Barbara Liskov

2008 Turning award winner for her work on basically inventing the idea of data abstraction — for "programming language and system design, especially related to data abstraction, fault tolerance, and distributed computing"

One of the first women to earn a CS PhD in the world

- Professor @ MIT
- PhD from Stanford, 1968 (!)
- Research keywords: OOP, data abstraction, programming languages
The introduction should set up the problem (bit), introduce the solution (flip the bit), and present the solution concretely (instantiate the bit flip).

While this general structure is the same, the specifics look different based on if you have an old problem / new method or new problem / old method.
Assignment 3

You will write an Introduction to a paper for your project

Outline the introduction

Turn the outline into text

700-900 words

Due in 2 weeks: Thurs 5/5 11am

Details posted on course website

If you and your mentor are still thinking of a project, treat the Week 6 deadline as when you decide your direction for technical work
Congrats on finishing assignment 2!

Extra credit opportunity: If you turn in your introduction outline by next Weds, and come to your CA’s OH for feedback, you’ll get 1 extra credit point

50% points back for future assignments: We’ll release related works grades by Monday night. If you want to earn back (50%) lost points, make the changes your grader suggests and go to their OH to show them the revised draft
What problem are we solving?

“But how do we start?”

“I’m feeling so lost.”

“I thought of an important reason that this won’t work.”

“It’s not working yet. I’m not sure that we’re making progress.”
Today: vectoring

What is vectoring?
How do we vector effectively?
What goes wrong if we don’t vector?
Bernstein theory of faculty success

To be a top-tier faculty member, you need to master two skills that operate in a tight loop with one another.

**Vectoring**: identifying the biggest dimension of risk in your project right now

**Velocity**: rapid reduction of risk in the chosen dimension
What is Vectoring?
What research is not

1. Figure out what to do.
2. Do it.
3. Publish.

What research is

Research is an iterative process of exploration, not a linear path from idea to result [Gowers 2000]
Research is rarely linear

“OK, we have a good idea. Let’s build it / model it / prove it / get training data.”

“I spent some time thinking about this and hacking on it, and it’s not going to work: it has a fatal flaw.”

Treating your research goal as a project spec and executing it

These viewpoints are rigid and don’t allow for the flexibility (and self forgiveness!) that research requires.
As an example...

Jingyi’s current project: making drawn 2D clothes look more realistic

<table>
<thead>
<tr>
<th>Apr 2021</th>
<th>Aug 2021</th>
<th>Jan 2022</th>
<th>Apr 2022</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Brainstormed project idea</td>
<td>• Initial experiments (mostly failed)</td>
<td>• Thought numerical optimization would help make failed experiments look better</td>
<td>• Manually did more experiments for yet another new method</td>
</tr>
<tr>
<td>• Decided inputs &amp; outputs of system</td>
<td>• More experiments using a different tech stack (seemed maybe promising, 80% of the way there...?)</td>
<td>• Development work on getting an automated pipeline</td>
<td>• Implementing new method to be automatic</td>
</tr>
<tr>
<td>• Planned out project to take a year</td>
<td></td>
<td>• Pipeline worked, but results still were not good (very discouraging after a month+ of coding)</td>
<td>• Keep chugging along!</td>
</tr>
</tbody>
</table>
Ideas aren’t project specs

Should do
(Sketch)

Tempting to do
(Prototype)

This is all other points of a research project

This is the endpoint of a research project

[Buxton 2007, Sketching User Experiences]
Avoid jumping to the end

“OK, we have a good idea. Let’s build it / model it / prove it / get training data.”

“I spent some time thinking about this and hacking on it, and it’s not going to work: it has a fatal flaw.”

…before knowing what to refine!

……before identifying if that test or flaw is the right one to focus on!
Pick a vector

It may feel like we get stuck unable to solve the problem because we haven’t figured out everything else about it. There are too many open questions, and too many possible directions. The more dimensions there are, the harder gradient descent becomes.

Instead of doing trying to do everything at once (project spec), pick one dimension of uncertainty — one vector — and focus on reducing its risk and uncertainty.

To reduce uncertainty, it’s a better use of your time to make progress on a vector than it is to plan a grand timeline.
Example vectors

**Piloting:** Will this technique work at all? To answer this, we implement a basic version of the technique and manually put in data or interactions before programming them. (“Wizard of Oz” technique)

**Engineering:** Will this technique work with a realistic workload? To answer this, we need to find a representative, but smaller, test set.

