CSCI 567: Machine Learning

Vatsal Sharan

Fall 2022

Lecture 11, Nov 17

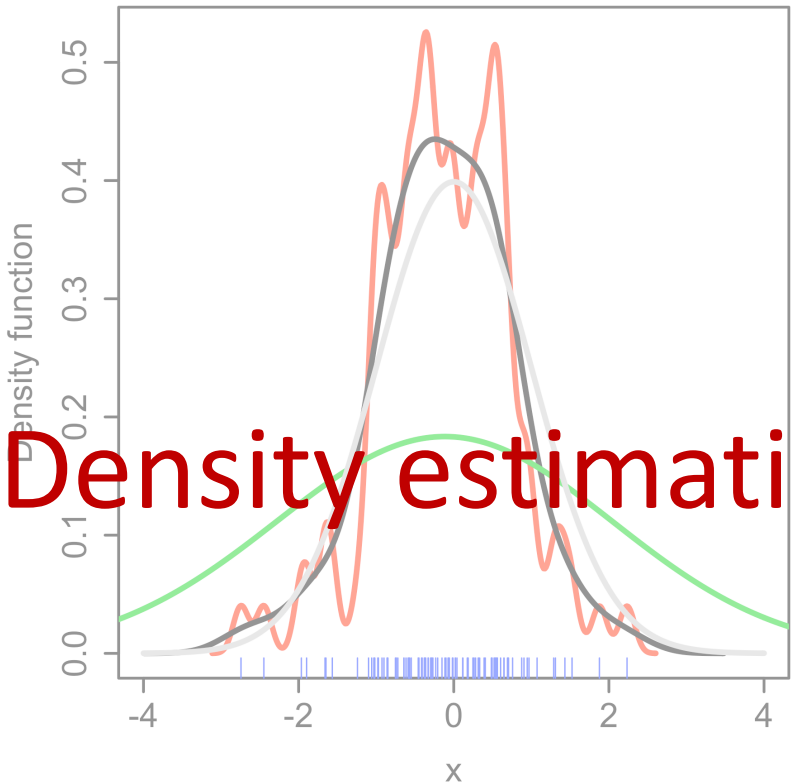
USC University of Southern California

Administrivia

- Quiz 2 is on December 1
  - More info will be posted on Ed later

- Today's plan:
  - Density estimation & Naïve Bayes
  - Multi-armed bandits
  - Responsible ML

Density estimation



Density estimation

- Introduction
- Parametric methods
- Non-parametric methods

Introduction

With clustering using GMMs, our high-level goal was the following:

Given a training set $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, **estimate a density function p that could have generated this dataset** (via $\boldsymbol{x}_i \overset{i.i.d.}{\sim} p$).

This is a special case of the general problem of *density estimation*, an important unsupervised learning problem.

Density estimation is useful for many downstream applications

- we have seen clustering already, will see more today
- these applications also *provide a way to measure quality of the density estimator*

Density estimation

- Introduction
- Parametric methods
- Non-parametric methods

Parametric methods: generative models

Parametric estimation assumes a generative model parametrized by $\boldsymbol{\theta}$:

$$p(\boldsymbol{x}) = p(\boldsymbol{x}\,;\boldsymbol{\theta})$$

Examples:

- GMM: $p(\boldsymbol{x}\,;\boldsymbol{\theta}) = \sum_{j=1}^{k} \pi_j N(\boldsymbol{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ where $\boldsymbol{\theta} = \{\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}$
- Multinomial: a discrete variable with values in $\{1, 2, \ldots, k\}$ s.t.

  $$p(x = j\,;\boldsymbol{\theta}) = \theta_j$$

  where $\boldsymbol{\theta}$ is a distribution over the k elements.

Size of $\boldsymbol{\theta}$ is independent of the size of the training set, so it's parametric.

Parametric methods: estimation

As usual, we can apply **MLE** to learn the parameters $\boldsymbol{\theta}$:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{n} \ln p(x_i \,; \boldsymbol{\theta})$$

i.i.d.

For some cases this is intractable and we can use algorithms such as EM to approximately solve the MLE problem (e.g. GMMs).

For some other cases this admits a simple closed-form solution (e.g. multinomial).

MLE for multinomials

The log-likelihood is

$$\sum_{i=1}^{n} \ln p(x = x_i \,; \boldsymbol{\theta}) = \sum_{i=1}^{n} \ln \theta_{x_i} = \sum_{i=1}^{n} \sum_{j=1}^{k} \mathbf{1}(x_i = j) \ln \theta_j$$

$$= \sum_{j=1}^{k} \sum_{i:x_i=j} \ln \theta_j = \sum_{j=1}^{k} z_j \ln \theta_j$$

where $z_j = |\{i : x_i = j\}|$ is **the number of examples with value j**.

The solution is simply

$$\theta_j = \frac{z_j}{n} \propto z_j,$$

i.e. the fraction of examples with value j. (See HW4 Q2.1)

Density estimation

- Introduction
- Parametric methods
- Non-parametric methods

Nonparametric methods

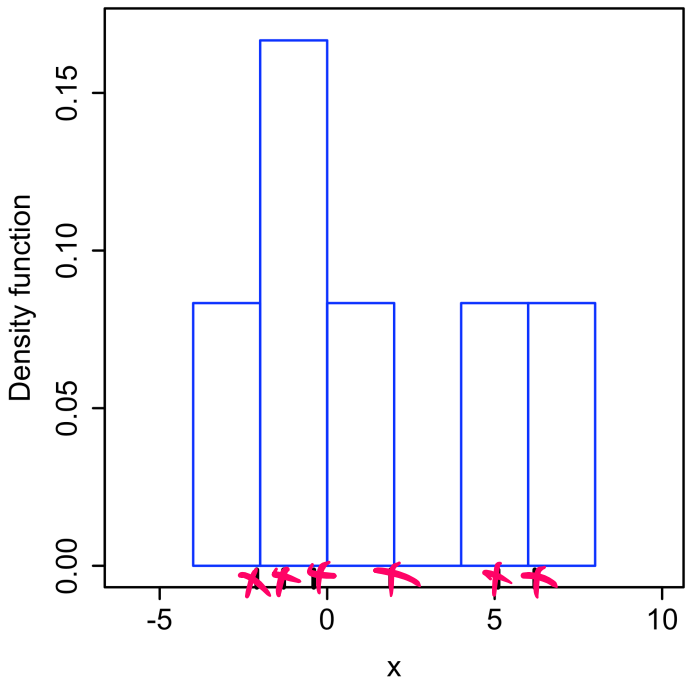
Can we estimate *without assuming a fixed generative model?*

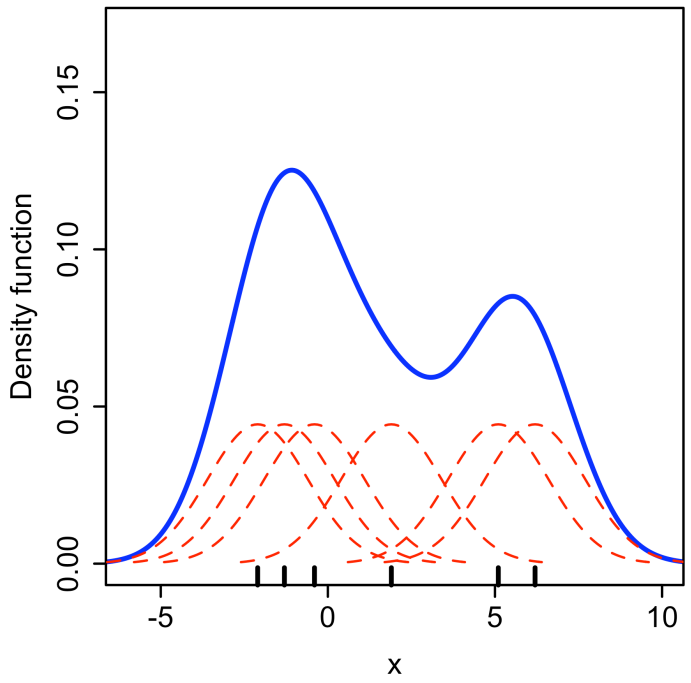
Kernel density estimation (KDE) provides a solution.

