Framing the problem and constructing the dataset

Szu-Chi Chung

Department of Applied Mathematics, National Sun Yat-sen University

Start from scratch!

- In the real world, you don't start from a dataset, you start from a problem
- Imagine that you're starting your own data science consulting shop. You put up a fancy website, you notify your network. The projects start rolling in:
  1. A personalized photo search engine for a picture-sharing social network—type in "wedding" and retrieve all the pictures you took at weddings
  2. Flagging spam and offensive text content among the posts of a chat app
  3. Building a music recommendation system for users of an online radio
  4. Detecting credit card fraud for an e-commerce website
  5. Predicting display ad click-through rate to decide which ad to serve to a given user at a given time
  6. Flagging anomalous cookies on the conveyor belt of a cookie-manufacturing line

The universal workflow of data science project

- The universal workflow of data science is broadly structured in three parts:
  1. **Define the task** — Understand the problem domain and the *business logic* underlying what the customer asked for. *Collect a dataset*, understand what the data represents, and choose how you will *measure success* on the task
  2. **Develop a model** — Prepare your data so that it can be processed by a machine learning model, select a model evaluation protocol and a simple baseline to beat, train a first model that has generalization power and that can *overfit*, and then regularize and tune your model until you achieve the best possible generalization performance
  3. **Deploy the model** — Present your work to stakeholders, ship the model to a web server, a mobile app, a web page, or an embedded device, monitor the model's performance in the wild, and start collecting the data you'll need to build the next-generation model

Recap

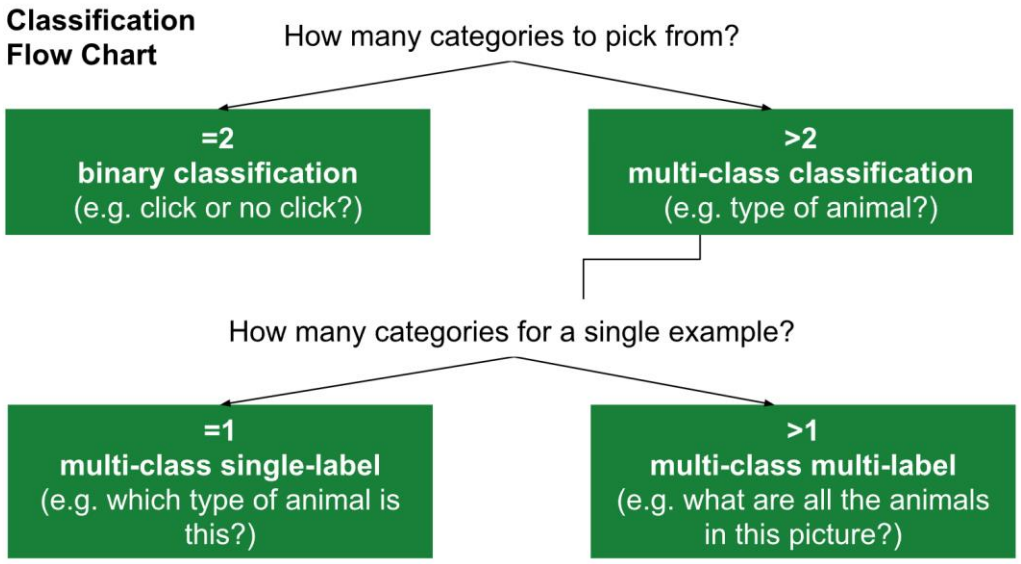
1. **Look at the big picture**
2. **Get the data**
3. Discover and visualize the data to gain insights
4. Prepare the data for Machine Learning algorithms
5. Select a model and train it
6. Fine-tune your model
7. Present your solution
8. Launch, monitor, and maintain your system

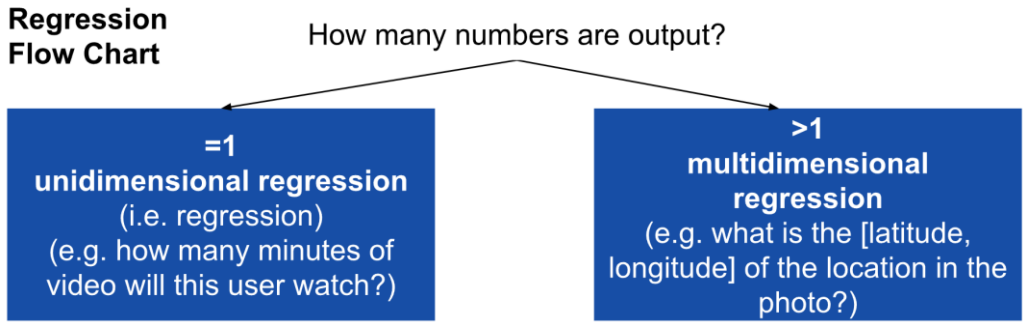
1. Frame the problem

- Framing a data science problem usually involves many detailed discussions with others. Here are the questions that should be on the top of your mind:
  - What are you trying to predict?
  - What type of machine learning task are you facing?
    - Is it binary classification? Multiclass classification? Multiclass, multilabel classification?
    - Scalar regression? Vector regression?
    - Something else, like clustering, visualization, or reinforcement learning? In some situations, machine learning isn't the best way to make sense of the data, and you should use something else, such as traditional statistical analysis

Frame the problem - Type of ML Problem

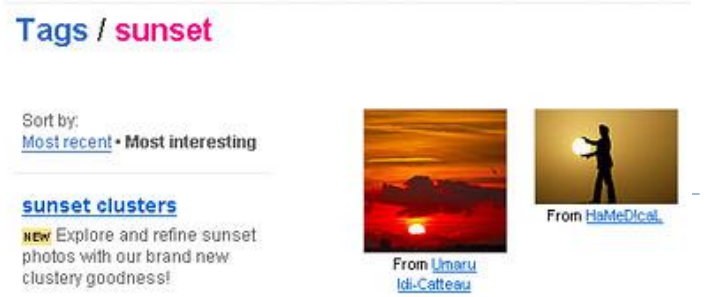
Type of ML Problem	Description	Example
Classification	Pick one of N labels	Cat, dog, horse, or bear
Regression	Predict numerical values	Click-through rate
Complex output	Create complex output	Natural language parse trees, image recognition bounding boxes
Clustering	Group similar examples	Most relevant documents (unsupervised)
Recommendation system (e.g. Association rule learning)	Infer likely association patterns in data	If you buy hamburger buns, you're likely to burger meat (unsupervised)





https://developers.google.com/machine-learning/problem-framing/

Frame the problem



- What type of the problem
  1. The photo search engine project is a multiclass, multilabel classification task
  2. The spam detection project is a binary classification task. If you set “offensive content” as a separate class, it’s a three-way classification task
  3. The music recommendation engine turns out to be better handled not via deep learning, but via matrix completion/factorization (collaborative filtering)
  4. The credit card fraud detection project is a binary classification task
  5. The click-through-rate prediction project is a scalar regression task
  6. Anomalous cookie detection is a binary classification task, but it will also require an object detection model as a first stage to correctly crop out the cookies in raw images

Frame the problem – Existing solution

- What do existing solutions look like?
  - Perhaps your customer already has a *handcrafted algorithm* that handles spam filtering or credit card fraud detection, with lots of nested *if* statements
  - Perhaps a human is currently in charge of manually handling the process under consideration — monitoring the conveyor belt at the cookie plant and manually removing the bad cookies
- Are there particular constraints you will need to deal with?
  - For example, perhaps the cookie-filtering model has such *latency* constraints that it will need to run on an embedded device at the factory rather than on a remote server. You should understand the full context in which your work will fit

