https://cs.umd.edu/researcharea/artificial-intelligence
- https://cs.umd.edu/researcharea/big-data
- https://cs.umd.edu/research-area/human-computer-interaction
- https://cs.umd.edu/research-area/data-science

IN THE WORKS …

~A dozen of you have approached me about doing independent research

- **Sometimes with an idea in mind**
- **Sometimes without an idea in mind**

If there is enough interest, I'd be happy to create an official course next (= Fall 2017) semester where:

- We meet infrequently;
- Get together in groups;
- Do a more intense version of a class project, where we create a new dataset and analyze it; and
- Write a paper about the experience.

Caveat: need to make sure I can actually do this … Piazza!

DEBUGGING DATA SCIENCE

Thanks to: https://alistapart.com, Zico Kolter

Traditional debugging

Traditional debugging of programs is relatively straightforward

You have some desired input/output pairs

You have a mental model (or maybe something more formal) of how each step in the algorithm “should” work

You trace through the execution of the program (either through a debugger or with print statement), to see where the state diverges from your mental model (or to discover your mental model is wrong)

Data science debugging

You have some desired input/output pairs

Your mental model is that an ML algorithm should work because … math? ... magic?

What can you trace through to see why it may not be working? Not very useful to step through an implementation of logistic regression…

Debugging data science vs. machine learning

Many of the topics here overlap with material on “debugging machine learning”

We are indeed going to focus largely on debugging data science prediction tasks (debugging web scraping, etc, is much more like traditional debugging)

But,

The first step of data science debugging

Step 1: determine if your problem is impossible

There are plenty of tasks that would be really nice to be able to predict, and absolutely no evidence that there the necessary signals to predict them (see e.g., predicting stock market from Twitter)

But, hope springs eternal, and it's hard to prove a negative…

A good proxy for impossibility

Step 1: ~~determine if your problem is impossible~~ see if *you* can solve your problem manually

Create an interface where you play the role of the prediction algorithm, you need to make the predictions of the outputs given the available inputs

To do this, you'll need to provide some intuitive way of visualizing what a complete set of input features looks like: tabular data for a few features, raw images, raw text, etc

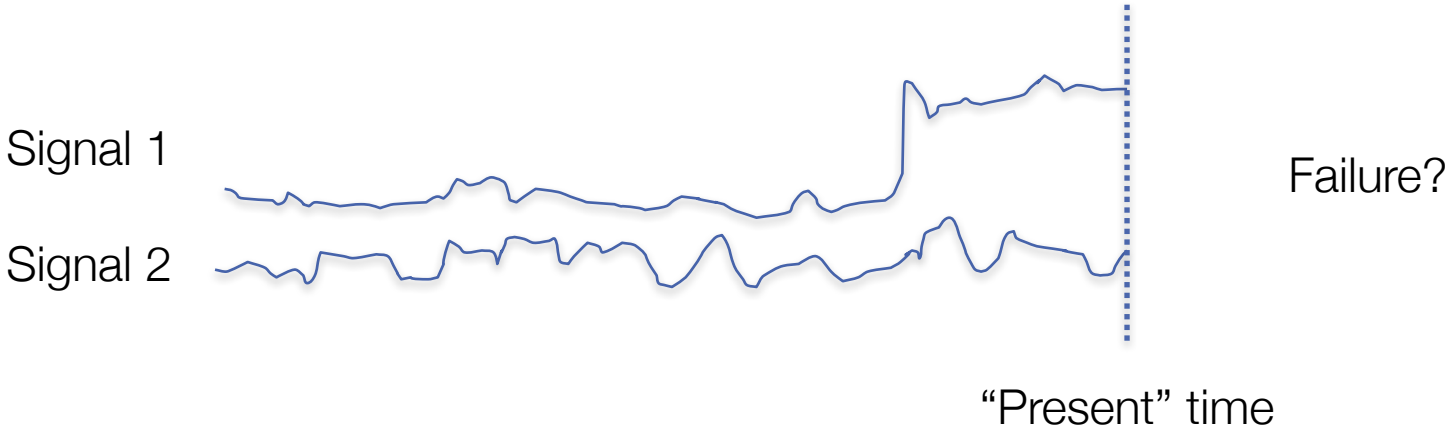
Just like a machine learning algorithm, you can refer to training data (where you know the labels), but you can't peak at the answer on your test/validation set

