# Introduction to Machine Learning Problem Framing

## Introduction

Welcome to Introduction to Machine Learning Problem Framing!

This course helps you frame machine learning (ML) problems.

This course does not cover how to implement ML or work with data.

## Objectives:

* Define common ML terms
* Describe examples of products that use ML and general methods of ML problem-solving used in each
* Identify whether to solve a problem with ML
* Compare and contrast ML to other programming methods
* Apply hypothesis testing and the scientific method to ML problems
* Have conversations about ML problem-solving methods

Diagrama

Descrição gerada automaticamente

# Common ML Problems

In basic terms, ML is the process of training a piece of software, called a model[[1]](#footnote-1), to make useful predictions using a data set. This predictive model can then serve up predictions about previously unseen data. We use these predictions to take action in a product; for example, the system predicts that a user will like a certain video, so the system recommends that video to the user.

Often, people talk about ML as having two paradigms, supervised and unsupervised learning. However, it is more accurate to describe ML problems as falling along a spectrum of supervision between supervised and unsupervised learning. For the sake of simplicity, this course will focus on the two extremes of this spectrum.

## What is Supervised Learning?

Supervised learning is a type of ML where the model is provided with labeled [[2]](#footnote-2)training data. But what does that mean?

For example, suppose you are an amateur botanist determined to differentiate between two species of the Lilliputian plant genus (a completely made-up plant). The two species look pretty similar. Fortunately, a botanist has put together a data set of Lilliputian plants she found in the wild along with their species name.

Here's a snippet of that data set:

|  |  |  |
| --- | --- | --- |
| Leaf Width | Leaf Length | Species |
| 2.7 | 4.9 | small-leaf |
| 3.2 | 5.5 | big-leaf |
| 2.9 | 5.1 | small-leaf |
| 3.4 | 6.8 | big-leaf |

Leaf width and leaf length are the **features** [[3]](#footnote-3)(which is why the graph below labels both of these dimensions as X), while the species is the label. A real life botanical data set would probably contain far more features (including descriptions of flowers, blooming times, arrangement of leaves) but still have only one label. Features are measurements or descriptions; the label is essentially the "answer." For example, the goal of the data set is to help other botanists answer the question, "Which species is this plant?"

This data set consists of only four examples. A real life data set would likely contain vastly more examples.

Suppose we graph the leaf width and leaf length and then color-code the species.

Gráfico

Descrição gerada automaticamente

Now that a model exists, you can use that model to classify new plants that you find in the jungle. For example:

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

To tie it all together, supervised machine learning finds patterns between data and labels that can be expressed mathematically as functions. Given an input feature, you are telling the system what the expected output label is, thus you are supervising the training. The ML system will learn patterns on this labeled data. In the future, the ML system will use these patterns to make predictions on data that it did not see during training.

An exciting real-world example of supervised learning is a [study from Stanford University](https://news.stanford.edu/2017/01/25/artificial-intelligence-used-identify-skin-cancer/) that used a model to detect skin cancer in images. In this case, the training set contained images of skin labeled by dermatologists as having one of several diseases. The ML system found signals that indicate each disease from its training set, and used those signals to make predictions on new, unlabeled images.

## Unsupervised Learning

In unsupervised learning, the goal is to identify meaningful patterns in the data. To accomplish this, the machine must learn from an unlabeled data set. In other words, the model has no hints how to categorize each piece of data and must infer its own rules for doing so.

In the following graph, all the examples are the same shape because we don't have labels to differentiate between examples of one type or another here:

Ícone

Descrição gerada automaticamente

Fitting a line to unlabeled points isn't helpful. We still end up with examples of the same shape on both sides of the line. Clearly we will have to try a different approach.

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

Here, we have two **clusters**[[4]](#footnote-4). (Note that the number of clusters is arbitrary). What do these clusters represent? It can be difficult to say. Sometimes the model finds patterns in the data that you don't want it to learn, such as stereotypes or bias.

