Lecture 11

Density estimation

Intro

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

Given a training set $x_1, \cdots, x_n$, estimate a density function p that could have generated this dataset (via $x_i \sim^{i.i.d.} p$).

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

Parametric methods: generative models

Parametric estimation assumes a generative model parametrized by θ:

$$p(x) = p(x; \theta)$$

Examples:

- GMM: $p(x;\theta) = \sum_{j=1}^{k} \pi_j N(x|\mu_j, \Sigma_j)$ where $\theta = \{\pi_j, \mu_j, \Sigma_j\}$
- Multinomial: a discrete variable with values in $\{1, 2, \cdots, k\}$ s.t.

  $$p(x = j; \theta) = \theta_j$$

  where θ is a distribution over the k elements.

Size of θ is independent of the size of the training set, so it's parametric.

Parametric methods: estimation

As usual, if i.i.d, we can apply MLE to learn the parameters θ:

$$\arg\max_{\theta} \sum_{i=1}^{n} \ln p(x_i; \theta)$$

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

$$\begin{aligned}\sum_{i=1}^{n} \ln p(x = x_i; \theta) &= \sum_{i=1}^{n} \ln \theta_{x_i} = \sum_{i=1}^{n} \sum_{j=1}^{k} \mathbb{1}(x_i = j) \ln \theta_j \\ &= \sum_{j=1}^{k} \sum_{i: x_i = j} \ln \theta_j = \sum_{j=1}^{k} z_j \ln \theta_j\end{aligned}$$

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 . (HW4 Q2.1)

Nonparametric methods

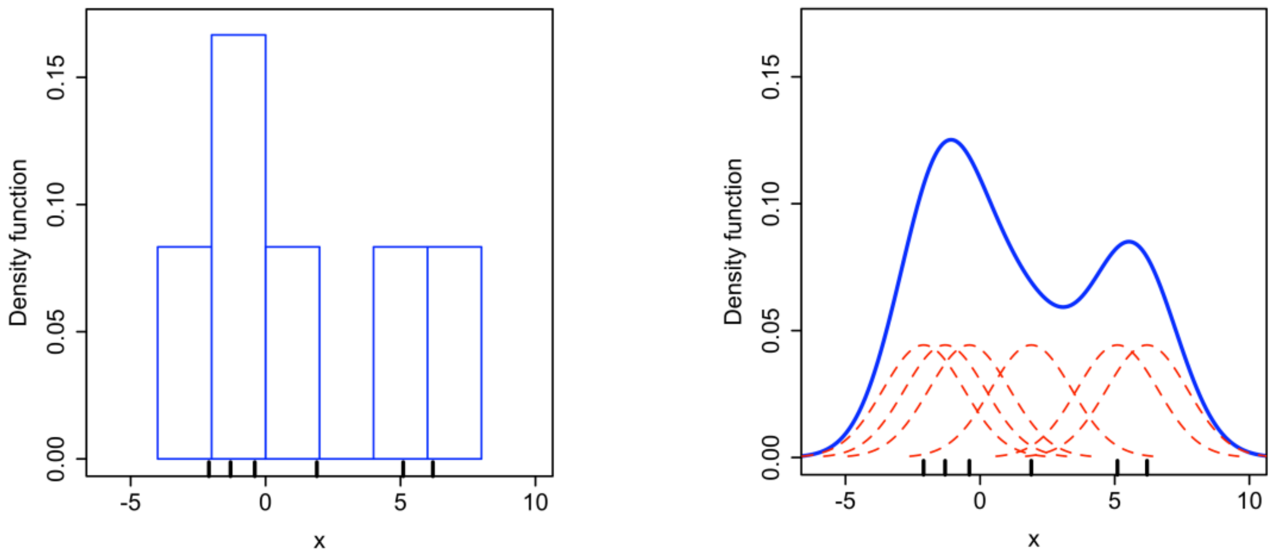
Can we estimate without assuming a fixed generative model?