- the approach is nonparametric: it keeps the entire training set
- we focus on the one-dimensional (continuous) case

High-level idea

- Construct something similar to a histogram:
- For each data point, create a “bump” (via a Kernel)
- Sum up or average all the bumps





Kernel

KDE with **a kernel** K: $\mathbb{R} \to \mathbb{R}$:

adding bump around x_i

$$p(x) = \frac{1}{n} \sum_{i=1}^{n} K(x - x_i)$$

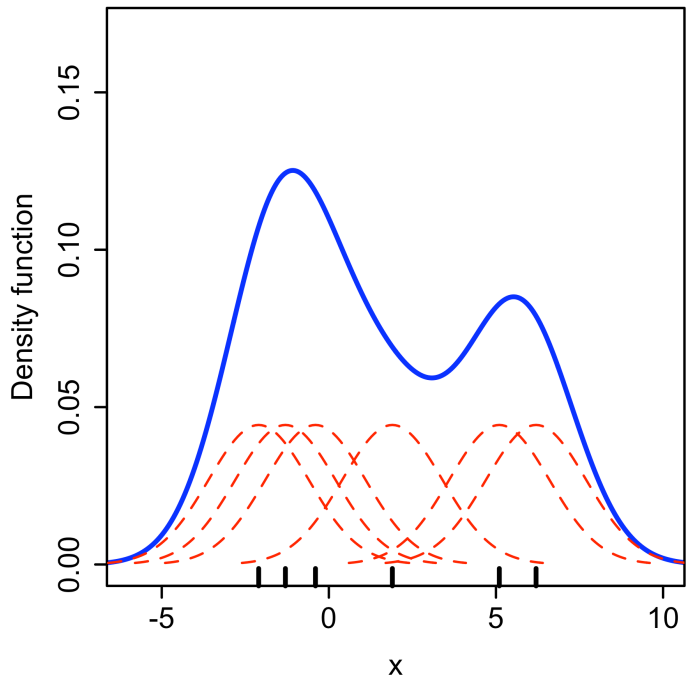
need to loop around all datapoints

shape of bump

e.g. $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$, the standard Gaussian density

Kernel needs to satisfy:

- symmetry: $K(u) = K(-u)$
- $\int_{-\infty}^{\infty} K(u)du = 1$, makes sure p is a density function.

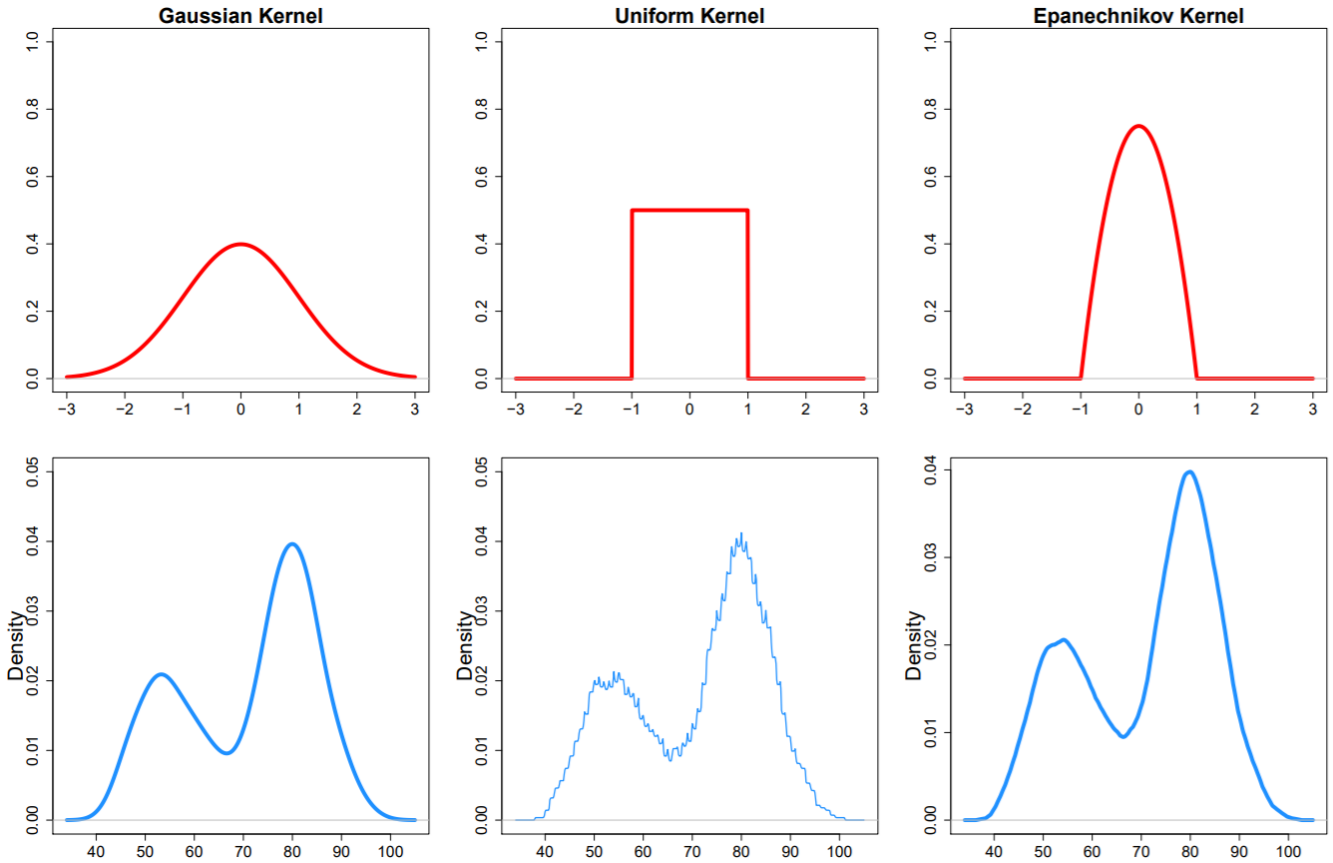


Different kernels

$$\frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$$

$$\frac{1}{2}\mathbb{I}[|u| \leq 1]$$

$$\frac{3}{4}\max\{1-u^2, 0\}$$



Bandwidth

If $K(u)$ is a kernel, then for any $h > 0$

$$K_h(u) := \frac{1}{h} K\left(\frac{u}{h}\right)$$

(stretching the kernel)

can be used as a kernel too (verify the two properties yourself)

$$\int_{-\infty}^{\infty} \frac{1}{h} K\left(\frac{u}{h}\right) du = 1$$

So general KDE is determined by both the kernel K and the bandwidth h

$$p(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

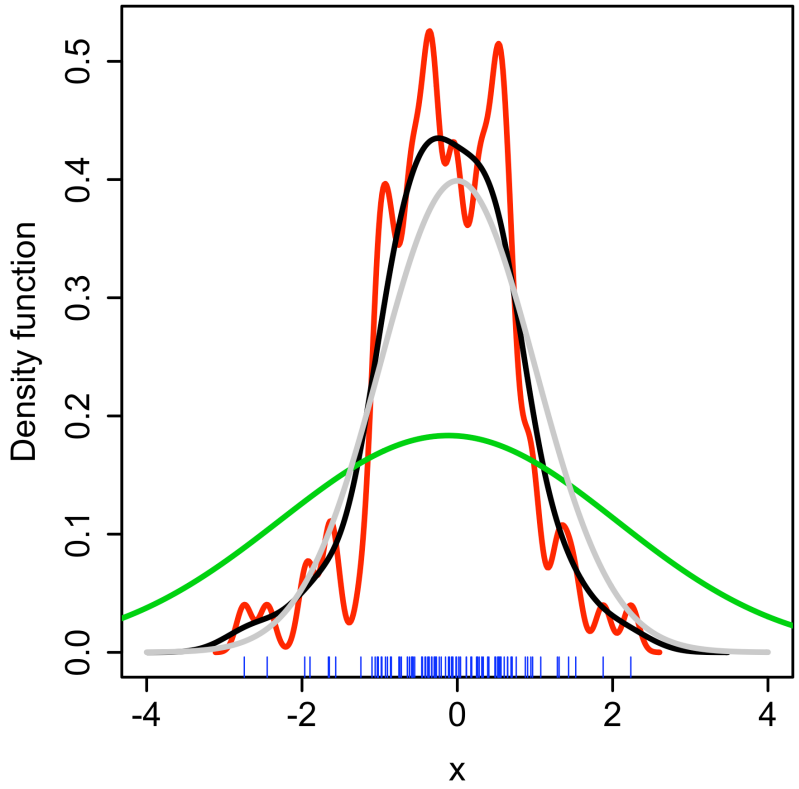
- x_i controls the center of each bump
- h controls the width/variance of the bumps

