Introduction to Machine Learning
Lecture 10: ML in Practice

Dr. Gabriel Stanovsky
Slides adapted from Prof. Matan Gavish and Prof. Shai Shalev-Shwartz

May 31, 2022

Suggested reading:

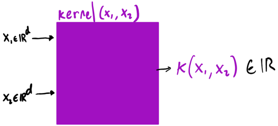
האוניברסיטה העברית בירושלים
THE HEBREW UNIVERSITY OF JERUSALEM
الجامعة العبرية في اورشليم القدس

Advanced Topics in ML

- **Advanced topics**
  - Last Week: Unsupervised learning
  - **Last Week: Kernel methods**
  - **Today: ML in practice**
- **Modern ML**
  - Gradient-based learning
  - Neural networks
- **Summary and Ethics**

Recap: Kernels

- **Kernel Trick**: Learn in high dimension without paying the full price



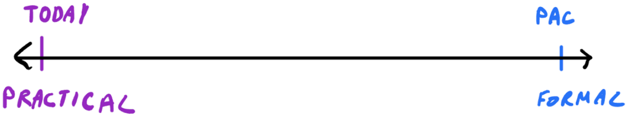
- We discussed two kernels
  - **Polynomial and Gaussian (RBF)**

- Enable linear solvers to do **non-linear separation** in feature space
  - Complexity depends on m, instead of d

- **Kernels lead to very expressive and powerful models**
  - and they're an example of metric learning

Today: ML in Practice

- **ML requires a dual set of skills**
  - Mathematical understanding and formulation
  - Expertise in data processing, coding, presenting results
- **This course aims to give you initial skills in both**
  - Math in most lectures and recitations
  - Hands-on exercises (there's no other way)
- **Today we'll focus on best practices for ML in practice**
  - Back to math next week



This will be a different lecture

- **Many examples**
- **(almost) No math**
- **Somewhat subjective** advice from my experience
  - Much under current research
  - No proofs, you're welcome to question my suggestions

What do I mean by “in practice”?

- Mostly research projects in **academia**, and a little bit of **tech**
  - With strong bias towards NLP
- Luckily, **applied ML is becoming more homogeneous**
  - With a strong ties between fields and across academia and tech
- ML in practice require **practice and expertise**
  - You’ll improve the more projects you take
  - E.g., our hackathon, data challenge, and others

Outline

- **In this course we mostly ignored the approach to a new problem**
  - Instead we focus on giving you an ML toolbelt
- **Next, we'll acknowledge an assumption we made in the course**
  - Then show how in practice things are more complicated
- **We'll give a set of best practices and advise**

Approaching a New Problem: Takeaways

- **Define setup:** Which elements of the problem require ML
- **Understand the features:** Feature and label types & possible ranges
- **Question the labels:** Estimate annotation quality and biases
- **Make sensible partitions:** Each part is representative of the whole

We started each lecture with a given setup.

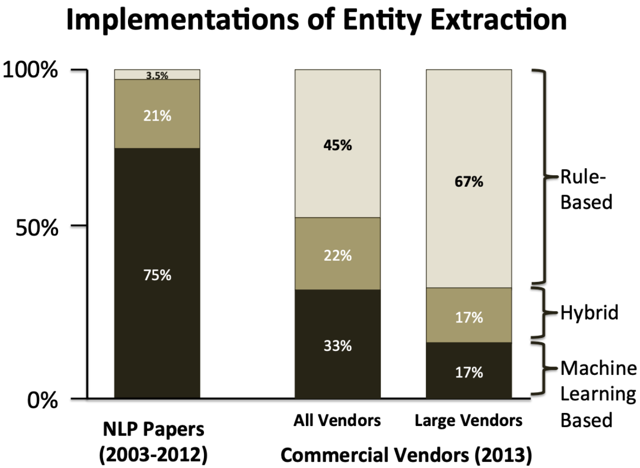
$$S_m \in \mathbb{R}^{m \times d}, \mathcal{Y} = \mathbb{R}$$

$$\hat{w} = \arg\min_{w} \in \mathbb{R}^{d+1}[L_{train}(w) + \lambda \cdot \mathcal{R}(w)]$$

- **This hides much of the (interesting) work**
  - In practice, there's a wide range for defining the setup
  - Can already decide the fate of the project

Do I need ML, and if so for which part?

