CAI 4104/6108 – Machine Learning Engineering:

ML Engineering (1)

Prof. Vincent Bindschaedler

Spring 2024

Administrivia

- Homework 0 is due **today** (by 11:59PM)

- **No class** on Friday (1/19)
  - Exercise 1 — I will **pre-record** it

Background & Notation

- Scalars: a, b, c (if domain not specified, assume real number)
- Vectors: $\boldsymbol{a}$, $\boldsymbol{b}$
  - $a^{(i)}$ or a_i is the i^{th} element of the vector $\boldsymbol{a}$; e.g.: if $\boldsymbol{b}$ =[7, 3] then b_1=7 and b_2=3
  - $|| \boldsymbol{a} ||$ is the Euclidean norm of the vector $\boldsymbol{a}$, i.e.: $|| \boldsymbol{a} || = [\sum_i (a_i)^2]^{1/2}$
    - More generally: $|| \boldsymbol{a} ||_p = (\sum_i |a_i|^p)^{1/p}$ (called L_p-norm)
  - Dot product: $\boldsymbol{a}\,\boldsymbol{b} = \sum_i a_i\, b_i$
  - $\boldsymbol{a}^T$ is the transpose of $\boldsymbol{a}$
- Matrices: $\boldsymbol{A}$, $\boldsymbol{B}$
  - $\boldsymbol{a_i}$ is the i^{th} row of $\boldsymbol{A}$
  - a_{ij} is the j^{th} element of the i^{th} row of $\boldsymbol{A}$
  - $\boldsymbol{A}^T$ is the transpose of $\boldsymbol{A}$
- Sets: $\mathcal{A}$, $\mathcal{B}$
  - $[n] = \{1, 2, ..., n\}$
  - Union: $\mathcal{A} \cup \mathcal{B}$; intersection $\mathcal{A} \cap \mathcal{B}$; set difference: $\mathcal{A} \setminus \mathcal{B}$; cardinality: $|\mathcal{A}|$

Background & Notation

- Functions: f: $\mathcal{X} \rightarrow \mathcal{Y}$
  - $\mathcal{X}$: domain; $\mathcal{Y}$: co-domain, range, or image set
  - Functions can have a global minimum (or maximum) and several local minimum (or maximum)
  - Derivatives and gradients
    - $f'(x)$ denotes the derivative of $f(x)$ [e.g., if $f(x)=x^2$ then $f'(x)=2x$]
    - If f takes multiple inputs (e.g., $f(x,y)$ or $f(\boldsymbol{x})$), then the gradient of f, denoted ∇f is the vector of partial derivatives
    - For example for $f(x,y)$: $\nabla f = [\partial f/\partial x, \partial f/\partial \mathrm{y}]$
- Random variables: X: $\Omega \rightarrow \mathcal{R}$
  - X is a function from the sample space to some output set $\mathcal{R}$
  - Discrete X: $\Pr\{X = x\}$ denotes the probability that X takes value $x \in \mathcal{R}$
    - The probability distribution of X is called the probability mass function (pmf)
  - Continuous X: $\Pr\{X = x\}=0$ for any specific value $x \in \mathcal{R}$
    - We have a probability density function (pdf) denoted $p(x)$
  - Expectation: $\mathbb{E}[X]$; Variance: $\mathrm{Var}(X)$