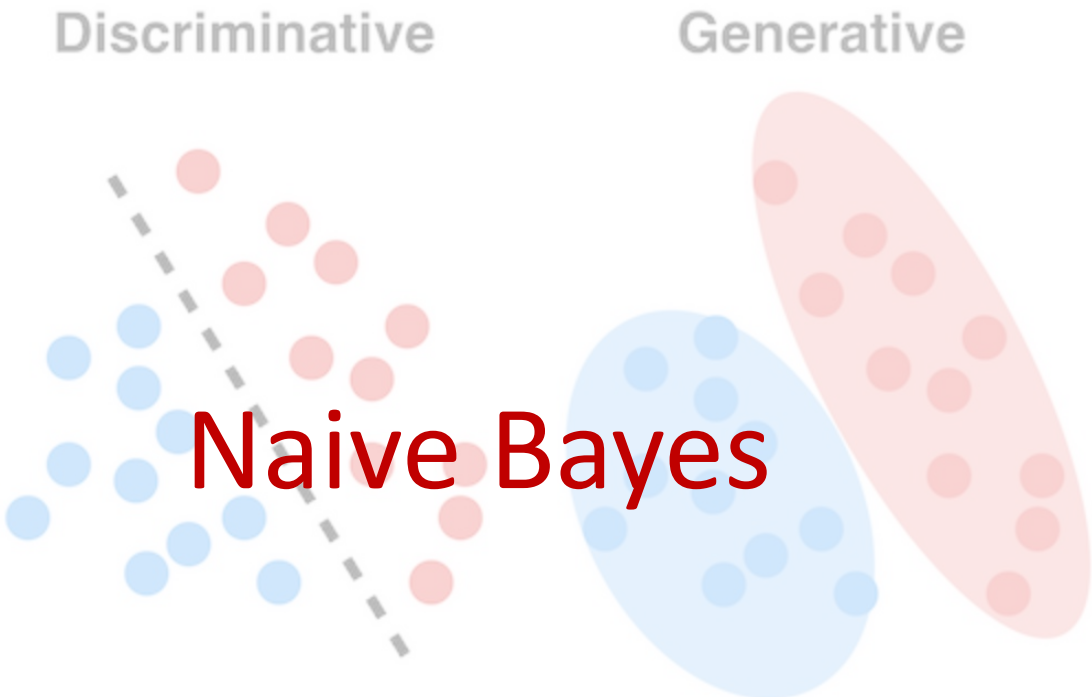
Bandwidth

Larger h means larger variance and smoother density

Gray curve is ground-truth

- Red: $h = 0.05$
- Black: $h = 0.337$
- Green: $h = 2$

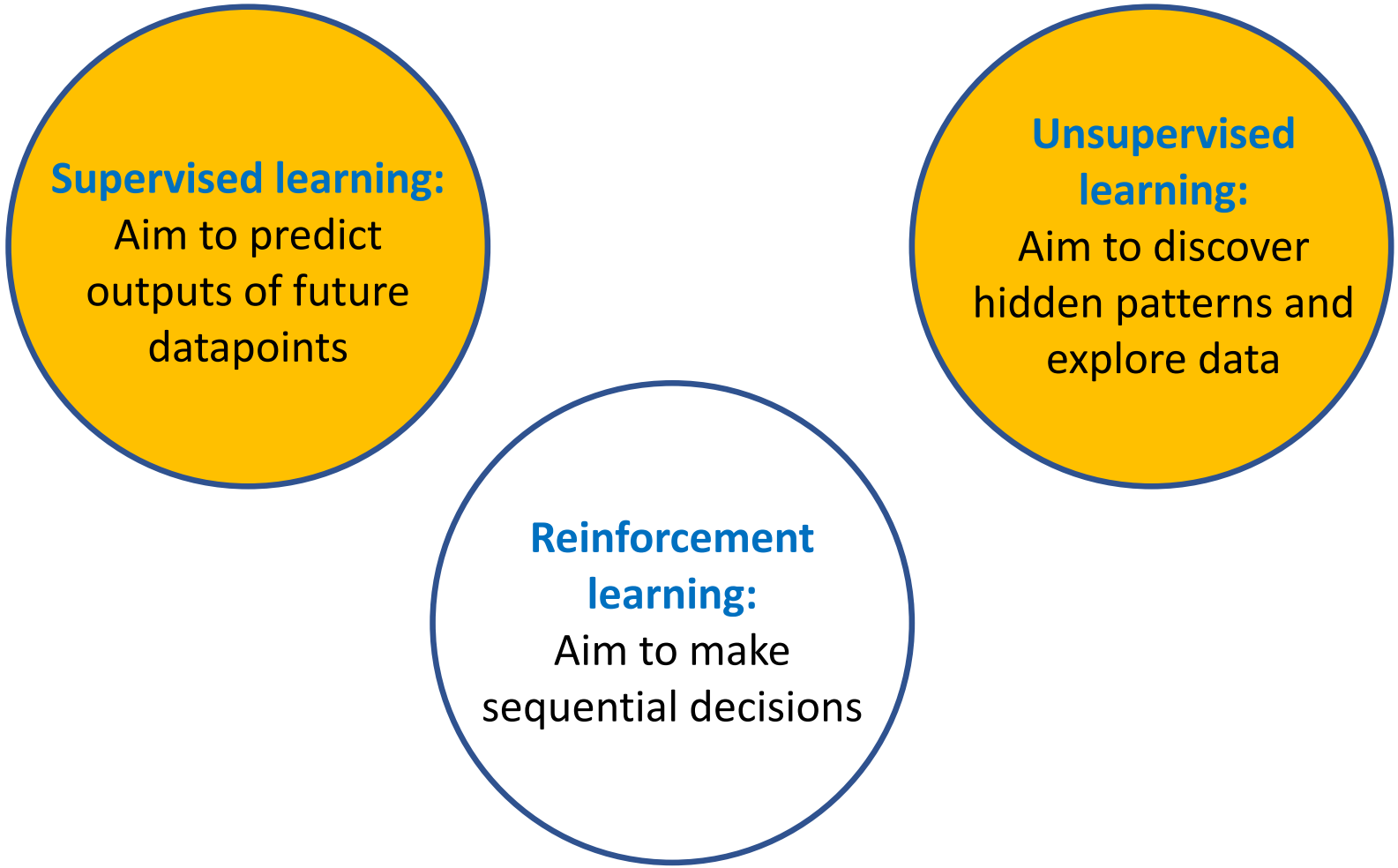




Naive Bayes

Image source

A simplistic taxonomy of ML



Naïve Bayes

- Motivation & setup
- Prediction with Naïve Bayes, and some connections

Bayes optimal classifier

Suppose $(\boldsymbol{x}, y)$ is drawn from a joint distribution p. The **Bayes optimal classifier** is

$$f^*(\boldsymbol{x}) = \operatorname*{argmax}_{c \in [\mathsf{C}]} p(c \mid \boldsymbol{x})$$

i.e. predict the class with the largest conditional probability.

p is of course unknown, but we can estimate it, which is *exactly a density estimation problem!*

Estimation

How to estimate a joint distribution? Observe we always have

$$p(\boldsymbol{x}, y) = p(y)p(\boldsymbol{x} \mid y)$$

C possible values

We know how to estimate $p(y)$ by now.

To estimate $p(\boldsymbol{x} \mid y = c)$ for some $c \in [\mathsf{C}]$, we are doing density estimation using data $\{\boldsymbol{x}_i : y_i = c\}$.

This is *not a 1D problem* in general.

A naïve assumption

Naive Bayes assumption: **conditioning on a label, features are independent**, which means

$(x_1, \ldots x_d)$ x_j is the jth coolinate of x

$$p(\boldsymbol{x} \mid y = c) = \prod_{j=1}^{d} p(x_j \mid y = c)$$

Now for each j and c we have a simple 1D density estimation problem!

Is this a reasonable assumption? Sometimes yes, e.g.

- use $\boldsymbol{x} = (\text{Height, Vocabulary})$ to predict $y = \text{Age}$
- Height and Vocabulary are dependent
- but conditioned on Age, they are independent!

More often this assumption is *unrealistic and "naive"*, but still Naive Bayes can work very well even if the assumption is wrong.

Example: Discrete features

Height: $\leq$3', 3'-4', 4'-5', 5'-6', $\geq$6' → 5 possible values

Vocabulary: $\leq$5K, 5K-10K, 10K-15K, 15K-20K, $\geq$20K → 5 values

Age: $\leq$5, 5-10, 10-15, 15-20, 20-25, $\geq$25 → 6 possible values

MLE estimation: e.g.

$$p(\text{Age} = \text{10-15}) = \frac{\#\text{examples with age 10-15}}{\#\text{examples}}$$

$$p(\text{Height} = \text{5'-6'} \mid \text{Age} = \text{10-15}) = \frac{\#\text{examples with height 5'-6' and age 10-15}}{\#\text{examples with age 10-15}}$$

→ conditional: joint / marginal

Discrete features: More formally

For a label $c \in [\mathsf{C}]$,

$$p(y = c) = \frac{|\{i : y_i = c\}|}{n}$$

For each possible value ℓ of a discrete feature j,

$$p(x_j = \ell \mid y = c) = \frac{|\{i : x_{i,j} = \ell, y_i = c\}|}{|\{i : y_i = c\}|}$$

jth feature of ith datapoint

density for the jth feature

Continuous features

If the feature is continuous, we can do

- parametric estimation, e.g. via a Gaussian

$$p(x_j = x \mid y = c) = \frac{1}{\sqrt{2\pi}\sigma_{c,j}} \exp\left(-\frac{(x - \mu_{c,j})^2}{2\sigma_{c,j}^2}\right)$$

  where $\mu_{c,j}$ and $\sigma_{c,j}^2$ are the empirical mean and variance of feature j among all examples with label c.