Ícone

Descrição gerada automaticamente

However, when new data arrives, we can categorize it pretty easily, assuming it fits into a known cluster. But what if your photo clustering model has never seen a pangolin before? Will the system cluster the new photo with armadillos or maybe hedgehogs? This course will talk more about the difficulties of unlabeled data and clustering later on.

Uma imagem contendo Diagrama

Descrição gerada automaticamente

### Reinforcement Learning

An additional branch of machine learning is reinforcement learning (RL). Reinforcement learning differs from other types of machine learning. In RL you don't collect examples with labels. Imagine you want to teach a machine to play a very basic video game and never lose. You set up the model (often called an agent in RL) with the game, and you tell the model not to get a "game over" screen. During training, the agent receives a reward when it performs this task, which is called a reward function. With reinforcement learning, the agent can learn very quickly how to outperform humans.

The lack of a data requirement makes RL a tempting approach. However, designing a good reward function is difficult, and RL models are less stable and predictable than supervised approaches. Additionally, you need to provide a way for the agent to interact with the game to produce data, which means either building a physical agent that can interact with the real world or a virtual agent and a virtual world, either of which is a big challenge. See this blog post by Alex Irpan for an overview of the types of problems currently faced in RL. Reinforcement learning is an active field of ML research, but in this course we'll focus on supervised solutions because they're a better known problem, more stable, and result in a simpler system.

For comprehensive information on RL, check out [Reinforcement Learning: An Introduction](https://mitpress.mit.edu/books/reinforcement-learning) by Sutton and Barto.

## Types of ML Problems

There are several subclasses of ML problems based on what the prediction task looks like. In the table below, you can see examples of common supervised and unsupervised ML problems.

|  |  |  |  |
| --- | --- | --- | --- |
| Type of ML Problem | Description | Example | Supervised / Unsupervised |
| Classification | Pick one of N labels | Cat, dog, horse, or bear | Supervised |
| Regression | Predict numerical values | Click-through rate | Supervised |
| Clustering | Group similar examples | Most relevant documents (unsupervised) | Unsupervised |
| Association rule learning | Infer likely association patterns in data | If you buy hamburger buns, you're likely to buy hamburgers (unsupervised) | Unsupervised |
| Structured output | Create complex output | Natural language parse trees, image recognition bounding boxes | Supervised |
| Ranking | Identify position on a scale or status | Search result ranking | Supervised |

## Check Your Understanding

Structured output

Complex outputs require complex labeled data. This is a supervised learning problem.

Tente novamente.

Regression

Regression requires labeled numerical data. This is a supervised learning problem.

Tente novamente.

Classification

Classification requires a set of labels for the model to assign to a given item. This is a supervised learning problem.

Tente novamente.

***Clustering***

***Clustering is typically done when labeled data is not available. This is an unsupervised learning problem.***

***Resposta correta.***

## Contrasting Cases

As you walk through each example, note the types of data used and how that data informed the product design and iterations. Think about how the examples compare to and contrast from each other. Click on each product name button to see more information below.

### Smart Reply

Suggested short responses to emails.

Smart Reply is an example of ML that utilizes Natural Language Understanding (NLU) and generation, sequence-to-sequence learning, to make replying to a flooded inbox far less painful.

* [Computer, respond to this email](https://research.googleblog.com/2015/11/computer-respond-to-this-email.html)
* [Smart Reply: Automated Response Suggestion for Email](http://www.kdd.org/kdd2016/papers/files/Paper_1069.pdf)(2016 article)

### YouTube Watch Next

YouTube Watch Next uses ML to generate the list of video recommendations after you've watched a video on YouTube. It is a large scale recommendation system using deep networks to generate and rank potential videos.

* [Deep Neural Networks for YouTube Recommendations](https://research.google.com/pubs/pub45530.html)

### Cucumber Sorting

See how a cucumber farmer is using machine learning to sort cucumbers by size, shape, color, and other attributes.