Frame the problem – Get the data/Check the assumptions

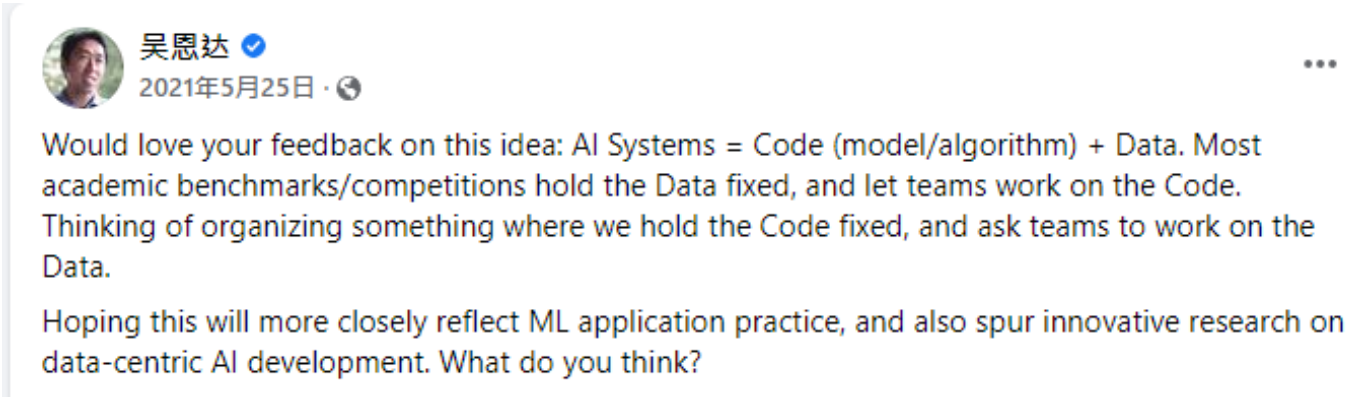
- Once you've done your research, you should know what *your inputs will be*, *what your targets will be*, and what broad type of machine learning task the problem maps to. Be aware of the hypotheses you're making at this stage:
  1. You hypothesize that your targets can be predicted given your inputs
  2. You hypothesize that the data that's available (or that you will soon collect) is sufficiently informative to learn the relationship between inputs and targets
  3. Not all problems can be solved with machine learning; just because you've assembled inputs X and targets Y doesn't mean X contains enough information to predict Y!
     - For instance, if you're trying to predict the movements of a stock on the stock market given its recent price history, you're unlikely to succeed, because price history doesn't contain much predictive information

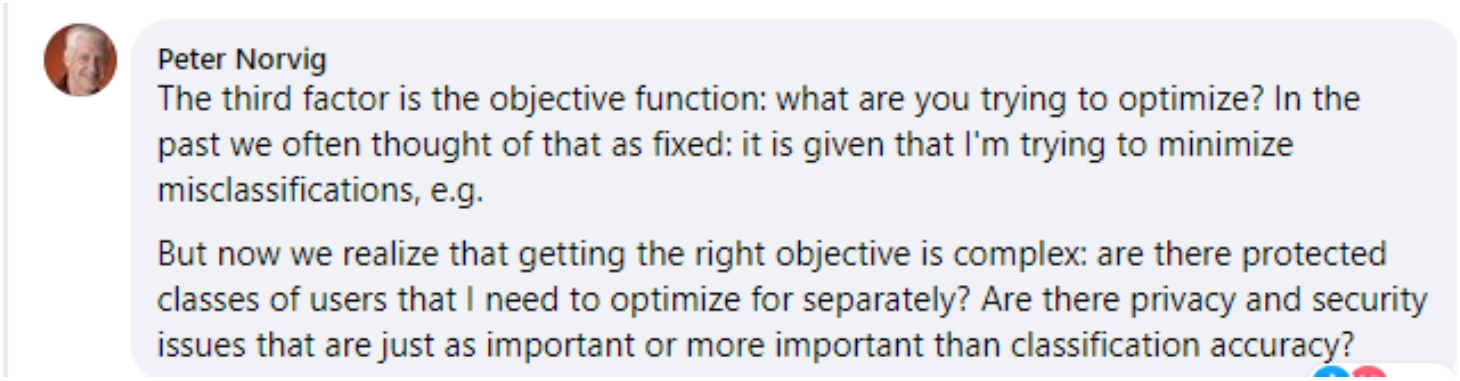
Frame the problem - Choose a measure of success

- To achieve success on a project, you must first define what you mean by success. Accuracy? Precision and recall?
  - Your metric for success will guide all of the choices you make throughout the project. It should align with your higher-level goals, such as the business success of your customer!
  - For balanced classification problems, where every class is equally likely, accuracy and the area under a *receiver operating characteristic (ROC)* curve, are common metrics. For class-imbalanced problems or multilabel classification, you can use precision and recall, as well as a weighted form of accuracy or F1 score
  - It isn't uncommon to have to define your own custom metric by which to measure success. To get a sense of the diversity of machine learning success metrics and how they relate to different problem domains, it's helpful to browse the data science competitions on Kaggle

Frame the problem - Choose a measure of success

- Data
- Model
- Objective function

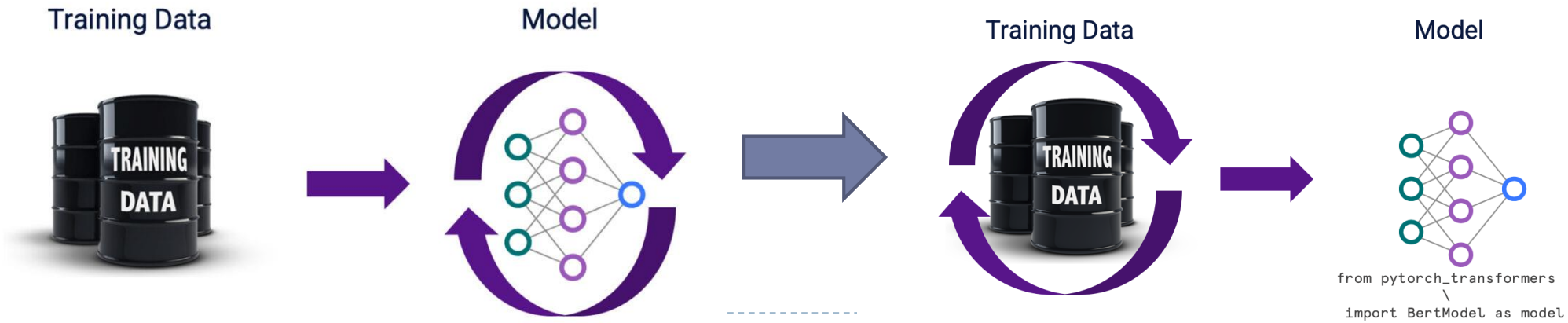




https://www.facebook.com/andrew.ng.96/posts/3982283021827575

Frame the problem - Data-Centric AI

- In most machine learning competitions, you are asked to build a high-performance model given a fixed dataset
  - However, machine learning has matured to the point that high-performance model architectures are widely available, while approaches to engineering datasets have lagged
    - The Data-Centric AI inverts the traditional format and instead asks you to improve a dataset given a fixed model
  - Provide you with a dataset to improve by applying data-centric techniques such as fixing incorrect labels, adding examples that represent edge cases, apply data augmentation, etc

