Introduction to Machine Learning

Sina Zarrieß

MSCJ Summer School 2020

- Who am I?
- Who are you?
- What are we doing here (today)?

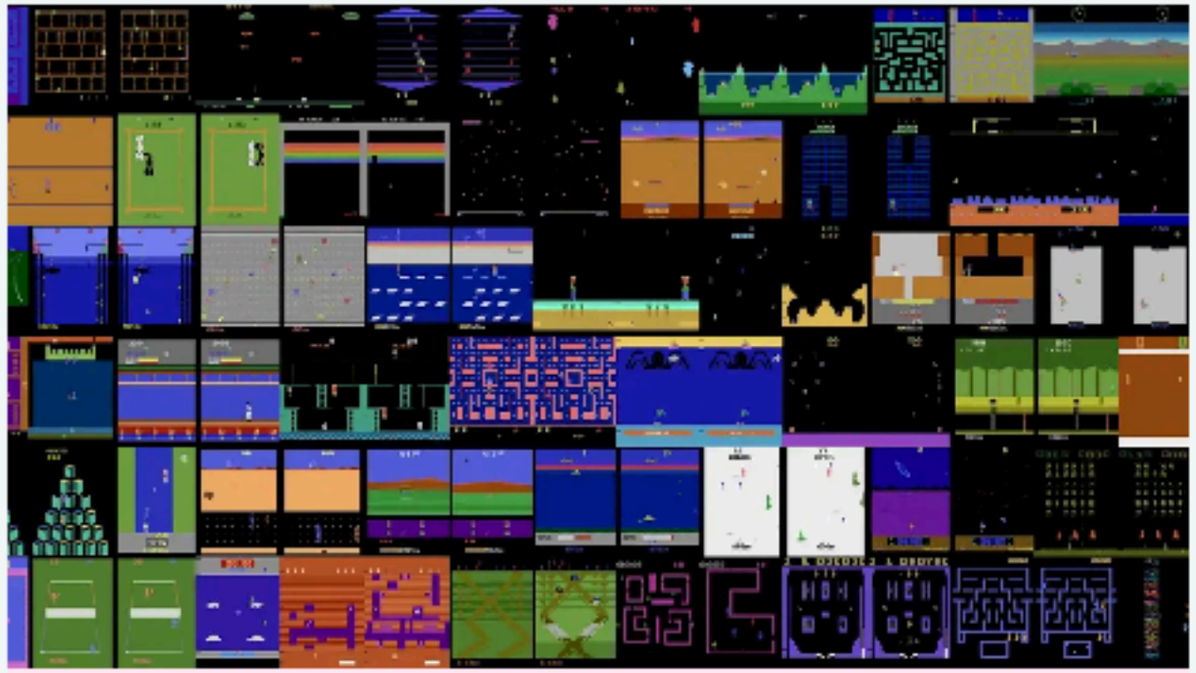
Schedule

- **Session 1 (11-13): A refresher on ML**
    - Some definitions, and hopefully some discussion
    - Goal: start talking to each other!
- **Session 2 (14-16): The Limits of Learning**
    - Evaluation and Explainability
    - Goal: know how to critically appraise a model
- **Session 3 (16-18): Explainable ML in Python**
    - Hands-on: scikit-learn, keras and lime
    - Goal: do it!

Preparation for Hands-on Session

- A python notebook and this presentation is here: https://github.com/sinazarriess/mlintro_mscj2020
- You need: numpy, scikit-learn, lime
- pip install lime, sklearn, numpy

What is Machine Learning?

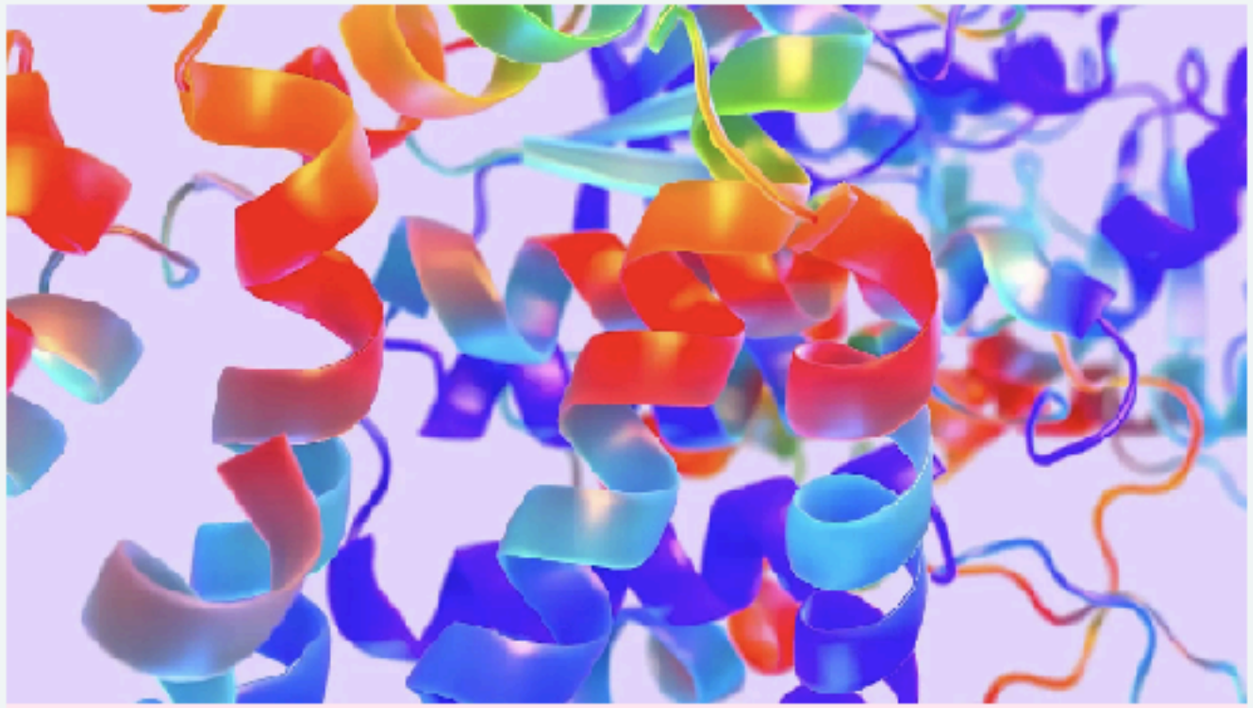


BLOG POST
RESEARCH

Agent57: Outperforming the Human Atari Benchmark

The Atari57 suite of games is a long-standing benchmark to gauge agent performance across a wide range of tasks. We've...

31 MAR 2020



BLOG POST
RESEARCH

AlphaFold: Using AI for scientific discovery

Our Nature paper describes AlphaFold, a system that generates 3D models of proteins that are far more accurate than any...

