

Beyond Binary Classification

Chris Piech
CS109, Stanford University

Review

Whats new on PSet7?

Pset 6 - Machine Learning Ge x +

← → ⌂ ⌂ cs109psets.netlify.app/fall24/pset6/calibration

PS6

1 ✓
2 ✓
3 ✓
4 ✓
5
6
7a
7b

Logistic Regression: Calibration

Imagine you use logistic regression to make a prediction for a datapoint with features x and your model returns a prediction of a 1. You hesitate. Does the model know its a 1 or is it just guessing? In a situation like this, it might be really important to also know how *confident* the model is in a prediction. For example, it might be critically important for the model to also report that it was 98% confident that the label was a 1.

Luckily, in the case of logistic regression, a concept of confidence is baked into the model. By the Logistic Regression assumption, we have a formula for computing $P(Y = 1|X = x)$:

$$P(Y = 1|X = x) = \sigma(\sum_i x_i \theta_i)$$

Great! But how can we test the trustworthiness of those output probabilities? More formally, how can we check if the values output for $P(Y = 1|X = x)$ are "calibrated"? To evaluate if a model is calibrated we first group datapoints together based on the $P(Y = 1|X = x)$ probability output by the model. Then, among those groups we check how often the label is 1. Here is an example of calibration curve which plots probability groupings on the x-axis and fraction of times that the label was 1 on the y-axis:

A scatter plot titled 'Calibration Curve' showing the relationship between predicted probability and observed fraction of times Y=1. The x-axis is labeled 'ofTimes Y = 1' and ranges from 0.6 to 1.0. The y-axis ranges from 0.0 to 1.0. Three data points are plotted with vertical error bars:

- At approximately 0.75 on the x-axis, the y-value is about 0.75.
- At approximately 0.85 on the x-axis, the y-value is about 0.85.
- At approximately 0.95 on the x-axis, the y-value is about 0.95.

The error bars indicate the range of predicted probabilities for each observed fraction of times Y=1.

Answer Editor Solution

Numeric Answer: Enter your answer Check Answer

Explanation: ⓘ

Block LaTeX ✓ Inline LaTeX Python Image

Previous Question Next Question

68

Whats new on PSet7?

Pset 6 - Machine Learning Ge X +

← → ⌂ ⌂ cs109psets.netlify.app/fall24/pset6/decision_tree_en... ⌂ ⌂ ⌂ Relaunch to update ⌂

PS6

Entropy and Decision Trees

For the heart dataset, a DecisionTree also performs really well at classifying healthy hearts. We would like to understand how the decision tree is making its decision.

A DecisionTree has been trained on heart-train dataset. By interrogating the trained model we learn that the decision tree has a root decision which splits datapoints on the value of "C.4". Datapoints with a C.4 value of 0 will get sorted to the left, datapoints with a C.4 value of 1 will get sorted to the right.

Here is a visualization of the root of the decision tree:

Above each node is the count of labels (y values) for datapoints in the training dataset. Specifically, in the original train dataset:

- There are 80 datapoints. 40 have a Label of 1, and the rest have a Label of 0.
- 52 points have a "C.4" value of 0 and will get sorted into the left child. Of those 52 points, 17 have a Label of 1 and the rest have a Label of 0.
- 28 points have a "C.4" value of 1 and will get sorted into the right child. Of those 28 points, 23 have a Label of 1.

Recall that a Decision Tree is trained to create "decision nodes" that maximize the reduction in expected entropy, which reflects the model's ability to

Answer Editor Solution

Numeric Answer: Enter your answer Check Answer

Explanation: ⓘ

Block LaTeX ✓ Inline LaTeX ✎ Python Image

Previous Question https://cs109psets.netlify.app/fall24/pset6/decision_tree_entropy Next Question

Whats new on PSet7?

The screenshot shows a web browser window titled "Pset 6 - Machine Learning Ge". The URL is "cs109psets.netlify.app/fall24/pset6/linear_regression_b". The page content is for "Linear Regression: Gaussian Errors".

PS6 sidebar:

- 1 (green checkmark)
- 2 (green checkmark)
- 3 (green checkmark)
- 4 (green checkmark)
- 5
- 6
- 7a
- 7b (highlighted)

Linear Regression: Gaussian Errors

For this problem, use the same Linear Regression model and Caltrain datasets as in the previous problem.

A major assumption of classic Linear Regression is that the model's prediction errors (sometimes called residuals, or how "off" your model is for each prediction it makes) are distributed as a Gaussian with a mean of zero. This assumption allows us to reason about how likely a model is to be significantly incorrect.

Let $Z \sim \mathcal{N}(\mu = 0, \sigma^2)$ be a random variable representing the difference between a model's prediction and the true value. We want to choose the value for σ^2 that best describes the model's prediction errors on our test data; that is, we want to find the MLE estimate for σ^2 .

Recall that in class, we derived the MLE estimate for the variance of a Gaussian. If we have n datapoints z_1, z_2, \dots, z_n , we would estimate σ^2 to be:

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (z_i - \mu)^2$$

Your Task

1. Using the Linear Regression model you trained in the previous problem and the data in caltrain-test.csv, compute the prediction error for each datapoint in the test set. Note that error can be positive or negative (you don't need to use the absolute value).
2. Estimate σ^2 for Z from these errors.
3. Use the distribution of Z to find the probability that the number of hourly Caltrain passengers predicted by your model will be off by more than 20 people.

[Previous Question](#) [Next Question](#)

https://cs109psets.netlify.app/fall24/pset6/linear_regression_b

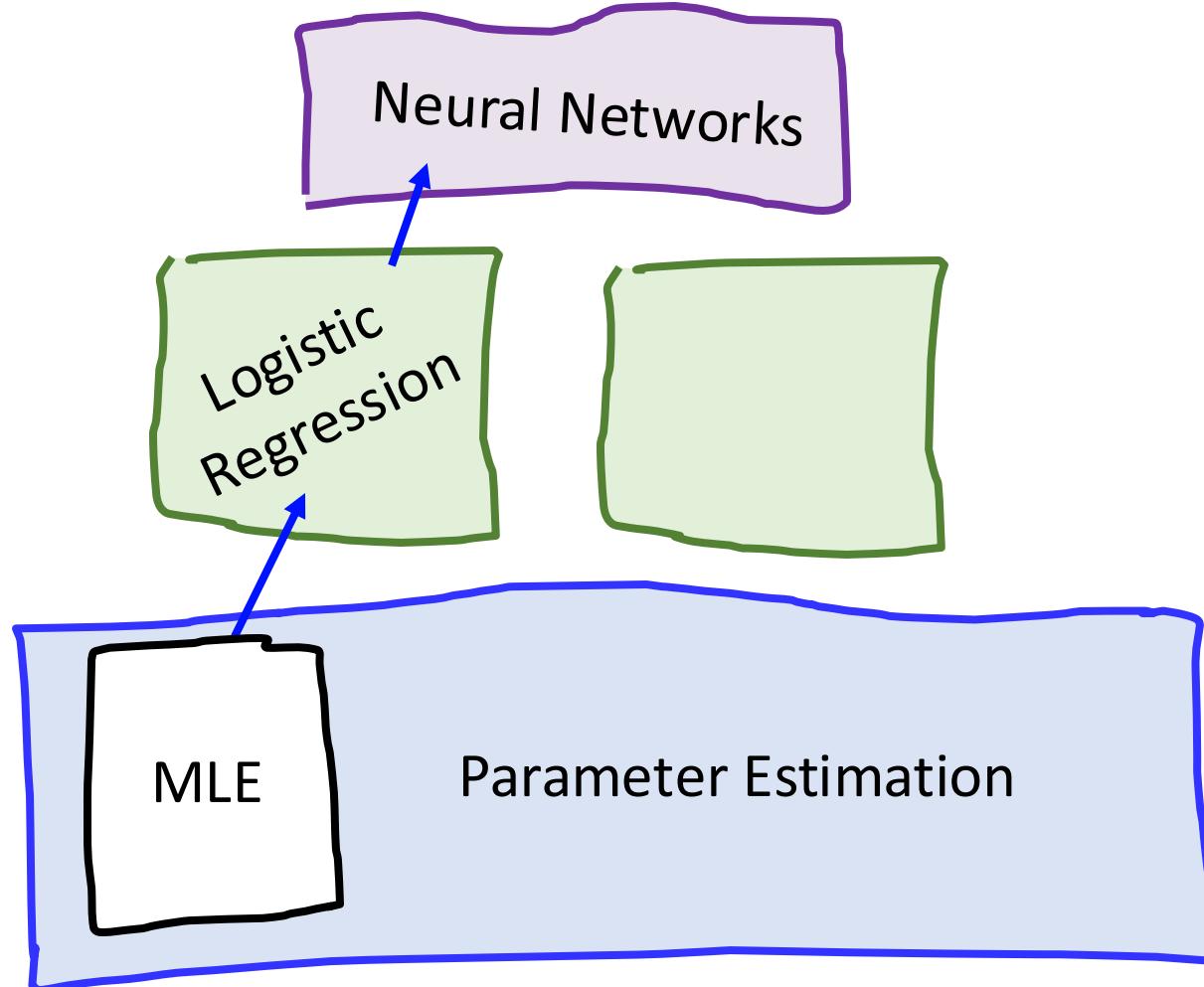
Review

Machine Learning in CS109

Great Idea

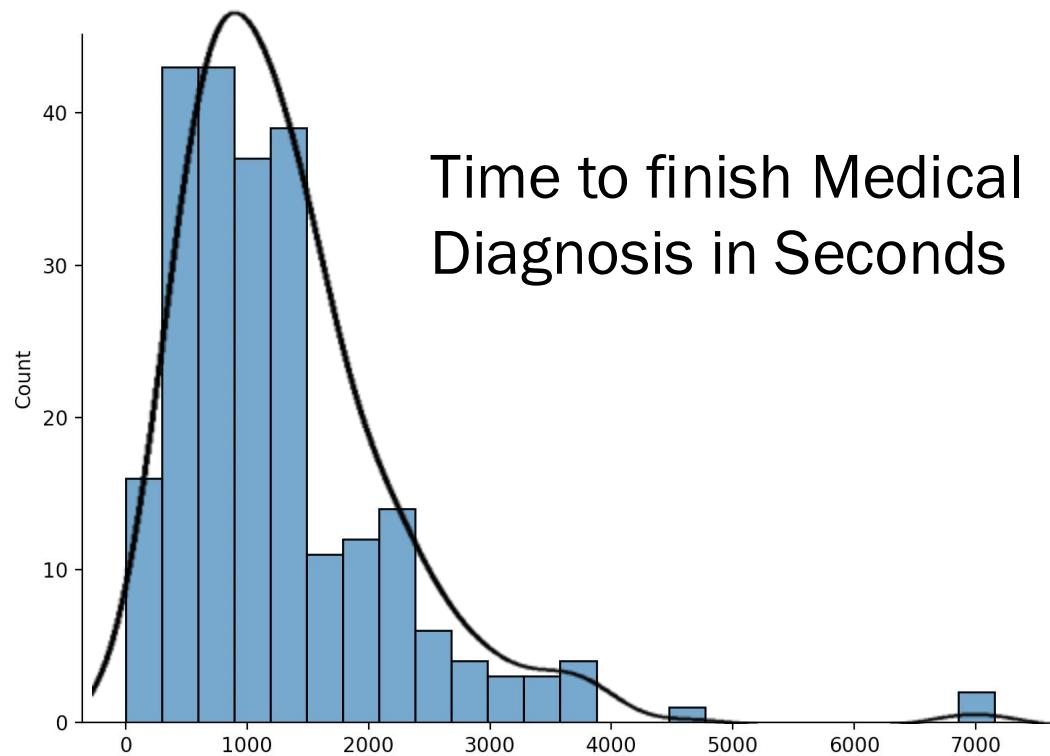
Core
Algorithms

Theory



MLE of Erlang

```
[3.002, 0.983, 2.186, 1.624, 3.997, 1.777,  
2.809, 0.42, 0.515, 1.582, 0.948, 0.458, 1.  
066, 0.8, 2.398, 0.794, 2.561, 2.61, 0.  
595, 3.897, 1.852, 1.182, 3.043, 0.905, 1.  
45, 0.405, 0.445, 2.103, 1.425, 3.12, 0.  
973, 1.056, 3.715, 2.952, 1.817, 2.686, 4.  
173, 0.358, 2.185, 2.581, 7.134, 0.206, 2.  
049, 0.896, 2.095, 4.39, 2.199, 3.434, 5.  
696, 0.819, 0.416, 1.571, 1.337, 2.79, 2.  
701, 3.061, 4.677, 0.671, 1.594, 3.586, 2.  
708, 1.417, 1.799, 1.137, 1.771, 2.12, 0.  
93, 6.835, 3.213, 2.541, 2.505, 1.257, 1.  
99, 1.5, 0.014, 3.856, 0.979, 2.413, 2.  
596, 1.653, 0.881, 4.457, 0.717, 3.305, 2.  
456, 3.462, 1.737, 0.968, 0.528, 0.18, 1.  
626, 2.224, 1.466, 1.6, 1.572, 0.12, 2.86,  
1.062, 2.139, 1.217]
```

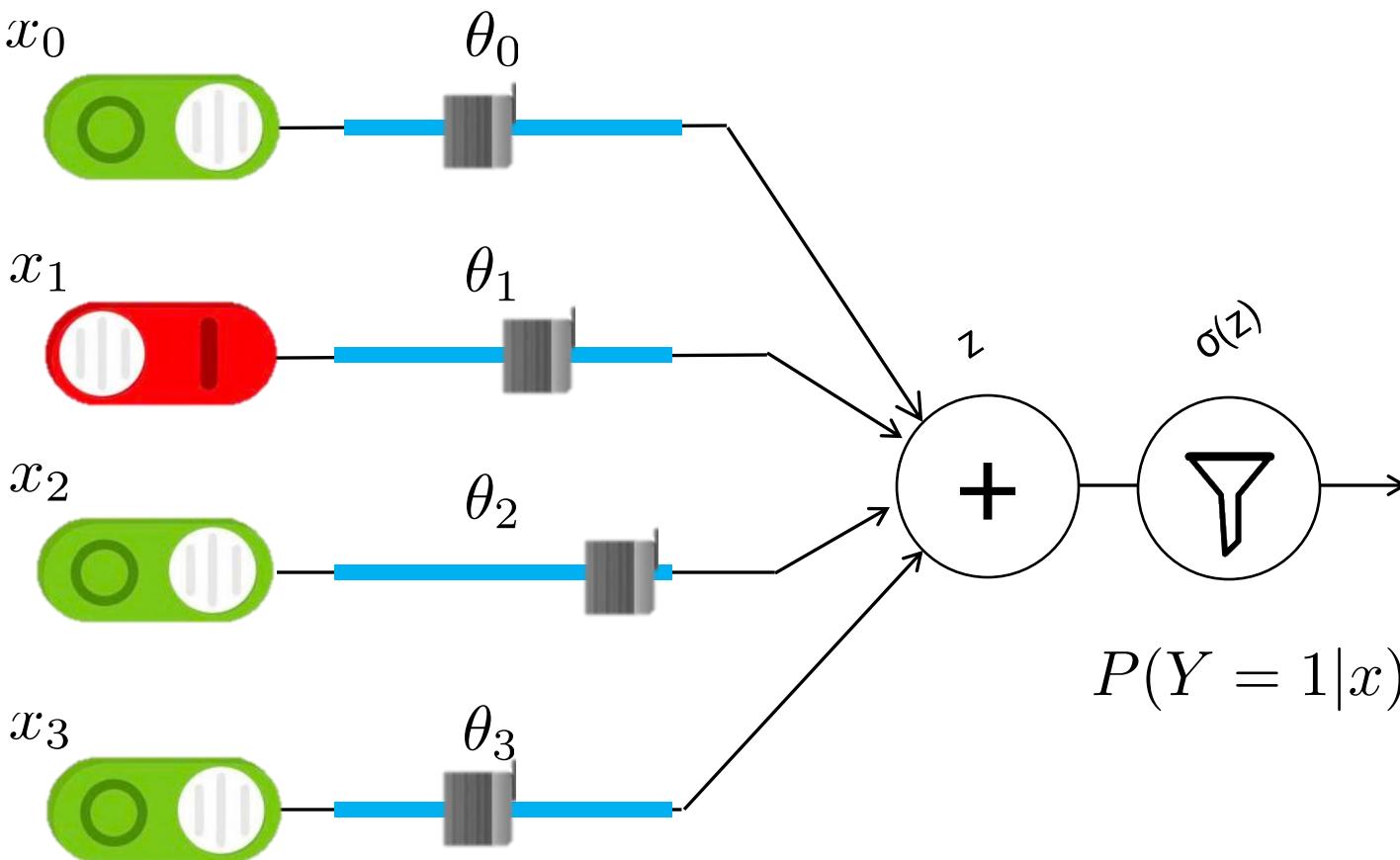


$$f(x) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!} = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{\Gamma(k)}$$