**Proving:** Does the limit exist that I suspect does? To answer this, we start by writing a proof for a simpler case.

**Designing:** What might this interaction look like to an end user? To answer this, we create a low-fi prototype.
Implications

The vectors under consideration will each imply building different parts of your system.

Rather than building them all at once, when you might have to change things later, vectoring instead implies that you start by reducing uncertainty in the most important dimension first — your “inner loop” — and then building out from there.
Vectoring algorithm

1. Generate questions
   Untested hunches, risky decisions, high-level directions

2. Rank your questions
   Which is most critical?

3. Pick one and answer it rapidly
   Answer only the most critical question
   (This is where velocity comes into play)
How to generate questions? Assumption map.

Assumption mapping is a strategy for articulating questions by ranking assumptions.
Assumption mapping

Known - Unknown

Important - Unimportant

Plan

Evaluate (prioritize answering)

Generate (Get more data as needed)

Defer (Distraction)
Assumption mapping: Example

Project: making drawn 2D clothing more realistic

How to leverage existing physics simulation engines
How to automatically generate sewing patterns of clothing (we’re using a different method now)

Kinds of example clothes we want to support and show

Important

Once we have the 3D simulation, how do we flatten it back to 2D?

For what cases does the method presented in [nearest neighbor paper] fail?

What should I show in the submission video?

Unimportant

How will professional artists use this research?

How to submit the paper

Phrase unknowns as questions!
Assumption mapping: Example

Try it yourself!

http://tiny.cc/cs197-vector

1. Plan
   - How to submit the paper
   - Kinds of example clothes we want to support and show

2. Defer (Distraction)
   - How to automatically generate sewing patterns of clothing (we're using a different method now)

3. Evaluate
   - Once we have the 3D simulation, how do we flatten it back to 2D?
   - For what cases does the method presented in [nearest neighbor paper] fail?

4. Generate
   - What should I show in the submission video?
   - How will professional artists use this research?

- How to leverage existing physics simulation engines
- (prioritize answering)
- (Get more data as needed)
Vectoring and velocity

The output of a vectoring decision should allow you to identify what is core and what is periphery to reducing uncertainty in your vector of choice.

You should be able to make strong assumptions and use temporary scaffolding for anything that’s periphery. (That’s the velocity skill.)

We’ll talk more about velocity next week.
Let’s Try It
While everyone thinks that trolling online is due to a small number of antisocial sociopaths, we had a hunch that “normal” people were responsible for much trolling behavior when triggered.

What’s our first step?

We have: dataset of 16M CNN.com comments (w/ troll flags), Mechanical Turk for studies
Possible vectors:

Do people really troll when pissed off?

Can we train a classifier to predict when someone would troll, and compare weights of personal history vs. other posts and title?

Does the same person troll more on certain (angry) topics than on other (boring) ones?
We thought that, in domains where ML still cannot succeed, we could draw on crowdsourcing to identify human-labeled predictive features. In other words, that people are great at identifying potentially informative features, but might be poor at weighing those features correctly to arrive at a prediction.

What's our first step?
Possible vectors:

Can people identify predictive features for a single domain, e.g., lie detection?

Can people estimate which features are going to be informative?

Would a hybrid classifier (human features and labels as input to an ML model) actually perform well?

ABSTRACT

We present hybrid crowd-machine learning classifiers: classification models that start with a written description of a learning goal, use the crowd to suggest predictive features and label data, and then weigh these features using machine learning to produce models that are accurate and use human-understandable features. These hybrid classifiers enable fast prototyping of machine learning models that can improve on both algorithms performance and human judgment, and accomplish tasks where automated feature extraction is not yet feasible. Flock, an interactive machine-learning platform, instantiates this approach. To generate informative features, Flock asks the crowd to compare paired examples, an approach inspired by analogical encoding. The crowd’s effort can be focused on specific subsets of the input space where feature-extracted features are not predictive, or instead used to partition the input space and improve algorithm performance in subregions of the space. An evaluation on six prediction tasks, ranging from detecting deception to differentiating impressionist artists, demonstrated that aggregating crowd features improves upon both asking the crowd for a direct prediction and off-the-shelf machine learning features by over 10%. Further, hybrid systems that use both crowd-annotated and machine-extracted features can outperform those that use either in isolation.