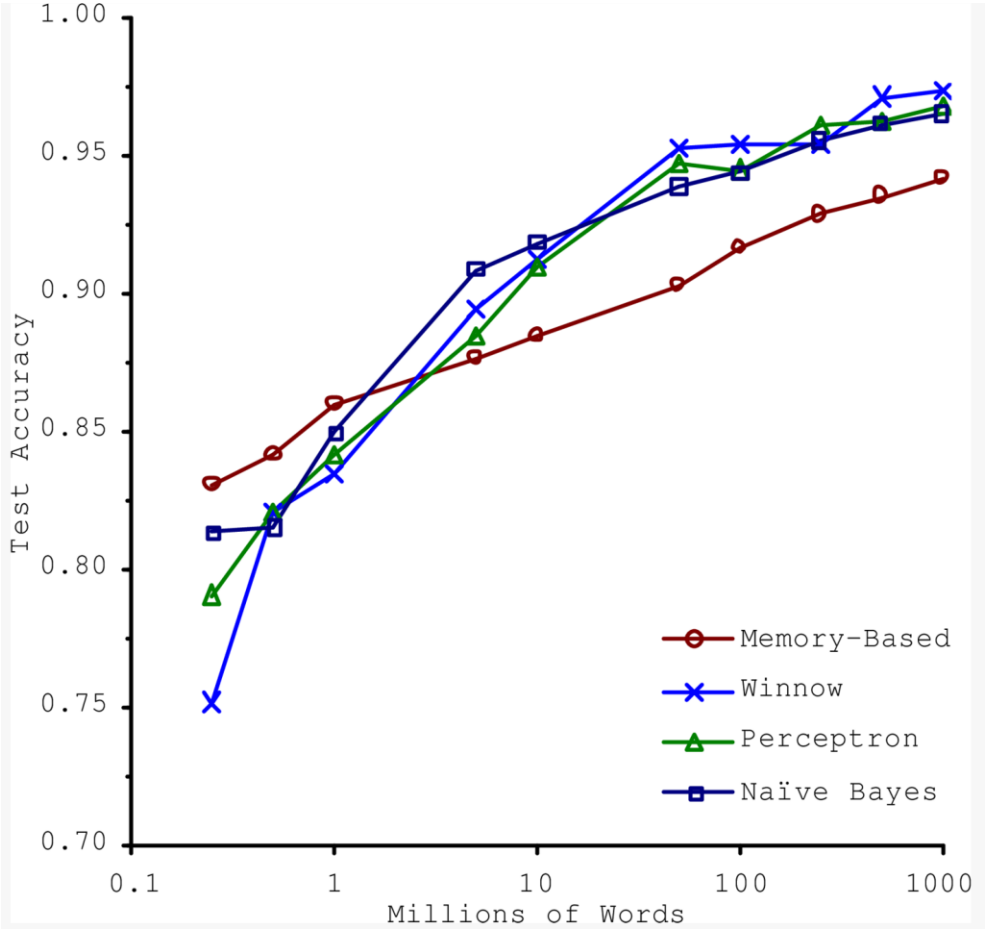


https://snorkel.ai/data-centric-ai/

2. Constructing a dataset - Insufficient Quantity of Training Data

1. Insufficient Quantity of Training Data

- You may have to resort to collecting and annotating new datasets yourself
- In 2001, Microsoft showed that different algorithms performed almost identically well on problems of natural language disambiguation once they were given enough data
- The idea that data matters more than algorithms for complex problems was popularized by a paper titled "The Unreasonable Effectiveness of Data" in 2009



Constructing a dataset - Nonrepresentative Training Data

2. Nonrepresentative Training Data

- In order to generalize well, it is crucial that your training data be representative of the new cases you want to generalize to

- To build a system to recognize pop music videos. One way to build your training set is to search for “pop music” on YouTube and use the resulting videos. But this assumes that YouTube’s search engine returns a set of videos that are representative

- It is crucial to use a training set that is representative of the cases you want to generalize to. This is often harder than it sounds: if the sample is too small, you will have *sampling noise* (i.e., nonrepresentative data as a result of chance), but even very large samples can be nonrepresentative if the sampling method is flawed. This is called *sampling bias*

Constructing a dataset - Nonrepresentative Training Data

- Models can only make sense of inputs that are similar to what they've seen before
  - Suppose you're developing an app where users can take pictures of a plate of food to find out the name of the dish. You train a model using pictures from an image-sharing social network that's popular with foodies. Your accuracy on the test set was well over 90%!
  - Come deployment time, feedback from angry users starts rolling in: your app gets the answer wrong 8 times out of 10. What's going on?
  - A quick look at user-uploaded data reveals that mobile picture uploads of random dishes from random restaurants taken with random smartphones look nothing like the professional-quality pictures you trained the model on: your training data wasn't representative of the production data!

Constructing a dataset - Nonrepresentative Training Data

- If possible, collect data directly from the environment where your model will be used. A movie review sentiment classification model should be used on new IMDB reviews, not on STEAM reviews, nor on Twitter (X)
  - If you want to rate the sentiment of a tweet, start by collecting and annotating actual tweets from a similar set of users as those you're expecting in production
  - If it's not possible to train on production data, then make sure you fully understand how your training and production data differ, and that you are actively correcting for these differences

https://www.reddit.com/r/IndieGaming/comments/5rftfn/that_feel_when_copypasta_jokes_ding_your_steam/

Constructing a dataset - Nonrepresentative Training Data

- A related phenomenon you should be aware of is *concept drift*. Concept drift occurs when the properties of the production data change over time, causing model accuracy to gradually decay
  - A music recommendation engine trained in the year 2013 may not be very effective today. Likewise, the IMDB dataset you worked with was collected in 2011, and a model trained on it would likely not perform as well on reviews from 2024 compared to reviews from 2012, as vocabulary, expressions, and movie genres evolve over time!
  - Concept drift is particularly acute in adversarial contexts like credit card fraud detection, where fraud patterns change practically every day. Dealing with fast concept drift requires constant data collection, annotation, and model retraining

Constructing a dataset - Poor-Quality Data

3. Poor-Quality Data

- If your training data is full of errors, outliers, and noise (e.g., due to poor-quality measurements), it will make it harder for the system to detect the underlying patterns

- It is worth the effort to spend time *cleaning up your training data*. Most data scientists spend a significant part of their time doing just that
  1. If some instances are clearly outliers, it may help to simply discard them or try to fix the errors
  2. If some instances are missing a few features (e.g., some customers did not specify their age), you must decide whether you want to ignore this attribute altogether, ignore these instances, fill in the missing values, or train one model with the feature and one model without it
  3. Duplicate examples. For example, a server mistakenly uploaded the same logs twice

Constructing a dataset - Poor-Quality Data

- Reliability refers to the degree to which you can trust your data. A model trained on a reliable dataset is more likely to yield useful predictions than a model trained on unreliable data. In measuring reliability, you must determine:
  1. How common are label errors? For example, if your data is labeled by humans, sometimes humans make mistakes
  2. Are your features noisy? For example, GPS measurements fluctuate. Some noise is okay. You'll never purge your data set of all noise. You can collect more examples too
  3. Is the data properly filtered for your problem? For example, should your dataset include search queries from bots? If you're building a spam-detection system, then likely the answer is yes, but if you're trying to improve search results for humans, then no

Constructing a dataset - main challenges of data science

4. Irrelevant Features

- "Garbage in, garbage out". Your system will only be capable of learning if the training data contains enough relevant features and not too many irrelevant ones. The so called *feature engineering*, involves the following steps:

1. *Feature selection* (selecting the most useful features to train on among existing features)
2. *Feature extraction* (combining existing features to produce a more useful one - dimensionality reduction algorithms can help)
3. Creating new features by gathering new data