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 $K\colon \mathbb{R} \to \mathbb{R}$:

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

K is adding bump around x_i , we need to keep all datapoints

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

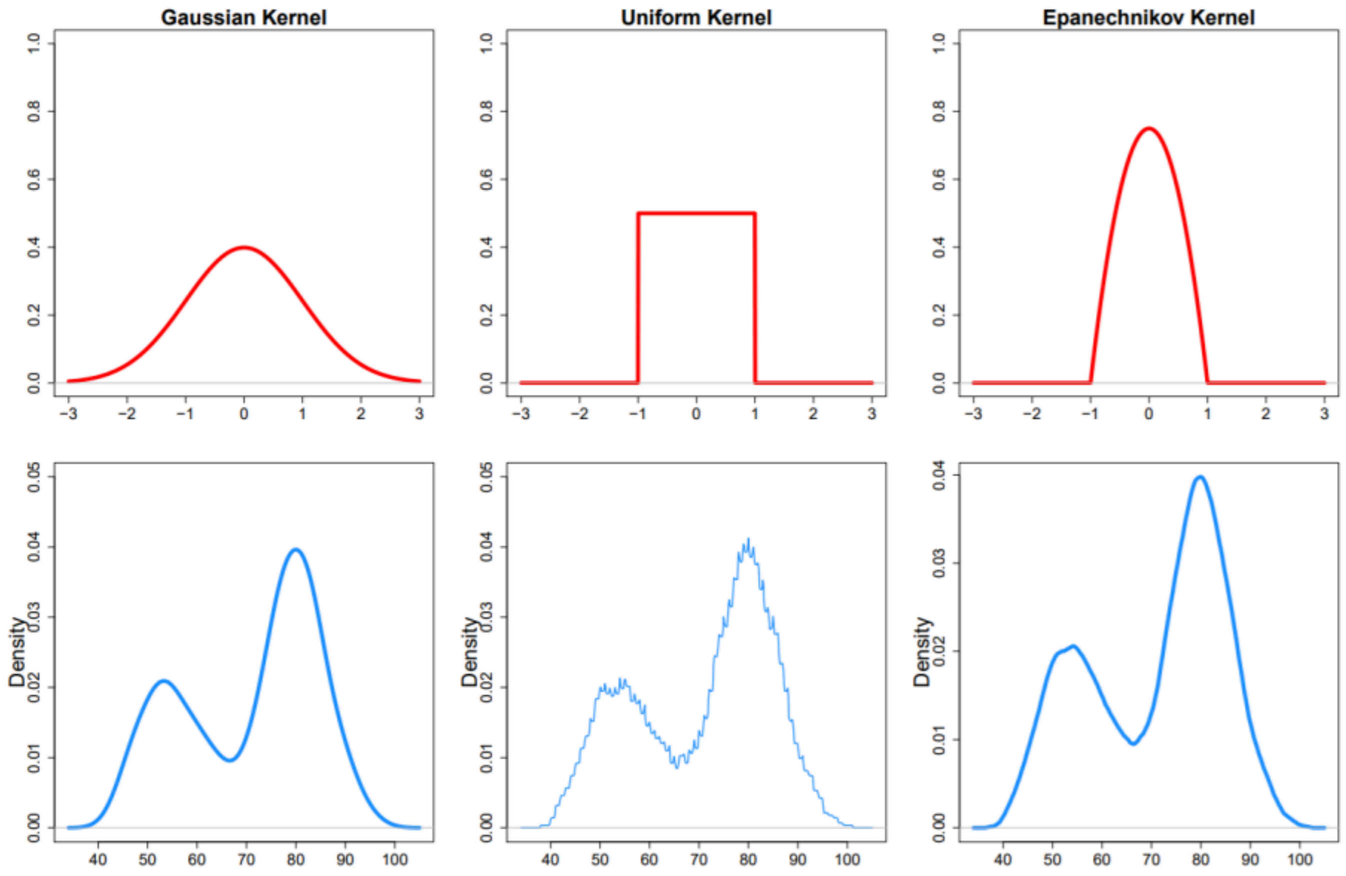
Kernel needs to satisfy:

- symmetry: $K(u) = K(-u)$

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

Different kernels

$$\frac{1}{\sqrt{2\pi}}e^{-\frac{u^2}{2}} \qquad \frac{1}{2}\mathbb{I}[|u| \leq 1] \qquad \frac{3}{4}\max\{1-u^2, 0\}$$



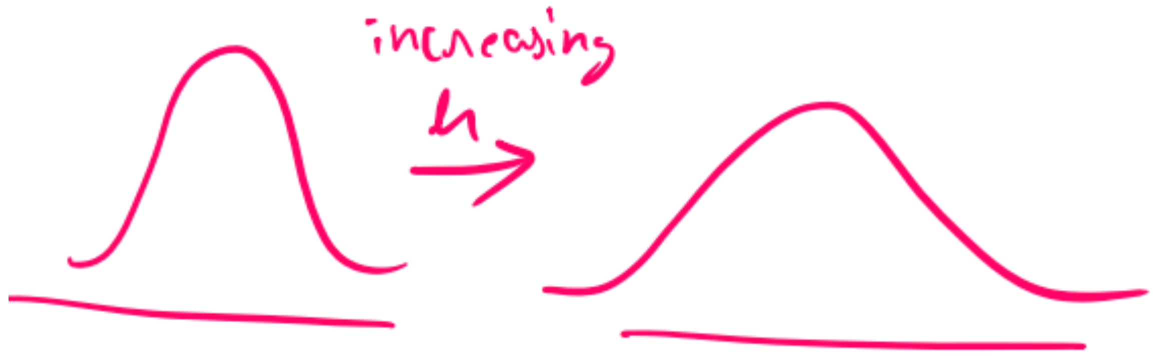
Bandwidth

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

$$K_{h(u)} = \frac{1}{h}K(\frac{u}{h}) \qquad (stretching\ the\ kernel)$$

can be used as a kernel too (verify the two properties yourself) $\int_{-\infty}^{\infty} = \frac{1}{h}K(\frac{u}{h})\mathrm{d}u = 1$.