Healthy Heart Classifier

	ROI 1	ROI 2	ROI m	Output
Heart 1	0	1	1	0
Heart 2	1	1	1	0
	⋮	⋮	⋮	⋮
Heart n	0	0	0	1

Logistic Regression



$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Logistic Regression

1

Make logistic regression assumption

$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$

Often call this
 \hat{y}

2

Calculate the log probability for all data

$$LL(\theta) = \sum_{i=0}^n y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

3

Get derivative of log probability with respect to thetas

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^n \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

Accuracy Comparison

Model	Train Accuracy	Test Accuracy
Baseline	0.6138	0.6300
Logistic Regression	0.7300	0.7200
Naive Bayes	0.7275	0.7200
Decision Tree	0.7975	0.6150
Random Forest	0.7950	0.7100
Gradient Boosting	0.7738	0.7250
AdaBoost	0.7588	0.7100

The Kaggle Champion



The **Gradient Boosting** is by far the most successful single model.

Especially the package XGB is used in pretty much every winning (and probably top 50%) solution. XGB has essentially become the first model you try and the best performing single model by the end.



Kaggle

<https://www.kaggle.com> › general ::

[What machine learning approaches have won most ... - Kaggle](#)

XGB is for
extreme
gradient
boosting

Wednesday: Three Guiding Questions

1. What are other models for classification?
2. For a dataset, how do you tell which one is best?
3. What are other validation metrics I might care about?

How **well calibrated** are my probabilities?

Measuring if Probabilities are Calibrated

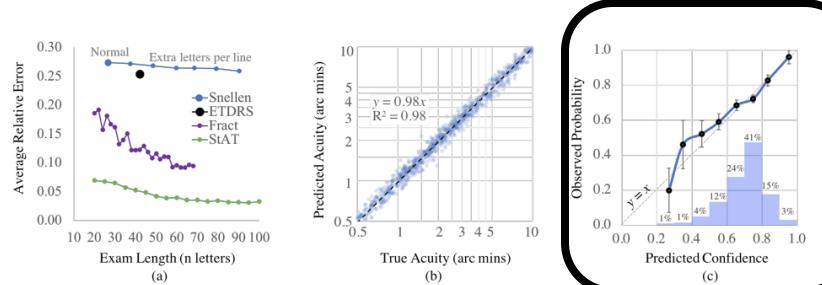


Figure 4: (a) The tradeoff between length of exam and error for the different algorithms. (b) A visualization of the predictions made by StACT. (c) Calibration test: StACT confidences correspond to how often it is correct.

4.2 Baseline Acuity Tests

We use the following baselines and prior algorithms to compare against the StACT algorithm.

Const Policy. This policy always predicts the most common visual acuity in our data i.e. the mode of the visual acuity prior. This serves as a true null model because it doesn't take patient responses into account at all.

Snellen and ETDRS. The Revised 2000 Series ETDRS charts and the Traditional Snellen Eye Chart were programmed so that we could simulate their response to different virtual patients. Both exams continue until the user incorrectly answers questions for more than half of the letters on a line. ETDRS has a function for predicted acuity score that takes into account both the last line passed, and how many letters were read on the last line not-passed. Both charts use 19 unique optotypes.

FrACT. We use an implementation of the FrACT algorithm (Bach and others 1996), with the help of code graciously shared by the original author. We also included the ability to learn the “*s*” parameter as suggested by the 2006 paper (Bach 2006), and verified that it improved performance.

5 Results and Evaluation

The results of the experiments can be seen in Table 1.

Accuracy and error. As can be seen from Table 1, the StACT test has substantially less error than all the other baselines. After 20 optotype queries, our algorithm is capable of predicting acuity with an average relative error of 0.069. This prediction is a 74% reduction in error from our implementation of the ubiquitous Snellen test (average error = 0.276), as well as a 67% reduction in error from the FrACT test (average error = 0.212). One possible reason for the improvement over FrACT is that the simulations used in our evaluations are based off the Floored-Exponential model that StACT uses. However, even when we evaluate StACT on simulations drawn from the FrACT logistic assumption we still achieve a 41% reduction. The improved accuracy of the StACT algorithm suggests our Bayesian approach

	μ Acuity Error	μ Test length
Const	0.536	0
Snellen [†]	0.264	27
ETDRS [†]	0.254	42
FrACT	0.212	20
StACT	0.069	20
StACT-noSlip	0.150	20
StACT-greedyMAP	0.132	20
StACT-logistic	0.125	20
StACT-noPrior	0.090	20
StACT-goodPrior	0.047	20
StACT-star	0.038	63

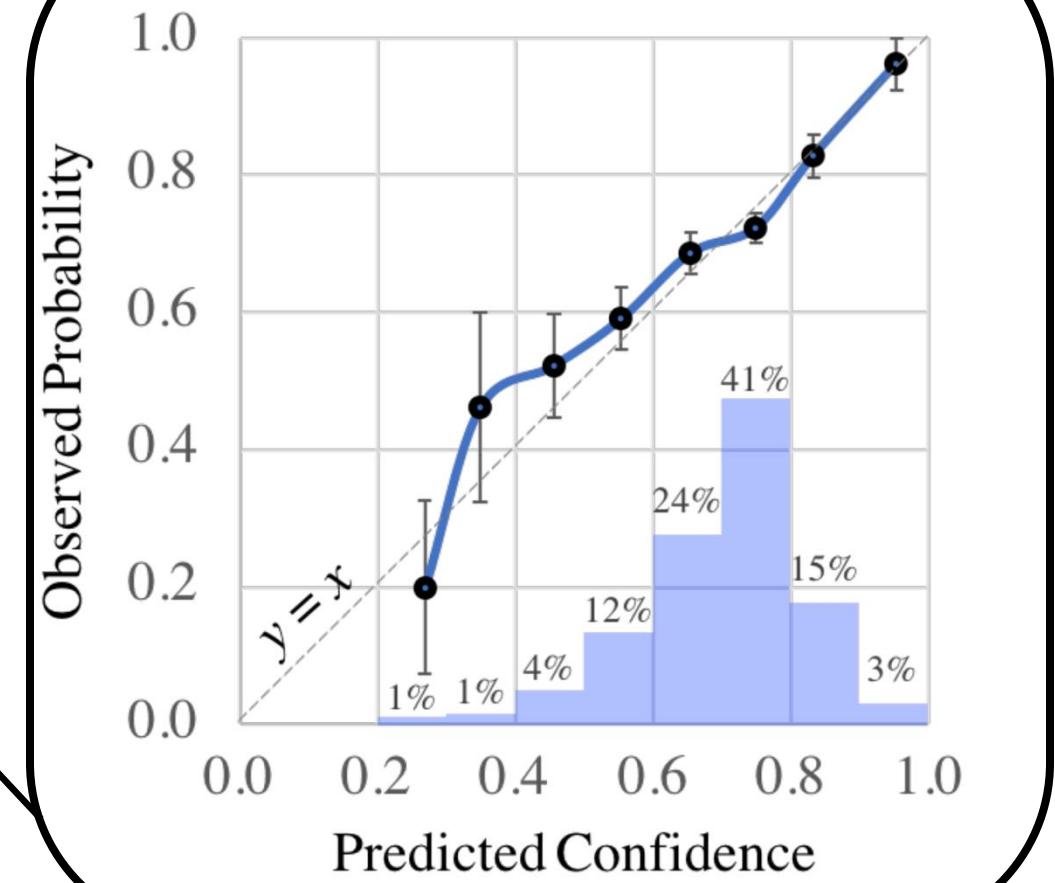
Table 1: Average relative error for each algorithm. Except for Snellen each test was allowed 20 letters. Results are average relative error after 1000 tests. [†] Snellen and ETDRS used 19 unique optotypes.

to measuring acuity is a fruitful proposal both because of our introduction of the floored exponential as well as our Thompson-sampling inspired algorithm to choose a next letter size.

Figure 4 (b) visualizes what StACT's small relative error means in terms of predictions. Each point in the plot is a single patient. The x-axis is the true acuity of the patient and the y-axis is the predicted accuracy. We can qualitatively observe that the predictions are often accurate, there are no truly erroneous predictions, and that the exam is similarly accurate for patients of all visual acuities.

Moreover, as seen in Figure 4 (a), StACT's significant improvement in error rate holds even when the length of the exam is increased. It is also evident that increasing exam length reduces our error rate: if we increase the exam length to 200 letters, the average error of StACT falls to 0.020. While this is highly accurate, its far too long an exam, even for patients who need to know their acuity to high precision.

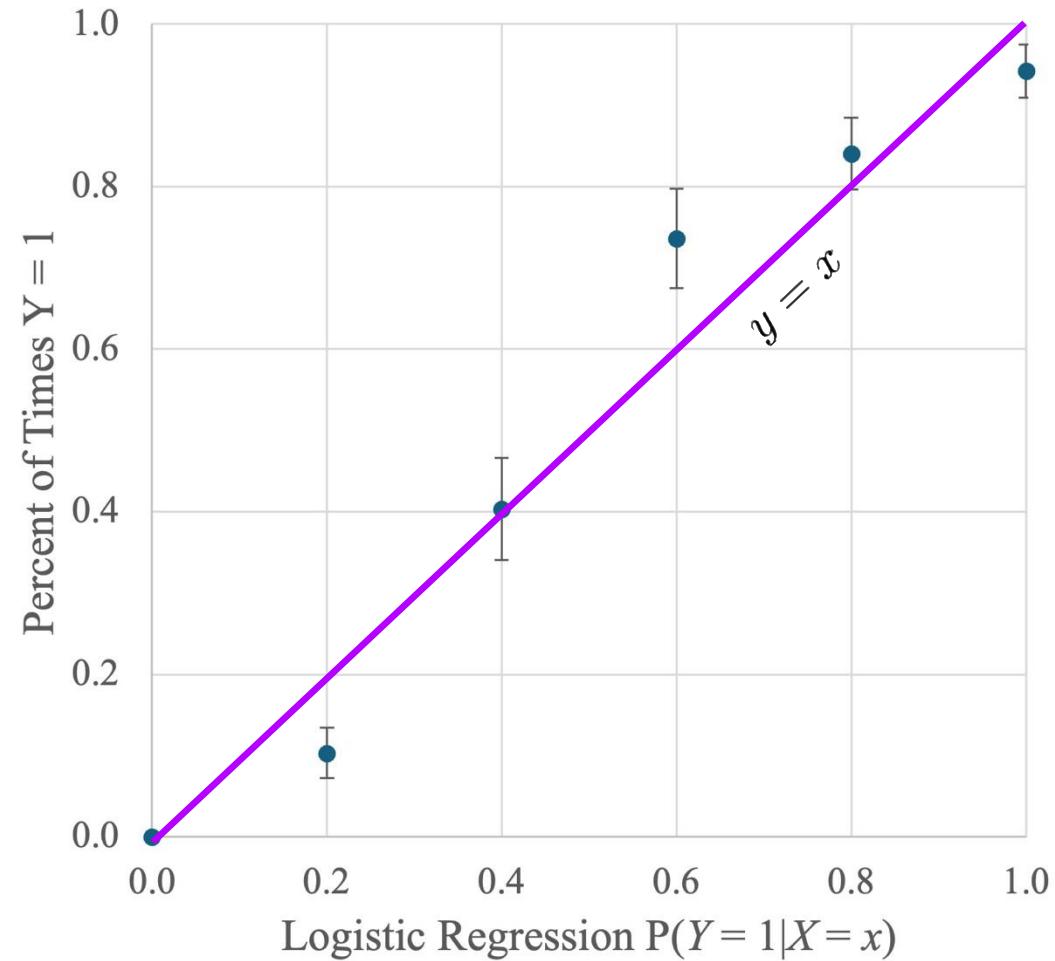
StACT Star Exam. Our primary experiments had a fixed



Logistic Regression Calibration Curve

$$x_1 \quad x_2 \quad x_{19} \quad y \quad \checkmark \quad P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

	A	B	T	U	V
1	col1	col2	col19	Label	LogRegPr
2	1	0	1	0	0.237
3	1	1	1	1	0.928
4	1	0	0	1	0.541
5	1	0	1	0	0.003
6	1	0	0	0	0.914
7	1	0	0	1	0.432
8	1	0	1	0	0.001
9	1	0	1	1	0.530
10	0	0	1	0	0.090
11	1	0	1	0	0.439
12	1	0	1	0	0.032
13	1	0	1	0	0.114
14	0	0	1	1	0.848
15	1	0	1	0	0.025
16	1	0	1	0	0.180
17	0	0	1	1	0.672
18	1	0	1	1	0.531
19	1	0	1	0	0.012
20	1	0	1	1	0.560
21	1	0	1	1	0.502



End Review

One Final Measure....

Is the Model Fair?

Two Philosophic Values of Fairness

Procedural Fairness:

Focuses on the decision-making or classification process, ensures that the algorithm does not rely on unfair features.

Distributive Fairness:

Focuses on the decision-making or classification *outcome*, ensures that the distribution of good and bad outcomes is equitable.

Two Philosophic Values of Fairness

Procedural Fairness:

Focuses on the decision-making or classification process, ensures that the algorithm does not rely on unfair features.

Distributive Fairness:

Focuses on the decision-making or classification *outcome*, ensures that the distribution of good and bad outcomes is equitable.