An example: predictive maintenance

An example task: you run a large factory and what to predict whether any given machine will fail within the next 90 days

You're given signals monitoring the state of this device

Your interface: visualize the signals (but not whether there was a failure or not), and see if you can identify whether or not a machine is about to fail?



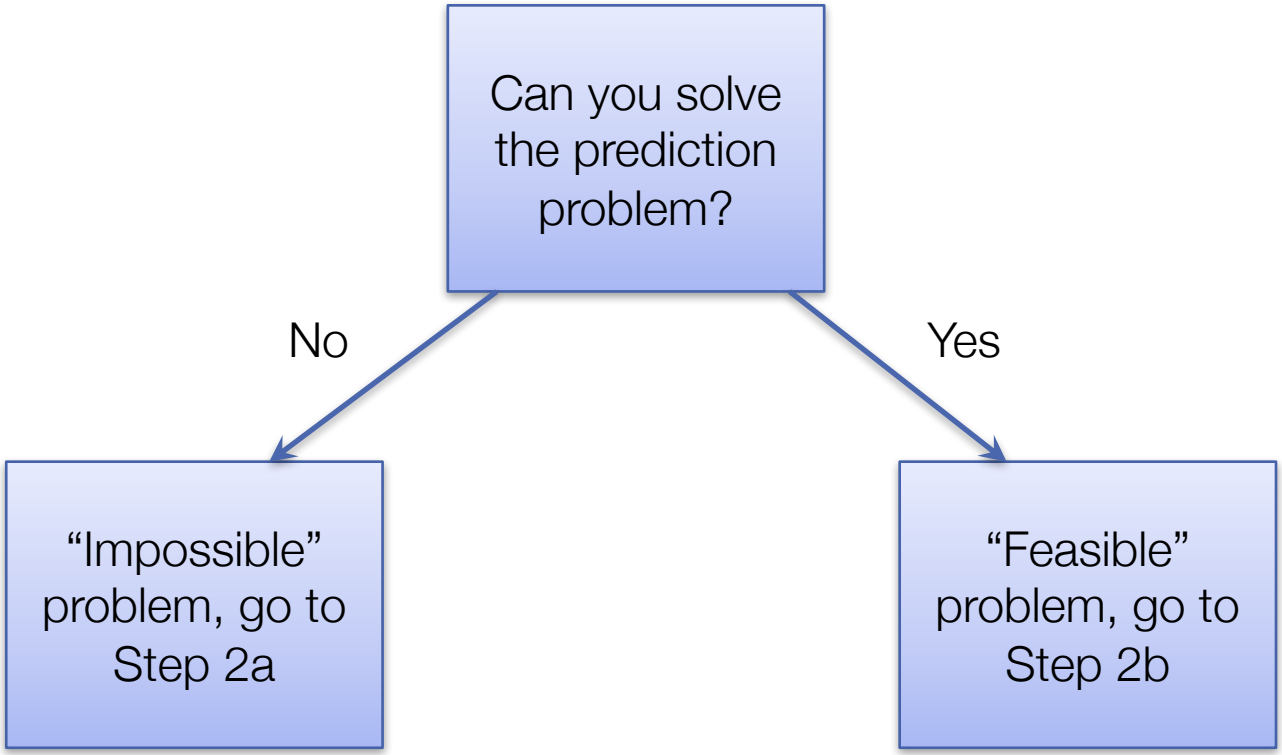
What about “superhuman” machine learning

It’s a common misconception that machine learning will *outperform* human experts on most tasks

In reality, the benefit from machine learning often doesn’t come from superhuman performance in most cases, it comes from the ability to scale up expert-level performance extremely quickly

If you can’t make good predictions, neither will a machine learning algorithm (at least the first time through, and probably always)

Decision diagram



Dealing with “impossible” problems

So you’ve built a tool to manually classify examples, run through many cases (or had a domain expert run through them), and you get poor performance

What do you do?