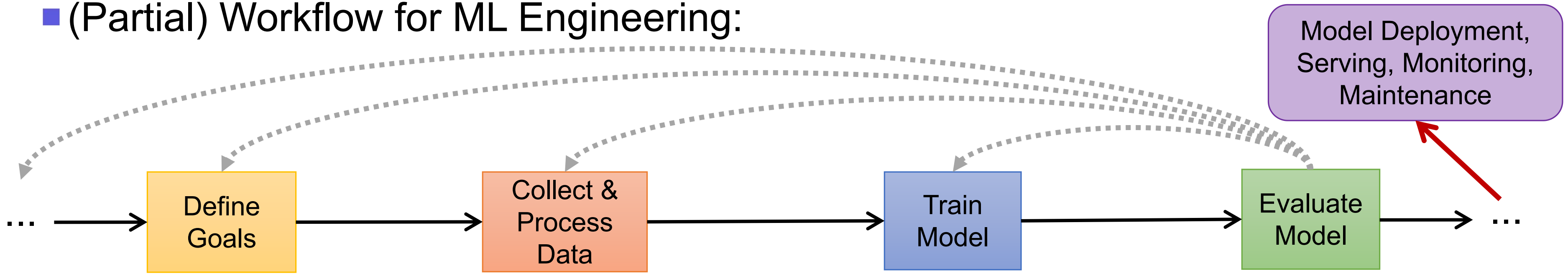
Background & Notation

- Machine Learning
  - Training dataset with n examples and m features: $\boldsymbol{X}$ ($n \times m$ matrix)
    - $\boldsymbol{x_i}$ represents the i^{th} example; $x_{i,j}$ or $x_i^{(j)}$ is the j^{th} feature value
  - If supervised learning, then there is a corresponding set of labels $\boldsymbol{y}$ ($n \times 1$ vector)
    - If doing classification, then the label of the i^{th} example y_i would be an integer (or encoded as one)
    - If doing regression, then y_i could be a real number
  - Alternatively the training dataset may be denoted as: $\{(\boldsymbol{x_1}, y_1),(\boldsymbol{x_2}, y_2),\ldots,(\boldsymbol{x_n}, y_n)\}$

  - Model: $\boldsymbol{\theta} \in \boldsymbol{\Theta}$
    - Prediction function $f_{\theta}(\boldsymbol{x})$ or $h_{\theta}(\boldsymbol{x})$
    - Loss function or cost function: L_{θ} or $J(\boldsymbol{\theta})$
      - For example: L_2 loss (aka "squared error loss"): $L = [y - f_{\theta}(\boldsymbol{x})]^2$

Machine Learning Engineering

- (Partial) Workflow for ML Engineering:



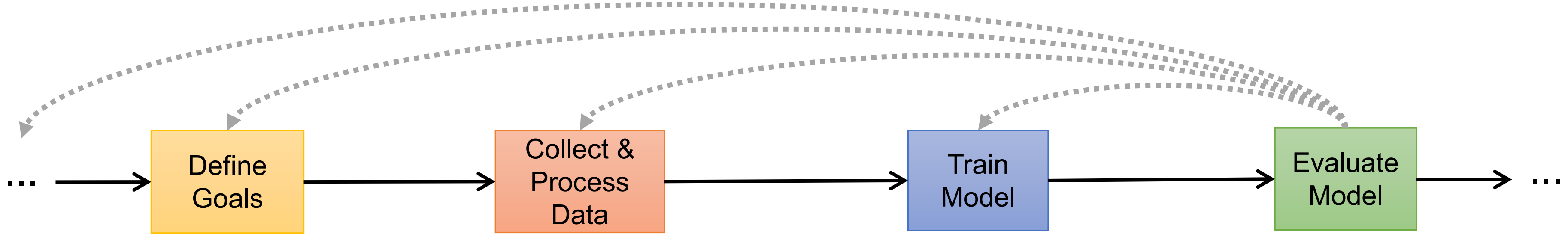
- When to use ML?
- What problems can it solve?
- Why do ML projects fail?
- What can ML not do?

- How to get data?
- Dealing with noisy data?
- How much data is needed?
- How to extract features?

- What is the task?
- What kind of model?
- How to tune hyperparameters?

- Best practices to evaluate the model?
- What metric(s) to use?
- What is the baseline?
- Underfitting/overfitting?

- (Partial) Workflow for ML Engineering:



- When to use ML?
- What problems can it solve?
- Why do ML projects fail?
- What can ML not do?

- How to get data?
- Dealing with noisy data?
- How much data is needed?
- How to extract features?

- What is the task?
- What kind of model?
- How to tune hyperparameters?

- Best practices to evaluate the model?
- What metric(s) to use?
- What is the baseline?
- Underfitting/overfitting?

When Should We Use ML?