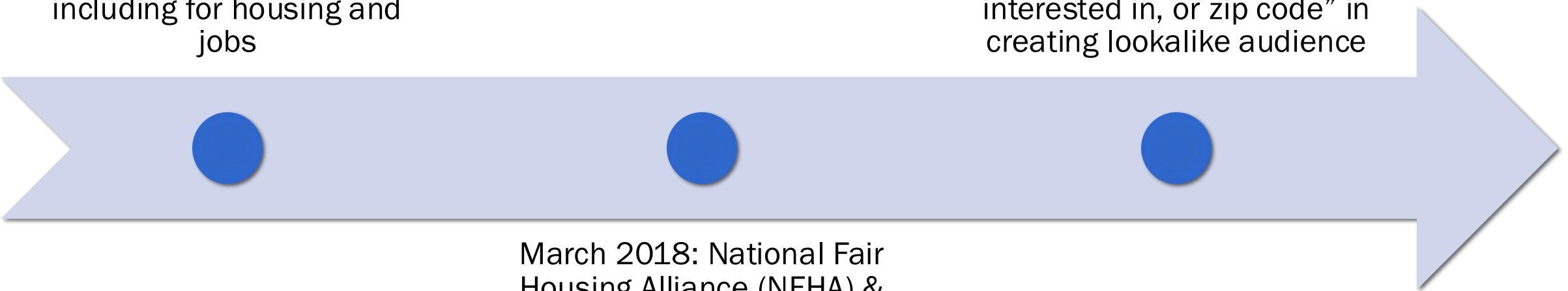
Fairness through unawareness



Case Study: Facebook Ads & Job/Housing Recommendations

Facebook creates “Lookalike” feature for advertisers: upload a “source list” and find users with “common qualities” to target ads, including for housing and jobs

March 2019: As part of settlement, Facebook agrees not to use “age, gender, relationship status, religious views, school, political views, interested in, or zip code” in creating lookalike audience



March 2018: National Fair Housing Alliance (NFHA) & other civil rights groups sue Facebook over violations of the Fair Housing Act

Facebook Input Lookalikes

The screenshot shows the 'Create a Lookalike Audience' interface in the Facebook Ads Manager. It consists of three main steps:

- 1 Select Your Lookalike Source**: A dropdown menu labeled 'Select an existing audience or data source' with a 'Create New Source' button.
- 2 Select Audience Location**: A dropdown menu showing 'Countries > North America' and 'United States' selected. A search bar below it says 'Search for regions or countries'.
- 3 Select Audience Size**: A slider labeled 'Number of lookalike audiences' set to 1. The slider scale ranges from 0% to 8%, with '2.3M' displayed at the 1% mark. A note below states: 'Audience size ranges from 1% to 10% of the combined population of your selected locations. A 1% lookalike consists of the people in your lookalike source. Increasing the percentage creates a bigger, broader audience.'

The screenshot shows the 'Create a Special Ad Audience' interface in the Facebook Ads Manager. It follows a similar three-step process:

- 1 Select Your Source**: A dropdown menu labeled 'Select an existing audience or data source'.
- 2 Select Audience Location**: A dropdown menu showing 'Countries > North America' and 'United States' selected. A search bar below it says 'Search for regions or countries'.
- 3 Select Audience Size**: A slider labeled 'Number of Special Ad Audiences' set to 1. The slider scale ranges from 0% to 8%, with '2.3M' displayed at the 1% mark. A note below states: 'Audience size ranges from 1% to 10% of the combined population of your selected locations. A 1% Special Ad Audience consists of the most similar online behavior to your source. Increasing the percentage creates a bigger, broader audience.'

New “Special Ad” Audiences Still Biased

Gender: Equally Biased

Age: Almost as Biased

Race: more difficult to measure given the tools provided but still somewhat biased

Political Views: Less Biased

Sapiezynski et. al 2019,

<https://sapiezynski.com/papers/sapiezynski2019algorithms.pdf>

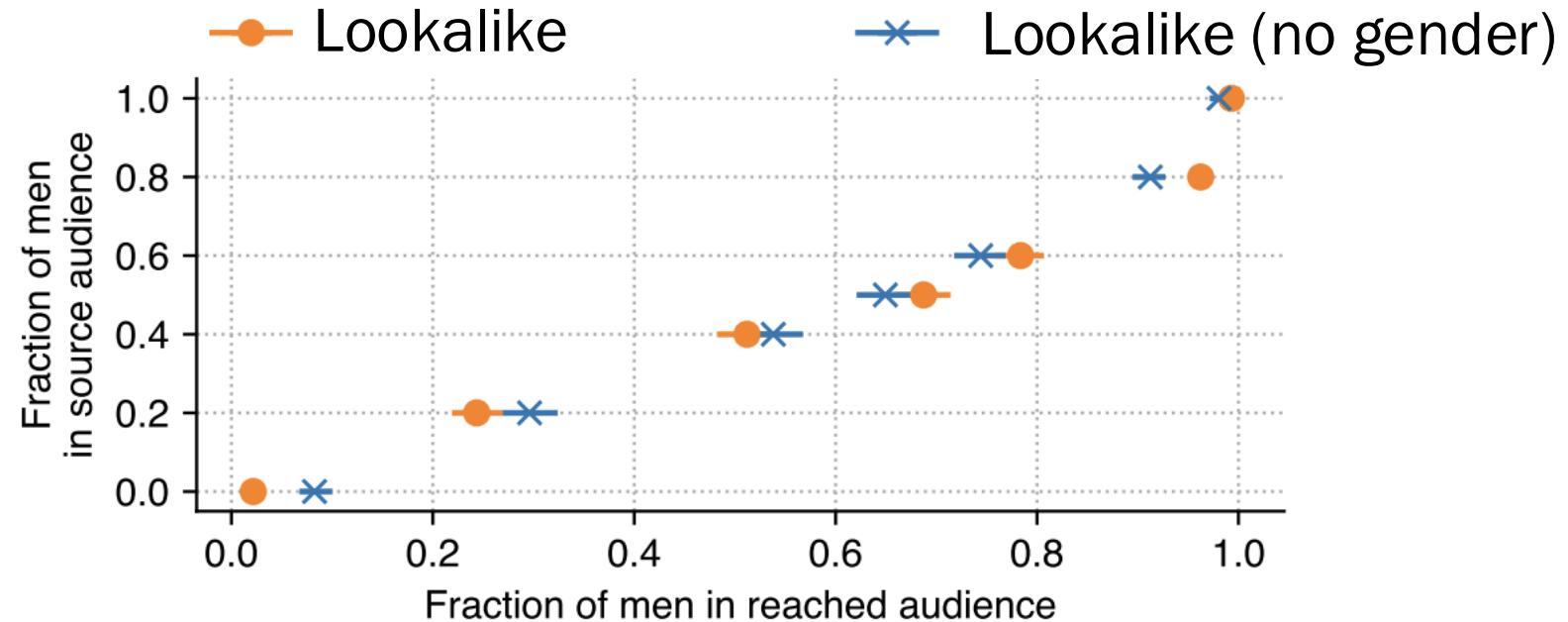
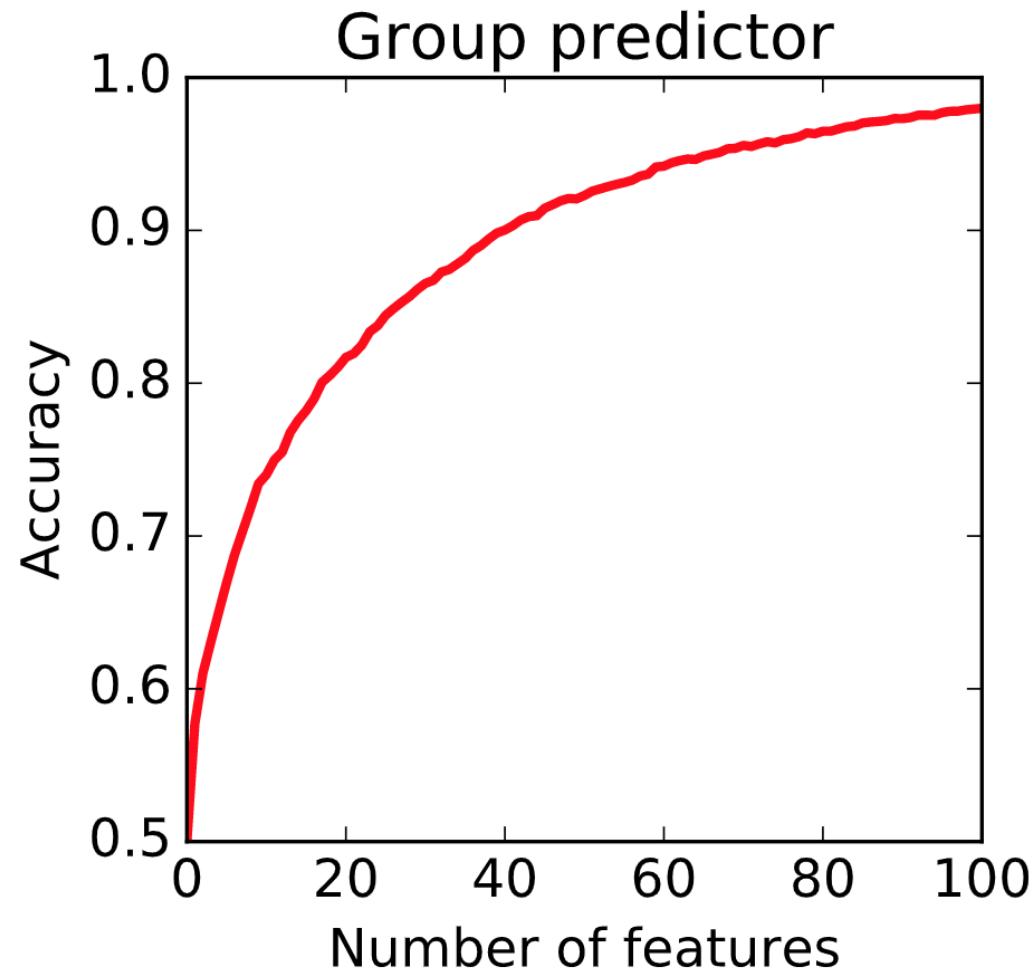


Figure 2: Gender breakdown of ad delivery to Lookalike and Special Ad audiences created from the same source audience with varying fraction of male users, using the same ad creative. We can observe that both Lookalike and Special Ad audiences reflect the gender distribution of the source audience, despite the lack of gender being provided as an input to Special Ad Audiences.

Yo, Piotr, you got your axis backwards 😊



Many Features = Accurate Group Prediction

Sensitive attributes are often “redundantly encoded” in the dataset

Many of the features or datapoints are correlated with the sensitive attribute

Two Philosophic Values of Fairness

Procedural Fairness:

Focuses on the decision-making or classification process, ensures that the algorithm does not rely on unfair features.

Distributive Fairness:

Focuses on the decision-making or classification *outcome*, ensures that the distribution of good and bad outcomes is equitable.



Fairness through unawareness
(facebook example shows this is hard)

Let's Try Fairness Through Awareness!

Awareness of what?

Fairness Through Awareness Terms

D : protected demographic

G : guess of your model (aka \hat{y})

T : the true value (aka y)

$D = 0$

$D = 1$

	$G = 0$	$G = 1$
$T = 0$	0.21	0.32
$T = 1$	0.07	0.28

	$G = 0$	$G = 1$
$T = 0$	0.01	0.01
$T = 1$	0.02	0.08

Distributive Fairness #1: Parity

Fairness definition #1: Parity

An algorithm satisfies “parity” if the probability that the algorithm makes a positive prediction ($G = 1$) is the same regardless of begin conditioned on demographic variable.

D : protected demographic

G : guess of your model (aka y hat)

T : the true value (aka y)

$$P(G=1|D=1) = P(G = 1 \mid D = 0)$$

Distributive Fairness #2: Calibration

Fairness definition #2: Calibration

An algorithm satisfies “calibration” if the probability that the algorithm is correct ($G = T$) is the same regardless of demographics.

D : protected demographic

G : guess of your model (aka \hat{y})

T : the true value (aka y)

$$P(G = T|D = 0) = P(G = T|D = 1)$$

Calibration (Relaxed)

Fairness definition #2: Calibration

An algorithm satisfies “calibration” if the probability that the algorithm is correct ($G = T$) is the same regardless of demographics.

D : protected demographic

G : guess of your model (aka y hat)

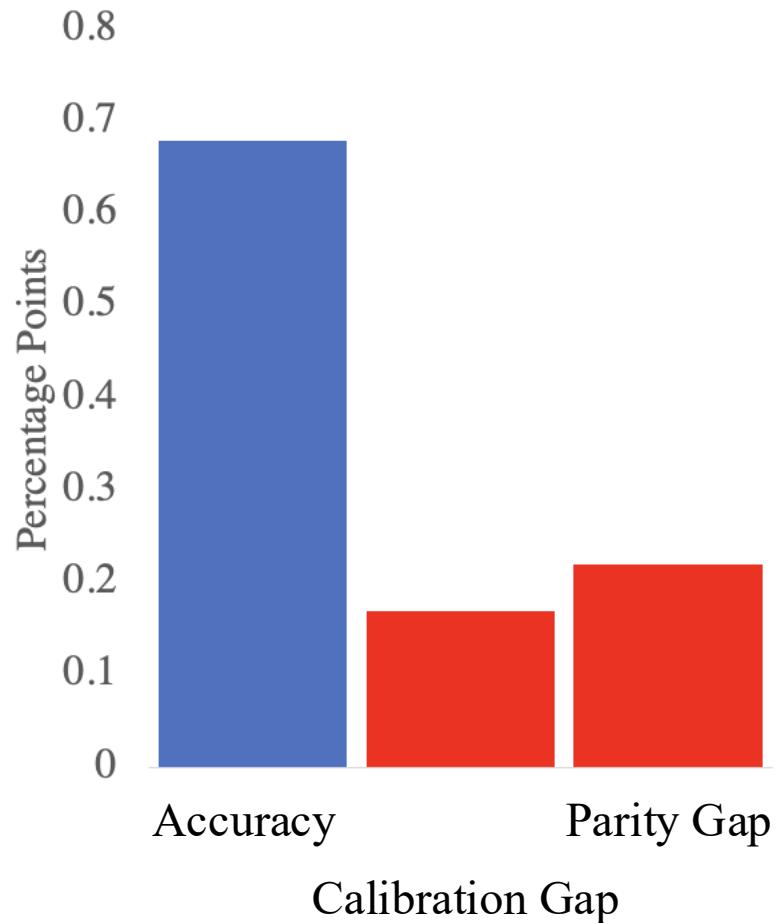
T : the true value (aka y)

$$\frac{P(G = T|D = 1)}{P(G = T|D = 0)} \geq 1 - \epsilon \quad \text{Where epsilon} = 0.2$$

US legal standard: “disparate impact,” also known as the 80% rule.