* [How a Japanese cucumber farmer is using deep learning and TensorFlow](https://cloud.google.com/blog/big-data/2016/08/how-a-japanese-cucumber-farmer-is-using-deep-learning-and-tensorflow)

## Thought Questions

Think about the similarities and differences between each of the above cases. Click on the plus icon to expand the section and reveal the answers.

#### What user problem did these systems solve?

In all three cases there was motivation to build an ML system to address a real problem users were facing.

* Smart Reply: responding to emails can take up too much time
* YouTube: there are too many videos on YouTube for one person to navigate and find videos they like
* Cucumber sorter: the cucumber sorting process is burdensome

#### What does output from these systems look like?

Each is a bit different.

* Smart Reply: three short suggested responses at the bottom of an email
* YouTube: suggested videos along the right-hand side of the screen
* Cucumber sorter: directions to a robot arm that sorts cucumbers into their correct categories

#### What data sources were used?

In all three cases the large amounts of historical data had information closely tied to what we wanted to do.

* Smart Reply: conversation data (email messages and responses)
* YouTube: watch time, click-through rate, watch history, search history
* Cucumber sorter: exemplary cucumber data (size, shape, weight, etc.)

## The ML Mindset

"Machine Learning changes the way you think about a problem. The focus shifts from a mathematical science to a natural science, running experiments and using statistics, not logic, to analyse its results." - Peter Norvig - Google Research Director

In traditional software engineering, you can reason from requirements to a workable design, but with machine learning, it will be necessary to experiment to find a workable model.

Many machine learning systems produce models that encode knowledge and intelligence by interpreting signals differently than humans do. A neural network might interpret a word via an embedding, so "tree" is understood as something like, [0.37, 0.24, 0.2] and "car" as [0.1, 0.78, 0.9]. The neural network might use these representations to do accurate translations or sentiment analysis, but a human looking at the embeddings would find them very hard to understand. This can make machine intelligence difficult, but not impossible, for humans to understand and evaluate.

Models will make mistakes that are difficult to debug, due to anything from skewed training data to unexpected interpretations of data during training. Furthermore, when machine-learned models are incorporated into products, the interactions can be complicated, making it difficult to predict and test all possible situations. These challenges require product teams to spend a lot of time figuring out what their machine learning systems are doing and how to improve them.

## Look at the Example of Google Photos

The [video](https://youtu.be/bHvf7Tagt18) in this section dives into how ML powers Google Photos: ML powers the search behind Google Photos to classify people, places, and things.

This example demonstrates that we can teach a model to recognize cats in photos, but it is difficult to know what features the model uses to determine something is in fact a cat. This uncertainty can feel a little uncomfortable at times if you are used to determining every detail of your code's behavior.

Check out the links below for more information on the progress and impact of Google Photos.

|  |
| --- |
| Extra Resources |
| * [Wired Article](https://www.wired.com/2015/06/how-googles-new-photos-app-can-tell-cats-from-dogs/) |
| * [Google Blog at launch](https://www.blog.google/products/photos/picture-this-fresh-approach-to-photos/) |
| * [Google Blog 1 year later](https://www.blog.google/products/photos/google-photos-one-year-200-million/) |
| * [Google Blog 2 years later](https://www.blog.google/products/photos/google-photos-500-million-new-sharing/) |

## Experimental Design Primer

### Get Comfortable with Some Uncertainty

Beyond simply thinking about problems differently, implementing ML is different than traditional programming. In traditional programming, you have set parameters and you understand how everything should behave. With ML, the non-coding work can be very complicated, but you'll usually write far less code.

Will you end-up with a usable model? You don't really know at the start.

### Scientific Method

To address the challenges of transitioning to ML, it is helpful to think of the ML process as an experiment where we run test after test after test to converge on a workable model. Like an experiment, the process can be exciting, challenging, and ultimately worthwhile.

|  |  |
| --- | --- |
| Step | Example |
| 1. Set the research goal. | I want to predict how heavy traffic will be on a given day. |
| 2. Make a hypothesis. | I think the weather forecast is an informative signal. |
| 3. Collect the data. | Collect historical traffic data and weather on each day. |
| 4. Test your hypothesis. | Train a model using this data. |
| 5. Analyze your results. | Is this model better than existing systems? |
| 6. Reach a conclusion. | I should (not) use this model to make predictions, because of X, Y, and Z. |
| 7. Refine hypothesis and repeat. | Time of year could be a helpful signal. |

## Identifying Good Problems for ML

This section examines the characteristics of good ML problems.