3. Collect a dataset

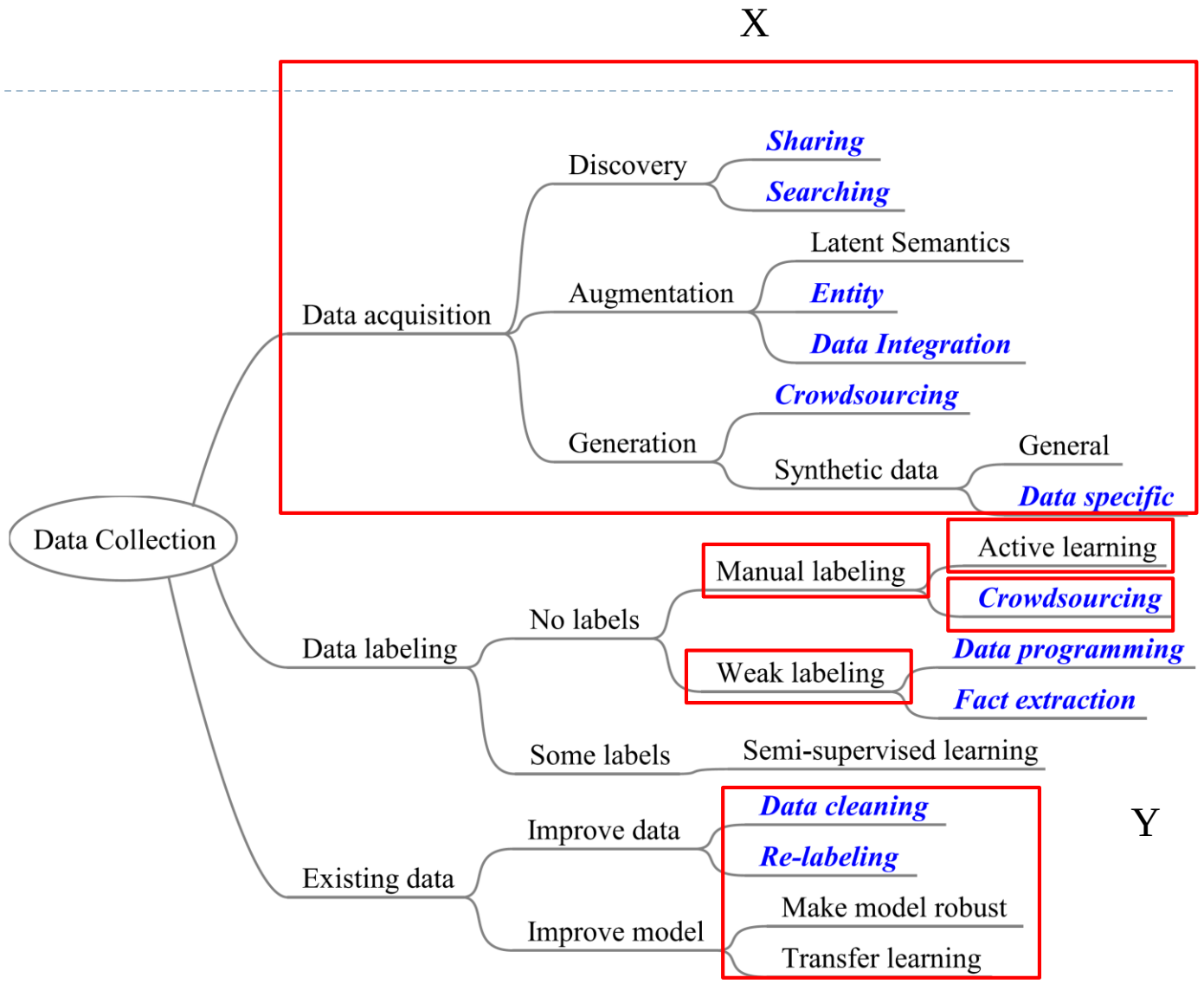
- Once you understand the nature of the task and you know what your inputs and targets are going to be, it's time for data collection
  - The photo search engine project requires you to first select the set of labels you want to classify - you settle on 10,000 common image categories. Then you need to manually tag hundreds of thousands of your past user-uploaded images with labels from this set
  - For the music recommendation engine, you can just use the "likes" of your users. No new data needs to be collected. Likewise for the click-through-rate prediction project: you have an extensive record of click-through rate for your past ads, going back years
  - For the cookie-flagging model, you will need to install cameras above the conveyor belts to collect tens of thousands of images, and then someone will need to manually label these images. You should be able to train people to do it

Collect a dataset

- Short summary
  - If you're doing supervised learning, then once you've collected inputs (such as images) you're going to need annotations for them (such as tags for those images) - the targets you will train your model to predict
    - Sometimes, annotations can be retrieved automatically like the previous examples
  - Check for target leaking: the presence of features in your data that provide information about the targets and which may not be available in production. If you're training a model on medical records to predict whether someone will be treated for cancer in the future, and *the records include the feature "this person has been diagnosed with cancer*," then your targets are being artificially leaked into your data
    - Always ask yourself, is every feature in your data something that will be available in the same form in production?

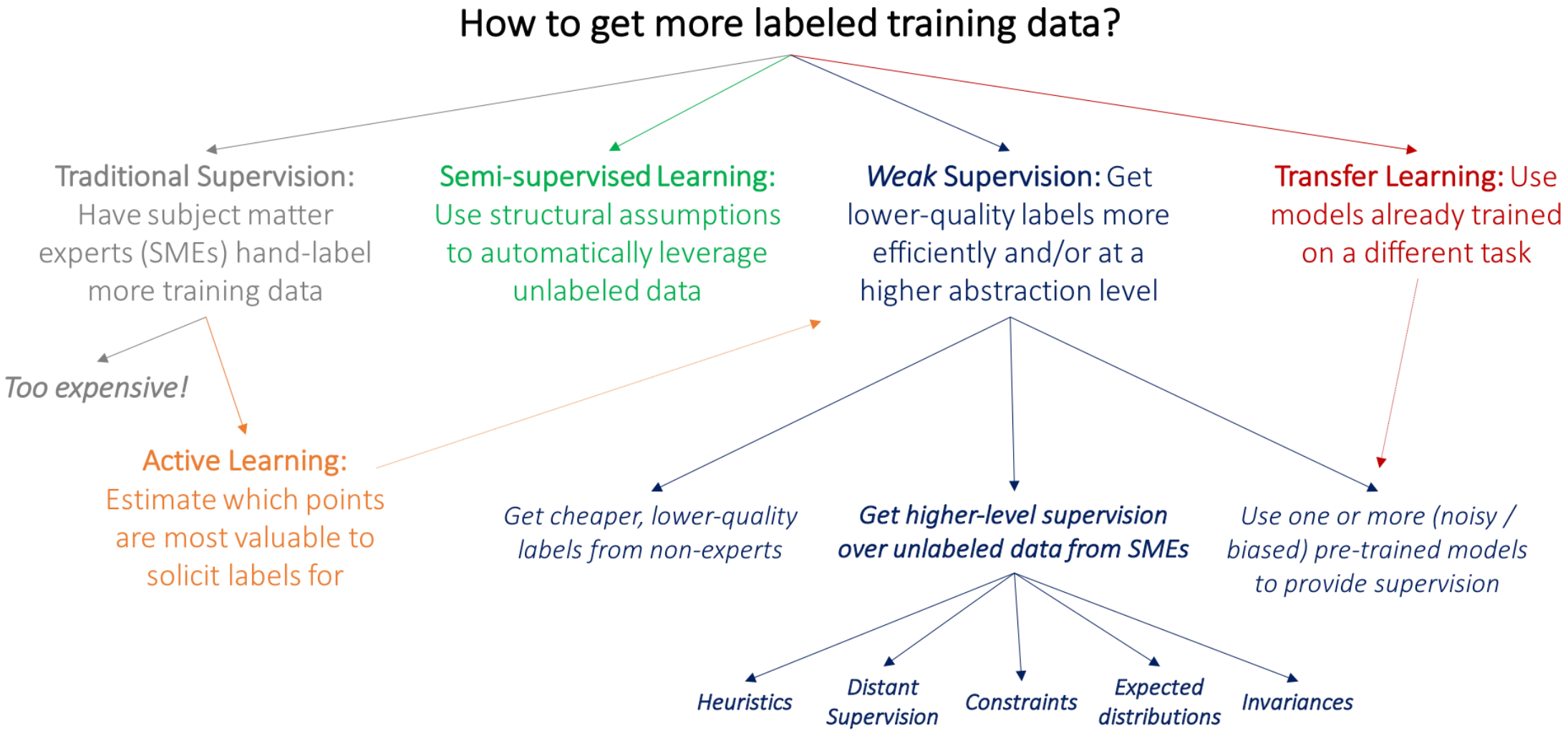
Collect a dataset

- Leverage existing search engine and public API
  - Many organizations provide public APIs for accessing their data - for example, the Twitter API or the NY Times API
- Crawler may be needed for gathering data



https://arxiv.org/pdf/1811.03402.pdf

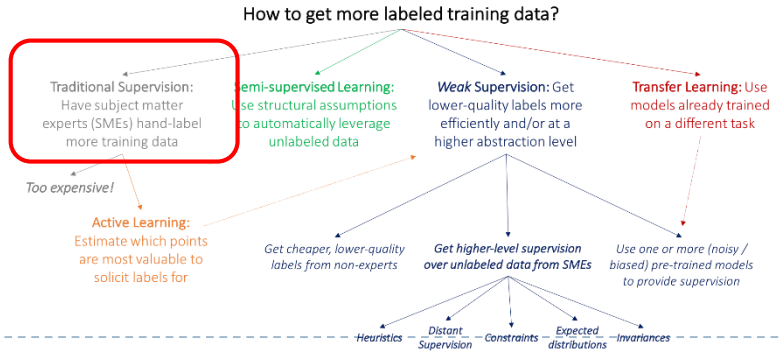
How to construct a supervised dataset?



https://www.snorkel.org/blog/weak-supervision

Labeling the dataset

- Your data annotation process will determine the quality of your targets, which in turn determine the quality of your model. Carefully consider the options you have available:
  - Should you annotate the data yourself?
  - Should you use a crowdsourcing platform like Mechanical Turk to collect labels?
  - Should you use the services of a specialized data-labeling company?