COMPAS: Biased Against Black Inmates

Before: Compas is Biased



Train bias out

Advanced Idea: Adversarial Learning

Achieving Fairness through Adversarial Learning: an Application to Recidivism Prediction

Christina Wadsworth

Stanford University

Stanford, CA

cwads@cs.stanford.edu

Francesca Vera

Stanford University

Stanford, CA

fvera@cs.stanford.edu

Chris Piech

Stanford University

Stanford, CA

piech@cs.stanford.edu



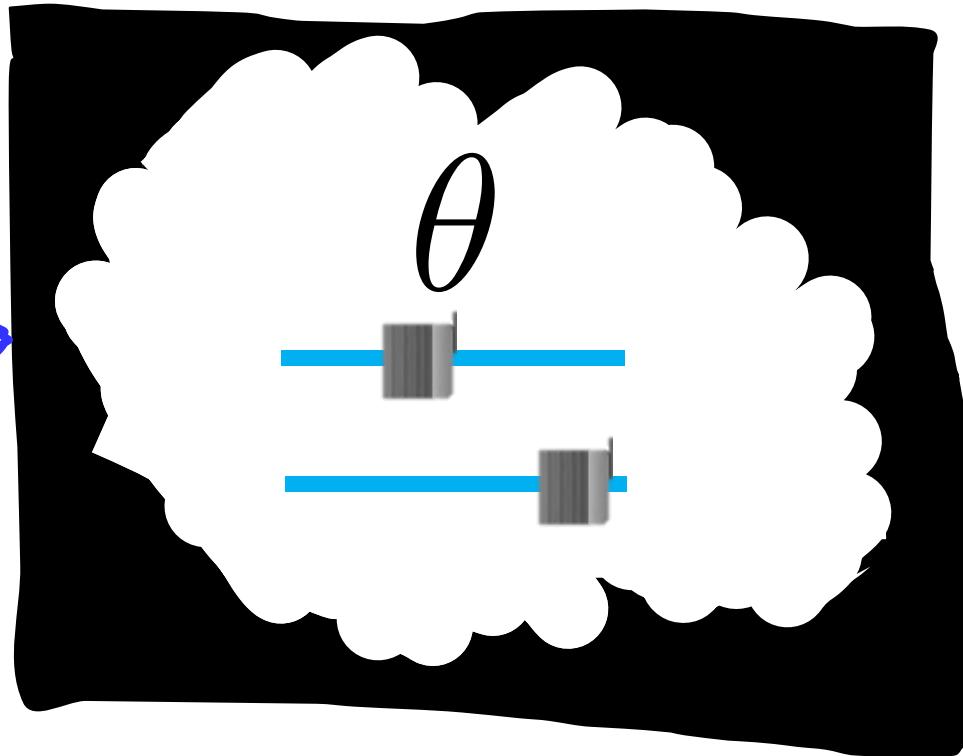
220+ citations

Seniors at the time
they wrote it

COMPAS: Predicting “Recidivism”

X

Data about an inmate:
Their zip code,
past crimes, etc



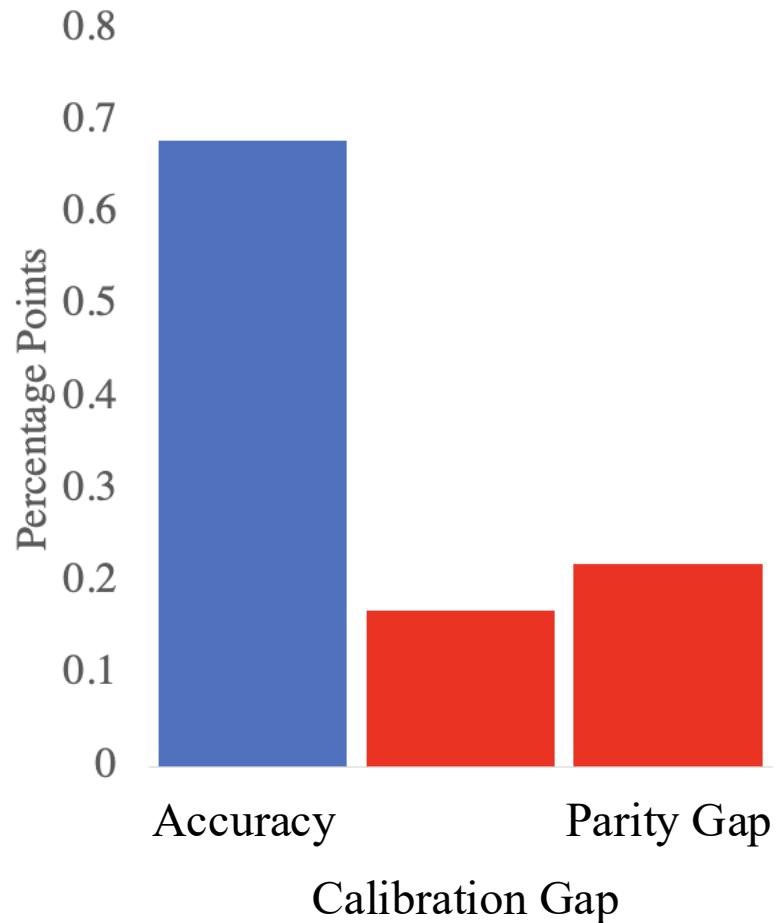
$$\hat{y} = 0$$

Will they commit a crime again

Was in use in California and Florida

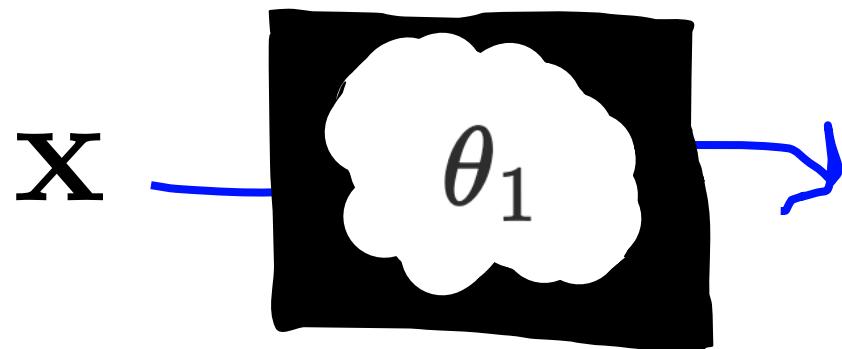
COMPAS: Biased Against Black Inmates

Before: Compas is Biased

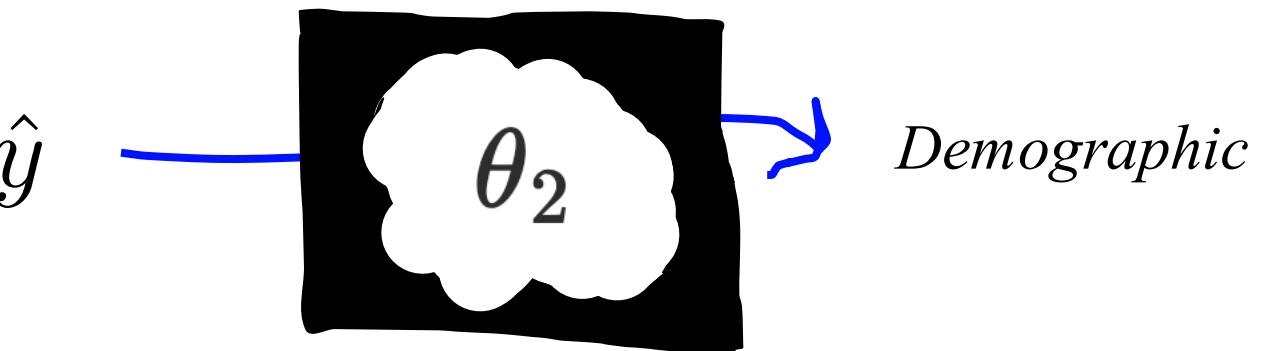


Can We Train Out Bias?

Model 1: Prediction



Model 1: Extract Demographic



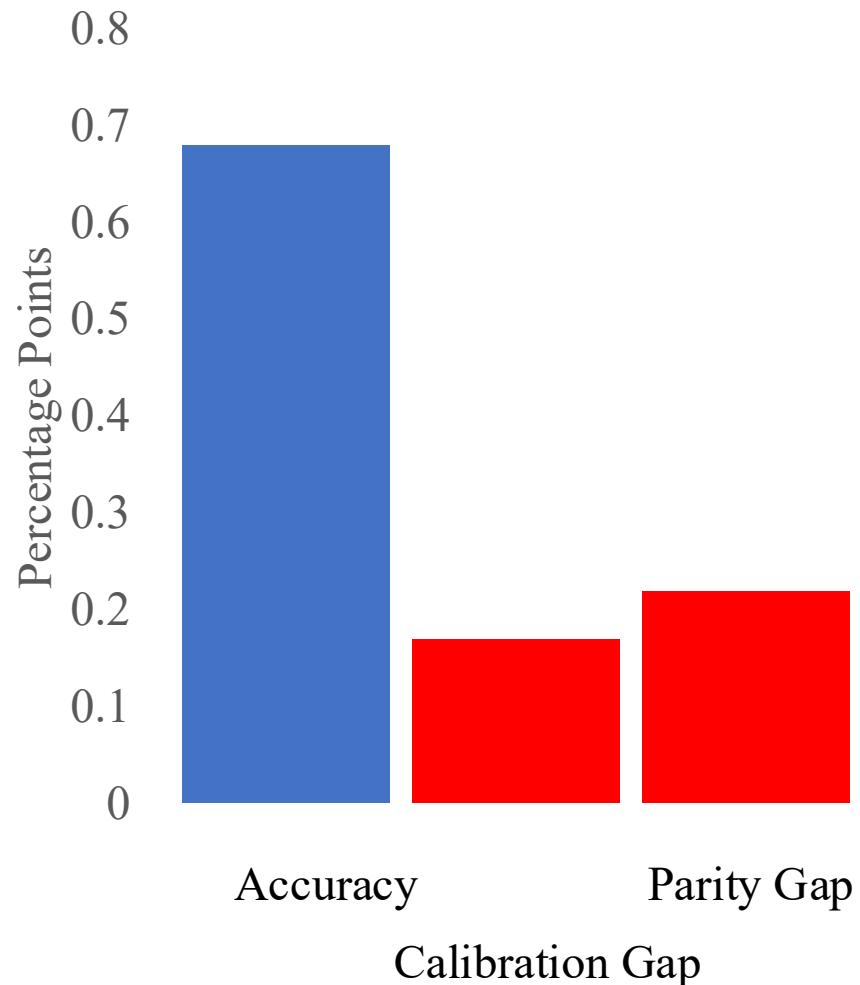
*Model 1 should
be accurate* *Model 2 should
be inaccurate*

$$\theta_1 = \operatorname{argmax}_{\theta_1} L_1(\theta_1) - L_2(\theta_2)$$

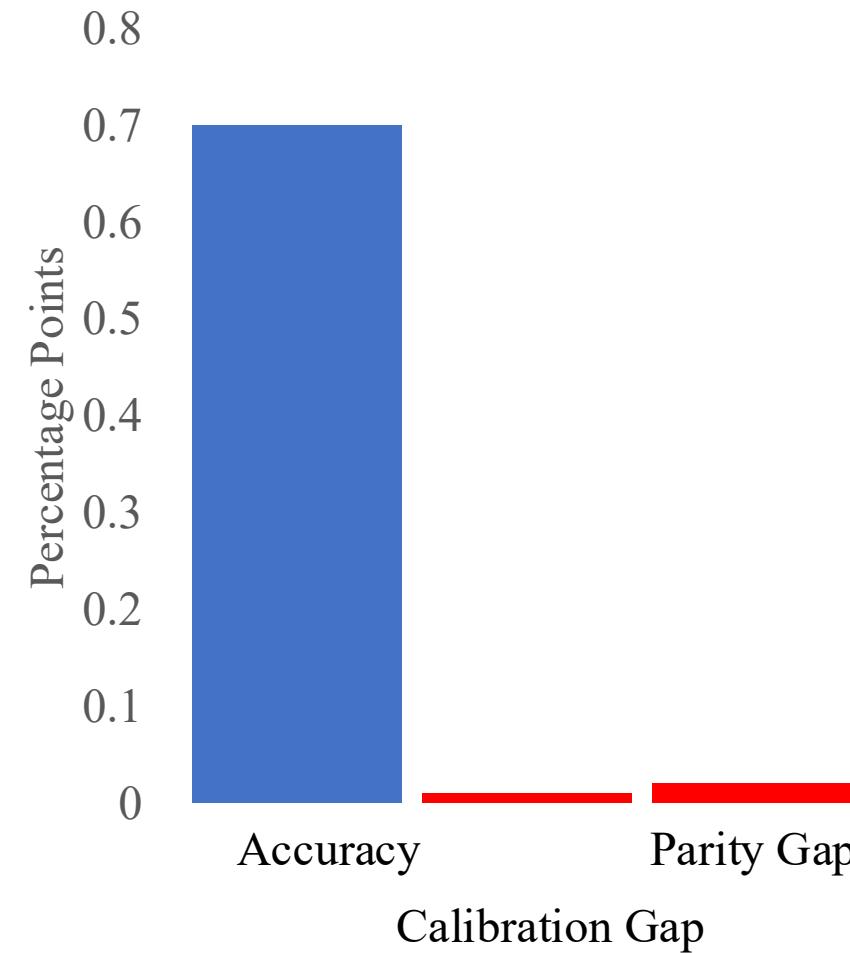
*note in the paper these were neural nets

Can We Train Out Bias?