- or nonparametric estimation, e.g. via a Kernel K and bandwidth h:

$$p(x_j = x \mid y = c) = \frac{1}{|\{i : y_i = c\}|} \sum_{i:y_i=c} K_h(x - x_{i,j})$$

Naïve Bayes

- Motivation & setup
- Prediction with Naïve Bayes, and some connections

Naïve Bayes: Prediction

After learning the model

$$p(\boldsymbol{x}, y) = p(y) \prod_{j=1}^{d} p(x_j \mid y)$$

the **prediction** for a new example $\boldsymbol{x}$ is

$$\begin{aligned}
\underset{c \in [C]}{\operatorname{argmax}}\ p(y = c \mid \boldsymbol{x}) &= \underset{c \in [C]}{\operatorname{argmax}}\ p(\boldsymbol{x}, y = c) \\
&= \underset{c \in [C]}{\operatorname{argmax}} \left(p(y = c) \prod_{j=1}^{d} p(x_j \mid y = c) \right) \\
&= \underset{c \in [C]}{\operatorname{argmax}} \left(\ln p(y = c) + \sum_{j=1}^{d} \ln p(x_j \mid y = c) \right) .
\end{aligned}$$

$p(x) \cdot p(y=c \mid x)$

$= p(y=c) \cdot p(x \mid y=c)$

Naive Bayes assumption

prior

likelihood

Example: Discrete features

For **discrete features**, plugging in previous MLE estimation gives

$$\begin{aligned}
&\operatorname*{argmax}_{c \in [\mathsf{C}]} \; p(y = c \mid \boldsymbol{x}) \\
&= \operatorname*{argmax}_{c \in [\mathsf{C}]} \left(\ln p(y = c) + \sum_{j=1}^{d} \ln p(x_j \mid y = c) \right) \\
&= \operatorname*{argmax}_{c \in [\mathsf{C}]} \left(\ln |\{i : y_i = c\}| + \sum_{j=1}^{d} \ln \frac{|\{i : x_{i,j} = x_j, y_i = c\}|}{|\{i : y_i = c\}|} \right)
\end{aligned}$$

What is Naïve Bayes learning?

Observe again for the case of continuous features with a Gaussian model, if we **fix the variance for each feature to be σ** (i.e. not a parameter of the model any more), then the prediction becomes

$$\begin{aligned}
&\operatorname*{argmax}_{c\in[\mathsf{C}]} p(y=c \mid \boldsymbol{x}) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]} \left(\ln |\{i : y_i = c\}| - \sum_{j=1}^{d} \left(\ln \sigma + \frac{(x_j - \mu_{c,j})^2}{2\sigma^2} \right) \right) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]} \left(\ln |\{i : y_i = c\}| - \sum_{j=1}^{d} \frac{\mu_{c,j}^2}{2\sigma^2} + \sum_{j=1}^{d} \frac{\mu_{c,j}}{\sigma^2} x_j \right) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]} \left(w_{c0} + \sum_{j=1}^{d} w_{cd} x_j \right) = \operatorname*{argmax}_{c\in[\mathsf{C}]} \boldsymbol{w}_c^{\mathsf{T}} \boldsymbol{x}
\end{aligned}$$

(x_j^2 is constant across classes)

(*linear classifier!*)

where we denote $w_{c0} = \ln |\{i : y_i = c\}| - \sum_{j=1}^{d} \frac{\mu_{c,j}^2}{2\sigma^2}$ and $w_{cd} = \frac{\mu_{c,j}}{\sigma^2}$.

Connection to logistic regression

Moreover, by a similar calculation you can verify

$$p(y = c \mid \boldsymbol{x}) \propto e^{\boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{x}}$$

This the **softmax** function, *the same model we used for the probabilistic interpretation of logistic regression!*

So what is different then? They **learn the parameters in different ways**:

- both via MLE, one on $p(y = c \mid \boldsymbol{x})$, the other on $p(\boldsymbol{x}, y)$
- solutions are different: logistic regression has no closed-form, naive Bayes admits a simple closed-form

we said $p(y=c|x)$ for $c=\{\pm 1\}$
$= \sigma(y w^T x)$

Connection to logistic regression

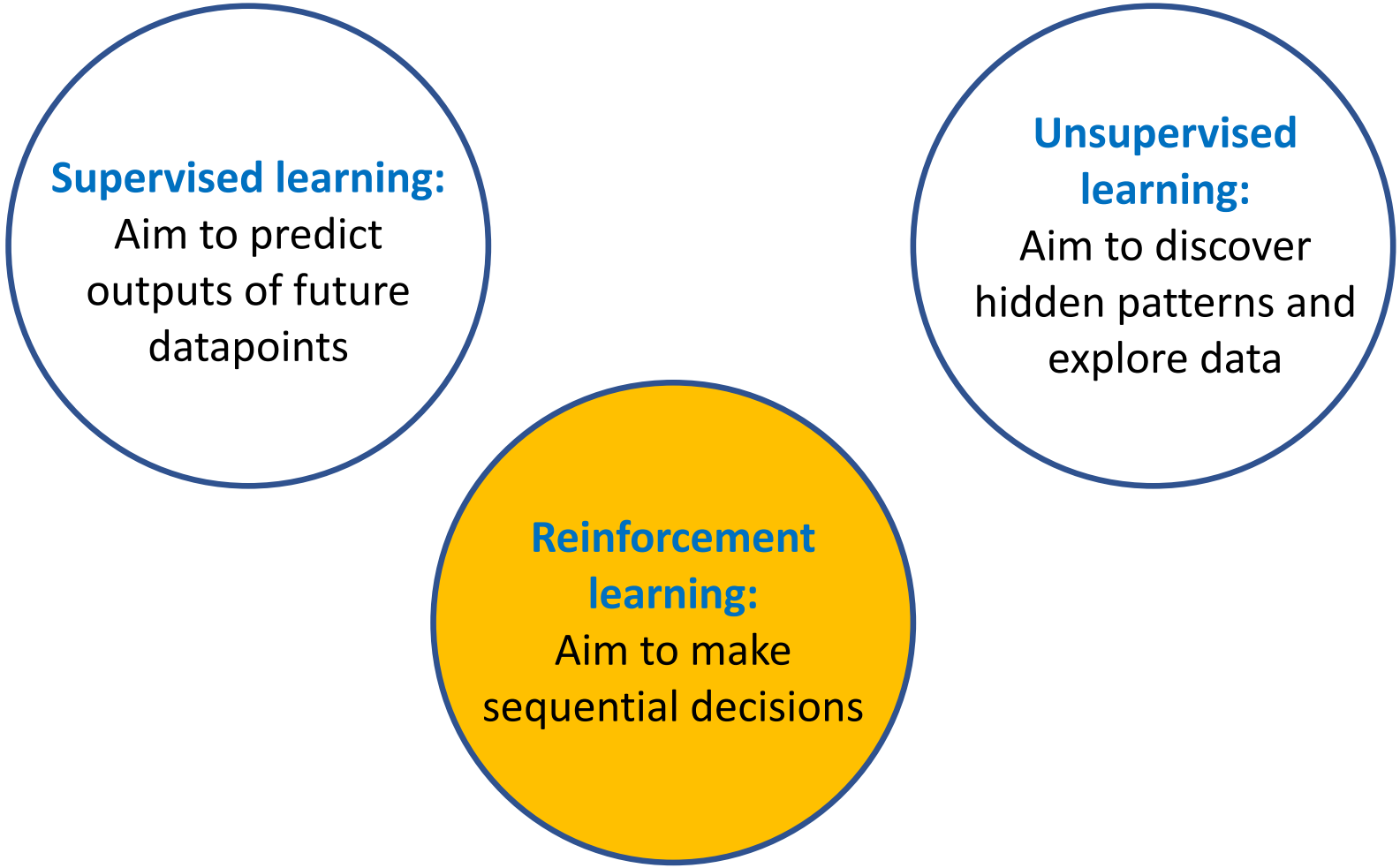
	Logistic regression	Naive Bayes
Model	conditional $p(y \mid \boldsymbol{x})$	joint $p(\boldsymbol{x}, y)$
Learning	MLE (can also be viewed as minimizing logistic loss)	MLE
Accuracy	usually better for large n	usually better for small n

Discriminative model

Generative model

Multiarmed bandits

A simplistic taxonomy of ML



Multi-armed bandits

- Motivation & setup
- Exploration vs. Exploitation

Decision making

Problems we have discussed so far:

- start with a fixed training dataset
- learn a predictor from the data or discover some patterns in the data

But many real-life problems are about **learning continuously**:

- make a prediction/decision
- receive some feedback
- repeat

Broadly, these are called **online decision making problems**.

Examples

Amazon/Netflix/MSN **recommendation systems**:

- a user visits the website
- the system recommends some products/movies/news stories
- the system observes whether the user clicks on the recommendation

Playing games (Go/Atari/StarCraft/...) or **controlling robots**:

- make a move
- receive some reward (e.g. score a point) or loss (e.g. fall down)
- make another move...