Labeling the dataset

- To pick the best option, consider the constraints you're working with:
  - Do the data labelers need to be experts, or could anyone annotate the data? The labels for a cat-versus-dog image classification problem can be selected by anyone, but those for a dog breed classification task require specialized knowledge. Meanwhile, annotating CT scans of bone fractures pretty much requires a medical degree!

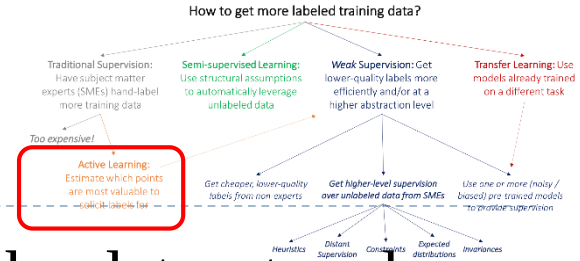
- If you decide to label your data in-house, ask yourself what software you will use to record annotations. Productive data annotation software will save you a lot of time, so it's worth investing in it early in a project

Manual labeling - Workflow setup

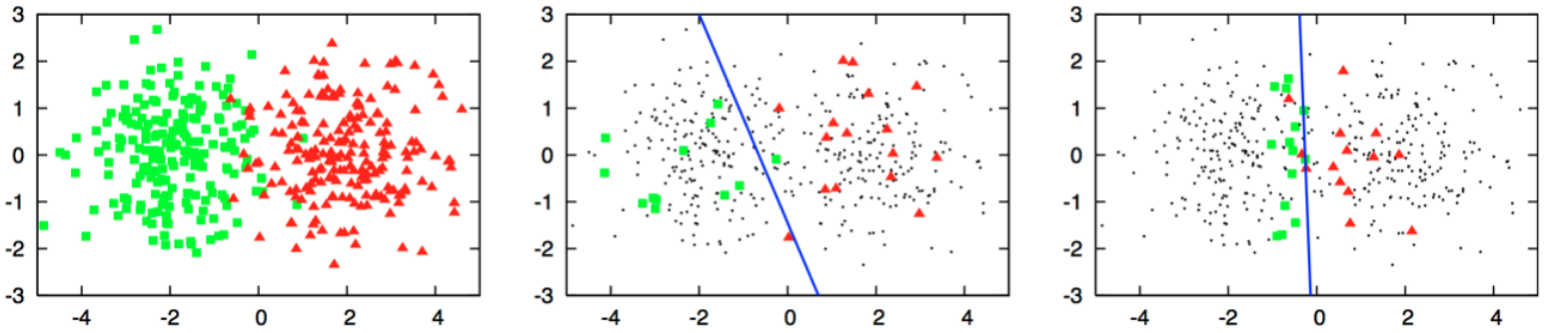
- Compose highly detailed labeling instructions for annotators
  - Examples of each labeling scenario
  - The course of action for discrepancies
- Create a quality assurance (QA) workflow
  - Separate from labeling workflow
  - Consensus algorithm or review by others

https://madewithml.com/courses/mlops/labeling/

4. Active learning (AL)



- Many framework employ active learning to iteratively label the dataset and evaluate the model
  - Learning algorithms can actively query the user/experts for labels. This type of iterative supervised learning is called *active learning*.
  - The number of examples to learn a concept can often be much lower than the number required in normal supervised learning!



All 400 samples labeled

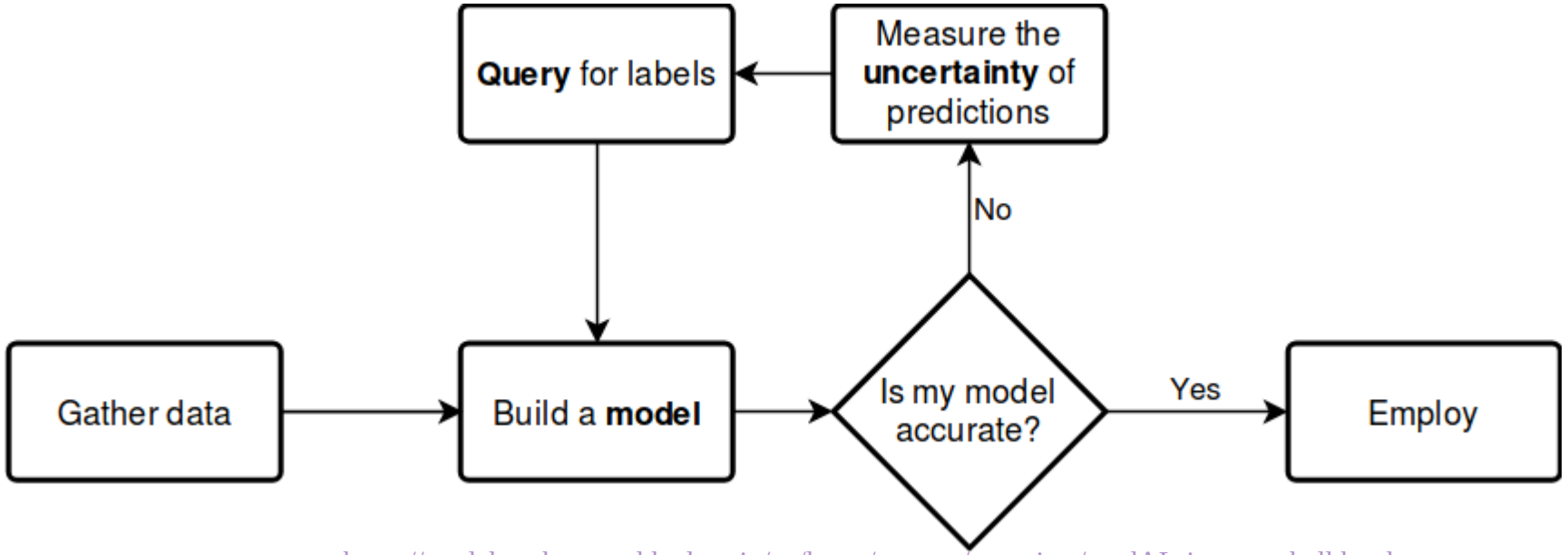
30 samples labeled randomly (acc = 0.7)

30 samples labeled w/ uncertainty sampling (acc = 0.9)

https://madewithml.com/courses/mlops/labeling/

Active learning

- In general, AL is a framework allowing you to increase performance by intelligently querying you to label the most informative instances
  - The key components of any workflow are the uncertainty measure you use and the query strategy you apply to request labels

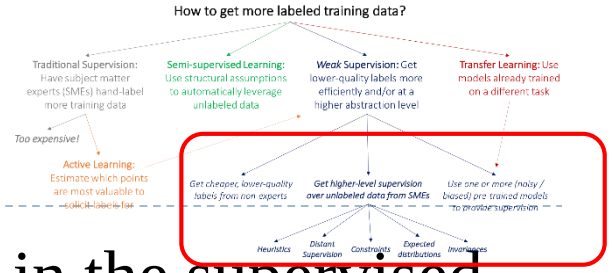


https://modal-python.readthedocs.io/en/latest/content/overview/modAL-in-a-nutshell.html

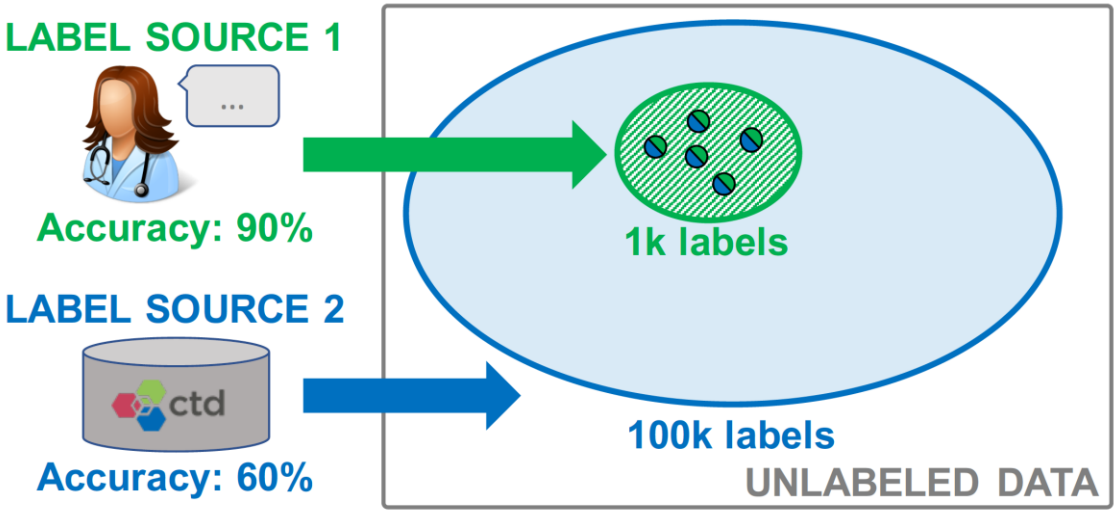
Active learning – classification example