Before: Compas is Biased



After: Gaps are reduced



Their Conclusion

DON'T USE BLACK
BOX ALGORITHMS TO
MAKE RECIDIVISM
PREDICTIONS

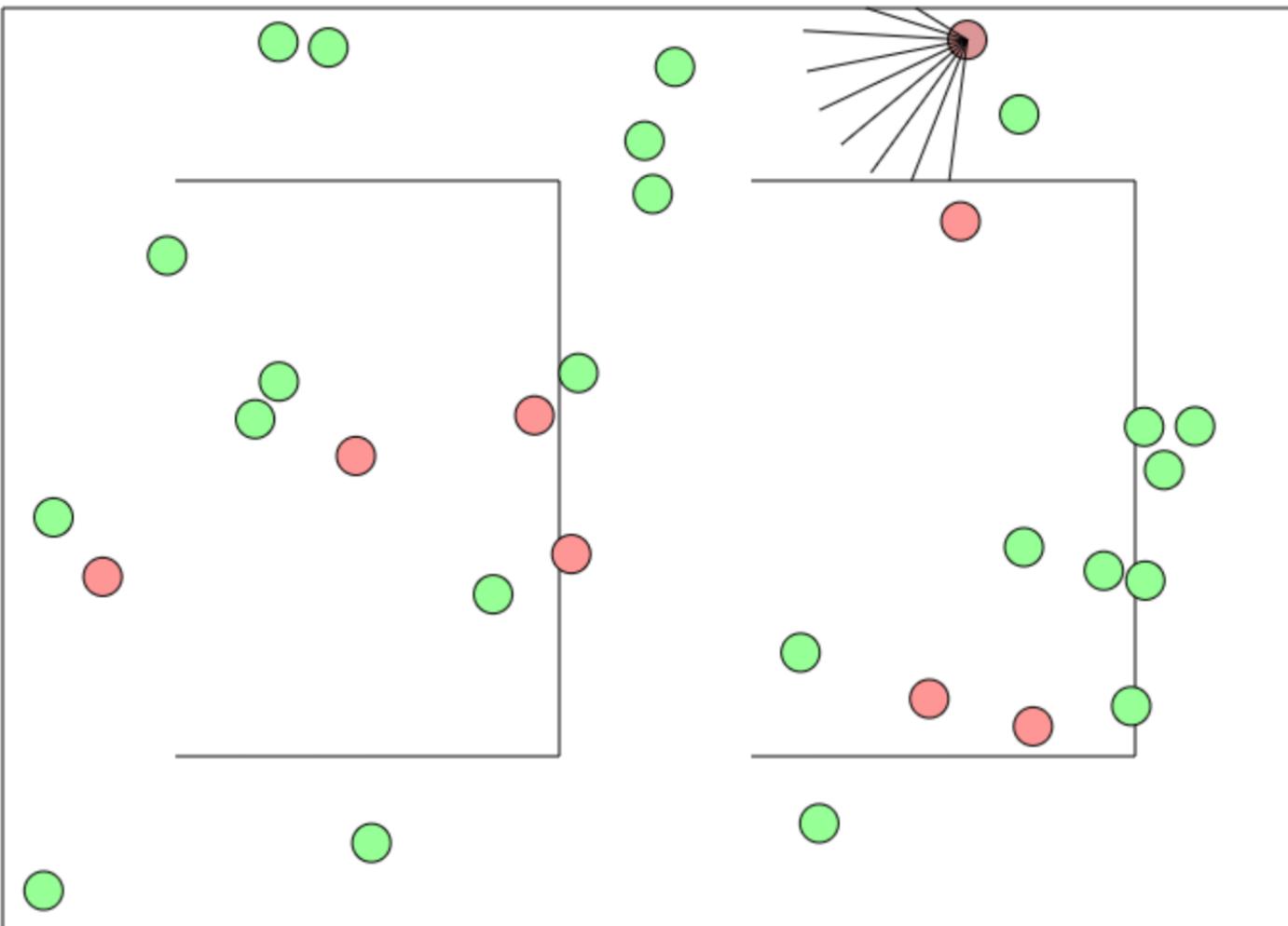
Many ways to evaluate models:

Test Accuracy!
Calibration!
Fairness!

But wait....

Is all of ML Classification?

Lets start training a Critter



<http://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html>

Stanford University

Types of Machine Learning Tasks

Multi-Class
Classification

Regression

Reinforcement
Learning

Generation

Types of Machine Learning Tasks

Multi-Class
Classification

Regression

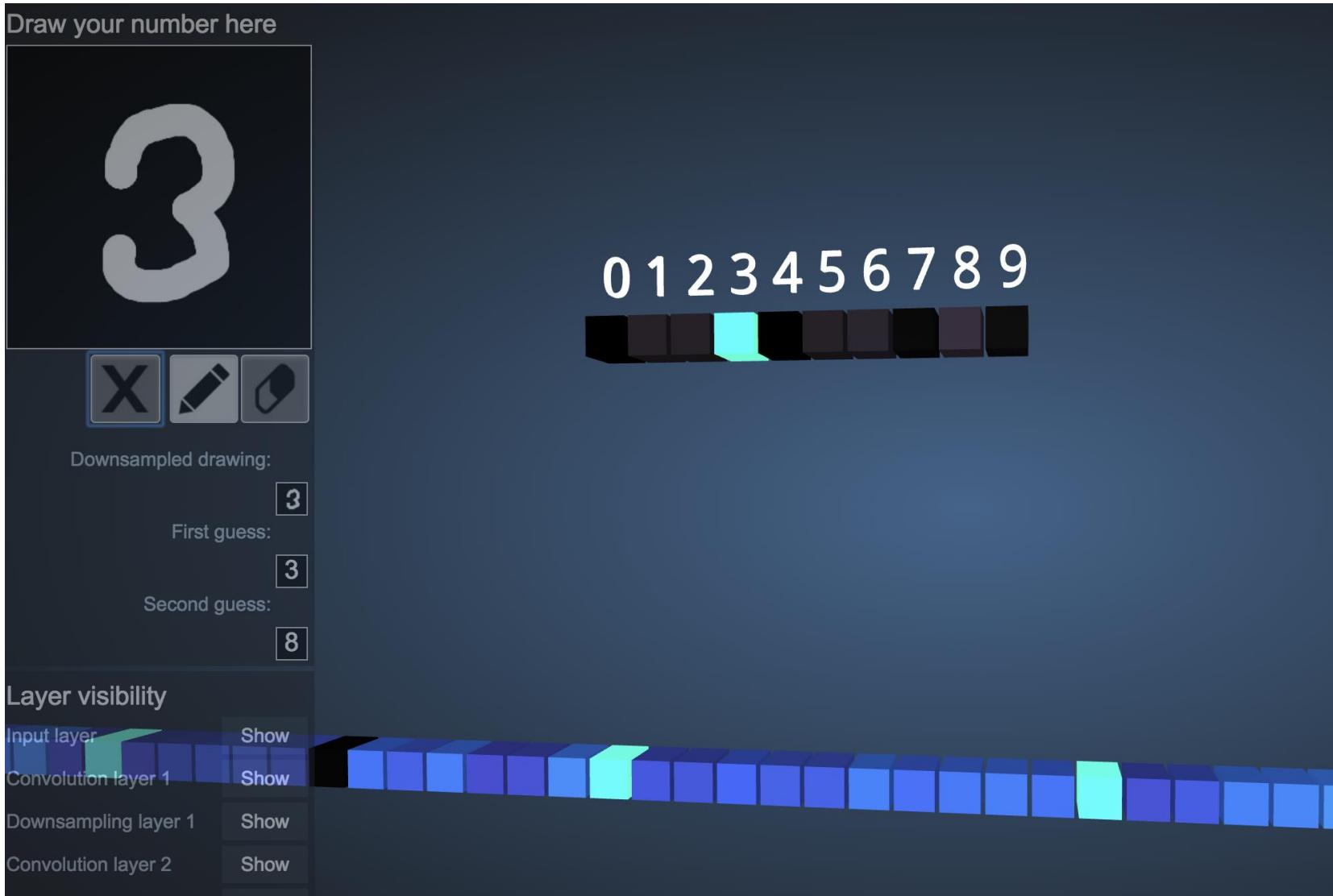
Today

Reinforcement
Learning

Generation

Beyond Binary Classification

Multiple Outputs



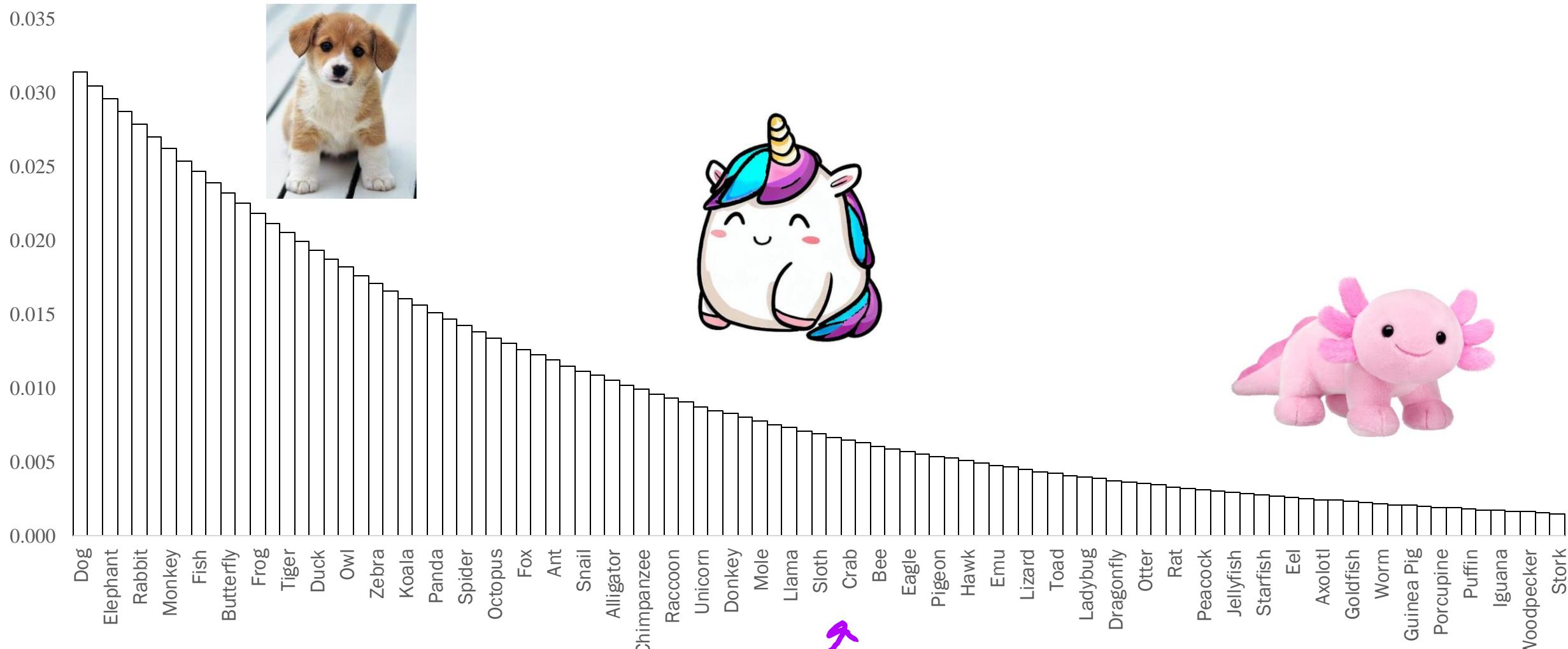
The Categorical

	Binary results (eg coin)	More than two outcomes (eg dice)
One experiment	Bernoulli	???
Many experiments	Binomial	Multinomial

The Categorical

	Binary results (eg coin)	More than two outcomes (eg dice)
One experiment	Bernoulli	Categorical (or just Random Variable)
Many experiments	Binomial	Multinomial

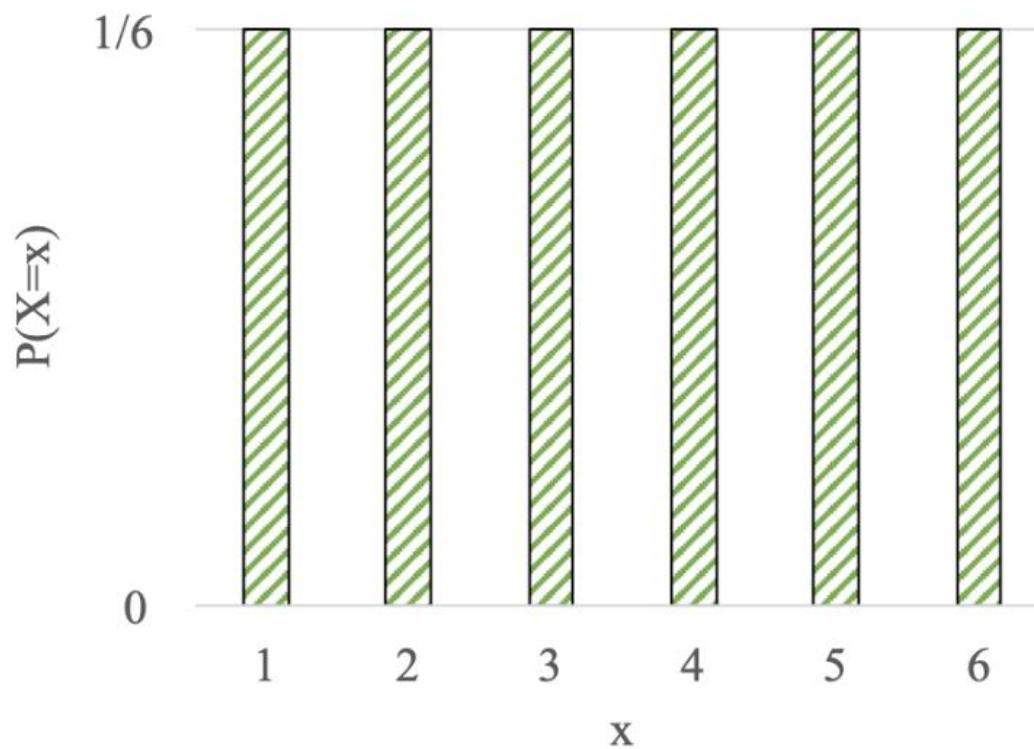
Categorial Example from Class: Thinking of an Animal