- **Hype around everything ML and AI**
  - But what you learned before ML still holds
  - Often termed **rule based**
- There's nothing bad about using rule-based approaches
  - **Explainability** is a huge advantage
- **Often a good approach would be a combination of ML and rules**
  - Real-world problems are likely multi-faceted



- Solutions that work in practice often need a combination of ML and rules

Chiticariu et al., 2013

Example: Identifying Punishment in Court Decisions (Wenger et al., 2021)

- **Input**: Court cases in natural language
- **Output**: Months of imprisonment given as punishment
- **Approach: A combination of rules and ML**
  - Write rules to identify the sentence in which the decision is given
  - Supervised learning to identify court decision in that sentence
- **The bulk of research went into finding this paradigm**

What constitutes a good ML problem?

- **There's good signal in examples**
  - But it's hard to come up with concrete rules
- **There's some inherent variability...**
  - E.g., the problem itself seems non-deterministic
  - Or there's noise in the instruments that is hard to model
- **...but not too much of it**
  - If there's no signal, we'll end up with bad results
- **We can think of indicative features**
  - That we think will generalize, but not overfit
- **And which we have (or can get) good amount of**
  - Small number of examples would lead to overfitting

Once we decide on scope

- **What's the scientific/business context?**
  - In which scenarios is the model expected to work?
  - How will performance be measured?
  - Is the training data representative?
- **Answers will affect many design choices**
  - E.g., Loss, or regularization term, evaluation metric
- **Extensive knowledge of the domain is crucial**
  - Often the job is to bridge between ML and the specific field
  - E.g., medicine, linguistics, biology, and more

We treated $\mathcal{X}$ as numerical values, detached of physical meaning

- In practice, crucial to know what features represent
- Can help in building models
  - Identifying correlated features
  - Designing appropriate regularization
  - Detecting overfitting and overestimation

Example: Categorical vs. Continuous Features

	X_1	X_2	X_3	X_4
sample 1	2.0	3.0	2.5	yes
sample 2	1.7	3.8	9.1	no
⋮				

Ask each feature if it's ...

- **Categorical:** how many levels? ordered or unordered?
- **Continuous:** what are the allowed values?