- *Classification uncertainty* is the uncertainty of classification defined by
  - $U(x) = 1 - P(\hat{y}|x)$
  - When querying for labels based on this measure, the strategy selects the sample with the *highest uncertainty*
- *Classification margin* is the difference in probability of the first and second most likely prediction, that is, it is defined by
  - $M(x) = P(\hat{y}_1|x) - P(\hat{y}_2|x)$
  - When querying for labels, the strategy selects the sample with the *smallest margin*, since the smaller the decision margin is, the more unsure the decision
- *Classification entropy* is defined by
  - $H(x) = -\sum_k p_k \log(p_k)$ where p_k is the probability of the sample belonging to k-th class
  - The closer the distribution to *uniform (higher entropy)*, the larger the entropy and will be selected as query

5. Weak supervision



- In the weak supervision setting, our objective is the same as in the supervised setting, however instead of a ground-truth labeled training set we have:
  - Unlabeled data $X_u = x_1, \ldots, x_N$
  - One or more weak supervision sources $\tilde{p}_i(y|x), i = 1 \ldots M$ provided by an expert, such that each one has:
    - A **coverage set** C_i, which is the set of points x over which it is defined
    - An accuracy, defined as the expected probability of the true label y^* over its coverage set, which we assume is < 1.0



https://www.snorkel.org/blog/snorkel-programming

Weak supervision

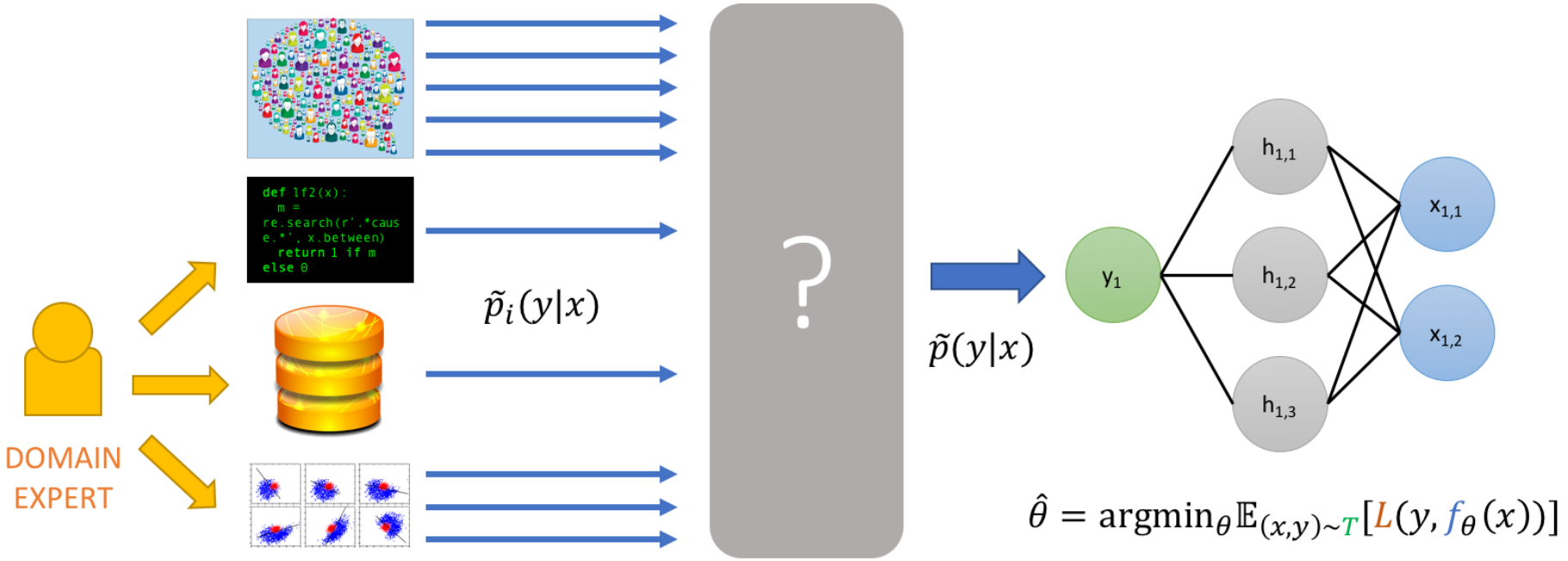
- In general, we are motivated by the setting where these weak label distributions serve as a way for human supervision to be provided more cheaply and efficiently:
  - **Higher-level, less precise** supervision (e.g. heuristic rules, expected label distributions)
  - **Cheaper, lower-quality** supervision (e.g. crowdsourcing)
  - Taking opportunistic advantage of **existing resources** (e.g. knowledge bases, pre-trained models)

Pattern Matching	If a phrase like "send money" is in a email
Boolean Search	If unknown_sender AND foreign_source
DB Lookup	If sender is in our Blacklist.db
Heuristics	If SpellChecker finds 3+ spelling errors
Legacy System	If LegacySytem votes spam
Third Party Model	If TweetSpamDetector votes spam
Crowd Labels	If Worker #23 votes spam

https://snorkel.ai/programmatic-labeling/

Weak supervision

- The core technical challenge is to unify and model these disparate sources
  - Refer to https://dcai.csail.mit.edu/2024/dataset-creation-curation/



https://www.snorkel.org/blog/weak-supervision

Conclusion

- When you take on a new machine learning project, first define the problem at hand:
  - Understand the broader context of what you're setting out to do—what's the end goal and what are the constraints?
  - Collect and annotate a dataset; make sure you understand your data in depth
  - Choose how you'll measure success for your problem—what metrics will you monitor on your validation data?
  - Active learning and weak supervision are two crucial techniques in the supervised setting
- If you would like to construct dataset for different kinds of tasks, refer to the libraries listed in appendix
- Once you understand the problem and you have an appropriate dataset, you can then *prepare and clean* your data!

References

[1] Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition Chapter 1

[2] Deep learning with Python, 2nd Edition Chapter 6

[3] https://developers.google.com/machine-learning/problem-framing/

[4] https://developers.google.com/machine-learning/data-prep/

[5] https://madewithml.com/courses/mlops/labeling/

[6] https://www.snorkel.org/blog/weak-supervision

[7] https://dcai.csail.mit.edu/2024/dataset-creation-curation/

Appendix

Resources and Libraries

- Crawler
  - Scrapy: An open source framework for extracting the data you need from website
  - Beautiful Soup: a Python library for pulling data out of HTML and XML files
- Data augmentation
  - https://madewithml.com/courses/mlops/augmentation/
  - albumentations
  - nlpaug
- Labeling data
  - https://landing.ai/blog/tips-for-a-data-centric-ai-approach
  - https://github.com/EthicalML/awesome-production-machine-learning?tab=readme-ov-file#data-labelling-and-synthesis

Resources and Libraries

- General
  - Labelbox: the data platform for high quality training and validation data for AI applications.
  - Label Studio: a multi-type data labeling and annotation tool with standardized output format
  - modAL: a modular active learning framework for Python
  - Snorkel: a modular libraries for performing weak supervision, augmentation or slicing
  - Cleanlab: The standard data-centric AI package for data quality and machine learning with messy, real-world data and labels. CROWDLAB, ActivlLab
- Natural language processing
  - Doccano: an open source text annotation tool for text classification, sequence labeling and sequence to sequence tasks
  - BRAT: a rapid annotation tool for all your textual annotation needs