Multiarmed bandits: Motivation

Imagine going to a casino to play a slot machine

- it robs you, like a "bandit" with a single arm

Of course there are many slot machines in the casino

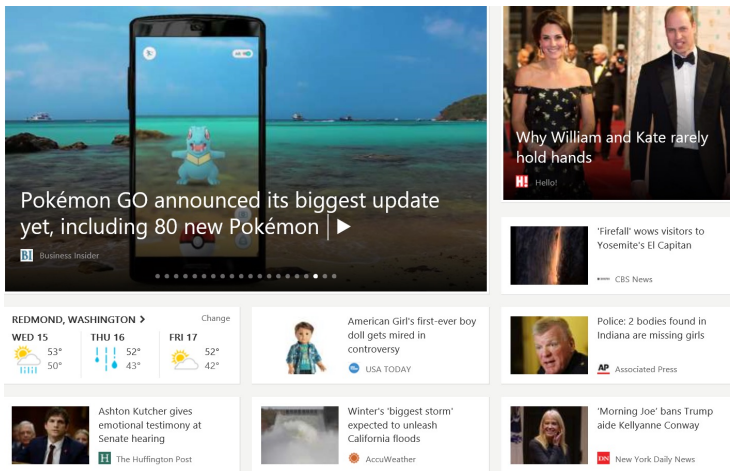
- like a bandit with multiple arms (hence the name)
- **if I can play 10 times, which machines should I play?**

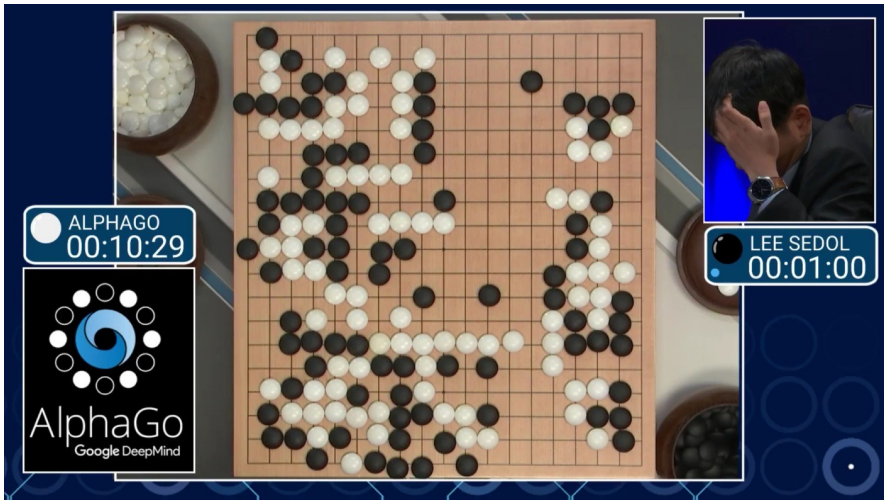
Applications

This simple model and its variants capture many real-life applications:

- recommendation systems, each product/movie/news story is an arm
  (Microsoft MSN employs a variant of bandit algorithm)

- game playing, each possible move is an arm
  (AlphaGo has a bandit algorithm as one of the components)





Applications

From a thesis defense yesterday…

BanditPAM: Almost Linear Time k-Medoids Clustering via Multi-Armed Bandits

Mo Tiwari
Department of Computer Science
Stanford University
motiwari@stanford.edu

MABSplit: Faster Forest Training Using Multi-Armed Bandits

Mo Tiwari[1] **Ryan Kang**[1,*] **Je-Yong Lee**[2,*] **Sebastian Thrun**[1]

Chris Piech[1] **Ilan Shomorony**[#,3] **Martin Jinye Zhang**[#,4]

Formal setup

There are K **arms** (actions/choices/...)

The problem proceeds in rounds between the environment and a learner: for each time $t = 1, \ldots, T$

- the environment decides the reward for each arm $r_{t,1}, \ldots, r_{t,K}$
- the learner picks an arm $a_t \in [K]$
- the learner observes the reward for arm a_t, i.e., r_{t,a_t}

↑ reward for each arm at time t.

Importantly, *learner does not observe rewards for arms not selected!*

This kind of limited feedback is usually referred to as bandit feedback

Evaluating performance

What should be the goal here?

Maximizing total rewards $\sum_{t=1}^{T} r_{t,a_t}$ seems natural.

But the absolute value of rewards is not meaningful, instead we should compare it to some *benchmark*. A classic benchmark is

$$\max_{a \in [K]} \sum_{t=1}^{T} r_{t,a}$$

i.e. the largest reward one can achieve by always playing a fixed arm

So we want to minimize

$$\max_{a \in [K]} \sum_{t=1}^{T} r_{t,a} - \sum_{t=1}^{T} r_{t,a_t}$$

This is called the **regret**: *how much I regret not sticking with the best fixed arm in hindsight?*

Environments

How are the rewards generated by the environments?

- they could be generated via some fixed distribution
- they could be generated via some changing distribution
- they could be generated even completely arbitrarily/adversarially

We focus on a simple setting:

- rewards of arm a are i.i.d. samples of Ber(μ_a), that is, $r_{t,a}$ is 1 with prob. μ_a, and 0 with prob. $1-\mu_a$, independent of anything else.
- each arm has a different mean $(\mu_1, \ldots, \mu_K)$; the problem is essentially about finding the best arm $\text{argmax}_a \mu_a$ as quickly as possible

Empirical means

Let $\hat{\mu}_{t,a}$ be the **empirical mean** of arm a up to time t:

$$\hat{\mu}_{t,a} = \frac{1}{n_{t,a}} \sum_{\tau \leq t : a_\tau = a} r_{\tau,a}$$

where

$$n_{t,a} = \sum_{\tau \leq t} \mathbb{I}[a_\tau == a]$$

is the **number of times** we have picked arm a.

Concentration: $\hat{\mu}_{t,a}$ should be close to μ_a if $n_{t,a}$ is large

Multi-armed bandits

- Motivation & setup
- Exploration vs. Exploitation

Exploitation only

Greedy:

Pick each arm once for the first K rounds.

For $t = K+1, \ldots, T$, pick $a_t = \text{argmax}_a \ \hat{\mu}_{t-1,a}$.

What's wrong with this greedy algorithm?

Consider the following example:

- $K = 2, \mu_1 = 0.6, \mu_2 = 0.5$ (so arm 1 is the best)
- suppose the algorithm first picks arm 1 and sees reward 0, then picks arm 2 and sees reward 1 (this happens with decent probability) $(0.4 \times 0.5 = 0.2)$
- the algorithm will never pick arm 1 again!

The key challenge

All bandit problems face the same **dilemma**:

Exploitation vs. Exploration trade-off

- on one hand we want to exploit the arms that we think are good
- on the other hand we need to explore all arms often enough in order to figure out which one is better
- so each time we need to ask: *do I explore or exploit? and how?*

We next discuss **three ways** to trade off exploration and exploitation for our simple multi-armed bandit setting.

A natural first attempt

Explore–then–Exploit:

Input: a parameter $T_0 \in [T]$

Exploration phase: for the first T_0 rounds, pick each arm for T_0/K times

Exploitation phase: for the remaining $T - T_0$ rounds, stick with the empirically best arm $\operatorname{argmax}_a \ \hat{\mu}_{T_0,a}$

Parameter T_0 clearly controls the exploration/exploitation trade-off

Explore-then-Exploit: Issues

It's pretty reasonable, but the disadvantages are also clear:

- not clear how to tune the hyperparameter T_0
- in the exploration phase, even if an arm is clearly worse than others based on a few pulls, it's still pulled T_0/K times
- clearly it won't work if the environment is changing

A slightly better algorithm

ϵ-Greedy Pick each arm once for the first K rounds.

For $t = K+1, \ldots, T$,

- with probability ϵ, explore: pick an arm uniformly at random
- with probability $1-\epsilon$, exploit: pick $a_t = \text{argmax}_a \ \hat{\mu}_{t-1,a}$

Pros

- always exploring and exploiting
- applicable to many other problems
- first thing to try usually

Cons

- need to tune ϵ
- same uniform exploration

Is there a more adaptive way to explore?

More adaptive exploration

A simple modification of "Greedy" leads to the well-known:

Upper Confidence Bound (UCB) algorithm

For $t = 1, \ldots, T$, pick $a_t = \text{argmax}_a \ \text{UCB}_{t,a}$ where

$$\text{UCB}_{t,a} := \hat{\mu}_{t-1,a} + 2\sqrt{\frac{\ln t}{n_{t-1,a}}}$$

initialize all estimates to be same

if $n_{t-1,a}$ is small, $\text{UCB}_{t,a}$ will be large

- the first term in $\text{UCB}_{t,a}$ represents exploitation, while the second (bonus) term represents exploration
- the bonus term is large if the arm is not pulled often enough, which encourages exploration (adaptive due to the first term)
- a parameter-free algorithm, and *it enjoys optimal regret!*

Upper confidence bound

Why is it called upper confidence bound?

One can prove that with high probability,

$$\mu_a \leq \text{UCB}_{t,a}$$

so $\text{UCB}_{t,a}$ is indeed an upper bound on the true mean.