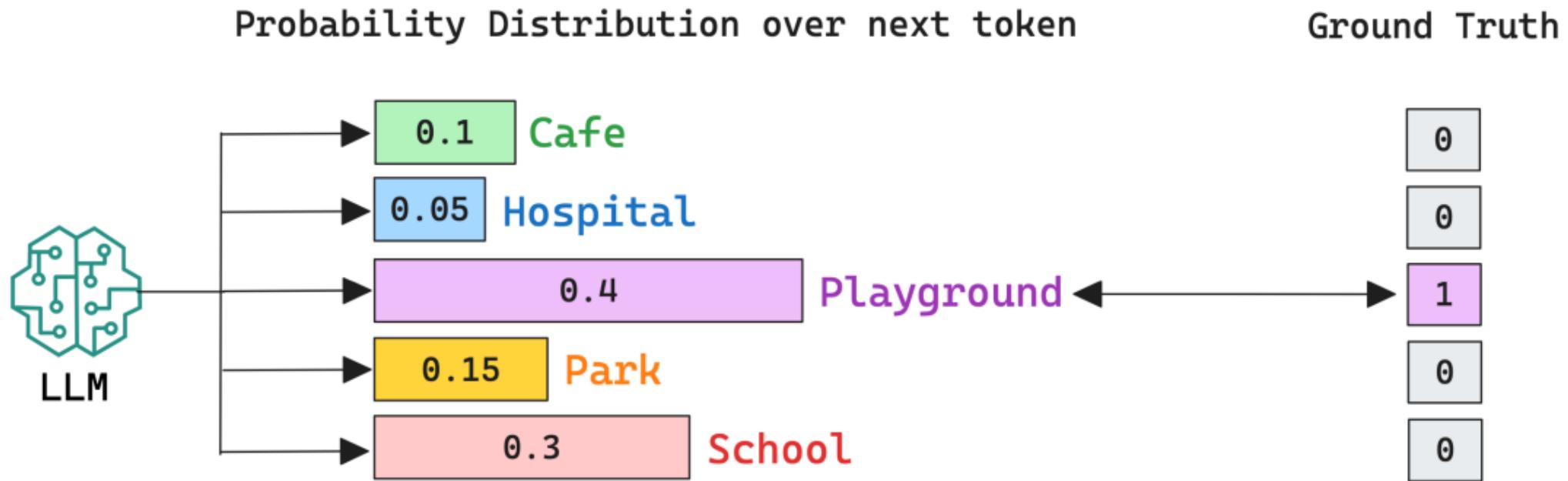
Notice that these are not numbers



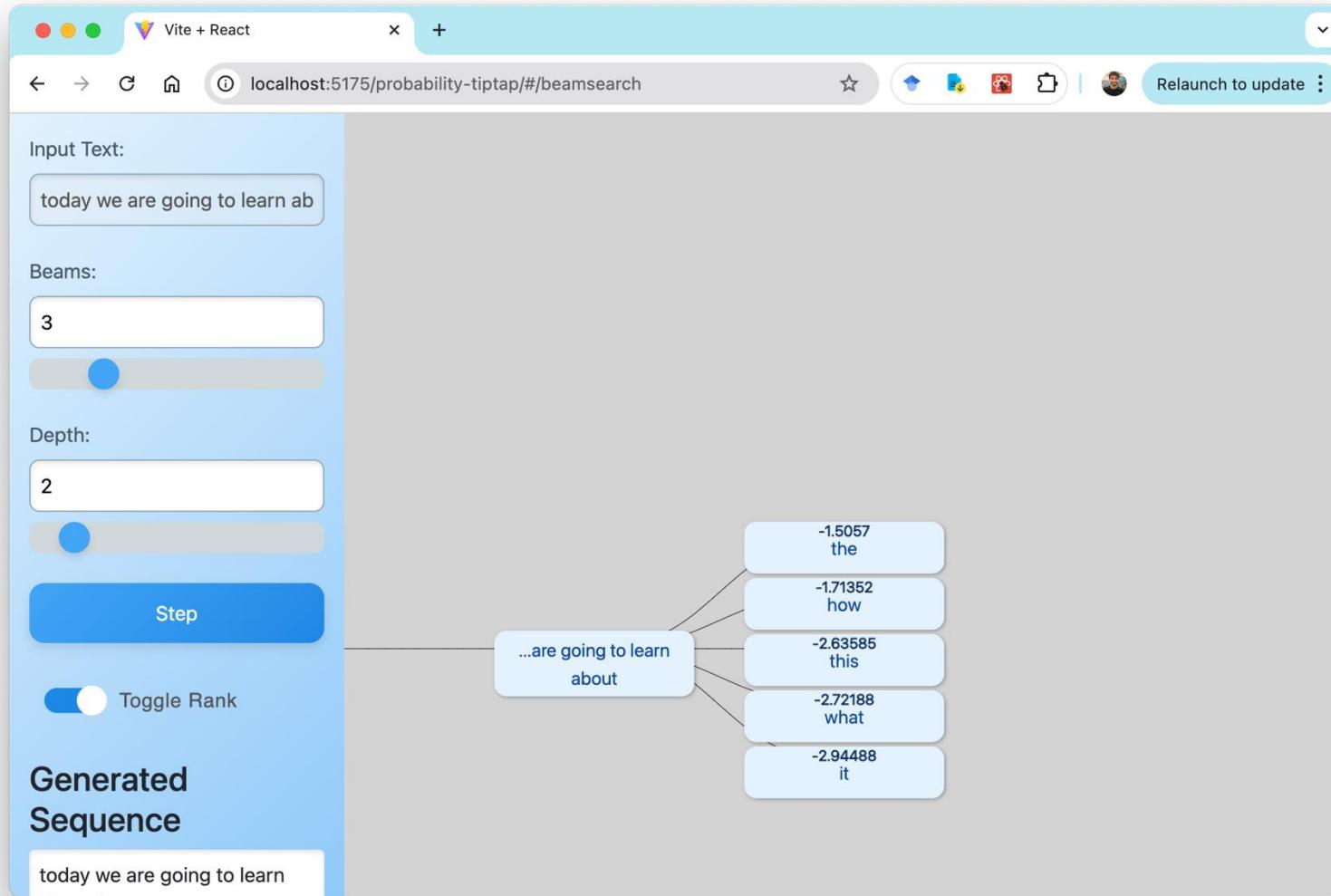
Categorial Example from Class: Single Dice Roll



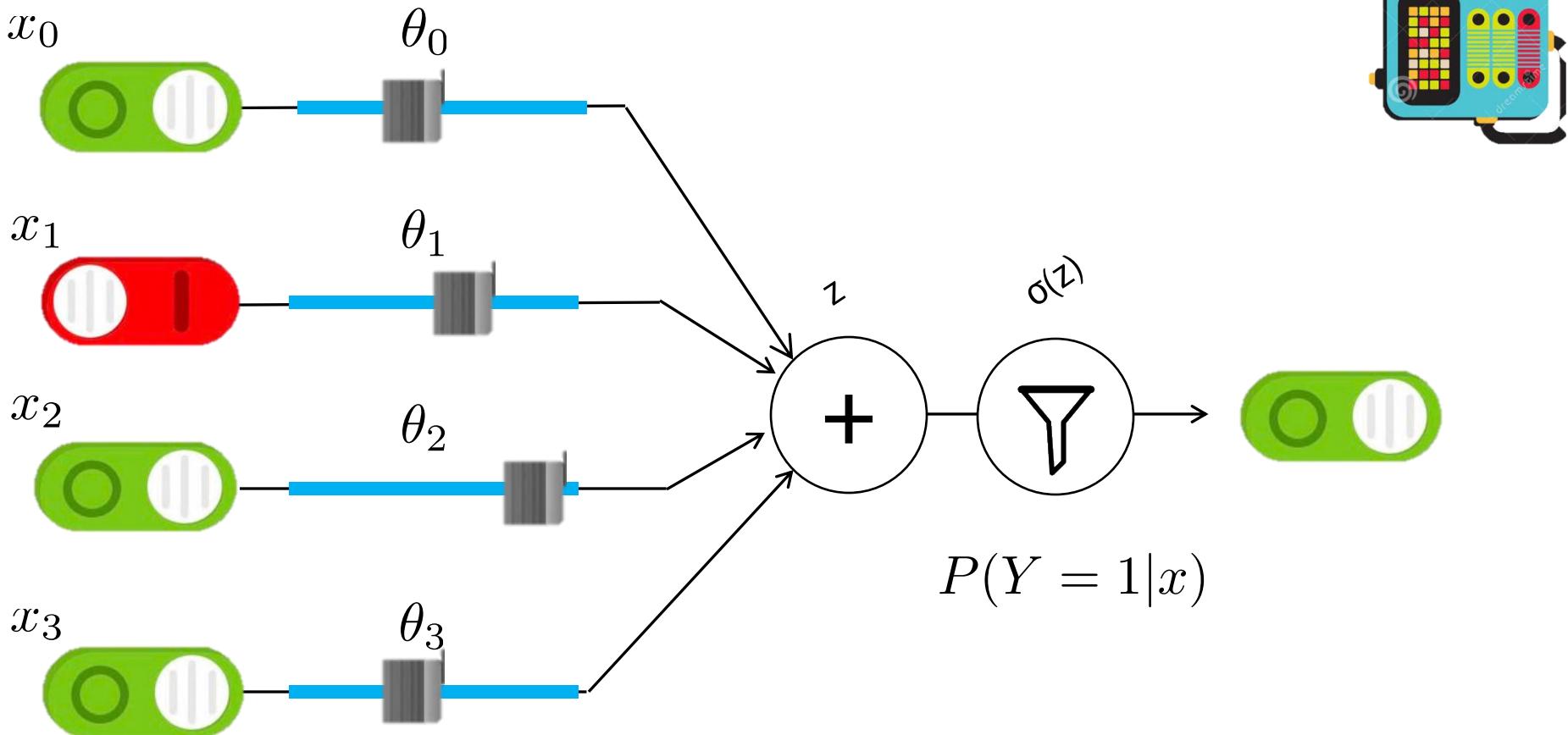
Output of an LLM is a Categorical for Next Token



Output of an LLM is a Categorical for Next Token



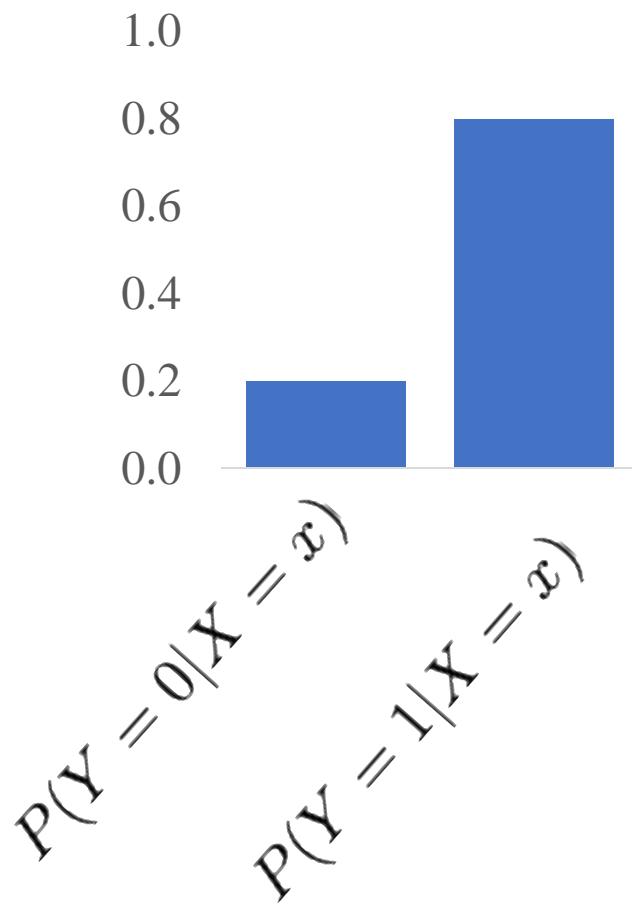
Logistic Regression to Predict a Categorical?



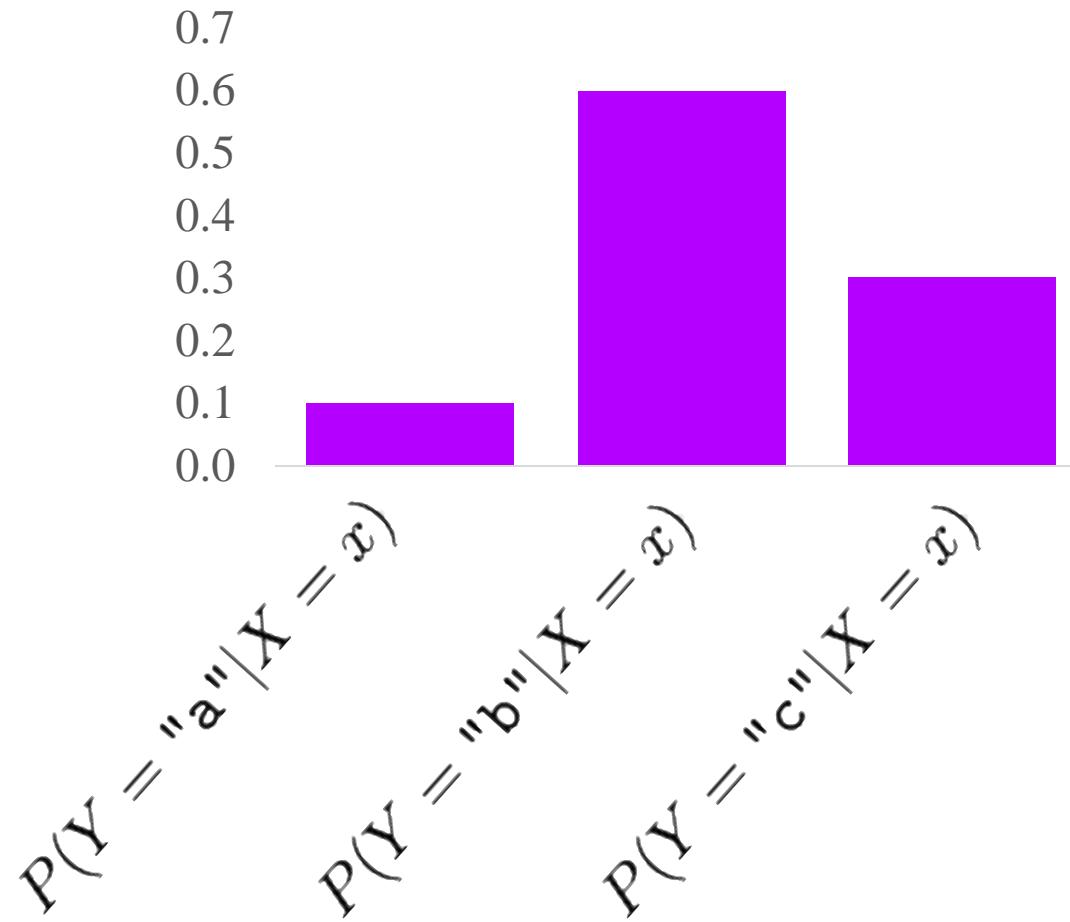
$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Logistic Regression to Predict a Categorical?

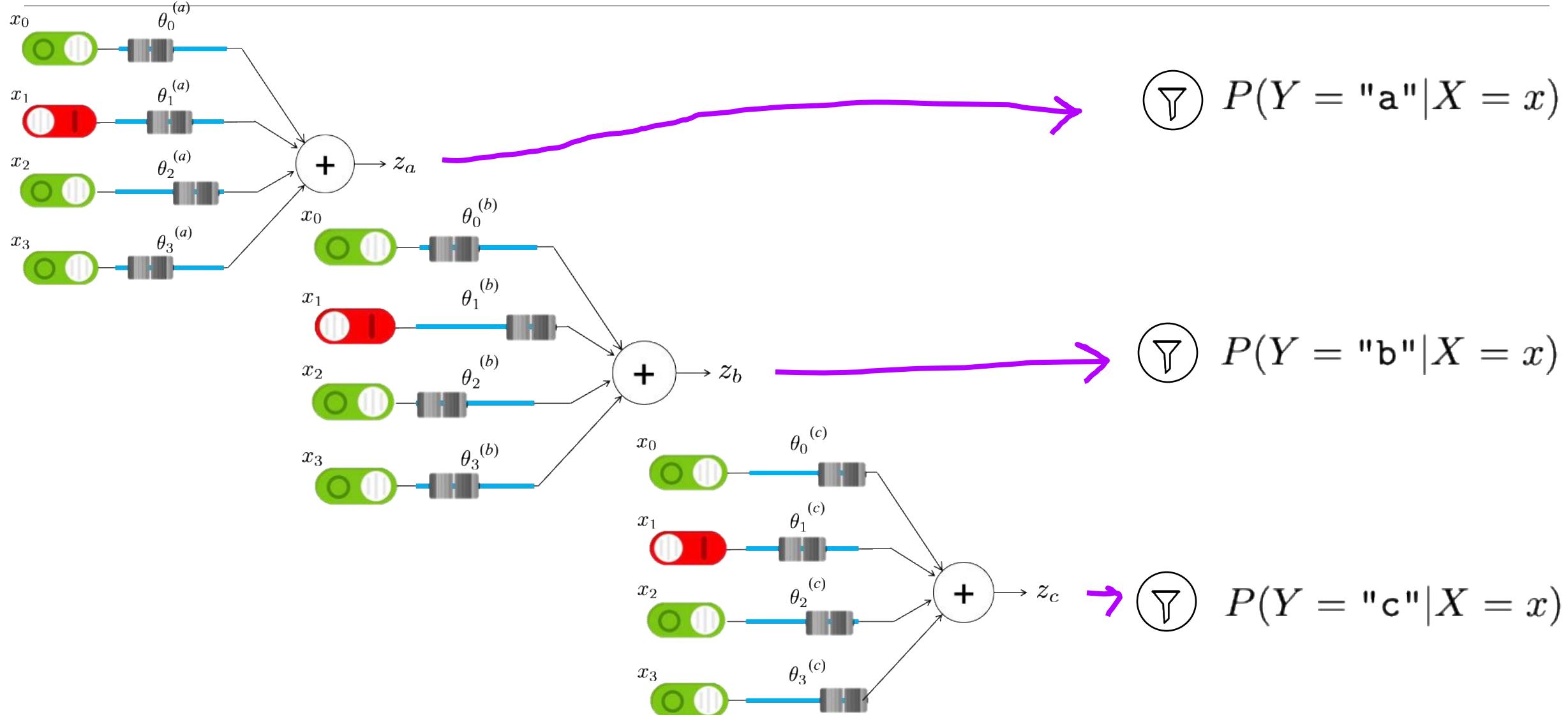
Standard Logistic Regression



Multi-Class Logistic Regression



Logistic Regression to Predict a Categorical



Categorical Classification?