- Consider using ML when:
  - The problem is too complex for hardcoding rules
    - There could be too many rules, the rules may be unknown/hard to specify, or there could be too many parameters determine the rules
    - For example: what rules would you use for spam filtering?
  - The problem deals with an unstudied/understudied phenomenon
    - Discovering unknown patterns in the data, maybe patterns difficult for humans to see
    - For example: predicting human behavior
  - The problem calls for automating some decision/prediction
    - In order to reduce human workload or amount of data to look through
    - For example: anomaly detection in intrusion detection system
  - The problem (or some aspect of it) is changing frequently
    - The data for a problem could be constantly changing (e.g., network traffic patterns)
    - We can (re)train model on new data, or we can use online/incremental learning

Many Reasons Not To Use ML

- Cannot get the (right) data or enough of it
- Problem does not require learning from data
- You can solve the problem in other ways
  - By developing new algorithms, using software development, etc.
- Cannot afford the cost of a mistake
- Judged unethical/undesirable to use ML
  - Should we use ML to determine prison/jail time?
- You need explanations, not just predictions
  - The problem requires explaining some phenomon (e.g., physics)
    - ML is **not** magic: it doesn't have deep insights about the natural world!
    - It's (just) a set of techniques and tools to make accurate predictions from data
  - *Note: there is active research on interpretable/explanable ML; but many models are black boxes!*
- Many others...

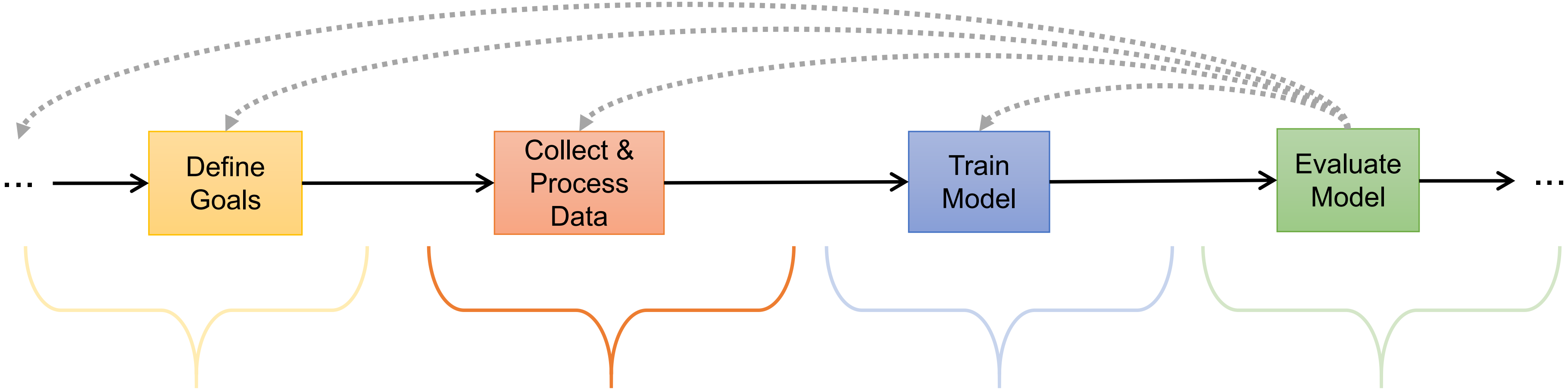
Why Do ML Projects Fail?

- According to VentureBeat AI, 87% of data science projects fail
  - *Source: https://venturebeat.com/2019/07/19/why-do-87-of-data-science-projects-never-make-it-into-production/*

- Why do ML projects fail?
  - Many reasons
    - Not having people with the right expertise; insufficient computing infrastructure
  - Bad data, or no data, or not enough data, or biased data
  - Don’t have the right data; or right data does not exist
  - ML is the wrong approach
  - Goal is technically infeasible
  - Lack of metrics for success or bad metrics
    - E.g., lack of baseline
  - Many others...

Machine Learning Engineering

- (Partial) Workflow for ML Engineering:



- When to use ML?
- What problems can it solve?
- Why do ML projects fail?
- What can ML not do?

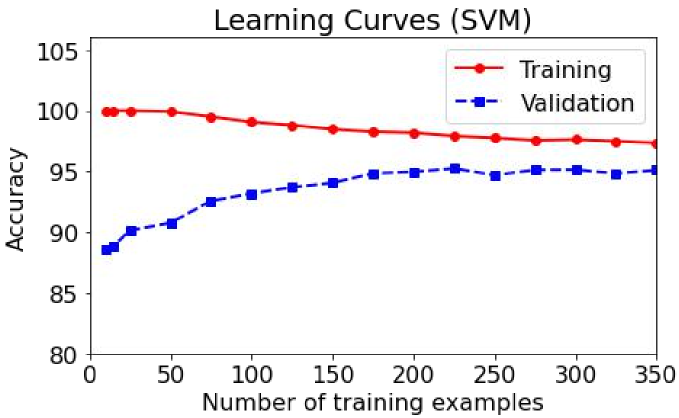
- How to get data?
- Dealing with noisy data?
- How much data is needed?
- How to extract features?