### Clear Use Case

***Start with the problem, not the solution. Make sure you aren't treating ML as a hammer for your problems.***

Focus on problems that would be difficult to solve with traditional programming. For example, consider Smart Reply. The Smart Reply team recognized that users spend a lot of time replying to emails and messages; a product that can predict likely responses can save user time. Another example is in Google Photos, where the business problem was to find a specific photo by keyword search without manual tagging.

Imagine trying to create a system like Smart Reply or Google Photos search with conventional programming. There isn't a clear approach. By contrast, machine learning can solve these problems by examining patterns in data and adapting with them. Think of ML as just one of the tools in your toolkit and only bring it out when appropriate.

With these examples in mind ask yourself the following questions:

1. What problem is my product facing?
2. Would it be a good problem for ML?

***Don't ask the questions the other way around!***

## Know the Problem Before Focusing on the Data

Be prepared to have your assumptions challenged.

If you understand the problem clearly, you should be able to list some potential solutions to test in order to generate the best model. Understand that you will likely have to try out a few solutions before you land on a good working model.

Exploratory data analysis can help you understand your data, but you can't yet claim that patterns you find generalize until you check those patterns against previously unseen data. Failure to check could lead you in the wrong direction or reinforce stereotypes or bias.

## Lean on Your Team's Logs

***ML requires a lot of relevant data.***

Data collected specifically for your task is going to be the most useful. In practice, you may not be able to do this, and you'll rely on whatever data you can get that's close enough. That's fine as long as you're aware of the cost, and as you can eventually get product logs, you can use those to build something more targeted to your task.

How much is "a lot?" That depends on the problem, but more data typically improves your model and therefore your model's predictive power. A good rule of thumb is to have at least thousands of examples for basic linear models, and hundreds of thousands for neural networks. If you have less data, consider a non-ML solution first.

## Predictive Power

***Your features contain predictive power.***

Suppose you're trying to predict which horses will perform well in a race. You decide to tackle the problem with ML and use the horse’s eye color as a feature. You reason that eye color predicts which horses are prone to eye disease, which in turn could predict a horse’s speed and stamina. Maybe you’re wrong and you'll reject the hypothesis later based on evidence; that is, perhaps using eye color as a feature does not improve your model.



You should not try to make ML do the hard work of discovering which features are relevant for you. If you simply throw everything at the model and see what looks useful, your model will likely wind up overly complicated, expensive, and filled with unimportant features. In smaller datasets, you have a higher chance that a feature will be correlated with your label by chance within your sample of data. If you try lots of features without a hypothesis, you'll falsely believe these are relevant signals for your model. You wouldn't catch this until you tried to make predictions with your model and realized it did not generalize.

1. **Model**: The representation of what a machine learning system has learned from the training data. Within *TensorFlow*, model is an overloaded term, which can have either of the following two related meanings:

   The TensorFlow graph that expresses the structure of how a prediction will be computed.

   The particular weights and biases of that TensorFlow graph, which are determined by training. [↑](#footnote-ref-1)
2. **Label**: In supervised learning, the "answer" or "result" portion of an example. Each example in a labeled dataset consists of one or more features and a label. For instance, in a housing dataset, the features might include the number of bedrooms, the number of bathrooms, and the age of the house, while the label might be the house's price. In a spam detection dataset, the features might include the subject line, the sender, and the email message itself, while the label would probably be either "spam" or "not spam." [↑](#footnote-ref-2)
3. **Feature**: An input variable used in making predictions. [↑](#footnote-ref-3)
4. Note: While it is very common, clustering is not the only type of unsupervised learning. [↑](#footnote-ref-4)