Softmax is a generalization of the sigmoid function that squashes a K-dimensional vector \mathbf{z} of arbitrary real values to a K-dimensional vector $\text{softmax}(\mathbf{z})$ of real values in the range $[0, 1]$ that **add up to 1**.

$$P(Y = i | \mathbf{X} = \mathbf{x}) = \text{softmax}(\mathbf{z})_i$$

Sigmoid is to Bernoulli as Softmax is to Categorical

$$\text{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_{j=1}^n \exp(z_j)}$$

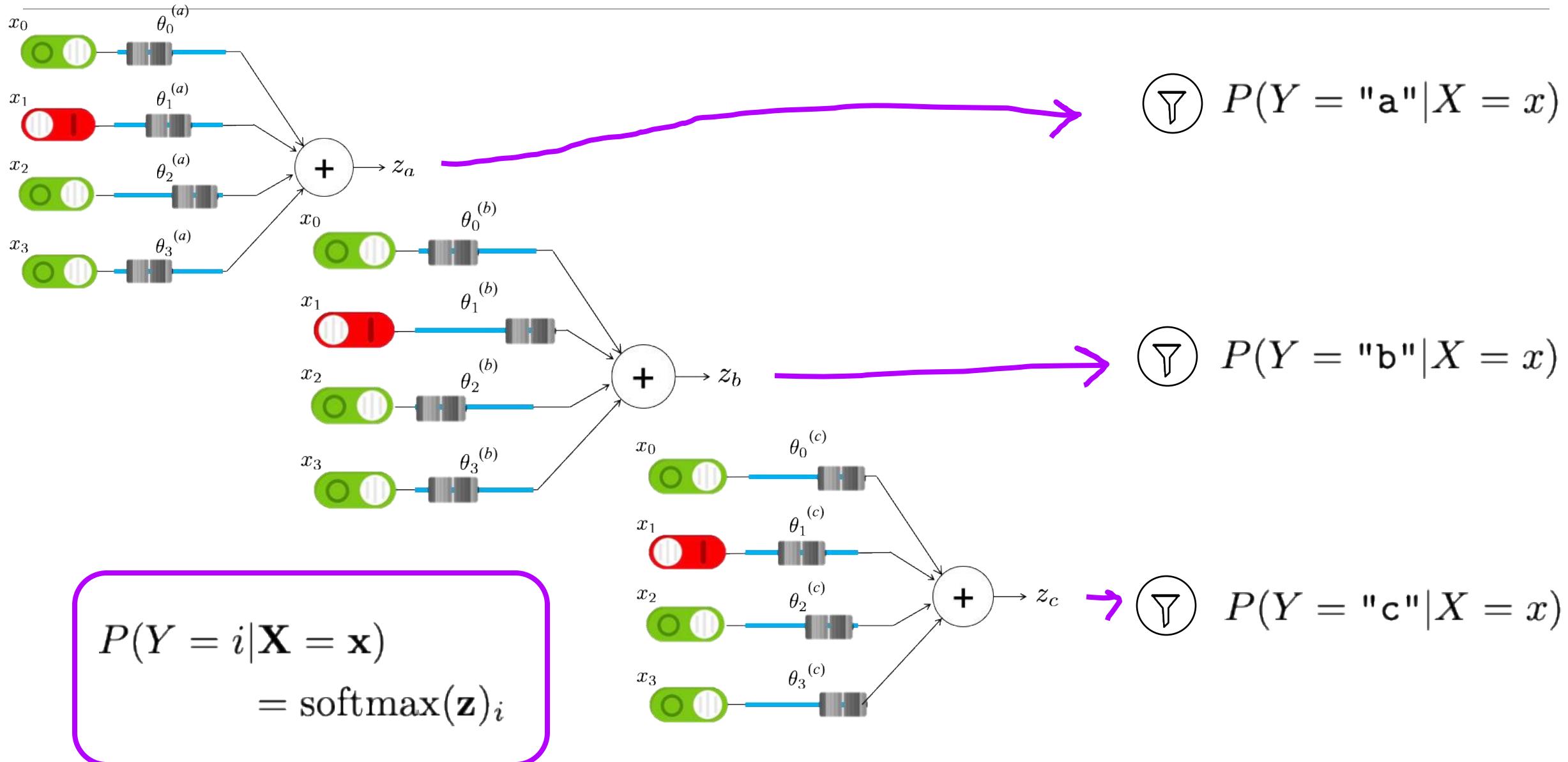
Understanding Softmax

```
9  def softmax(list_values):
10     """
11     Compute the softmax of a list of numbers.
12     >>> softmax([1, 2, 3])
13     [0.09, 0.24, 0.67]
14     >>> softmax([5, 2, 8])
15     [0.02, 0.00, 0.98]
16     """
17
18     # take the exp of each value in the list
19     raw_exp_values = []
20     for value in list_values:
21         raw_exp_values.append(math.exp(value))
22
23     # normalize the list
24     sum_exp_values = sum(raw_exp_values)
25     softmax_values = []
26     for value in raw_exp_values:
27         softmax_values.append(value / sum_exp_values)
28
29     return softmax_values
```

List: 7 7 7 7
[0.25, 0.25, 0.25, 0.25]

List: 2 8 5 6
[0.0, 0.84, 0.04, 0.11]

What is Log Likelihood?



What is Log Likelihood?

$$L(\theta) = \prod_i P(Y=y|X=x)$$

$$\left\{ \begin{array}{ll} P(Y='a'|X=x) & \text{if truth is 'a'} \\ P(Y='b'|X=x) & \text{if truth is 'b'} \\ P(Y='c'|X=x) & \text{if truth is 'c'} \end{array} \right.$$

$$LL(\theta) = \sum_i \log P(Y=y|X=x)$$

$$= \sum_i \log \text{softmax}(\mathbf{z})_y$$

Types of Machine Learning Tasks

Multi-Class
Classification

Regression

Reinforcement
Learning

Generation

Regression: Predicting Real Numbers

	Opposing team ELO	Points in last game	At Home?	Output
Game 1	84	105	1	 # Points
Game 2	90	102	0	95
		⋮		⋮
Game n	74	120	0	115

Same Notation for Training Data

Training Data: assignments all random variables \mathbf{X} and \mathbf{Y}

Assume IID data:

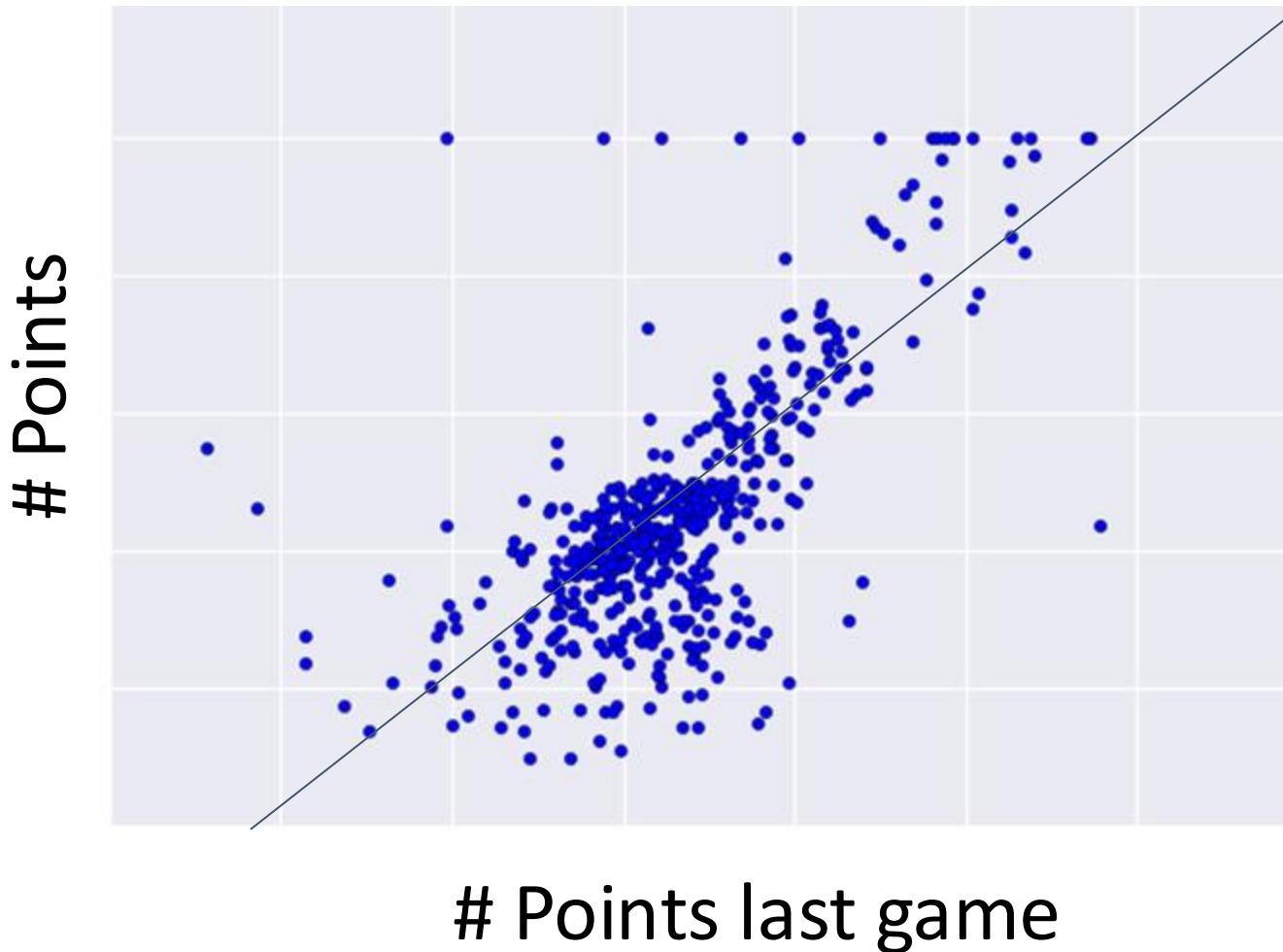
$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(n)}, y^{(n)})$$

n training datapoints

$$m = |\mathbf{x}^{(i)}|$$

Each datapoint has m features and a single output

Linear Regression with one Input



Linear Regression Model

$$\hat{y}^{(i)} = \theta^T x^{(i)}$$

output i

input i

parameters

The diagram illustrates the formula for a linear regression model. At the top, the text "output i" is written in blue, with a blue arrow pointing down to the term $\hat{y}^{(i)}$. To the right of the equals sign, the text "input i" is written in blue, with a blue arrow pointing down to the term $x^{(i)}$. Below the equals sign, the text "parameters" is written in blue, with a blue arrow pointing up to the term θ^T .

Linear Regression Model

$$y^{(i)} = \theta^T x^{(i)} + Z$$

output i

input i

parameters

random noise

The diagram illustrates the components of the linear regression model equation. The output i is labeled above $y^{(i)}$. The input i is labeled above $x^{(i)}$. A bracket under θ^T is labeled 'parameters'. A bracket under Z is labeled 'random noise'.

Linear Regression Model

$$y^{(i)} = \theta^T x^{(i)} + Z$$

output i input i

parameters random noise

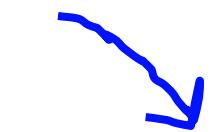
Noise is N with mean 0

$Z \sim N(0, \sigma^2)$

The diagram illustrates the components of a linear regression model. The equation $y^{(i)} = \theta^T x^{(i)} + Z$ is shown with annotations: 'output i ' points to $y^{(i)}$, 'input i ' points to $x^{(i)}$, 'parameters' points to θ^T , and 'random noise' points to Z . To the right, the text 'Noise is N with mean 0' is followed by the equation $Z \sim N(0, \sigma^2)$.

Linear Regression Model

output i



$$y^{(i)} = \theta^T x^{(i)} + Z$$

noise



noise is N with mean 0

$$Z \sim N(0, \sigma^2)$$

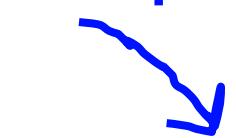
1. Linear Transform

If X is a Normal such that $X \sim N(\mu, \sigma^2)$ and Y is a linear transform of X such that $Y = aX + b$ then Y is also a Normal where:

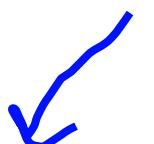
$$Y \sim N(a\mu + b, a^2\sigma^2)$$

Linear Regression Model

output i



noise



noise is N with mean 0

$$y^{(i)} = \underline{\theta^T x^{(i)}} + Z$$

$$Z \sim N(0, \sigma^2)$$

Output is normal too:

$$y^{(i)} \sim N(\underline{\theta^T x^{(i)}}, \underline{\sigma^2})$$

Log Likelihood

$$y^{(i)} = \theta^T x^{(i)} + Z \quad Z \sim N(0, \sigma^2)$$

Assume: $y^{(i)} \sim N(\theta^T x^{(i)}, \sigma^2)$

Data: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots (\mathbf{x}^{(n)}, y^{(n)})$

$$\begin{aligned} LL(\theta) &= \sum_{i=1}^n \log [f(y^{(i)})] \\ &= \sum_{i=1}^n \log \left[\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{y^{(i)} - \mu}{\sigma} \right)^2} \right] \\ &= \sum_{i=1}^n \log \left[\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{y^{(i)} - \theta^T x^{(i)}}{\sigma} \right)^2} \right] \\ &= \sum_{i=1}^n \log \left[\frac{1}{\sigma\sqrt{2\pi}} \right] - \frac{1}{2} \left(\frac{y^{(i)} - \theta^T x^{(i)}}{\sigma} \right)^2 \end{aligned}$$

Normal distribution PDF

Substitute in the mean

Apply the log

Optimization

$$y^{(i)} = \theta^T x^{(i)} + Z \quad Z \sim N(0, \sigma^2)$$

Log likelihood: $LL(\theta) = \sum_{i=1}^n \log \left[\frac{1}{\sigma\sqrt{2\pi}} \right] - \frac{1}{2} \left(\frac{y^{(i)} - \theta^T x^{(i)}}{\sigma} \right)^2$

$$\underset{\theta}{\operatorname{argmax}} LL(\theta) = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^n \log \left[\frac{1}{\sigma\sqrt{2\pi}} \right] - \frac{1}{2} \left(\frac{y^{(i)} - \theta^T x^{(i)}}{\sigma} \right)^2$$

Simplify

$$= \underset{\theta}{\operatorname{argmax}} - \sum_{i=1}^n \frac{1}{2} \left(\frac{y^{(i)} - \theta^T x^{(i)}}{\sigma} \right)^2$$

Simplify

$$= \underset{\theta}{\operatorname{argmax}} - \sum_{i=1}^n (y^{(i)} - \theta^T x^{(i)})^2$$

Hey it's the sum of squared errors!