15 JAN 2020

https://deepmind.com/blog, September 4

KÜNSTLICHE INTELLIGENZ

Multitalent für Sprache

Der neue Textgenerator GPT-3 kann Shakespeare nachahmen, Programmcode schreiben und Fremdsprachen sowie Rechtsparagrafen übersetzen. Aber versteht er auch, was er tut?

von Alexander Graf

Ein Roboterkind liest

IMAGES / ISTOCK (AUSSCHNITT)

Die Vergleiche können kaum groß genug sein: »Ich habe das Gefühl, die Zukunft gesehen zu haben«, schrieb ein kalifornischer Tech-Unternehmer vor einigen Tagen bei Twitter. Andere ließen verlauten, das neue Tool werde die Welt komplett verändern. Grund für diese Euphorie ist GPT-3 (Generative Pretrained Transformer 3), ein auf künstlicher Intelligenz basierendes Sprachprogramm. Dessen Betaversion ist derzeit für ausgewählte Testnutzer zugänglich, und seitdem sorgen Videos und Screenshots von den außergewöhnlichen Fähigkeiten des Tools in den sozialen Netzwerken für Aufregung.

https://www.spektrum.de/news/kuenstliche-intelligenz-der-textgenerator-gpt-3-als-sprachtalent/1756796

Schedule

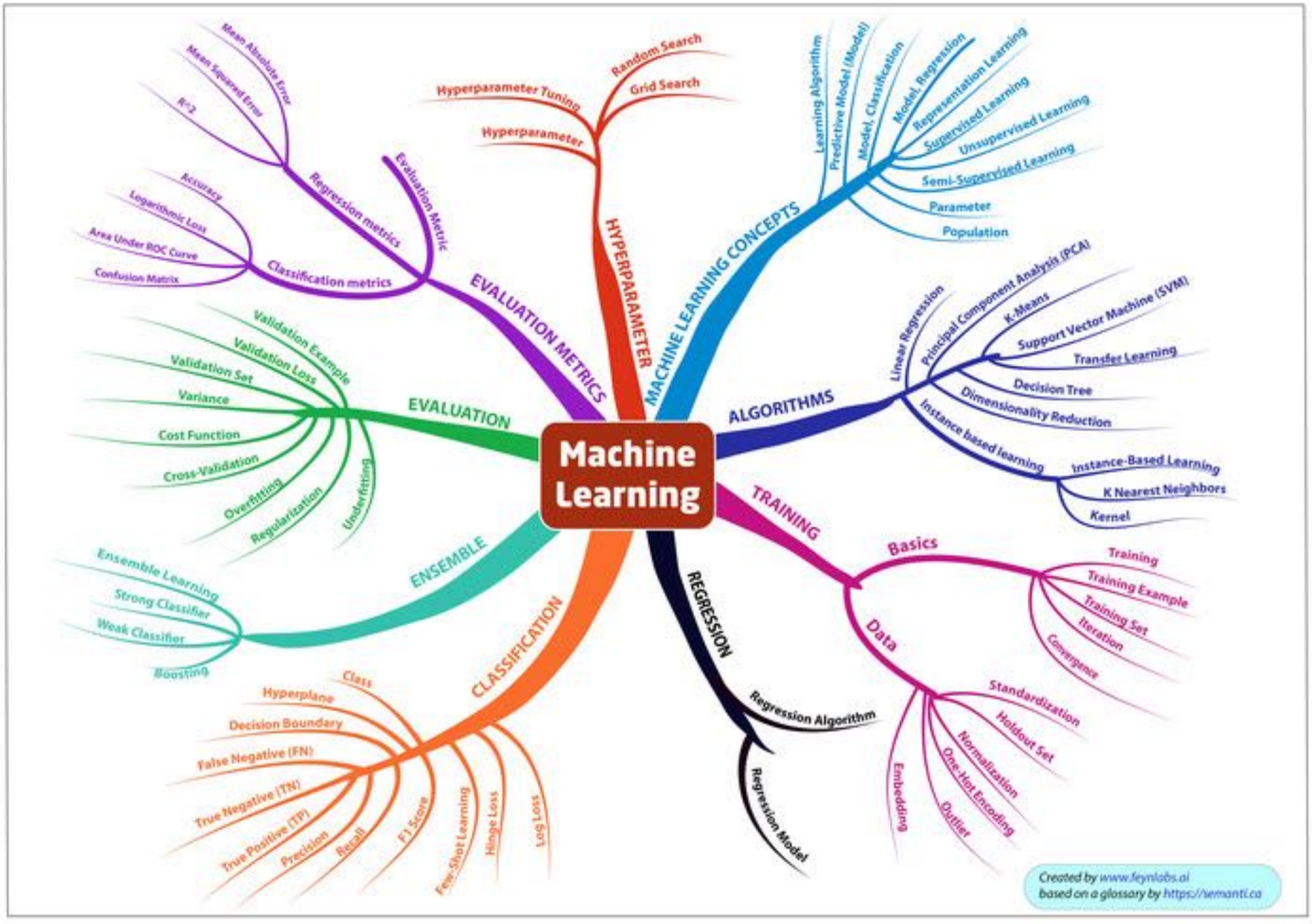
- **Session 1 (11-13): A refresher on ML**
    - Some definitions, and hopefully some discussion
    - Goal: start talking to each other!
- **Session 2 (14-16): The Limits of Learning**
    - Evaluation and Explainability
    - Goal: know how to critically appraise a model
- **Session 3 (16-18): Explainable ML in Python**
    - Hands-on: scikit-learn, keras and lime
    - Goal: do it!

So what is Machine Learning?

- Many many things!
- theory, algorithms, learning styles, applications, agents, optimization, data collection & data management, feature selection, tuning, evaluation …

- It’s not easy to find a good overview!

https://en.wikipedia.org/wiki/Outline_of_machine_learning#Machine_learning_algorithms



https://www.datasciencecentral.com/profiles/blogs/learn-machinelearning-coding-basics-in-a-weekend-glossary-and

Let’s draw our own map!

- Try to think of all (important) ML concepts you know and organize them somehow
- Work in small groups (3 people), 20-30min (?)
- We’ll then try to combine all your maps into one big map, 20-30min (?)

Session 1

- ML refresher: basic terms, definitions, algorithms (1h)

- An ML map (45min)

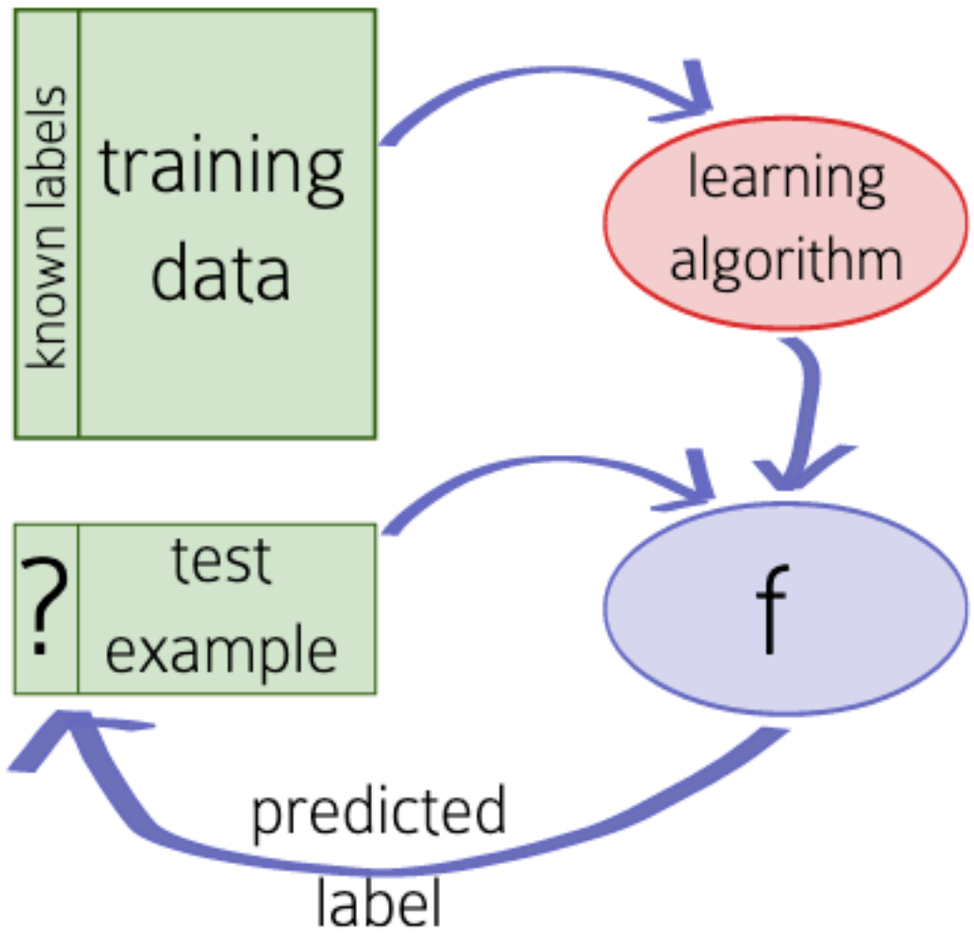
Definitions of ML

- Machine Learning is the study of computer algorithms that improve automatically through experience.
(Tom Mitchell, 1997, http://www.cs.cmu.edu/~tom/mlbook.html)

- At a basic level, machine learning is about predicting the future based on the past.
(Hal Daumé, http://ciml.info/)

A slightly more formal definition

- The goal of inductive machine learning is to take some training data and use it to induce a function f. This function f will be evaluated on the test data. The machine learning algorithm has succeeded if its performance on the test data is high.