- What is the task?
- What kind of model?
- How to tune hyperparameters?

- Best practices to evaluate the model?
- What metric(s) to use?
- What is the baseline?
- Underfitting/overfitting?

Collecting Data

- Many questions:
  - Where is the data coming from? Is it reliable?
  - Does it need to be anonymized? (What can happen if we don't?)
  - Is the data noisy/incomplete? Is it missing values?
  - Is the data biased in some way?
    - Is the data expected to be similar to the data available after the system is deployed?
- How much data do we need?
  - More than you expect
  - Rules of thumb:
    - More examples/instances than attributes/features
      - *Some models (e.g., SVM) still work with few examples but lots of features*
    - Many examples (e.g., 30+ or 50+) for each class
  - How to answer this in practice? => learning curves
    - You'll be able to try this in assignment1



Dealing With Unclean Data

- The data is noisy
  - For example: some attribute values/features are wrong
    - Some learning algorithms are more sensitive than others
- The data has missing features values
  - Mitigation strategies
    - Algorithmically: use a learning algorithm that can deal with this (e.g., some Decision Trees implementations)
    - Removal: discard examples with missing values
    - (If feature is categorical/nominal) Treat missing as value: replace missing values with a “special value” (e.g., -1)
    - Imputation: use a data imputation technique (e.g., resampling from marginals, replace with average)
- The data contains duplicates
- The data is imbalanced
  - Weighting classes (if allowed by learning algorithm); oversampling; undersampling
- The data is incomplete or not representative?
  - Find better data!

Too Little Data? Too Much Data?

- Data Augmentation
  - Idea: augment the existing training data by using the training data!
    - Can have a **major** impact on model performance
    - But it won't help if you truly don't have enough data!
  - For example: for images: rotation, scaling, crop, flip, etc.
  - Data augmentation techniques are usually specific to the data type

- Data Sampling
  - If you have a lot of data, it may not be practical (and necessary) to use all of it
  - Different strategies for sampling
    - Random sampling: e.g., pick a uniformly random subset of instances, pick each instance independently with some probability p
    - Stratified sampling: divide dataset into groups (strata), then randomly sample from each stratum
      - Note: this can be used to reduce bias (i.e., make the dataset more representative)

Data Augmentation?

- Data Augmentation can be thought of a regularization technique

Single sample

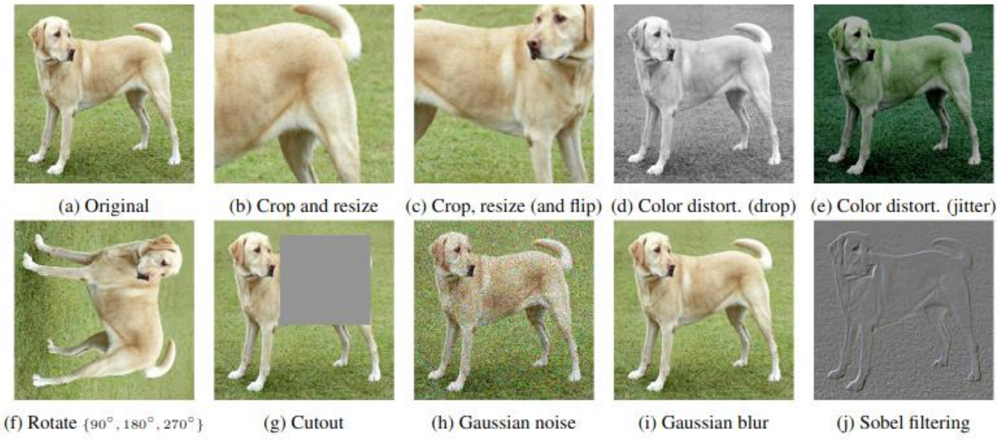
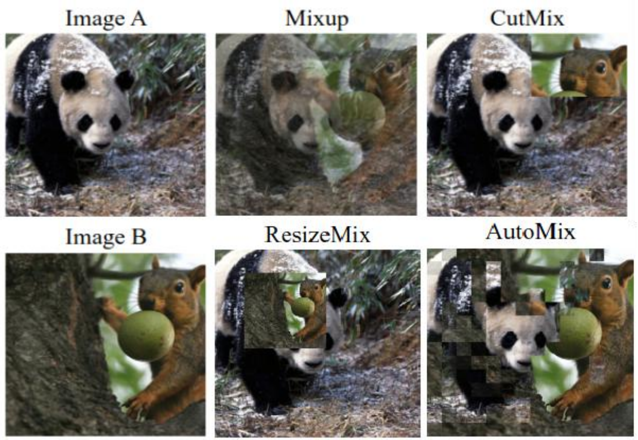