Example: `mtcars` dataset

```
> mtcars
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

- Which features are
  - **Categorical?** - ordered (big vs small) / unordered (yellow vs red)
  - **Continuous?**

Understanding the Data

- **Do we have meaningful feature names?**
- **What are the feature units?**
- **Who chose which features to collect?**
- **Can we ask for other features?**

Annotation Quality

In this course we trust the labels $\mathcal{Y}$ as absolute truth

- **labels Y are often noisy in practice**
  - E.g., due to error in annotation or disagreement
  - Note this is different from noise in samples $\mathcal{X}$
  - May fail *any* learning algorithm
  - **Look at train annotations, do you agree with them?**
- **Inter-annotator agreement** is a common data quality measure in AI
  - Useful when the task itself may be subjective

Inter-Annotator Agreement

- For example, Sentiment analysis in text
  - **This movie was terrible!**
  - **This movie was great!**
  - **Q: I like the leading actor, but the director was terrible**
- Solution: measure **agreement** between two human annotators
  - For example, ask two annotators to annotate the same set
  - Measure % in which they agree

Cohen's Kappa as a Measure of Agreement

- Say we measure agreement on 80% of the data
- **Q: More impressive if there were 1000 possible labels?**
- **Cohen's Kappa** takes random chance into account

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

- Where:
  - p_o = observed agreement (e.g., 80%)
  - p_e = expected agreement of a random coin toss
- So for balanced classes we get:
  - 2 classes - $p_e = 0.5 \Rightarrow \kappa = 0.6$
  - 1000 classes - $p_e = 0.5 \Rightarrow \kappa = 0.799$

Annotation Bias

- Beyond noise, labels are bound to introduce **human biases**
  - E.g., by the process

  - Or the world

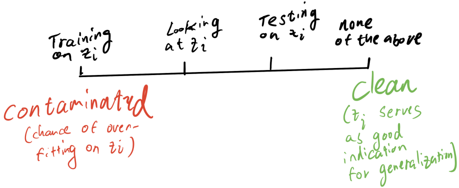
- **ML does not distinguish between these biases and signal**
  - Instead, optimizes loss in whatever way possible
  - Crucial to align mathematical loss with our actual purpose

Mark Yatskar et al., Shahar Levy et al.

Data Partitions

We assumed we have a train set S_m and sometimes a test set

- In practice, we often have a single annotated dataset
- We need to split it to achieve best generalization
  - and also be able to estimate the performance
  - Recall train-dev-test, cross-fold validation, bootstrap
- Each partition needs to be representative of the whole



For example: Imbalanced Classes

- **Sometimes the data is uneven**
  - E.g., most people won't have a heart attack
- **Need to divide while keeping the original class balance**
- **Q: What happens if don't do this?**
  - **Q: Severe discrepancies between train and test performance**

Approaching a New Problem: Takeaways

- **Define setup:** Which elements of the problem require ML
- **Understand the features:** Feature and label types & possible ranges
- **Question the labels:** Estimate annotation quality and biases
- **Make sensible partitions:** Each part is representative of the whole

The different stages of developing a model

Preprocessing → EDA → Baseline → model selection

The different stages of developing a model

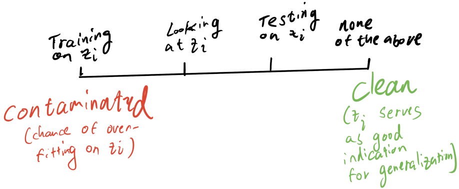
Preprocessing $\rightarrow$ EDA $\rightarrow$ Baseline $\rightarrow$ model selection

Preprocessing

- Large-scale data is bound to have errors and inconsistencies
- **Preprocessing: preparing the data before doing ML**
  - We'll see missing and corrupt values, and creating new features
  - All of these will be in the hackathon
  - As well as most other real-world problems
- **Much of the success of ML relies on it**
  - Give it plenty of time and thought

Preprocessing After Partitioning

- **Always develop preprocessing on the training set**
- **Q: What if you develop preprocessing on the entire dataset?**
  - You'll get acquainted with the test set
  - Thus contaminating it



Dealing with Missing Feature Values

- **Several reasons for missing feature values:**
  - Error in collection process (e.g., doctor forgot to mark)
  - Error in coding
- **First, identify how a missing feature value is coded in your data**
  - `n/a`, `0`, `-99`, `-1`
  - Requires understanding of feature semantics
- **Most models won't deal correctly with such values**
- We need to choose whether
  - To guess missing values (termed **imputation**)
  - Discard features with many missing values

Data Imputation

- **Data Imputation:** replacing missing data with substituted values
- **A broad field with many approaches**
- **Q: Can you think of an approach for data imputation?**
  - Filling in with average values
  - Extrapolation
  - Many more (see numpy, pandas in python)

Outliers and Corrupt Values in the Data

- **Corrupt values often appear in data**
  - People who are 7000 or -3 years old
  - Can manifest in subtle, domain-specific ways
  - E.g., male patients with ovary conditions
- **We need to choose a policy for corrupt values**
  - Similarly to missing values - impute or discard
- Some ML models are more **robust** to outliers
  - Think of the bias-variance tradeoff

Creating Features

- **Some data is not naturally in $\mathbb{R}^d$ (e.g. it's text or graph)**
- **Mapping it to $\mathbb{R}^d$ is called Euclidean Embedding**
  - E.g., in current NLP, word embedding is a must-have preprocessing step
- **A topic for advanced ML courses**

Preprocessing: Only in Code

- **The same preprocessing steps must be applied to train-dev-test**
  - But we only see the training
- **Preprocessing should always be done in code, never manually**
  - Apply same preprocessing to all data
  - **Never impute missing values / correct outliers manually**
- **Q: What's the danger in applying manual preprocessing?**
  - Over-estimation of performance
  - As other data partitions won't receive the same treatment

The different stages of developing a model

Preprocessing $\rightarrow$ EDA $\rightarrow$ Baseline $\rightarrow$ model selection

Exploratory Data Analysis

- After preprocessing a dataset, we usually start exploring
- **Exploratory Data Analysis (EDA):** looking for patterns, with no specific hypotheses
- **Q: Which data partition should we explore?**
  - **Only the training set**

EDA: Summary Statistics

- Collect statistics to get a grasp on data behavior
  - **Typical value:** Mean, Median, Mode
  - **Measure of variability:** Standard deviation, IQR
  - **Measure of relationship:** Covariance, correlation between features

Example: Pima dataset

- Native Arizona women (Pima tribe) often suffer from type-II diabetes
- Dataset consists of 768 records with 9 features:

Pregnancies	Number of times pregnant
Glucose	Plasma glucose after 2 hours in tolerance test
Blood pressure	Diastolic blood pressure (mm Hg)
Skin thickness	Triceps skin fold thickness (mm)
Insulin	2-Hour serum insulin (mu U/ml)
BMI	Body mass index (weight in kg/(height in m)^2)
Diabetes pedigree	Family history function
Age	Age (years)
Sick	Class variable (0 or 1)

Example: Pima Indians dataset