(Hal Daumé, http://ciml.info/)

Training Data

class A

class B

Test Data

A or B ?

Training Data

class A

class B

Test Data ?

A or B ?

Memorization vs. Generalization

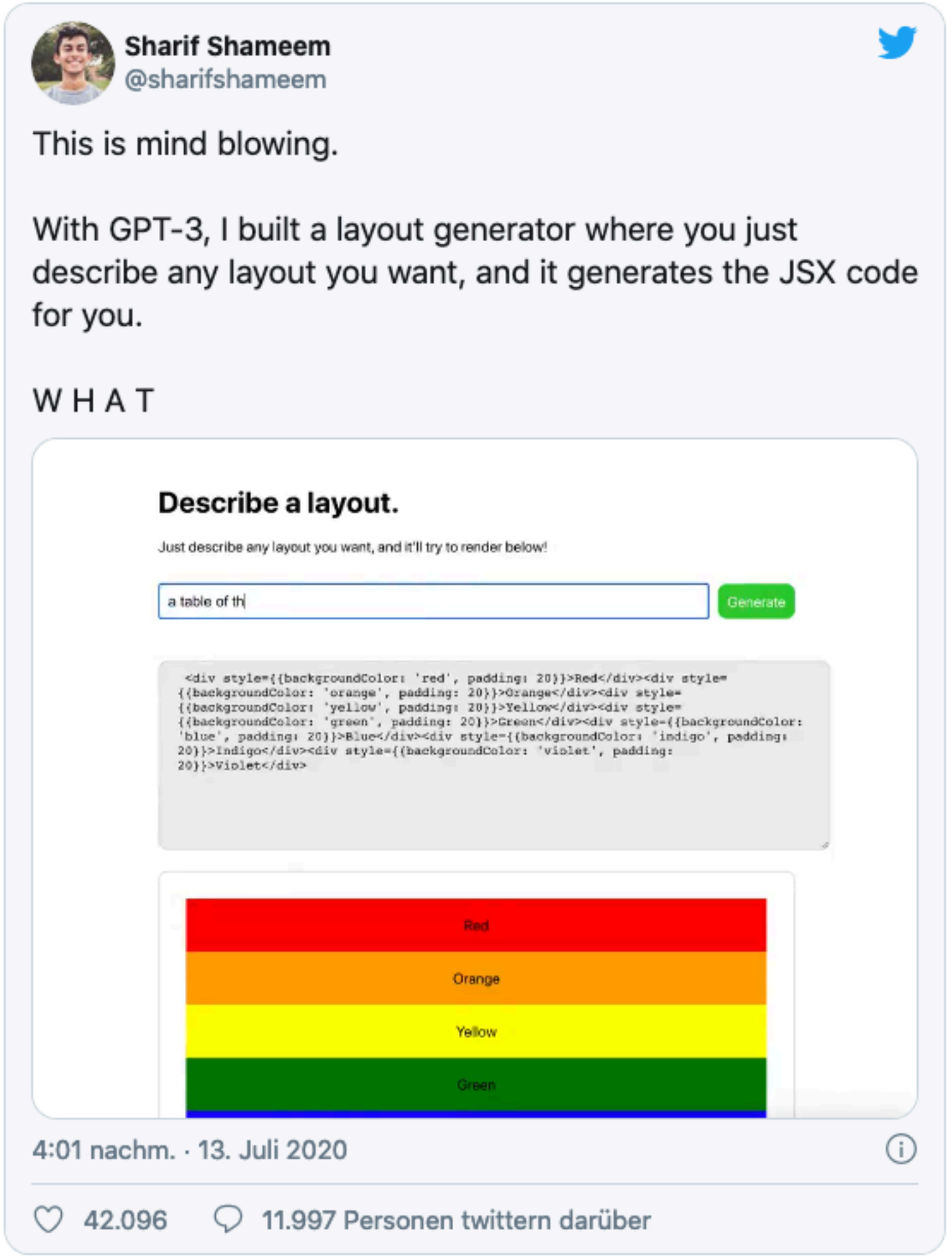
- Machine learning means building algorithms that have the ability to **generalize**!

- ML algorithms are used to **predict** labels (or other things) for **unseen examples**.

- Formulations of ML come with formalizations of generalization **performance**.

Side-note: Google’s recent language models

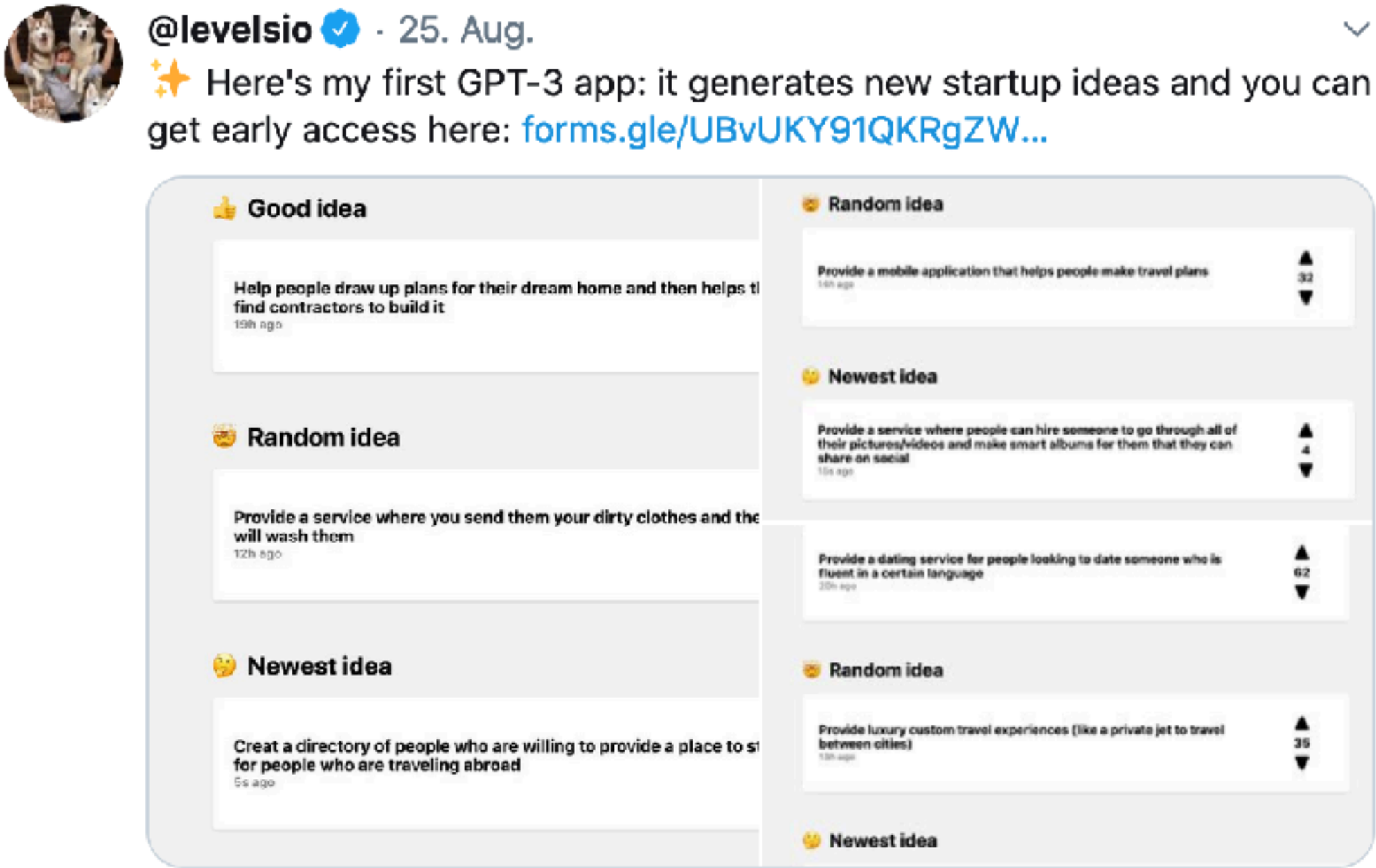
- GPT-3: „Language Models are Few-Shot Learners“, https://arxiv.org/pdf/2005.14165.pdf