Resources and Libraries

- Computer vision
  - LabelImg: a graphical image annotation tool and label object bounding boxes in images
  - CVAT: a free, online, interactive video and image annotation tool for computer vision
  - VoTT: an app for building end-to-end object detection models from images and videos
  - remo: an app for annotations and images management in computer vision
  - pigeonXT: quickly annotate a dataset from the comfort of your Jupyter notebook
- Audio
  - Audino: an open source audio annotation tool for voice activity detection (VAD), speaker identification, automated speech recognition, emotion recognition tasks, etc

Framing the problem: Prediction or inference

- Prediction: In many situations, a set of inputs X are readily available, but the output Y cannot be easily obtained; we can then use $\hat{f}$ as follows

$$\hat{Y} = \hat{f}(X)$$

  - In this setting, $\hat{f}(X)$ is often treated as a black box
  - There will be reducible and irreducible error
    - Reducible error can be potentially improved by using the most appropriate machine learning technique to estimate f
    - Irreducible error may contain unmeasured variables that are useful in predicting Y: since we don't measure them, f cannot use them for its prediction. It may also contain unmeasurable variation
  - We will focus on the part of the reducible error

Framing the problem: Prediction or inference

- Inference: We are often interested in understanding the association between Y and $X_1, \ldots, X_P$. In this situation, we wish to estimate f, but our goal is not necessarily to make predictions for Y.
  - Which predictors are associated with the response?
  - What is the relationship between the response and each predictor?
  - Can the relationship between Y and each predictor be adequately summarized using a linear equation, or is the relationship more complicated?
- We will see a number of examples that fall into the prediction setting, the inference setting, or a combination of the two

Note on ethics

- You may sometimes be offered ethically dubious projects, such as "building an AI that rates the trustworthiness of someone from a picture of their face."
  - First of all, the validity of the project is in doubt: it isn't clear why trustworthiness would be reflected on someone's face. Second, such a task opens the door to all kinds of ethical problems. Collecting a dataset for this task would amount to recording the biases and prejudices of the people who label the pictures. The models you would train on such data would merely encode these same biases into a black-box algorithm that would give them a thin veneer of legitimacy
  - Your model would be laundering and operationalizing at scale the worst aspects of human judgement, with negative effects on the lives of real people
- Technology is never neutral. If your work has any impact on the world, this impact has a moral direction: technical choices are also ethical choices. Always be deliberate about the values you want your work to support

Tabular data

- It is a 2-dimensional data structure that can store data of different types (including characters, integers, floating-point values, categorical data and more) in columns
  - It is similar to a spreadsheet, a SQL table or the data.frame in R
    - https://pandas.pydata.org/docs/getting_started/index.html
    - https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf
  - Rows indicating records (cases) and columns indicating features (variables)

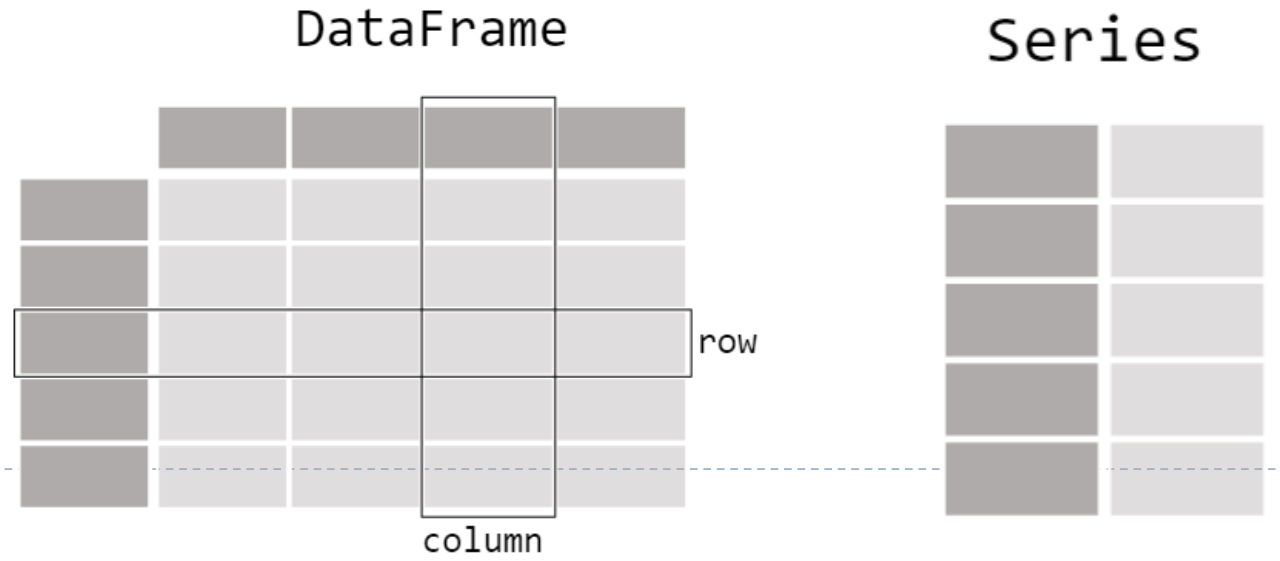
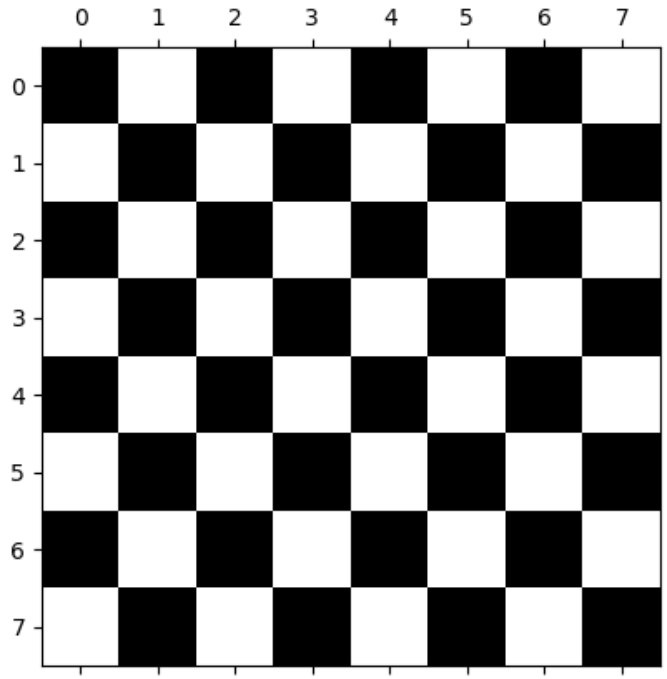


Image data

- Images are represented as multi-dimensional arrays
  - The origin is located in top-left corner
  - 0 is for black value and 255 (or 1.0) is for white value

Image:	np.ndarray
pixels:	array values: a[2, 3]
channels:	array dimensions
image encoding:	dtype (np.uint8, np.float)
filters:	functions (numpy, skimage, scipy)

Image type	Coordinates
2D grayscale images	(row, column)
2D multichannel images	(row, column, channel)
batch of 2D grayscale images	(batch, row, column)
2D multichannel images	(batch, row, column, channel)