You do *not* try to throw more, bigger, badder, machine learning algorithms at the problem

Instead you need to change the problem by: 1) changing the input (i.e., the features), 2) changing the output (i.e., the problem definition)

Changing the input (i.e., adding features)

The fact that we can always add more features is what makes these problems “impossible” (with quotes) instead of impossible (no quotes)

You can always hold out hope that you just one data source away from finding the “magical” feature that will make your problem easy

But you probably aren’t… adding more data is good, but:

1. Do spot checks (visually) to see if this new features can help *you* differentiate between what you were previously unable to predict

2. Get advice from domain experts, see what sorts of data source they use in practice (if people are already solving the problem)

Changing the output (i.e., changing the problem)

Just make the problem easier! (well, still need to preserve the character of the data science problem)

A very useful procedure: instead of trying to predict the future, try to predict what an expert would predict given the features you have available

E.g., for predictive maintenance this shifts the question from: “would this machine fail?” to “would an expert choose to do maintenance on this machine?”

With this strategy we already have an existence proof that it’s feasible

Changing the output #2

Move from a question of getting “good” prediction to a question of characterizing the uncertainty of your predicts

Seems like a cop-out, but many tasks are *inherently* stochastic, the best you can do is try to quantify the likely uncertainty in output given the input

E.g.: if 10% of all machines fail within 90 days, it can still be really valuable to predict if whether a machine will fail with 30% probability

Dealing with feasible problems

Good news! Your prediction problem seems to be solvable (because you can solve it)

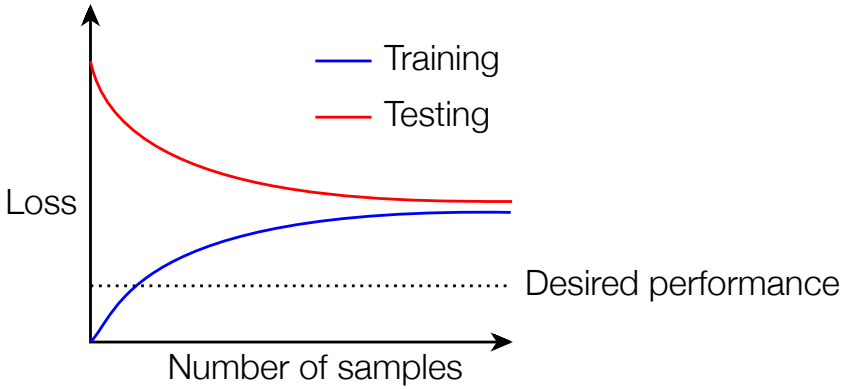
You run your machine learning algorithm, and find that it doesn't work (performs worse than you do)

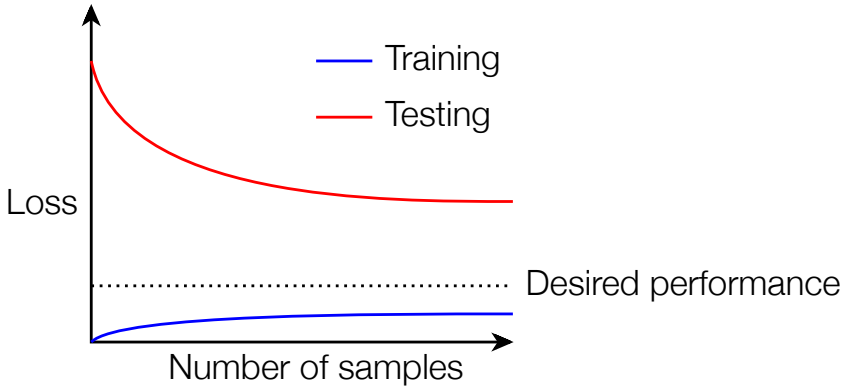
Again, you can try just throwing more algorithms, data, features, etc, at the problem, but this is unlikely to succeed