if h increases,

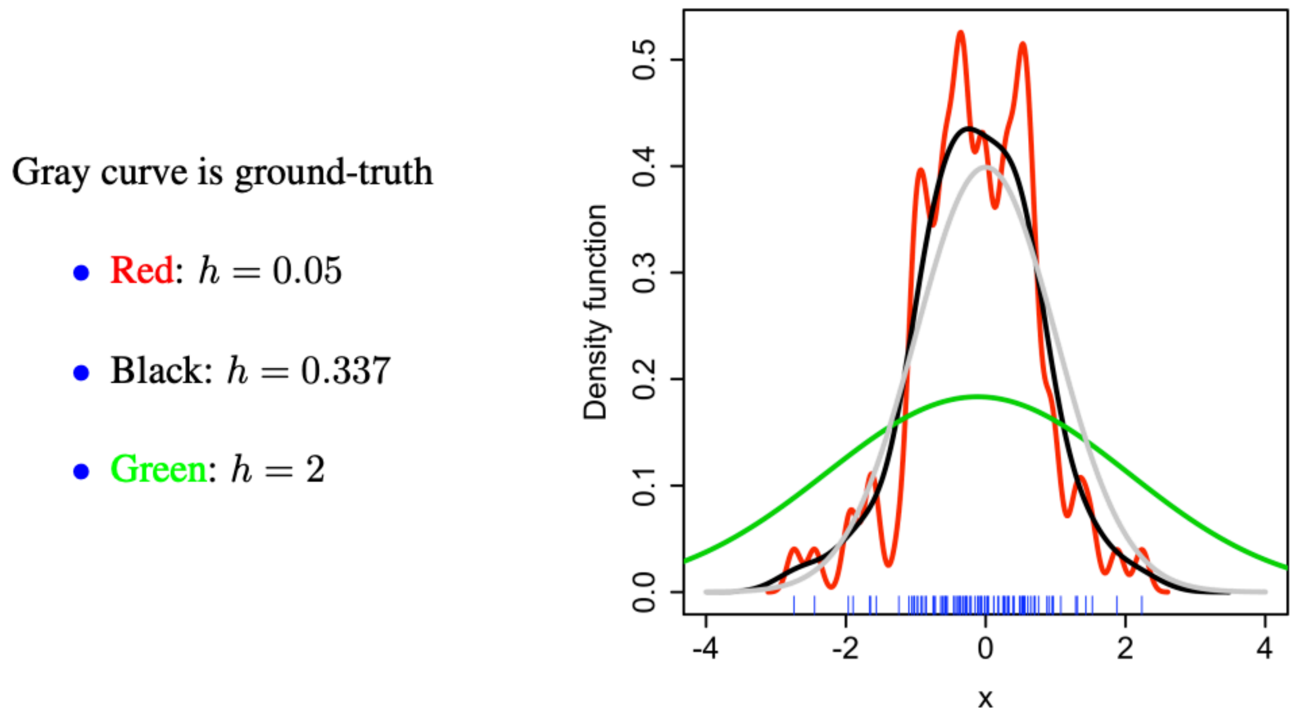


So general KDE is determined by both the kernel K and the bandwidth h :

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

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

Larger h means larger variance and smoother density.



Naive Bayes

Suppose (x, y) is drawn from a joint distribution p. The Bayes optimal classifier is

$$f^*(x) = \arg\max_{c \in [C]} p(c|x)$$

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

$$p(x, y) = p(y)p(x|y)$$

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

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

This is not a 1D problem in general.

A naive assumption

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

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

x_j is the j-th coordinate of 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 $x =$ (Height, Vocabulary) to predict $y =$ 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: $\leqslant 3', 3' - 4', 4' - 5', 5' - 6', \geqslant 6'$ (5 possible values)

Vocabulary: $\leqslant 5K, 5K - 10K, 10K - 15K, 15K - 20K, \geqslant 20K$ (5 values)

Age: $\leqslant 5, 5 - 10, 10 - 15, 15 - 20, 20 - 25, \geqslant 25$ (6 possible values)

MLE estimation: e.g.

$$\begin{aligned} &p(Age = 10 - 15) = \frac{\#examples\ with\ age\ 10 - 15}{\#examples} \\ &p(Height = 5' - 6'|Age = 10 - 15) \\ &= \frac{\#examples\ with\ height\ 5' - 6'\ and\ age\ 10 - 15}{\#examples\ with\ age\ 10 - 15} \qquad (conditional = \frac{joint}{marginal}) \end{aligned}$$

Discrete features: More formally

For a label $c \in [C]$

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

For each possible value l of a discrete feature j ,

$$p(x_i = l|y = c) = \frac{|\{i : x_{i,j} = l, y_i = c\}|}{|\{i : y_i = c\}|}$$

$p(x_i = l|y = c)$ is the density for the j-th feature, $x_{i,j}$ means the j-th feature of i-th datapoint.

Continuous features

If the feature is continuous, we can do

- parametric estimation, e.g. via a Gaussian

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

  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 :

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

Naive Bayes: Prediction

After learning the model

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

the prediction for a new example x is

$$\begin{aligned}
\arg\max_{c\in[C]} p(y = c|x) &= \arg\max_{c\in[C]} \frac{p(x, y = c)}{p(x)} = \arg\max_{c\in[C]} p(y = c) \cdot p(x|y = c) \\
&= \arg\max_{c\in[C]} \left(p(y = c) \prod_{j=1}^{d} p(x_j|y = c)\right) \quad (Naive\ Bayes\ assumption, priors \cdot lielihood) \\
&= \arg\max_{c\in[C]} \left(\ln p(y = c) + \sum_{j=1}^{d} \ln p(x_j|y = c)\right)
\end{aligned}$$