GPT-3's training data

Dataset	Quantity (tokens)
Common Crawl (filtered)	410 billion
WebText2	19 billion
Books1	12 billion
Books2	55 billion
Wikipedia	3 billion

How GPT-3 is tested (sometimes)



Is there a problem?

4 Measuring and Preventing Memorization Of Benchmarks

Since our training dataset is sourced from the internet, it is possible that our model was trained on some of our benchmark test sets. Accurately detecting test contamination from internet-scale datasets is a new area of research without established best practices. While it is common practice to train large models without investigating contamination, given the increasing scale of pretraining datasets, we believe this issue is becoming increasingly important to attend to.

GPT-3 operates in a somewhat different regime. On the one hand, the dataset and model size are about two orders of magnitude larger than those used for GPT-2, and include a large amount of Common Crawl, creating increased potential for contamination and memorization. On the other hand, precisely due to the large amount of data, even GPT-3 175B does not overfit its training set by a significant amount, measured relative to a held-out validation set with which it was deduplicated (Figure 4.1). Thus, we expect that contamination is likely to be frequent, but that its effects may not be as large as feared.

Seriously?

Memorization vs. Generalization

- Machine learning means building algorithms that have the ability to **generalize**!

- ML algorithms are used to **predict** labels (or other things) for **unseen examples**.

- Formulations of ML come with formalizations of generalization **performance**.

- **NEVER EVER TOUCH YOUR TEST DATA**

More definitions: loss

- **Loss**: we need a function that measures the performance of our model with respect to the ``truth", this is our **loss function** l

Regression: **squared loss** $\ell(y, \hat{y}) = (y - \hat{y})^2$
or **absolute loss** $\ell(y, \hat{y}) = |y - \hat{y}|$.

Binary Classification: **zero/one loss** $\ell(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{otherwise} \end{cases}$

More definitions: data generating distribution

- We assume that there is a distribution D over pairs (x,y) that defines the data we expect to see

- D assigns high probability to reasonable (x,y) pairs and low probability to unreasonable (x,y) pairs

- Our training data is a random sample from D!

A formal definition of the learning problem

- Based on our training data, which is drawn from D, we need to induce a function f that maps new inputs x’ to predictions y’.

- The key property of f is that it should do well on future examples that are also drawn from D.

- F should minimize the expected loss over D with respect to the loss function l as much as possible:

$$\epsilon \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell(y, f(\boldsymbol{x}))\right] = \sum_{(x,y)} \mathcal{D}(\boldsymbol{x}, y)\ell(y, f(\boldsymbol{x}))$$

Minimizing Errors

- During training, we can always compute the training error:

$$\hat{\epsilon} \triangleq \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n))$$

- But what we want to minimize is the expected error!
- Why?

Training vs. Expected Error

- Minimizing training error is easy! We can simply memorize all examples in the training set.

- We want to learn a function that generalizes beyond the training data, i.e. minimizes the expected error.

- The ML challenge: we only have access to the training error, but care about the expected error.

- We should never expect a machine learning model to generalize beyond the data distribution D it has seen at training time.

Is there a problem?

4 Measuring and Preventing Memorization Of Benchmarks

Since our training dataset is sourced from the internet, it is possible that our model was trained on some of our benchmark test sets. Accurately detecting test contamination from internet-scale datasets is a new area of research without established best practices. While it is common practice to train large models without investigating contamination, given the increasing scale of pretraining datasets, we believe this issue is becoming increasingly important to attend to.

GPT-3 operates in a somewhat different regime. On the one hand, the dataset and model size are about two orders of magnitude larger than those used for GPT-2, and include a large amount of Common Crawl, creating increased potential for contamination and memorization. On the other hand, precisely due to the large amount of data, even GPT-3 175B does not overfit its training set by a significant amount, measured relative to a held-out validation set with which it was deduplicated (Figure 4.1). Thus, we expect that contamination is likely to be frequent, but that its effects may not be as large as feared.

Yes, there is a problem!

What does it mean to learn?

- Given …
  - a loss function L
  - a data generating distribution D
- … we want to compute a function f that minimizes the expected error over D with respect to L.

How do ML models predict the future from the past?

- Preprocessing: inputs are decomposed into **features**

- Most of the time, inputs to ML (x) are lists of feature values or vectors

- The geometric view of data: examples are points in high-dimensional space

Example: Iris data set

sepal_length	sepal_width	petal_length	petal_width	species
4.3 – 7.9	2 – 4.4	1 – 6.9	0.1 – 2.5	3 unique va
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa

https://www.kaggle.com/arshid/iris-flower-dataset

Anything can be vectorized

★ 2/10

From sublime to the ridiculous in 3 steps.
nigelmacdonald 4 January 2020

Started with so much promise and the first episode is really good. The second episode begins to take too many liberties with the source material and by the third is unrecognizable as even part of the same series. I can't remember a series which nosedived so rapidly.

599 out of 759 found this helpful. Was this review helpful? Sign in to vote

Permalink

Ask the domain experts!

How do ML models predict the future from the past?

- A simple approach: kNN (k nearest neighbours)
- All we need is a geometric distance function *d* (e.g. Euclidian distance)
- Training: we simply store the training set (x,y)
- Testing: for a test example x', we retrieve the k most similar training examples X according to *d*, and predict the majority label for x'

kNN

Algorithm 3 KNN-PREDICT($\mathbf{D}$, K, $\hat{\boldsymbol{x}}$)

$S \leftarrow [\,]$
for $n = 1$ **to** N **do**
  $S \leftarrow S \oplus \langle \mathrm{d}(\boldsymbol{x}_n, \hat{\boldsymbol{x}}), n \rangle$ // store distance to training example n
end for
$S \leftarrow \textsc{sort}(S)$ // put lowest-distance objects first
$\hat{y} \leftarrow 0$
for $k = 1$ **to** K **do**
  $\langle dist, n \rangle \leftarrow S_k$ // n this is the kth closest data point
  $\hat{y} \leftarrow \hat{y} + y_n$ // vote according to the label for the nth training point
end for
return $\textsc{sign}(\hat{y})$ // return $+1$ if $\hat{y} > 0$ and -1 if $\hat{y} < 0$

Inductive Bias

- In the absence of data that narrow down the relevant concept, what types of solutions is a learner going to prefer?

- *Every* ML approach has inductive biases!

Inductive Biases of kNN

- All features of an input are equally important

- The label of an example is similar to examples of nearby points

How do ML models predict the future from the past?