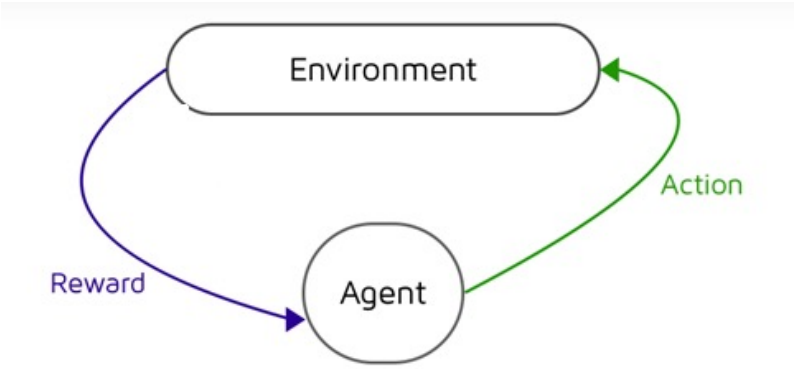
Another way to interpret UCB, "**optimism in face of uncertainty**":

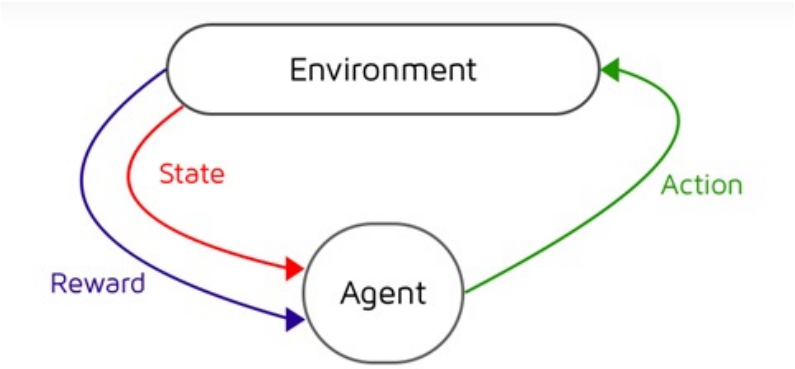
- true environment (best mean) is unknown due to randomness (**uncertainty**)
- have an upper bound (optimistic guess) on the expected reward of each environment, and pick best one according to upper bound (**optimism**)

This principle is useful for many other bandit problems.

Limitations of multi-armed bandits

Multi-armed bandit is among the simplest decision making problems with limited feedback.





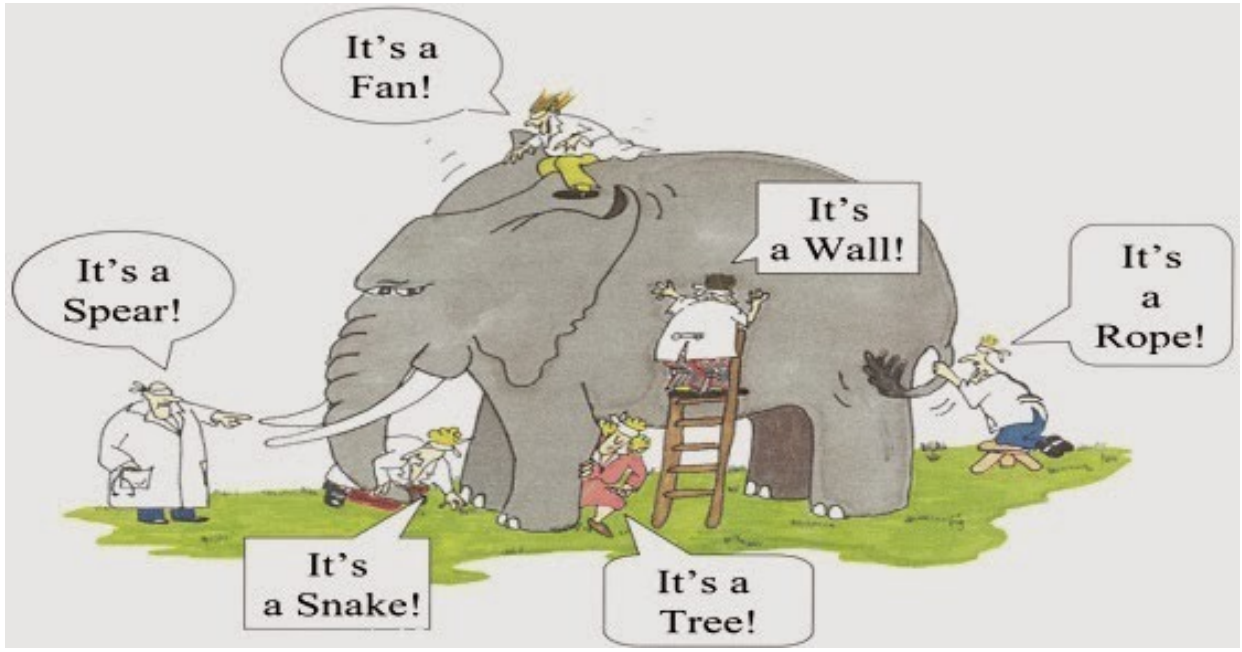
Often, it can be too simple to capture real-life problems. One important aspect it fails to capture is the "state" of the learning agent, which has impacts on the reward of each action.

- e.g. for Atari games, after making one move, the agent moves to a different state, with possible different rewards for each action

There are many other techniques and models in reinforcement learning (RL) which can deal with this issue.

Responsible ML

Machine Learning can be *brittle*



The Blind Men and the Elephant

It was six men of Indostan
To learning much inclined,
Who went to see the Elephant
(Though all of them were blind),
That each by observation
Might satisfy his mind.

The First approached the Elephant,
And happening to fall
Against his broad and sturdy side,
At once began to bawl:
"God bless me! but the Elephant
Is very like a WALL!"

….

Responsible ML

- ML models can be very sensitive to changes in the data distribution
- ML models can be biased against certain sub-populations
- …
- Conclusion

ML models can be very sensitive to changes in the data distribution

You saw a small example of this in the HW3 Bonus question:

ML models can latch onto spurious features to make predictions

Consider the following task:

Waterbird

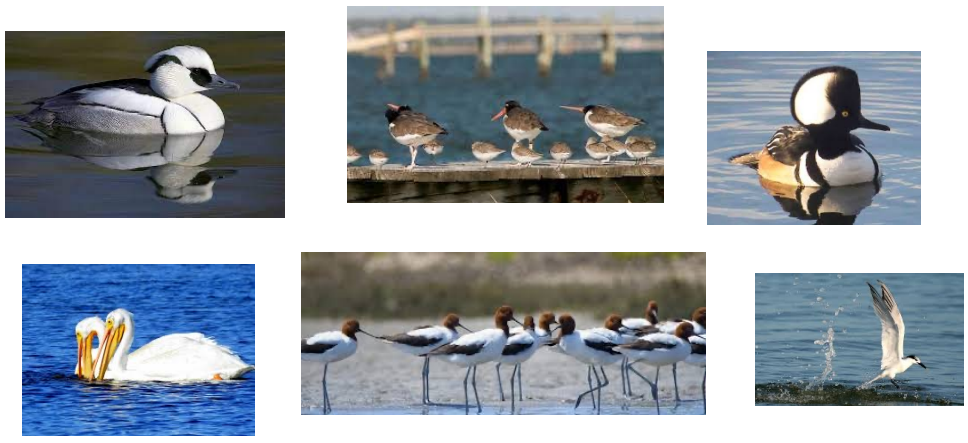
vs.

Landbird

Images courtesy of: University of Hawaii at Manoa, Greg Schechter (Flickr)

ML models can latch onto spurious features to make predictions

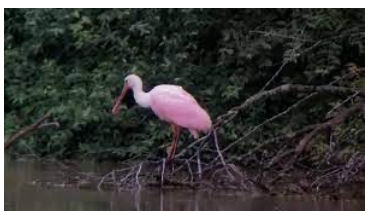
Most images of waterbirds are in water,
and landbirds are on land



Waterbirds vs. Landbirds

ML models can latch onto spurious features to make predictions

But this isn't always true!


Waterbirds

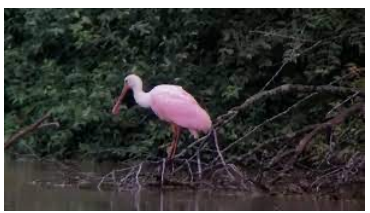
vs.