```
> summary(d)
   times.preg     plasma.glucose        BP              skin      
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00  
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00  
 Median : 3.000   Median :117.0   Median : 72.00   Median :23.00  
 Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54  
 3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00  
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00  
    insulin           BMI           pedigree           age       
 Min.   :  0.0   Min.   : 0.00   Min.   :0.0780   Min.   :21.00  
 1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437   1st Qu.:24.00  
 Median : 30.5   Median :32.00   Median :0.3725   Median :29.00  
 Mean   : 79.8   Mean   :31.99   Mean   :0.4719   Mean   :33.24  
 3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00  
 Max.   :846.0   Max.   :67.10   Max.   :2.4200   Max.   :81.00  
 sick   
 0:500  
 1:268  
```

More EDA Methods

- **Q: Did we already see EDA methods in the course?**
- **Clustering** and **dimensionality reduction** can reveal patterns in data
  - E.g., by looking at principal components in PCA

The different stages of developing a model

Preprocessing → EDA → Baseline → model selection

Baseline algorithm

- After exploring the data, we may have hypotheses regarding the data
  - E.g., which features are indicative of the label
  - What feature interactions are important
- **Baseline algorithm:** A naive approach against which we'll compare more complex models
  - Rule of thumb: Aim for either **low bias** or **low variance** baselines
  - E.g., deep vs. shallow decision trees
- **Baselines should be easy and quick to implement**

- The baseline gives us a sense of the problem complexity
- To estimate its performance we test it on a **held out set**
  - Either cross-validation or a predefined dev set
- If the baseline does very well - **the problem is relatively easy**
  - And we don't need to invest much effort into modelling
- Otherwise, **it serves as reference to more advanced approach**

The different stages of developing a model

Preprocessing $\rightarrow$ EDA $\rightarrow$ Baseline $\rightarrow$ model selection

Insert Here Everything We Learned

- **Goal: Improve over baseline**
    - **Bagging or boosting** the baseline
    - **Ensembeling** different learners
    - Experimenting with the **regularization term**
    - **Kernelizing** the baseline model
    - Any combination of these approaches
- Tune hyperparameters on the development set set

4 Stages of Model Development: Recap

cleaning & preparing data

Preprocessing → EDA → Baseline → model selection

Explore data to get insights

develop a naive model

Improve upon baseline by trying more complex approaches

- **In all 4 stages you'll need to deal with vast quantities of data**
- **E.g., millions of samples (Big Data)**
- **introduces a novel set of technical challenges**

The Missing Semester of your Education

- **Many tools and paradigms are crucial for everyday work**
  - Software development, in general and ML specifically
  - Efficient coding, software packages, source control, shell scripting
- **These usually don't fit the syllabus of any specific course**
  - Including IML
- **We'll give some pointers**

The Missing Semester of your Education

- I strongly recommend to checkout **The missing semester**[1]
- MIT Grad students initiative: a course covering these topics
- All material available online (lecture, handouts, exercises, etc.)

Schedule

- **1/13/20**: Course overview + the shell
- **1/14/20**: Shell Tools and Scripting
- **1/15/20**: Editors (Vim)
- **1/16/20**: Data Wrangling
- **1/21/20**: Command-line Environment
- **1/22/20**: Version Control (Git)
- **1/23/20**: Debugging and Profiling
- **1/27/20**: Metaprogramming
- **1/28/20**: Security and Cryptography
- **1/29/20**: Potpourri
- **1/30/20**: Q&A

Video recordings of the lectures are available on YouTube.

[1] `https://missing.csail.mit.edu/`

- For example, what's the problem with this code?