- A less simple approach: learn a hyperplane that separates our feature space into 2 halves (for binary classification)

Image source: Pixabay (Free license)

The Perceptron

- We learn the weight parameters W of linear function, that weights our features and adds them up:

$$a = \left[\sum_{d=1}^{D} w_d x_d \right] + b$$

- If the sum a (the activation) is < 0, we predict class A; if it is > 0, we predict class B

- We learn a weight vector *w*

Decision Boundary

$$\mathcal{B} = \left\{ \boldsymbol{x} \ : \ \sum_d w_d x_d = 0 \right\}$$

- The decision boundary is the set of points in our space that achieve 0 activation
- I.e. the vector x’ such dot-product(*w*,x’) = 0
- I.e. the vector that is perpendicular to *w*

(Simple) Gradient Descent

Algorithm 5 PERCEPTRONTRAIN(D, *MaxIter*)

```
w_d ← 0, for all d = 1 ... D                    // initialize weights
b ← 0                                           // initialize bias
for iter = 1 ... MaxIter do
    for all (x,y) ∈ D do
        a ← Σ_{d=1}^{D} w_d x_d + b             // compute activation for this example
        if ya ≤ 0 then
            w_d ← w_d + y x_d, for all d = 1 ... D   // update weights
            b ← b + y                           // update bias
        end if
    end for
end for
return w_0, w_1, ..., w_D, b
```

Algorithm 6 PERCEPTRONTEST($w_0, w_1, \ldots, w_D, b, \hat{x}$)

```
a ← Σ_{d=1}^{D} w_d x̂_d + b                     // compute activation for the test example
return SIGN(a)
```

Inductive Biases of Perceptron

- Perceptron always learns a linear decision boundary
- If the data is not linearly separable, perceptron will not converge
- For Perceptron, all decision boundaries that separate/do not separate the data are equal!

Summary

- Memorization, generalization, prediction
- Loss, training and expected error
- Features and vectorization
- Inductive bias
- Decision boundary

Let’s draw an ML map!

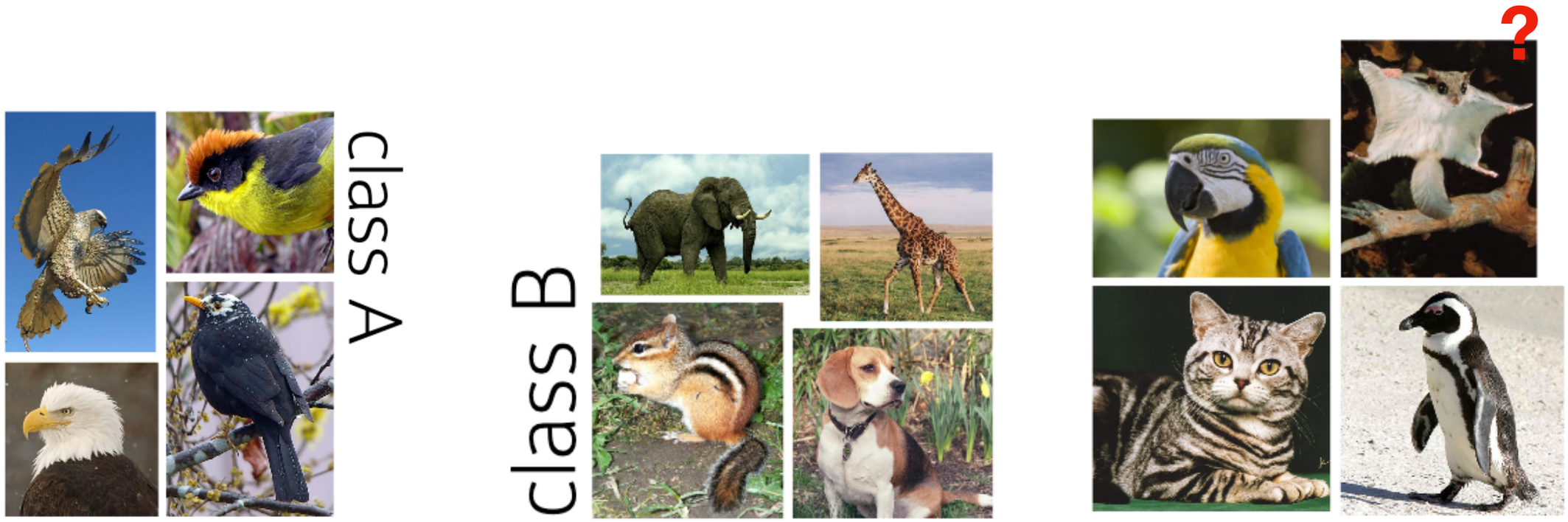
- Try to think of all (important) ML concepts you know and organize them somehow
- Work in small groups (3 people), 20-30min (?)
- We’ll then try to combine all your maps into one big map, 20-30min (?)

Break

Schedule

- **Session 1 (11-13): A refresher on ML**
    - Some definitions, and hopefully some discussion
    - Goal: start talking to each other!
- **Session 2 (14-16): The Limits of Learning**
    - Evaluation and Explainability
    - Goal: know how to critically appraise a model
- **Session 3 (16-18): Explainable ML in Python**
    - Hands-on: scikit-learn, keras and lime
    - Goal: do it!

The Limits of Learning



- Inductive bias: in the absence of training data that narrow down a prediction during testing, a model will prefer certain decisions over others due to inductive biases

- Even human learners have inductive biases!

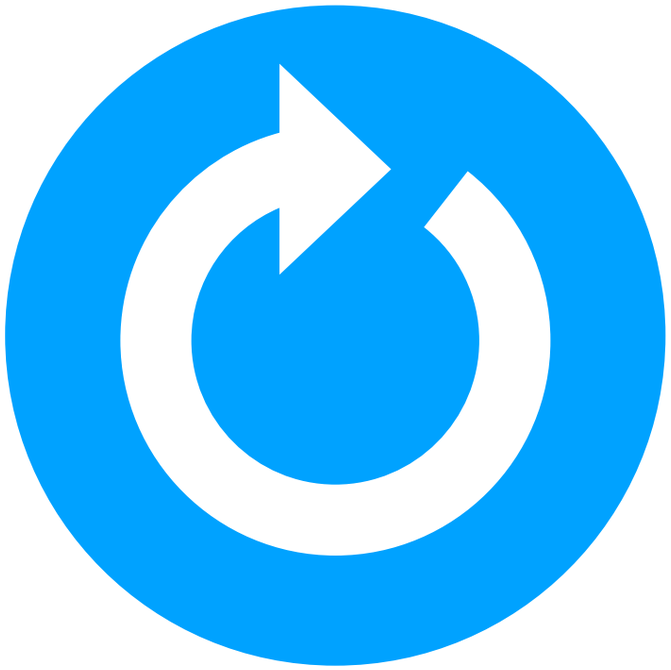
Why can ML fail?

- Noise in the data
- Not enough features
- Learning task is not well-defined (e.g. many examples with more than one solution)
- The inductive biases of your algorithm do not fit your task

Designing an ML application

1	real world goal	increase revenue
2	real world mechanism	better ad display
3	learning problem	classify click-through
4	data collection	interaction w/ current system
5	collected data	query, ad, click
6	data representation	bow^2, $\pm$ click
7	select model family	decision trees, depth 20
8	select training data	subset from april'16
9	train model & hyperparams	final decision tree
10	predict on test data	subset from may'16
11	evaluate error	zero/one loss for $\pm$ click
12	deploy!	(hope we achieve our goal)

Figure 2.4: A typical design process for a machine learning application.