Figure 4. Illustrations of the studied data augmentation operators. Each augmentation can transform data stochastically with some internal parameters (e.g. rotation degree, noise level). Note that we *only* test these operators in ablation, the *augmentation policy used to train our models* only includes *random crop (with flip and resize)*, *color distortion*, and *Gaussian blur*. (Original image cc-by: Von.grzanka)

Source: Chen et al. "A simple framework for contrastive learning of visual representations." ICML 2020.

Multi-sample



Source: Li et al. "Openmixup: Open mixup toolbox and benchmark for visual representation learning." arXiv preprint arXiv:2209.04851.

Extracting Features

- Feature Engineering
  - The process of transforming raw data into a dataset we can use
    - Time consuming process
    - Techniques are often specific to the data type you are dealing with (e.g., feature engineering for text)
  - Goal: select a set of features that are highly predictive
    - But also not correlated among each other. **Why?**
- Common Techniques
  - For numerical features:
    - Binning (aka bucketing): aggregate range of numerical values into discrete bins
    - Normalization: remap values onto a range such as [0,1] (min-max normalization) or [-1,1]
    - Standardization (aka z-score normalization): rescale feature values to follow a *standard normal distribution* (how? => subtract the mean and divide by the standard deviation)
  - For categorical features:
    - One-hot encoding: turn a categorical feature with k possible values into a vector of k binary features
    - Ordinal encoding: if values are ordered/ranked, values can encoded in order (e.g., using integers)

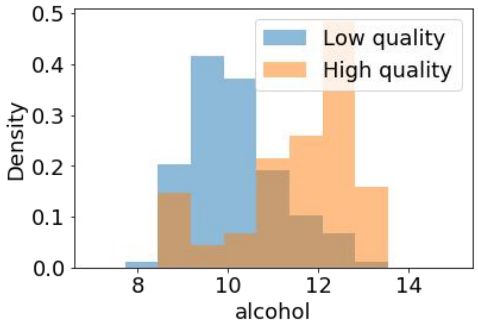
Feature Engineering

- For numerical features:
  - Binning (aka bucketing): aggregate range of numerical values into discrete bins
  - Normalization: remap values onto a range such as [0,1] or [-1,1]
  - Standardization (aka z-score normalization): rescale feature values to follow a *standard normal distribution* (how? => subtract the mean and divide by the standard deviation)
- For categorical features:
  - One-hot encoding: turn a categorical feature with k possible values into a vector of k binary features with hamming weight 1
  - Ordinal encoding: if values are ordered/ranked, values can encoded in order (e.g., as integers)

- Rules of thumb:
  - If you have a mix of numerical and categorical features: use one-hot encoding / ordinal encoding
  - Feature scaling (i.e., normalization & standardization) helps with most learning algorithms
    - For SVM (especially linear SVM), linear models (or if you are doing regularization): you should rescale features!

Best Practices

- Preprocessing, data cleaning, and feature engineering
  - Avoid manipulating the data in ad-hoc (i.e., non-reproducible ways) such as bash scripting, awk, manual editing, etc.
  - For reproducibility, it is best to implement all the steps in a (Python) script
    - Ideally, you want to develop a pipeline
- Visualizing data:
  - Before getting too far into data cleaning or training a model, take a look at the (training) data!
    - Visualizing the data may yield insights into what features to use and what kind of model to use
    - You can also look at feature statistics: mean, standard deviation, min, max, mode, missing values, etc.
  - Visualization examples:
    - Histograms,
    - Heatmaps, correlation plots,
    - Scatter plots, QQ-plots,
    - Class distribution plot per feature



Next Time

- Friday (1/19): Exercise 1
  - ***No Class - Pre-recorded***

- Upcoming:
  - Homework 0 (due today by 11:59pm)