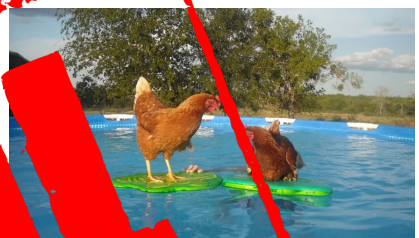


Landbirds

ML models can latch onto spurious features to make predictions

This is known as failure to distributional shifts





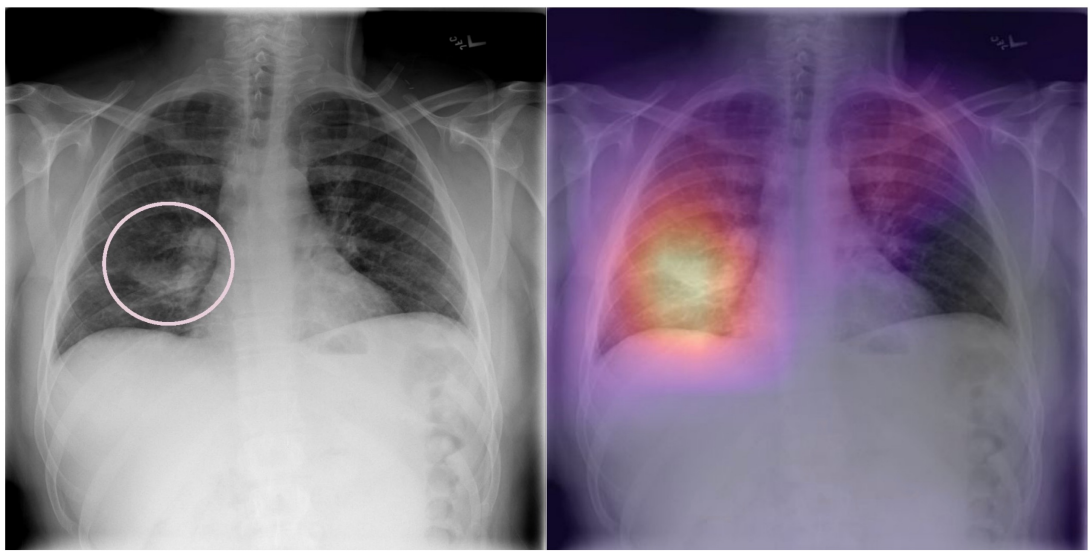
FAIL

Waterbirds vs. Landbirds

A real-world example

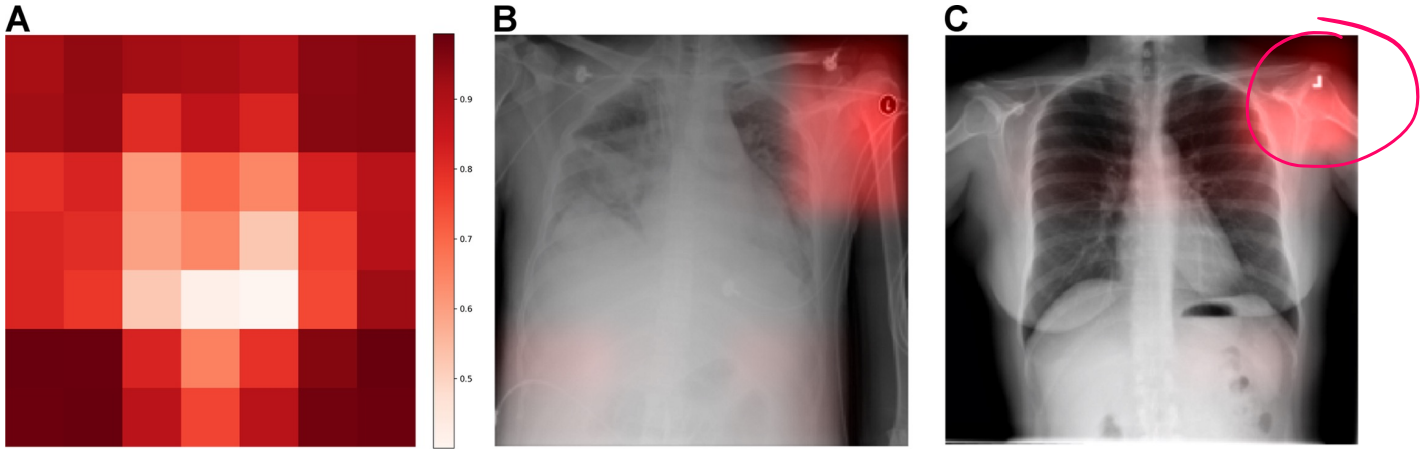
CNN models have obtained impressive results for diagnosing X-rays

E.g. *ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases*, Wang et a;. 2017



Source: *Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists,* Rajpurkar et al. 2018

But the models may not generalize as well to data from new hospitals because they can learn to pickup on spurious correlations such as the type of scanner and marks used by technicians in specific hospitals!



CNN to predict hospital system detects both general and specific image features.
(A) We obtained activation heatmaps from our trained model and averaged over a sample of images to reveal which subregions tended to contribute to a hospital system classification decision. Many different subregions strongly predicted the correct hospital system, with especially strong contributions from image corners. (B-C) On individual images, which have been normalized to highlight only the most influential regions and not all those that contributed to a positive classification, we note that the CNN has learned to detect a metal token that radiology technicians place on the patient in the corner of the image field of view at the time they capture the image. When these strong features are correlated with disease prevalence, models can leverage them to indirectly predict disease.

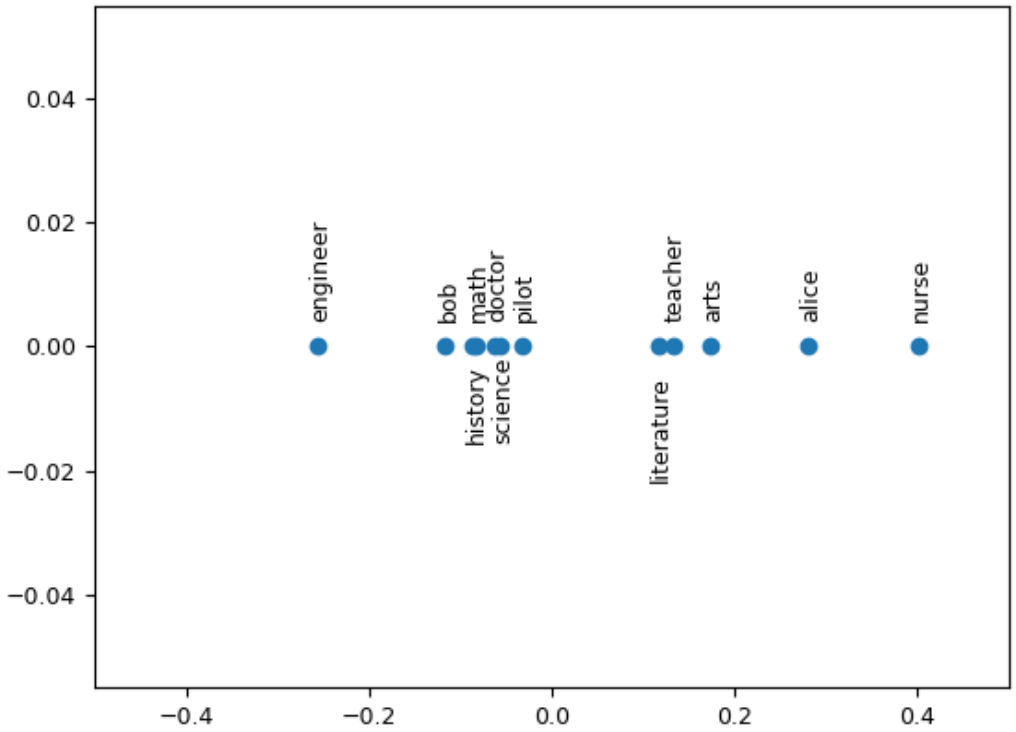
Source: *Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study,* Zech et al. 2018

Responsible ML

- ML models can be very sensitive to changes in the data distribution
- ML models can be biased against certain sub-populations
- …
- Conclusion

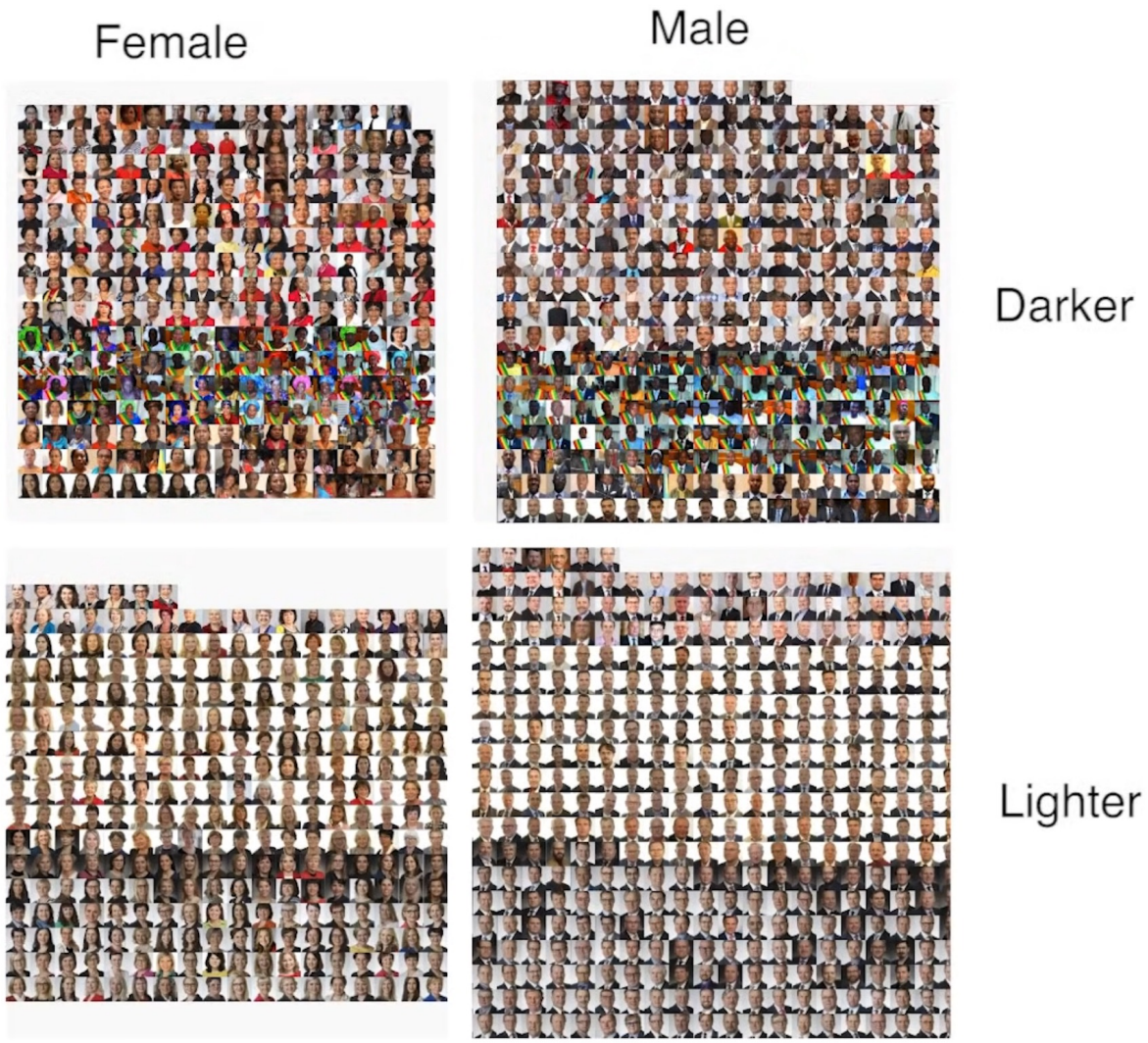
ML models can be biased against certain sub-populations

You saw a small example of this in the HW4 word embedding question:




The Gendershades Project

How well do facial recognition tools perform on various demographics?