```
def preprocess_file(file_name):
    """
    Run through the lines of a file and preprocess them
    """
    preprocessed_lines = []
    for line in open(file_name):
        preproc_line = clean_line(line)
        preprocessed_lines.append(preproc_line)

    return preprocessed_lines
```

  - Note we store all of the file contents in memory

- **Q: What happens when the file contains millions of samples?**
  - We'll quickly get an out-of-memory error

Prefer Lazy Evaluation

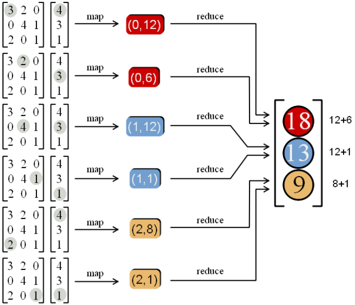
- **Lazy evaluation:** Delay evaluation until the value is needed
- In python this is done with the *yield* operator

```
def preprocess_file(file_name):
    """
    Run through the lines of a file and preprocess them
    """
    preprocessed_lines = []
    for line in open(file_name):
        preproc_line = clean_line(line)
        yield preproc_line
```

  - Produces iterator evaluating the next value only when needed
- Aim to hold the minimal units in memory at any given time

Parallelization

- **Performing math operations in parallel is key for time efficiency**
- **Q: What can we usually parallelize in algebraic computation?**



- **Map reduce, Spark, Hadoop are paradigms for such operations**

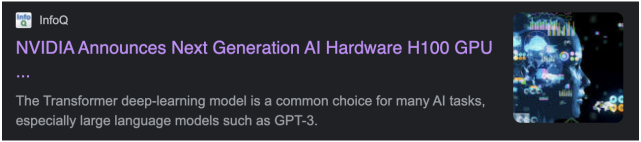
walletfox

CPU vs. GPU

- **Q: How many operations can we actually perform in parallel?**
  - Just as many cores we have on our machine
- **A few cores on laptop or PCs**
- **Small 100s on a high-capacity CPU server**
- **Graphical Processing Units (GPU) have 1000s of cores**

GPU

- **Originally developed for graphical rendering**
  - Which also have a lot of parallelizable algebraic operations
- **Ubiquitous in ML with the advent of parameter heavy models**
  - Speeds training a model by a factor of 10 or 100
  - There are now processors developed speficially for ML
- **Originally required a dedicated programming solution (CUDA)**
  - Many abstractions developed in code packages



Don't write your own learner from scratch

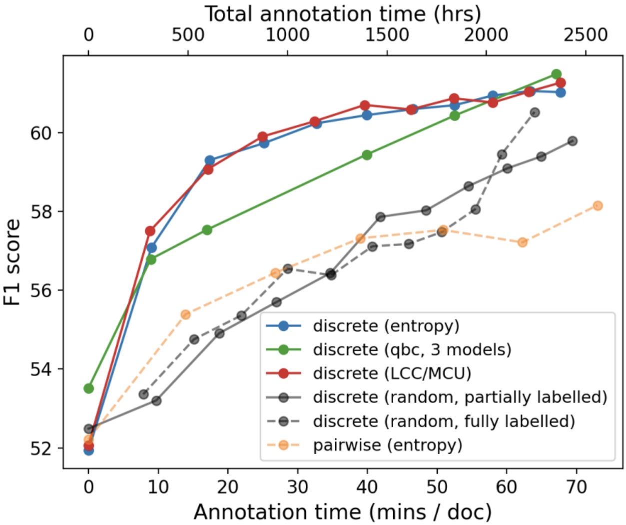
- Finally, familiarize with popular packages
  - Pandas
  - sklearn
  - matplotlib
  - numpy
  - tqdm
  - logging

Use the Test Set Once

- **We're ready to test our model against a truly held-out set** item **Goal: Understand where model works and where it doesn't**
  - "all models are wrong, some are useful"
- **Try to do this only once**
  - The more you use your test set, the more you'll overfit
- **We'll discuss best practices for reporting results**

Reporting Results

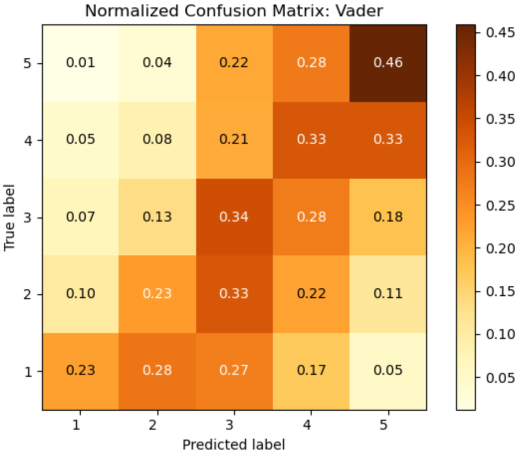
- Plot training and test error over model complexity and sample size



Li et al.

Reporting Results

- Plot a confusion matrix to better understand performance



Perform Error Analysis

- Sample 100s of examples **where your model was wrong**
- Manually go over them and find recurring patterns

Phenomenon	Passage Highlights	Question	Answer	Our model