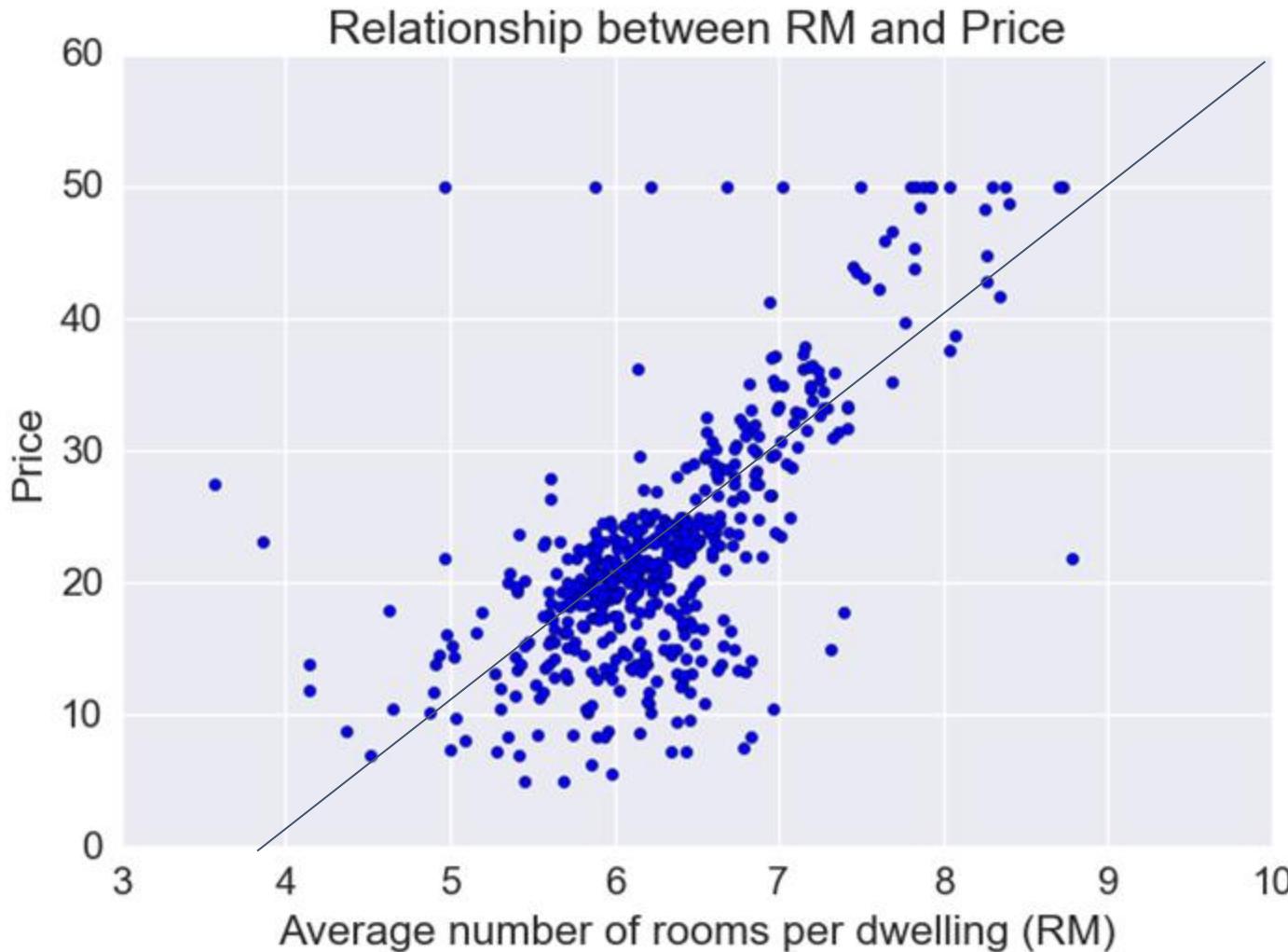
Derivative is Necessary for Gradient Ascent

$$\frac{\partial L(\theta)}{\partial \theta_j} = - \sum_{i=1}^n \frac{\partial}{\partial \theta_j} \left(y^{(i)} - \theta^T x^{(i)} \right)^2 \quad \text{Derivative of a sum}$$

$$= - \sum_{i=1}^n 2 \left(y^{(i)} - \theta^T x^{(i)} \right) (-x_j^{(i)}) \quad \text{Chain rule}$$

$$= \sum_{i=1}^n 2 \left(y^{(i)} - \theta^T x^{(i)} \right) \cdot x_j^{(i)} \quad \text{Simplify}$$

Linear Regression with one Input



Types of Machine Learning Tasks

Multi-Class
Classification

Regression

Reinforcement
Learning

Generation

Types of Machine Learning Tasks

Multi-Class
Classification

Regression

Reinforcement
Learning

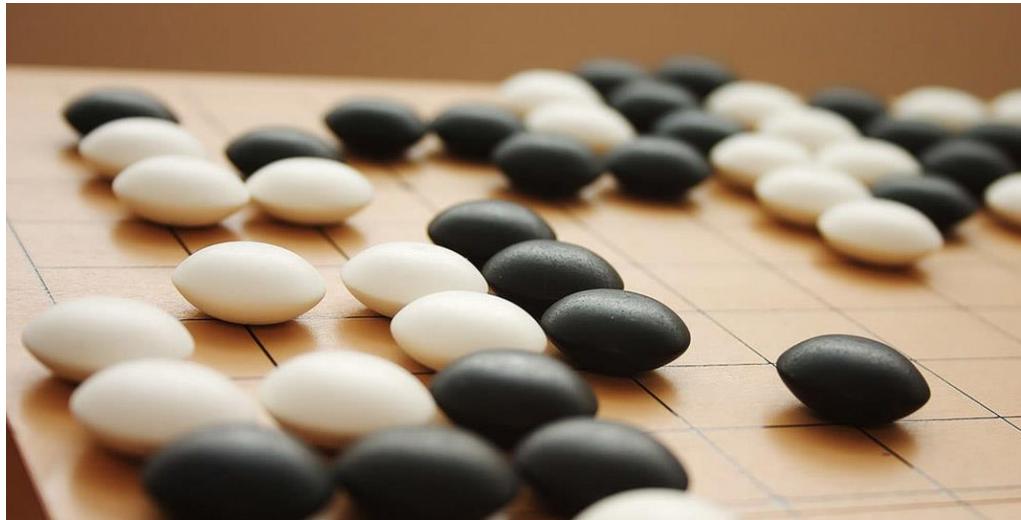
Generation

Making Decisions?

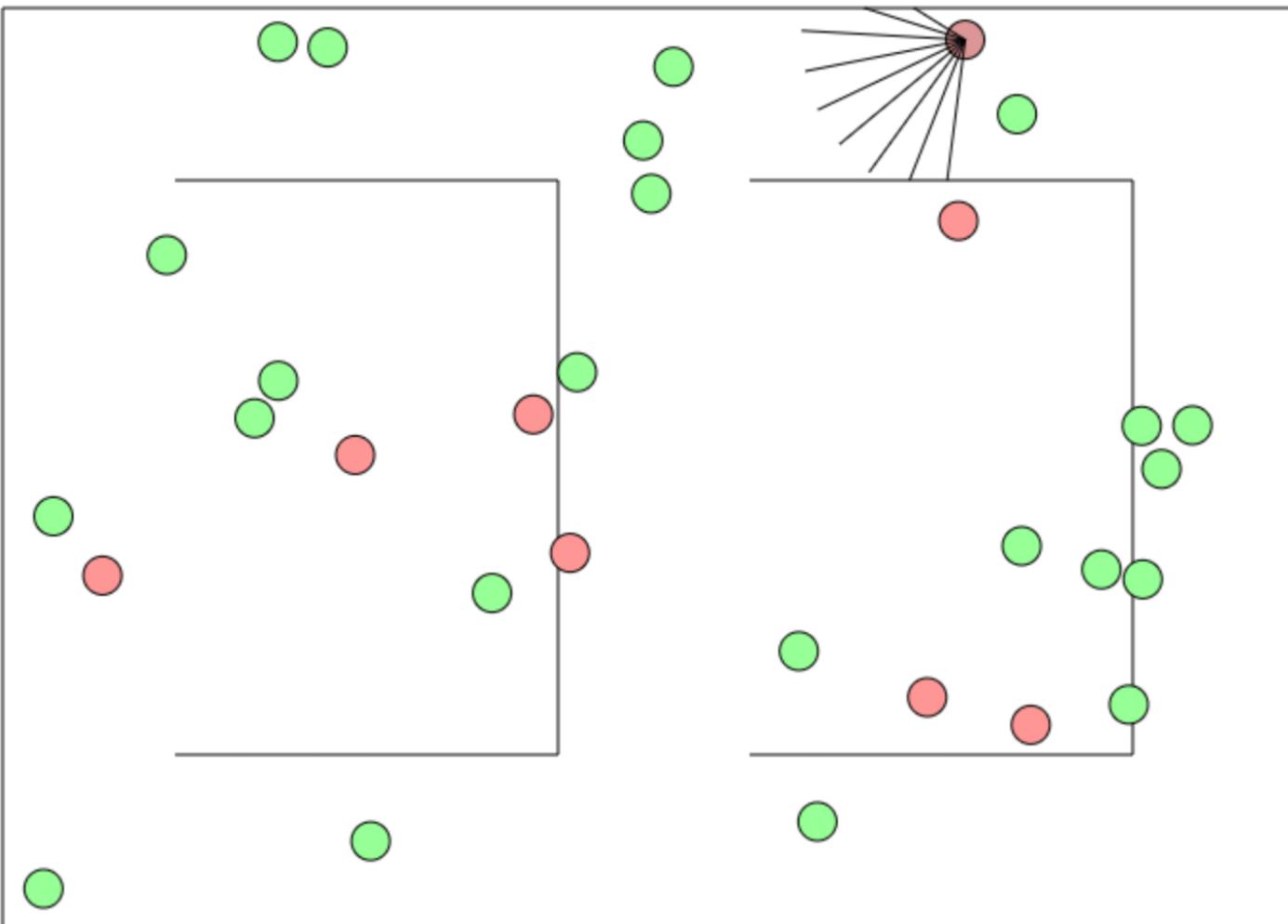


Deep Reinforcement Learning

Instead of having the output of a model be a probability you can make it an expectation.



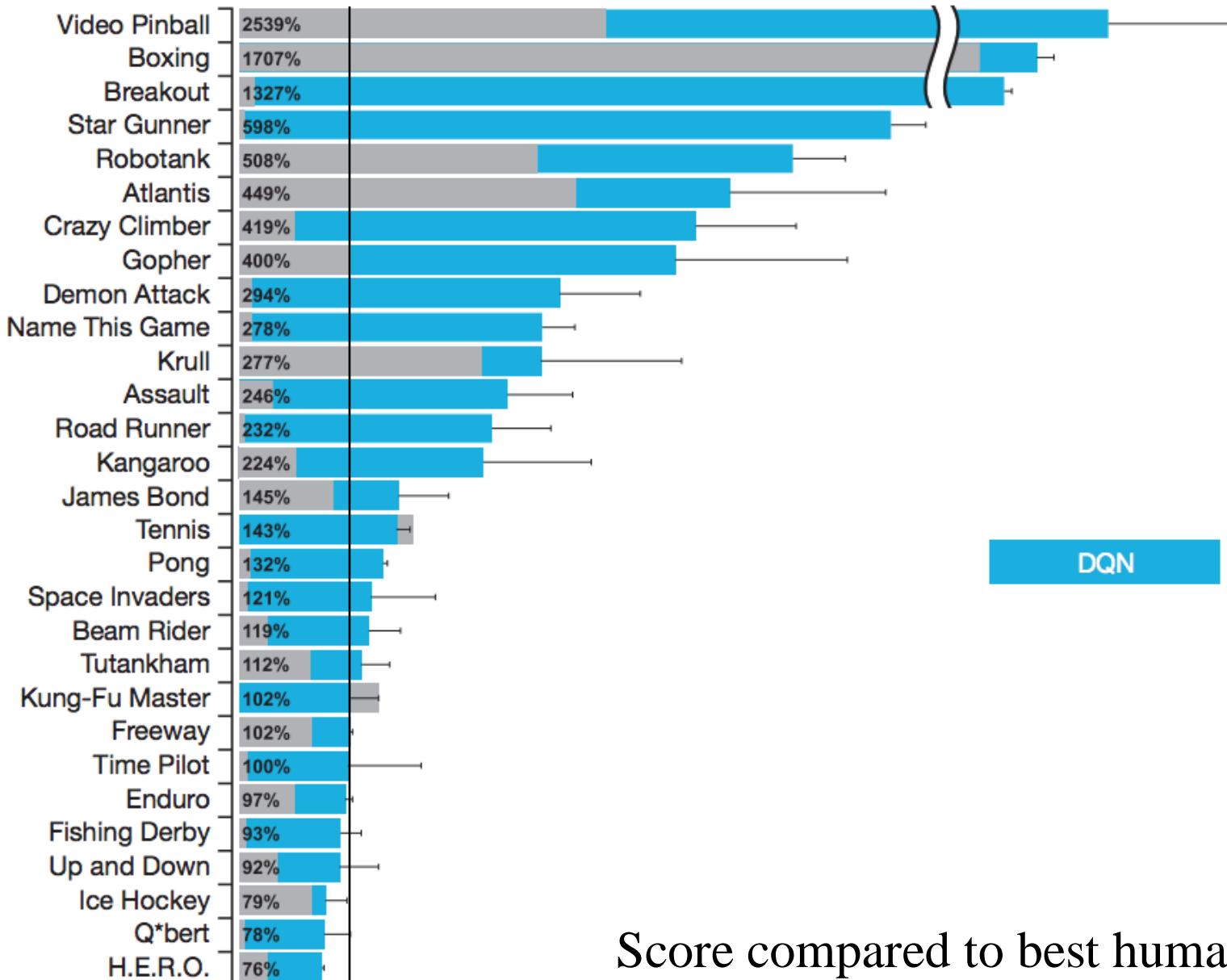
Deep Reinforcement Learning



<http://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html>

Stanford University

Deep Mind Atari Games



Score compared to best human

It is time for Deep Learning!