Overfitting and Underfitting

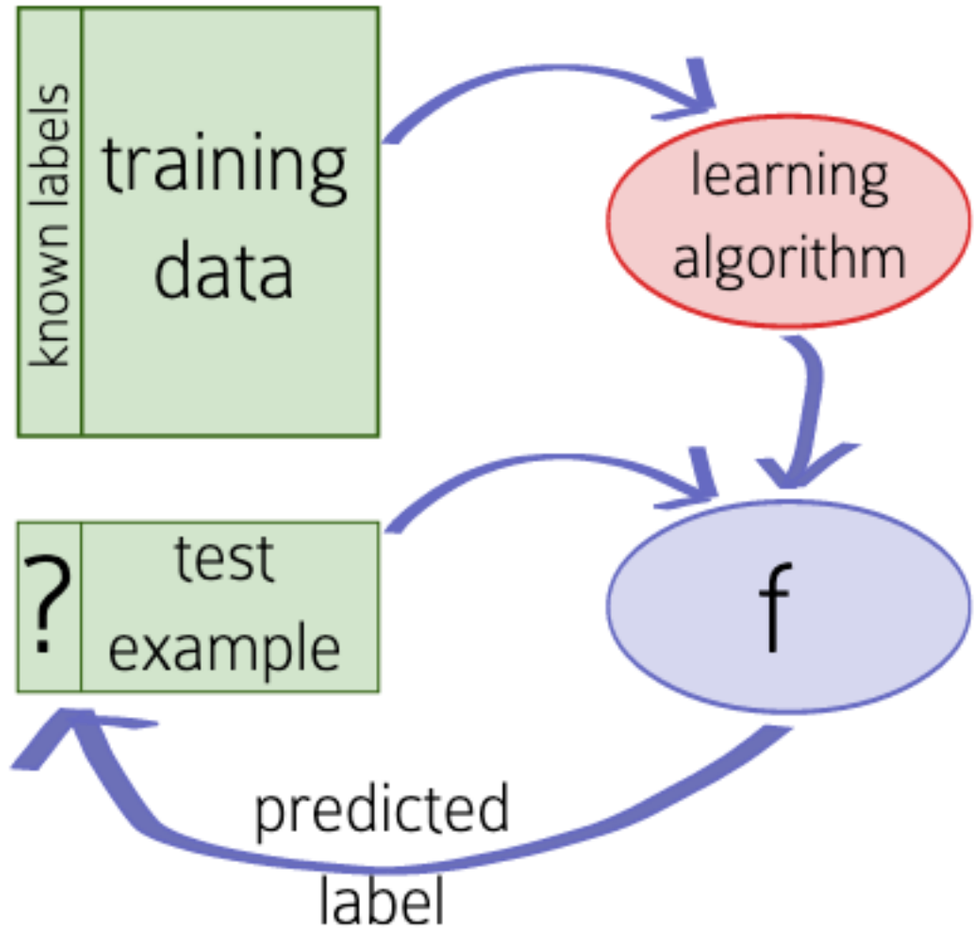
- Underfitting: your model had the opportunity to learn something but didn’t

- Overfitting: your model pays to much attention to idiosyncrasies in the data and does not generalize well

- How does overfitting happen?

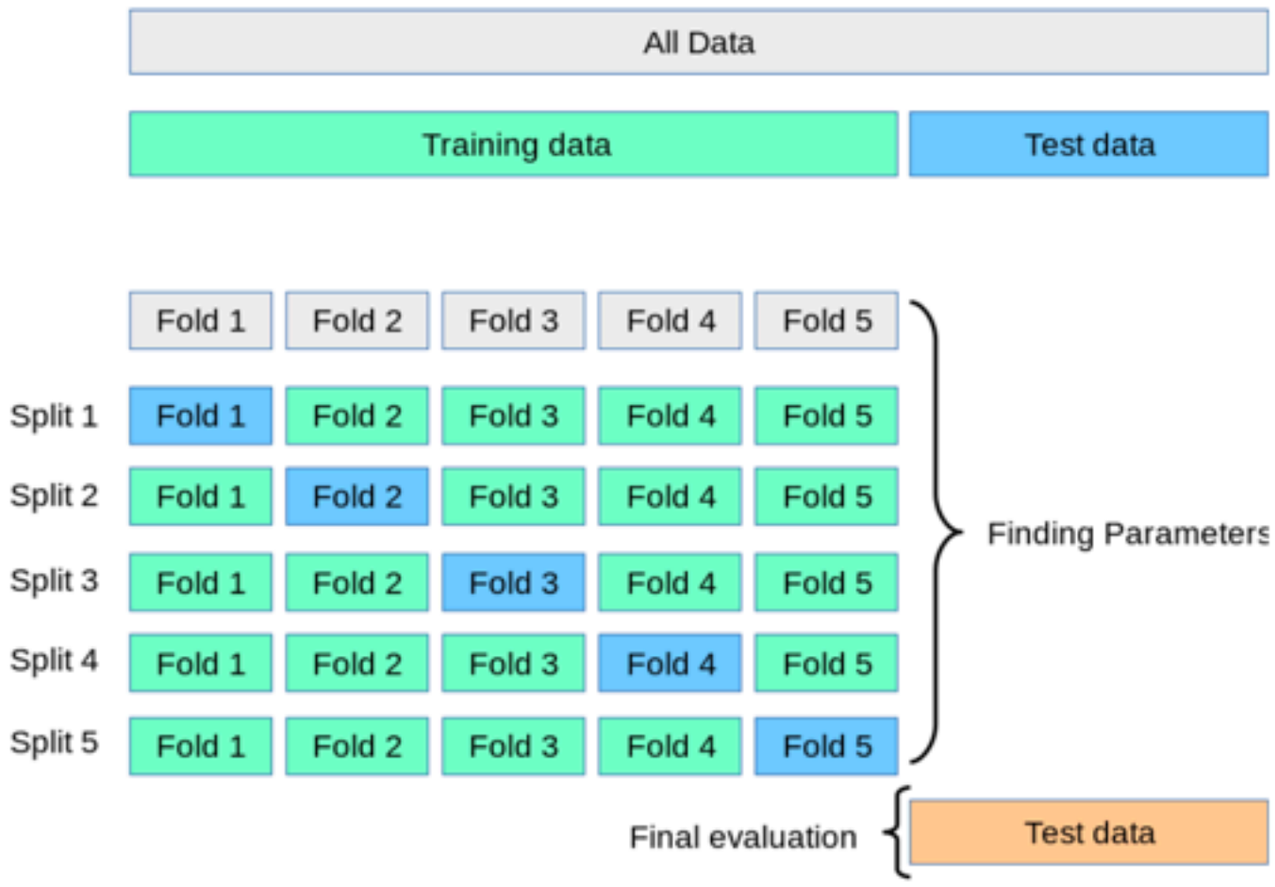
Evaluation

- ALWAYS test your model on a hold-out test set
- NEVER look at this test set



(Hal Daumé, http://ciml.info/)

Hold-out (Cross-)validation



- Test minor variants of your model via cross-validation
- Compare different models on the hold-out test data

Confusion Matrix

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

Confusion Matrix

- The most well-known evaluation metric in ML is accuracy:

$$acc = \frac{N\ correct\ predictions}{Total\ predictions}$$

- Always consider your majority baseline (or random baseline)!

Beyond Accuracy

- Accuracy helps when evaluating supervised classification
- Most other ML problems are difficult to evaluate!
- Accuracy does not tell you whether your model is useful for users (consider down-stream task-based evaluation or human evaluation)
- … and there is another problem

The story of Clever Hans

Explainability, Transparency, Interpretability, Trust, …

- We do not want our ML models to be like Clever Hans
- We want our models to be right for the right reasons
- We do not want our models to rely on spurious correlations, biases, idiosyncrasies in the data
- The bigger and deeper ML models get, the harder it is to evaluate them!

"Why Should I Trust You?" Explaining the Predictions of Any Classifier

Marco Tulio Ribeiro
University of Washington
Seattle, WA 98105, USA
marcotcr@cs.uw.edu

Sameer Singh
University of Washington
Seattle, WA 98105, USA
sameer@cs.uw.edu

Carlos Guestrin
University of Washington
Seattle, WA 98105, USA
guestrin@cs.uw.edu

ABSTRACT

Despite widespread adoption, machine learning models remain mostly black boxes. Understanding the reasons behind predictions is, however, quite important in assessing *trust*, which is fundamental if one plans to take action based on a prediction, or when choosing whether to deploy a new model. Such understanding also provides insights into the model, which can be used to transform an untrustworthy model or prediction into a trustworthy one.