Subtraction + Coreference	... Twenty-five of his 150 men were sick, and his advance stalled ...	How many of Bartolom de Amsqueta's 150 men were not sick?	125	145
Count + Filter	... Macedonians were the largest ethnic group in Skopje, with 338,358 inhabitants ... Then came ... Serbs (14,298 inhabitants), Turks (8,595), Bosniaks (7,585) and Vlachs (2,557) ...	How many ethnicities had less than 10000 people?	3	2
Domain knowledge	... Smith was sidelined by a torn pectoral muscle suffered during practice ...	How many quarters did Smith play?	0	2
Addition	... culminating in the Battle of Vienna of 1683, which marked the start of the 15-year-long Great Turkish War ...	What year did the Great Turkish War end?	1698	1668

Table 5: Representative examples from our model's error analysis. We list the identified semantic phenomenon, the relevant passage highlights, a gold question-answer pair, and the erroneous prediction by our model.

Dua et al.

Compare Across all Models

- Do all of these evaluation across all models you've tested

Create an Interactive Demo

- **Interactive demo is an excellent way to test models on various inputs**
  - and test different hyperparameter configurations
- **Used to be very time consuming to create**
- I strongly recommend learning **streamlit**[2]
  - Create useful demos in a few lines of code
- **Q: See a few demos?**

[2] https://streamlit.io

Reproducibility

- **Results should replicate across different runs, computers, data**
- **This turns out to be a very hard problem**

Write Reproducible Code

- **Use command line arguments**
  - For file names, hyperparameters, and anything configurable
  - Makes it easy to re-run your configuration again
- **Don't use Jupyter Notebooks for model development**
  - Great for teaching & visualization, **terrible** for reproducibility[3]

[3] `https://www.youtube.com/watch?v=7jiPeIFXb6U`

Reproducing Results is Hard: Controlling Randomness

- **Some models depend on random initialization**
  - **Q: Which ones did we see in the course?**
  - E.g., K-means clustering
  - Neural networks
- **Make sure to set seeds**
  - Lead to identical pseudo-random numbers over different runs
- **Yet some stubborn randomness persists**
  - GPUs are notorious for different results with same seed

Reproducing Results is Hard: Hardware Limitations

- **Large models developed in big tech companies**
  - E.g., BERT, GPT-3, Dall-e, . . .
- **Many labs can't train or even run these models**
  - GPU memory just too low
- **Green AI** (Roy Schwartz et al.)
  - Initiative for small efficient models fitting small hardware budget

- **We've covered 3 overarching themes**
  - Approaching a new problem
  - Developing a model
  - Reporting results
- **Discussed many pointers and good practices**
  - Hopefully useful as a reference and study guide
- **No replacement for experience**
  - Try to implement these concepts in as many projects as possible
  - Including **IML Hackathon 2022!**