Author Keywords
Crowdsourcing, interactive machine learning

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI); Miscellaneous

INTRODUCTION

Identifying predictive features is key to creating effective machine learning classifiers. Whether the task is link prediction or sentiment analysis, and no matter the underlying model, the “black art” of feature engineering plays a critical role in success [30]. Feature engineering is largely domain-specific, and success of machine learning systems depends on hours of experimentation. Often, the most predictive features only emerge empirically through trial and error. In this work we present a technique by which professionals can work collaboratively with non-experts in the design of features and the formulation of a hypothesis, effectively bridging the gap between human and machine.

Figure 1. Flock is a hybrid crowd-machine learning platform that exploits on-analogue encoding to guide crowds to generate effective features, then uses machine learning techniques to aggregate their labels, after many iterations [56]. And though feature engineers may have deep domain expertise, they are only able to incorporate features that are extractable via code.

However, embedding crowds outside of machine learning architecture opens the door to hybrid learners that can explore feature spaces that are largely unattainable by automatic extraction, then train models that use human-understandable features (Figure 1). Doing so enables fast prototyping of classifiers that can exceed both machine and expert performance. In this paper, we demonstrate classifiers that identify people who are lying, perform quality assessment of Wikipedia articles, and differentiate impressionist artists who use similar styles. Previous work that bridges crowdsourcing and machine learning has focused on optimizing the crowd’s efforts (e.g., [4, 21, 56]), we suggest that inverting the relationship and embedding crowd insight inside machine learning enables machine learning to be deployed for new kinds of tasks. We present Flock, an end-to-end machine learning platform that uses paid crowdsourcing to speed the prototyping loop and augment the performance of machine learning systems. Flock constructs a model for creating hybrid classifiers that intelligently combine crowd and machine features. The system allows users to rapidly author hybrid crowd-machine learners by structuring a feature nomination process using the crowd, aggregating the suggested features, then collecting labels on these new features. It loops and gathers more crowd features to improve performance on subsets of the space where the model is more ambiguous and improves. For instance, given a decision tree that uses machine-mutable features, Flock can dynamically grow sub-trees from nodes that have high classification error or even replace whole branches. In addition to
Why is vectoring so important?
Ideas rarely land exactly where you expect they will. It’s best to test the most critical assumptions quickly, so that you can understand whether your hunch will play out, and what problems are worth spending time solving.

Human creative work is best in a loop of reflection and iteration. Vectoring is a way to make sure you’re getting the most iteration cycles.
Re-vectoring

Often, after vectoring and reducing uncertainty in one dimension, it raises new questions and uncertainties.

In the next round of vectoring, you re-prioritize:

- If you get unexpected results and are confused (most of the time!), maybe it means you take a **new angle** to reduce uncertainty on a vector related to the prior one.

- If you answer your question to your own satisfaction (not completely, just to your satisfaction), you move on to the **next most important vector**
Magnitude of your vector

The result of vectoring should be something achievable in about a week’s time. If it’s not, you’ve picked too broad a question to answer.

If your vectoring for “Can normal people be responsible for a lot of the trolling online?” is “Can normal people be responsible for a lot of the trolling on CNN.com?”, you’re still way too broad.

That’s evidence that you’ve just rescaled your project, not picked a vector.
Project: making drawn 2D clothing more realistic

Vectors should be phrased as a question answerable with a week of work, from your important-unknown quadrant.

1. What existing methods exist for transferring 3D geometry to 2D, and how do they apply/not apply to our problem domain?

2. Important

3. After I implement the NN paper’s method for 2D->3D, what set of clothes should I test on?

4. Unknown

Once we have the 3D simulation, how do we flatten it back to 2D?

For what cases does the method presented in [nearest neighbor paper] fail?

What should I show in the submission video?

How will professional artists use this research?
On your slide, write 2-3 possible vectors.

http://tiny.cc/cs197-vector
Takeaways, in brief
1) Research is almost never a simple, linear process. It’s normal to feel nervous about all the possible things you can do to solve a problem.
2) Vectoring is a process of identifying the dimension of highest impact+uncertainty, and prioritizing that dimension.
3) Successful vectoring enables you to rapidly hone in on the core insight of your research project.
Your To-dos

1. Assignment 4 (due in 2 Thursdays)
   • Submit outline early and come to OH for feedback for extra credit
2. Continue to update weekly log
   • Try to write your planned vector each week!

Exit ticket: http://tiny.cc/cs197-week4
Computer Science Research

Slide content shareable under a Creative Commons Attribution-NonCommercial 4.0 International License.