In this work, we propose LIME, a novel explanation technique that explains the predictions of *any* classifier in an interpretable and faithful manner, by learning an interpretable model locally around the prediction. We also propose a method to explain models by presenting representative individual predictions and their explanations in a non-redundant way, framing the task as a submodular optimization problem. We demonstrate the flexibility of these methods by explaining different models for text (e.g. random forests)

how much the human understands a model's behaviour, as opposed to seeing it as a black box.

Determining trust in individual predictions is an important problem when the model is used for decision making. When using machine learning for medical diagnosis [6] or terrorism detection, for example, predictions cannot be acted upon on blind faith, as the consequences may be catastrophic.

Apart from trusting individual predictions, there is also a need to evaluate the model as a whole before deploying it "in the wild". To make this decision, users need to be confident that the model will perform well on real-world data, according to the metrics of interest. Currently, models are evaluated using accuracy metrics on an available validation dataset. However, real-world data is often significantly different, and further, the evaluation metric may not be indicative of the product's goal. Inspecting individual predictions and their explanations is a worthwhile solution, in addition to such metrics. In this case, it is important to aid users by suggesting

https://arxiv.org/abs/1602.04938

Beyond Accuracy: Behavioral Testing of NLP Models with CheckList

Marco Tulio Ribeiro
Microsoft Research
marcotcr@microsoft.com

Tongshuang Wu
Univ. of Washington
wtshuang@cs.uw.edu

Carlos Guestrin
Univ. of Washington
guestrin@cs.uw.edu

Sameer Singh
Univ. of California, Irvine
sameer@uci.edu

Abstract

Although measuring held-out accuracy has been the primary approach to evaluate generalization, it often overestimates the performance of NLP models, while alternative approaches for evaluating models either focus on individual tasks or on specific behaviors. Inspired by principles of behavioral testing in software engineering, we introduce CheckList, a task-agnostic methodology for testing NLP models. CheckList includes a matrix of general linguistic *capabilities* and *test types* that facilitate comprehensive test ideation, as well as a software tool to generate a large and diverse

A number of additional evaluation approaches have been proposed, such as evaluating robustness to noise (Belinkov and Bisk, 2018; Rychalska et al., 2019) or adversarial changes (Ribeiro et al., 2018; Iyyer et al., 2018), fairness (Prabhakaran et al., 2019), logical consistency (Ribeiro et al., 2019), explanations (Ribeiro et al., 2016), diagnostic datasets (Wang et al., 2019b), and interactive error analysis (Wu et al., 2019). However, these approaches focus either on individual tasks such as Question Answering or Natural Language Inference, or on a few capabilities (e.g. robustness), and thus do not provide comprehensive guidance on

BEST PAPER AT ACL 2020!

LIME - Explaining the predictions of any classifier

- Focus on explaining single predictions
- Pick a simple, interpretable model (i.e. linear model, decision tree)
- Locally approximate the global model with a simple model
- Explain the prediction by visualizing the weights of the simple, interpretable model

The intuition for LIME

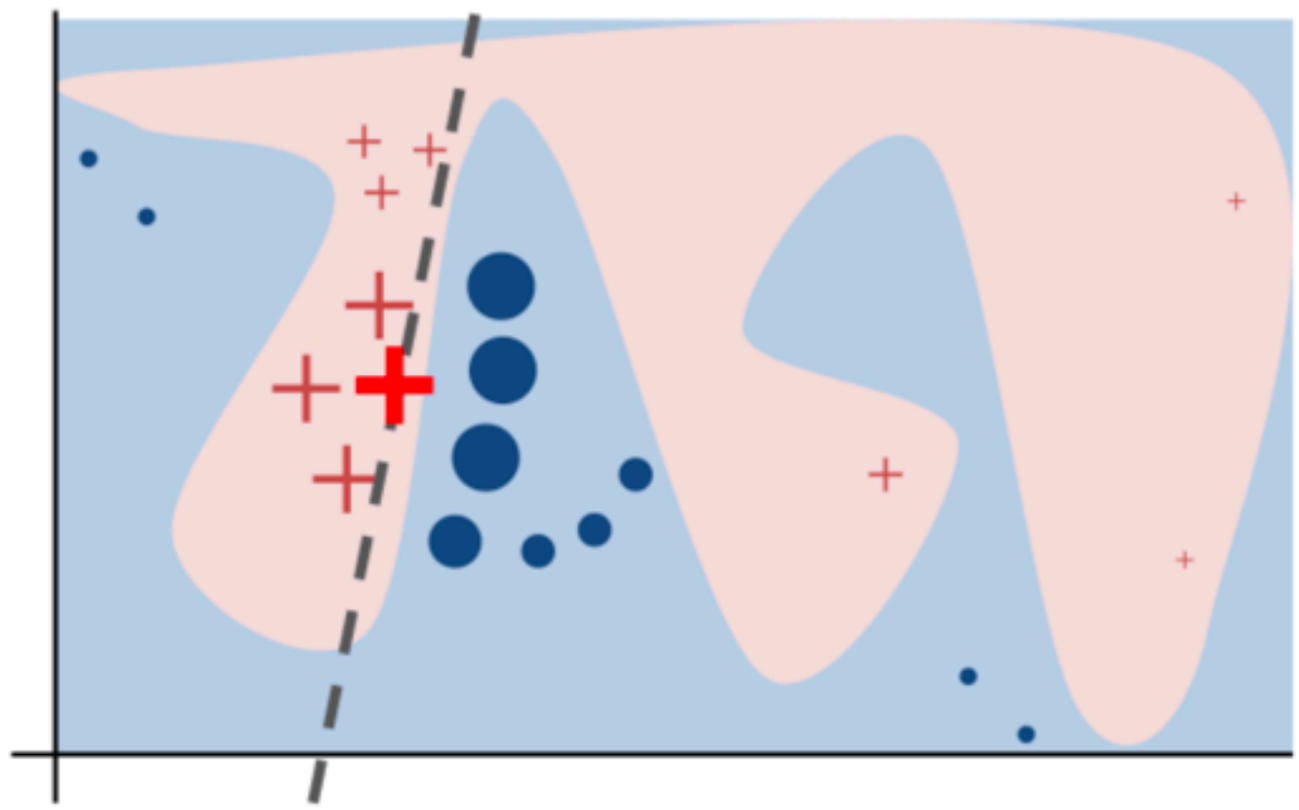


Figure 3: Toy example to present intuition for LIME. The black-box model's complex decision function f (unknown to LIME) is represented by the blue/pink background, which cannot be approximated well by a linear model. The bold red cross is the instance being explained. LIME samples instances, gets predictions using f, and weighs them by the proximity