Three essential computer vision tasks

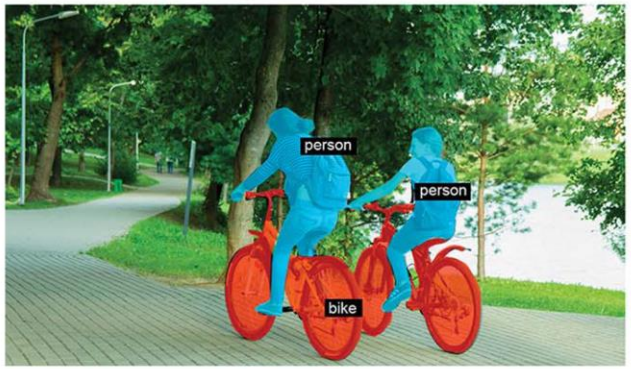
Single-label multi-class classification

- (•) Biking
- ( ) Running
- ( ) Swimming

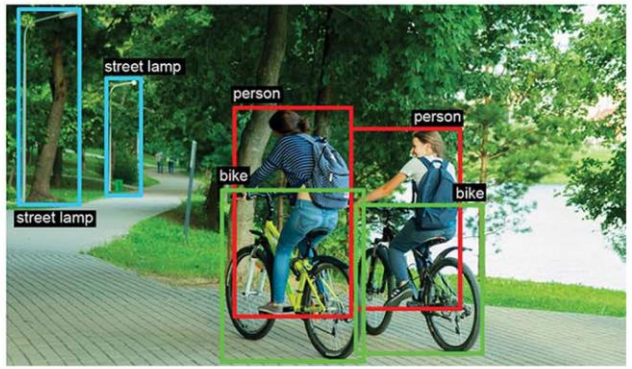
Multi-label classification

- [x] Bike
- [x] Person
- [ ] Boat
- [x] Tree
- [ ] Car
- [ ] House

Image segmentation

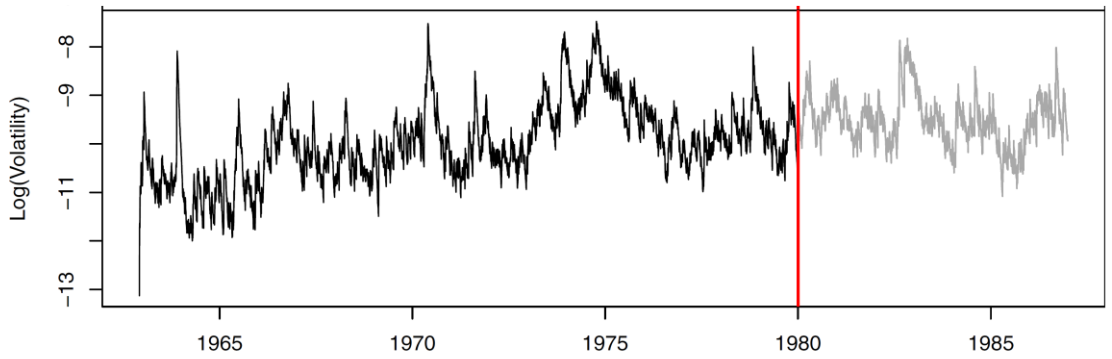


Object detection

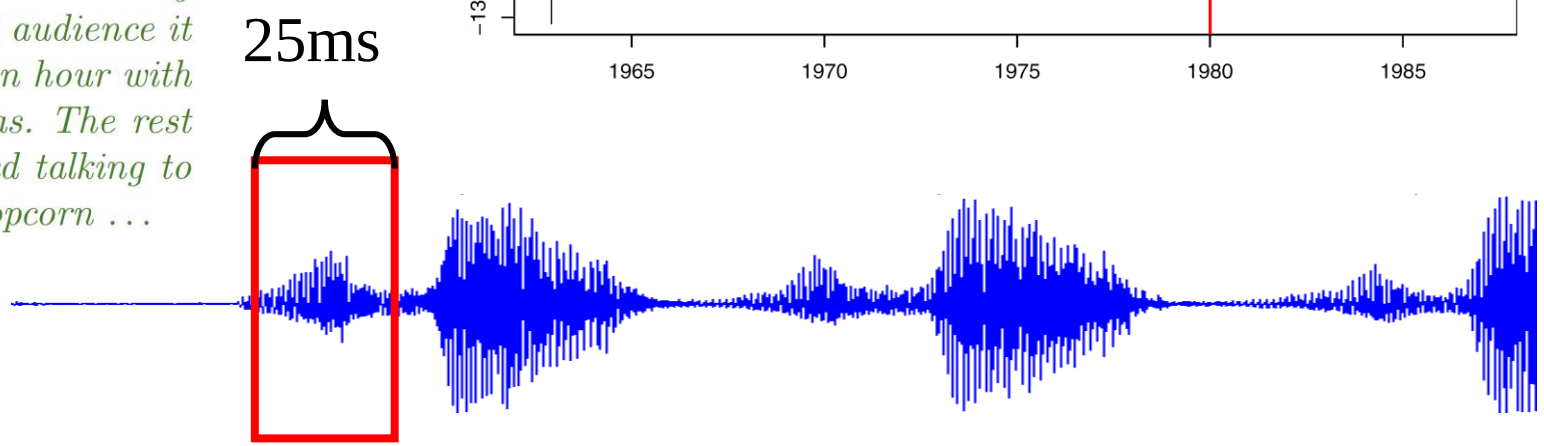


Sequence data

- Often data arise as sequences:
  - Documents are sequences of words, and their *relative positions* have meaning - NLP
  - Time-series such as weather data or financial indices
  - Recorded speech or music
  - Handwriting, such as doctor's notes



This has to be one of the worst films of the 1990s. When my friends & I were watching this film (being the target audience it was aimed at) we just sat & watched the first half an hour with our jaws touching the floor at how bad it really was. The rest of the time, everyone else in the theater just started talking to each other, leaving or generally crying into their popcorn . . .



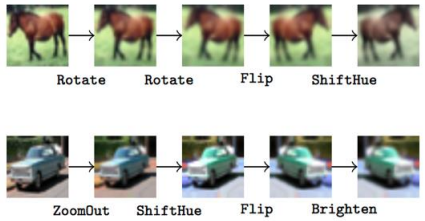
Weak supervision

- These weak label distributions could take one of many forms:
  1. **Weak Labels**: The weak label distributions could be deterministic functions – we might just have a set of noisy labels for each data point in C_i. These could come from crowd workers, the output of heuristic rules, or the result of distant supervision
  2. **Constraints**: We can also consider constraints represented as weak label distributions
  3. **Distributions**: We might also have direct access to a probability distribution. For example, we could have the posterior distributions of one or more weak (i.e. low accuracy/coverage) or biased classifiers, such as classifiers trained on different data distributions as in the *transfer learning* setting
  4. **Invariances**: Given a small set of labeled data, we can express functional invariances as weak label distributions – In this way we view techniques such as *data augmentation* as a form of weak supervision as well

Data augmentation

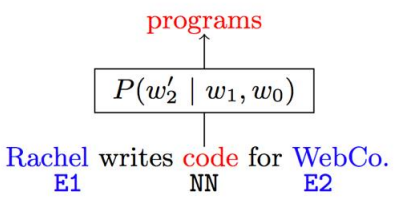
- A key challenge when training models is collecting a large, diverse dataset that sufficiently captures variability observed in the real world. Due to the cost of collecting and labeling data, data augmentation has emerged as an alternative
  - The central idea in data augmentation is to transform examples in the dataset in order to generate additional augmented examples that can then be added to the data. These additional examples typically increase the diversity of the data seen by the model, and provide additional supervision to the model

Images



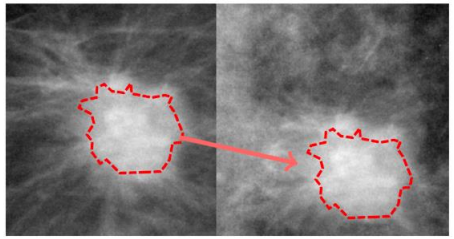
- Rotations
- Scaling / Zooms
- Brightness
- Color Shifts
- Etc…

Text



- Synonymy
- Positional Swaps
- Etc…

Medical



Domain-specific transformations.
Ex:
1. *Segment tumor mass*
2. *Move*
3. *Resample background tissue*
4. *Blend*

https://www.snorkel.org/blog/tanda

CROWDLAB (Classifier Refinement Of croWDsourced LABels)

- If we also want to account for feature values measured for each example (upon which the crowdsourced labels are anyway based), a natural way is to train a classification model to predict the labels for any example.
  - Such a model M can be trained with consensus labels derived via any method, such as simple majority-vote
- CROWDLAB estimates are based on the intuition that **we should rely on the classifier's prediction more for examples that are labeled by few annotators but less for examples labeled by many annotators** (with many annotations, simple agreement already provides a good confidence measure).

CROWDLAB (Classifier Refinement Of croWDsourced LABels)

- CROWDLAB relies on predicted class probabilities from any trained classifier model: $p_M = \hat{p}_{M(Y_i|X_i)}$. These are ideally held-out predictions produced for each example in a dataset via cross-validation.
- For a particular example i, we form an analogous "predicted" class probability vector p_i based on the annotation Y_{ij} from each annotator that labeled this example. CROWDLAB estimates a distribution for the true label via a simple weighted combination: $\Pr(Y_i) = \frac{1}{Z}\left(w_M \cdot p_M + \sum_{j \in J_i} w_j \cdot p_j\right)$
- The inter-annotator agreement should serve as a good proxy for consensus label quality when many annotators were involved