Instead you want to build diagnostics that can check what the problem may be

Characterizing bias vs. variance

Consider the training and testing loss of your algorithm (often plotting over different numbers of samples), to determine if you problem is one of high bias or high variance



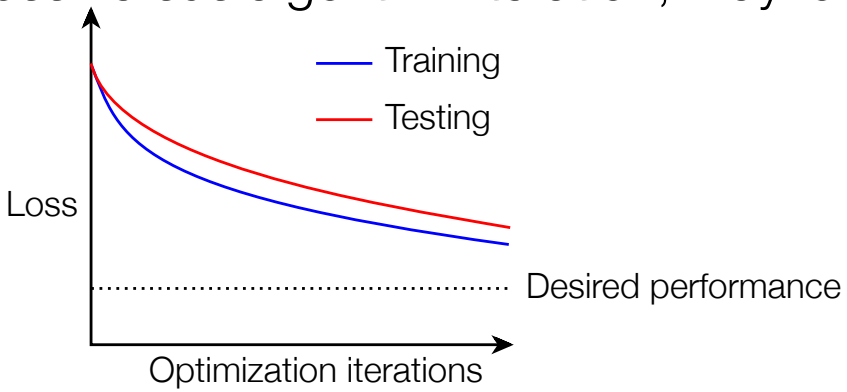


For high bias, add features based upon your own intuition of how you solved the problem

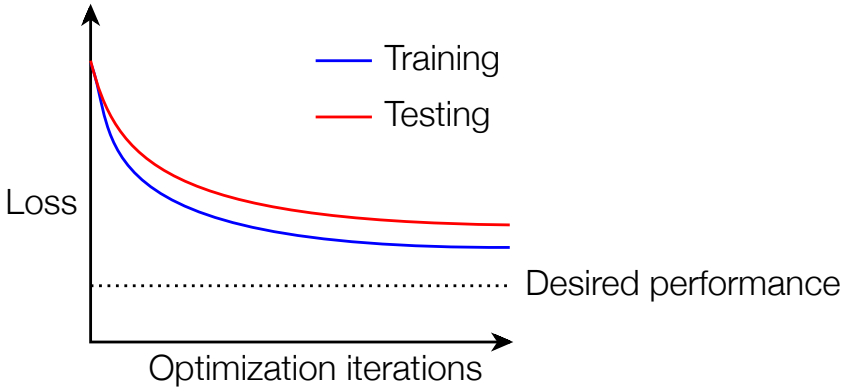
For high variance, add data or remove features (keeping features based upon your intuition)

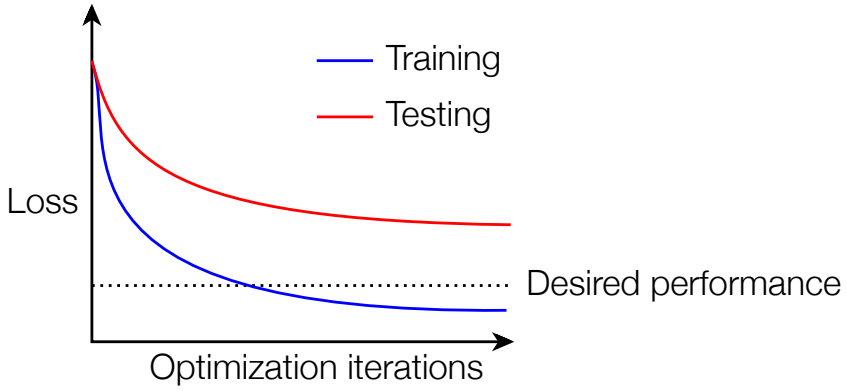
Characterizing optimization performance

It is a much less common problem, but you may want to look at training/testing loss versus algorithm iteration, may look like this:



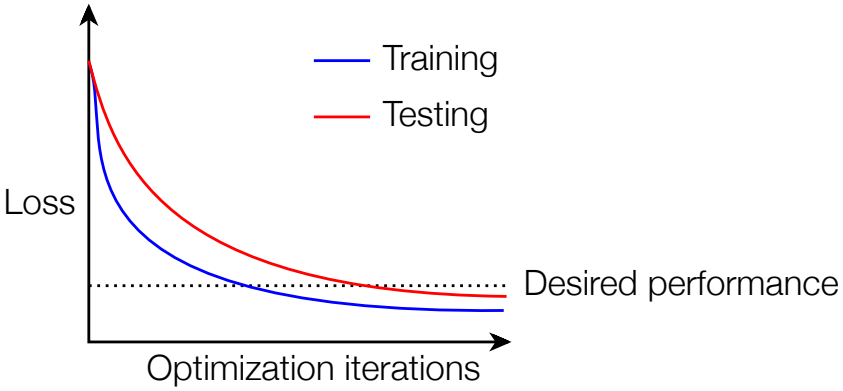
But it probably looks like this:

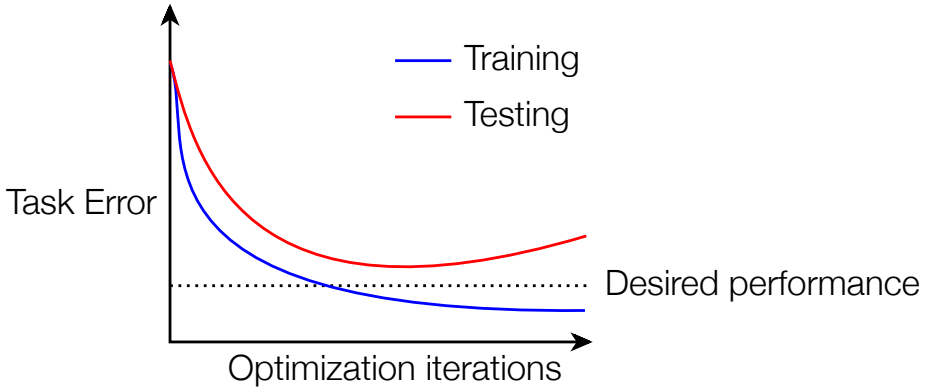




Consider loss vs. task error

Remember that machine learning algorithms try to minimize some loss, which may be different from the task error you actually want to optimize





This is common when dealing e.g. with imbalanced data sets for which cost of different classifications is very different

GET A PHD

AI-Hard problems: the difficulty of solving this problem is equivalent (at least) to solving "strong AI"

- "AI can perform any intellectual task a human can perform."
- Computer vision, natural language, MAS, RL, etc …

Your problem might be one of these.

- Go spend a life of luxury in academia or industry R&D making progress on these problems, or …
- … rephrase your problem and its output into something feasible.

THE DREAM

You run your ML algorithm(s) and it works well (?!)

Still: be skeptical …

Very easy to accidentally let your ML algorithm cheat:

- Peaking (train/test bleedover)
- Including output as an input feature explicitly
- Including output as an input feature implicitly

Try to solve the problem by hand;

Try to interpret the ML algorithm / output

Continue being skeptical. Always be skeptical.

SUMMARY

	ML predicts well	ML predicts poorly
You predict well	Be skeptical. Do automated analysis. Do human analysis.	"Feasible" problem: debug the **ML/analysis**.
You predict poorly	Make sure ML isn't cheating (peaking, …)	"Impossible" problem: debug the **problem**.

NEXT CLASS:

DEEP LEARNING