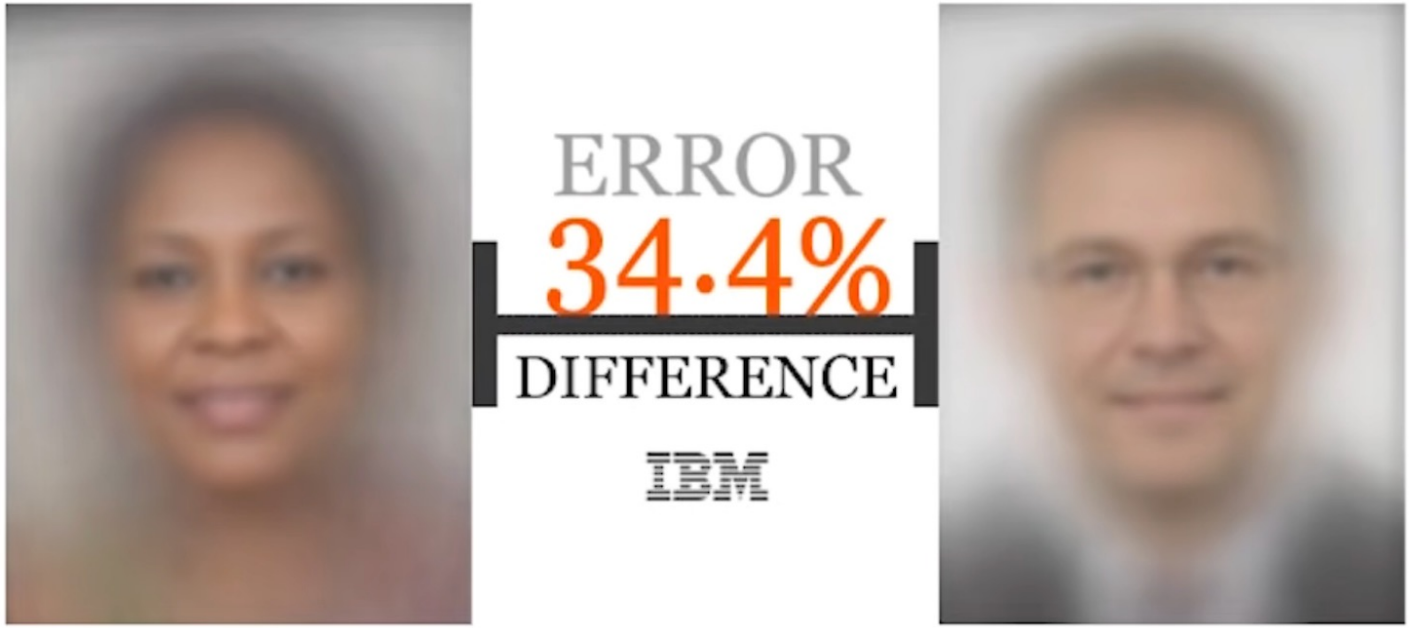


http://gendershades.org/

Ans: *Not very well*

	TYPE I	TYPE II	TYPE III	TYPE IV	TYPE V	TYPE VI
Microsoft	1.7%	1.1%	3.3%	0%	23.2%	25.0%
IBM	5.1%	7.4%	8.2%	8.3%	33.3%	**46.8%**
FACE++	11.9%	9.7%	8.2%	13.9%	32.4%	**46.5%**

Ans: *Not very well*



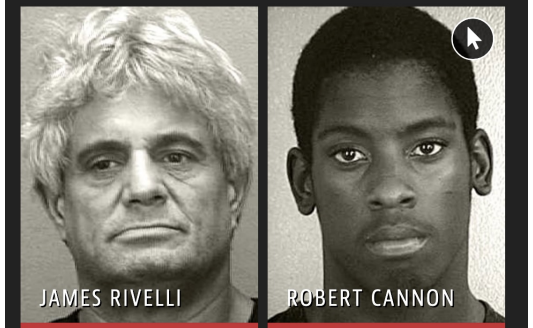
The COMPAS software

Bernard Parker, left, was rated high risk; Dylan Fugett was rated low risk. (Josh Ritchie for ProPublica)

Machine Bias

There's software used across the country to predict future criminals. And it's biased against blacks.

Two Shoplifting Arrests



After Rivelli stole from a CVS and was caught with heroin in his car, he was rated a low risk. He later shoplifted $1,000 worth of tools from a Home Depot.

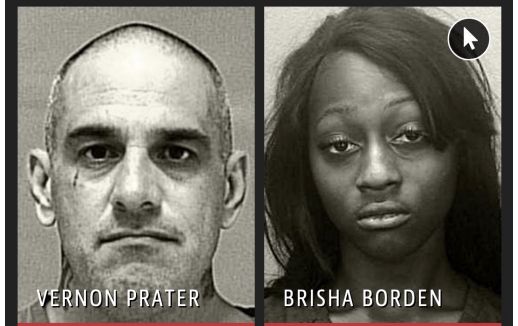
Two Drug Possession Arrests

Fugett was rated low risk after being arrested with cocaine and marijuana. He was arrested three times on drug charges after that.

Two DUI Arrests

Lugo crashed his Lincoln Navigator into a Toyota Camry while drunk. He was rated as a low risk of reoffending despite the fact that it was at least his fourth DUI.

Two Petty Theft Arrests



Borden was rated high risk for future crime after she and a friend took a kid's bike and scooter that were sitting outside. She did not reoffend.

https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

The New York Times

There Is a Racial Divide in Speech-Recognition Systems, Researchers Say

Technology from Amazon, Apple, Google, IBM and Microsoft misidentified 35 percent of words from people who were black. White people fared much better.

Give this article

Amazon's Echo device is one of many similar gadgets on the market. Researchers say there is a racial divide in the usefulness of speech recognition systems. Grant Hindsley for The New York Times

Link to article

MIT Technology Review

Featured Topics Newsletters Events Podcasts Sign in Subscribe

ARTIFICIAL INTELLIGENCE

LinkedIn's job-matching AI was biased. The company's solution? More AI.

ZipRecruiter, CareerBuilder, LinkedIn—most of the world's biggest job search sites use AI to match people with job openings. But the algorithms don't always play fair.

By Sheridan Wall & Hilke Schellmann June 23, 2021

Link to article

Aug 19, 2020 - Technology

How an AI grading system ignited a national controversy in the U.K.

Bryan Walsh, author of Axios Future

Illustration: Eniola Odetunde/Axios

A huge controversy in the U.K. over an algorithm used to substitute for university-entrance exams highlights problems with the use of AI in the real world.

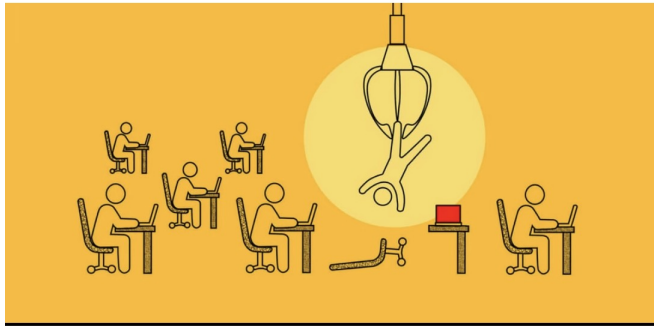
Link to article

TECHNOLOGY

It Was Supposed to Detect Fraud. It Wrongfully Accused Thousands Instead.

How Michigan's attempt to automate its unemployment system went horribly wrong

By Stephanie Wykstra and Undark



Link to article

Responsible ML

- ML models can be very sensitive to changes in the data distribution
- ML models can be biased against certain sub-populations
- …
- Conclusion

Going back to the beginning of Lecture 1..

This class:

- Understand the fundamentals
- Understand when ML works, its limitations, think critically

In particular,

- Study fundamental statistical ML methods (supervised learning, unsupervised learning, etc.)
- Solidify your knowledge with hand-on programming tasks
- Prepare you for studying advanced machine learning techniques

1. Examine your data
2. Examine your model

ML/AI can be very powerful, but should be used responsibly

Lots of great resources to learn more about best AI/ML practices.

Example:

https://ai.google/responsibilities/responsible-ai-practices/