A local explanation

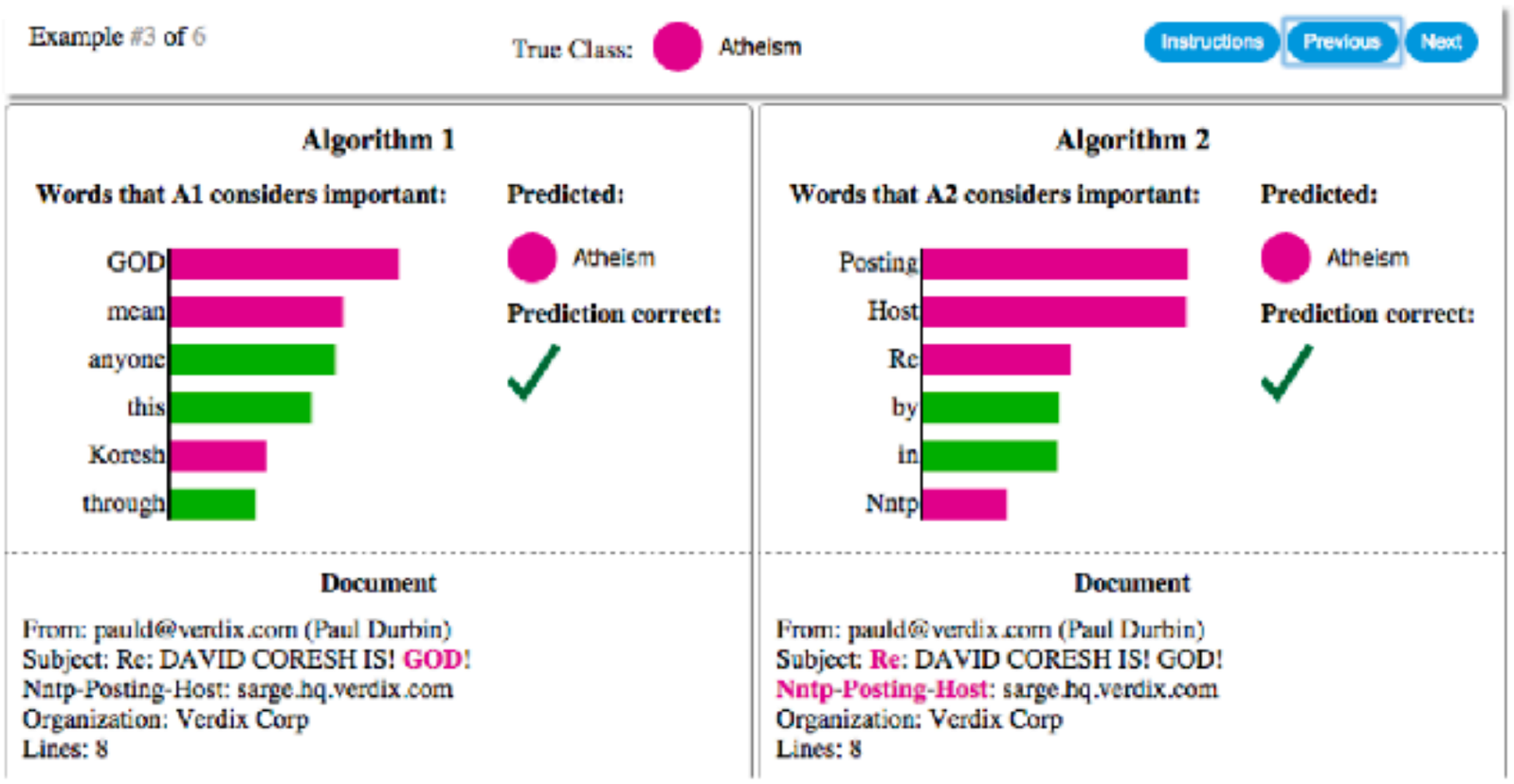


Figure 2: Explaining individual predictions of competing classifiers trying to determine if a document is about "Christianity" or "Atheism". The bar chart represents the importance given to the most relevant words, also highlighted in the text. Color indicates which class the word contributes to (green for "Christianity", magenta for "Atheism").

LIME algorithm

Algorithm 1 Sparse Linear Explanations using LIME

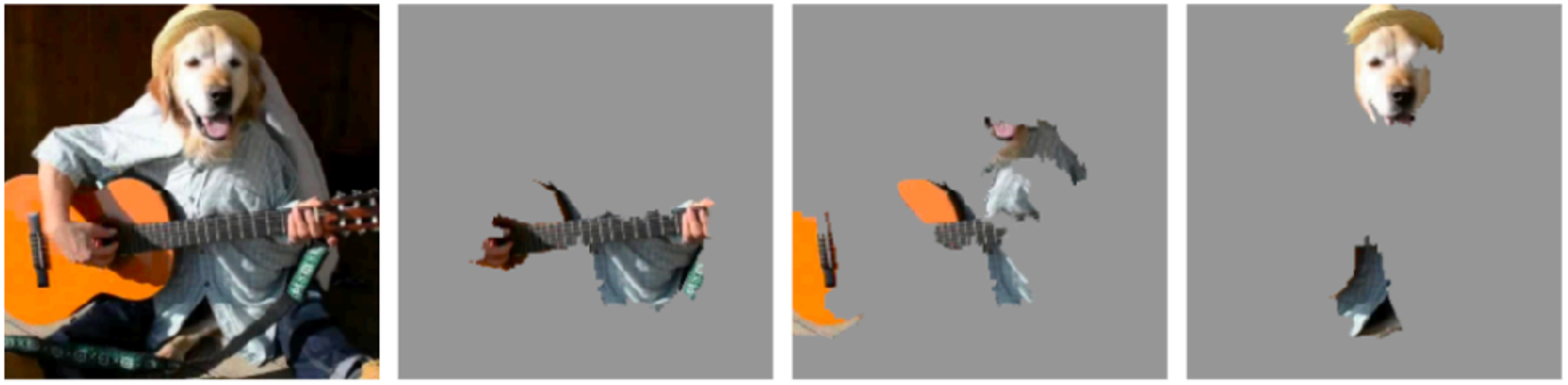
Require: Classifier f, Number of samples N
Require: Instance x, and its interpretable version x'
Require: Similarity kernel π_x, Length of explanation K

```
Z ← {}
for i ∈ {1, 2, 3, ..., N} do
    z'_i ← sample_around(x')
    Z ← Z ∪ ⟨z'_i, f(z_i), π_x(z_i)⟩
end for
w ← K-Lasso(Z, K)  ▷ with z'_i as features, f(z) as target
return w
```

- Locality-aware loss (f is the global model, g is the local model):

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z) \left(f(z) - g(z')\right)^2$$

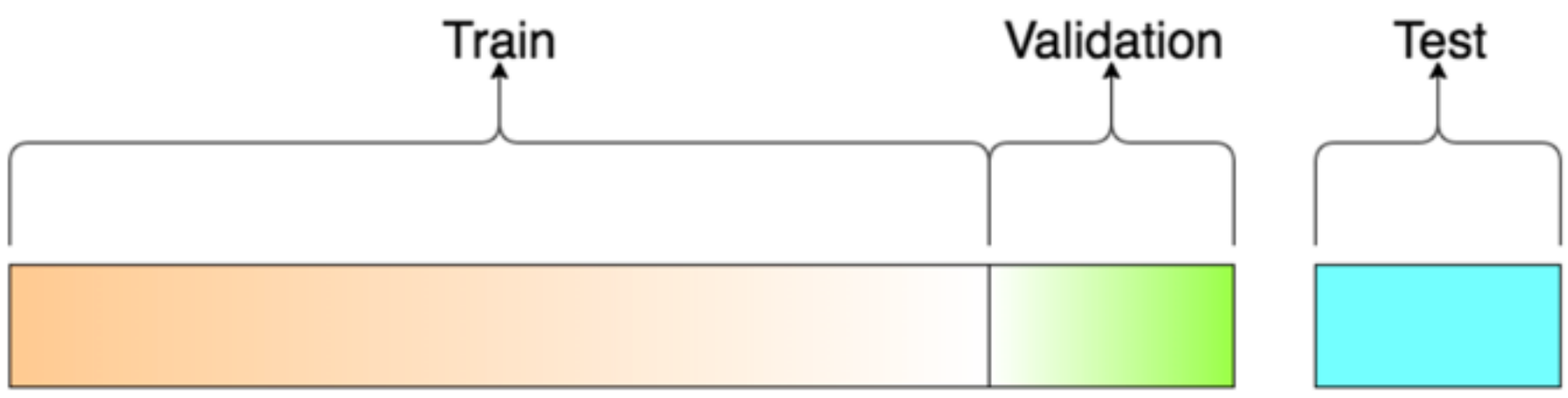
Explaining image classification



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$)**, "Acoustic guitar"** ($p = 0.24$) **and "Labrador"** ($p = 0.21$)

Never Ever Look at Your Test Data



A visualization of the splits

Summary

- ML often fails, for many reasons
- When building a model, you will need to tune and reiterate over a number of steps
- Be rigid! Always consider evaluation issues in this process!
- Spend as much time as you can on evaluation :-)

On to Hands-on!