For discrete features, plugging in previous MLE estimation gives

$$\begin{aligned}
\arg\max_{c\in[C]} p(y = c|x) &= \arg\max_{c\in[C]} p(x, y = c) \\
&= \arg\max_{c\in[C]} \left(\ln p(y = c) + \sum_{j=1}^{d} \ln p(x_j|y = c)\right) \\
&= \arg\max_{c\in[C]} \left(\ln |\{i : y_i = c\}| + \sum_{j=1}^{d} \ln \frac{|\{i : x_{i,j} = x_j, y_i = c\}|}{|\{i : y_i = c\}|}\right)
\end{aligned}$$

What is Naive Bayes learning? Connection to logistic regression

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

$$
\begin{aligned}
&\operatorname*{arg\,max}_{c\in[C]} p(y=c|x) \\
&= \operatorname*{arg\,max}_{c\in[C]} \left(\ln|\{i : y_i = c\}| - \sum_{j=1}^{d} \left(\ln\sigma + \frac{(x_j-\mu_{c,j})^2}{2\sigma^2} \right) \right) \qquad (x_j^2 \; is\; constant\; across\; all\; classes) \\
&= \operatorname*{arg\,max}_{c\in[C]} \left(\ln|\{i : y_i = c\}| - \sum_{j=1}^{d} \frac{\mu_{c,j}^2}{2\sigma^2} + \sum_{j=1}^{d} \frac{\mu_{c,j}}{\sigma^2} x_j \right) \\
&= \operatorname*{arg\,max}_{c\in[C]} \left(w_{c0} + \sum_{j=1}^{d} w_{cd} x_j \right) = \operatorname*{arg\,max}_{c\in[C]} w_c^T x \qquad (linear\; classifier)
\end{aligned}
$$

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

Moreover, by a similar calculation you can verify

$$p(y=c|x) \propto e^{w_c^T x}$$

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, logistic is on $p(y=c|x)$ (we said $p(y=c|x$ for $x=\{\pm1\} = \sigma(yw^T x)$.), the other(Bayes) is on $p(x,y)$.
- solutions are different: logistic regression has no closed-form, naive Bayes admits a simple closed-form. (GMM doesn't have closed-form, either)

	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

Logistic Regression is a discriminative model (learn the classifier hypo-plane for each class), Naive Bayes is a generative model (model the density for each class).

Multi-armed bandit (Reinforcement Learning)

Multi-armed bandits (Reinforcement Learning)

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.

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 and Applicaitons

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?

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)

Formal Setup

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

The problem proceeds in rounds between the environment and a learner:

for each time $t = 1, \cdots, T$

- the environment decides the reward for each arm $r_{t,1}, \cdots, r_{t,K}$ (reward for each arm at time t).
- the learner picks an arm $\alpha_t \in [K]$
- the learner observes the reward for arm at, i.e.

- the learner observes the reward for arm at, i.e., r_{t,α_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($\{0,1\}$) of arm a are i.i.d. samples of $Ber(\mu_a)$, 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 ($\mu_1, \cdots, \mu_K$); the problem is essentially about finding the best arm $\arg\max_a \mu_a$ as quickly as possible

Empirical means

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

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

Where

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

is the number of times we have picked arm 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.

Exploitation only

Greedy: Pick each arm once for the first K rounds.

For $t = K+1, \cdots, T$, pick $a_t = \arg\max_a \hat{\mu}_{t-1,a}$

What's wrong with this greedy algorithm?

Consider the following example:

- $K = 2, a_1 \sim Bern(\mu_1 = 0.6), a_2 \sim Bern(\mu_2 = 0.5)$ (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, opposed to optimal solution.

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.

Explore-then-Exploit

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 $a_t = \arg\max_a \hat{\mu}_{T_0,a}$.

Parameter T_0 clearly controls the exploration/exploitation trade-off.

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

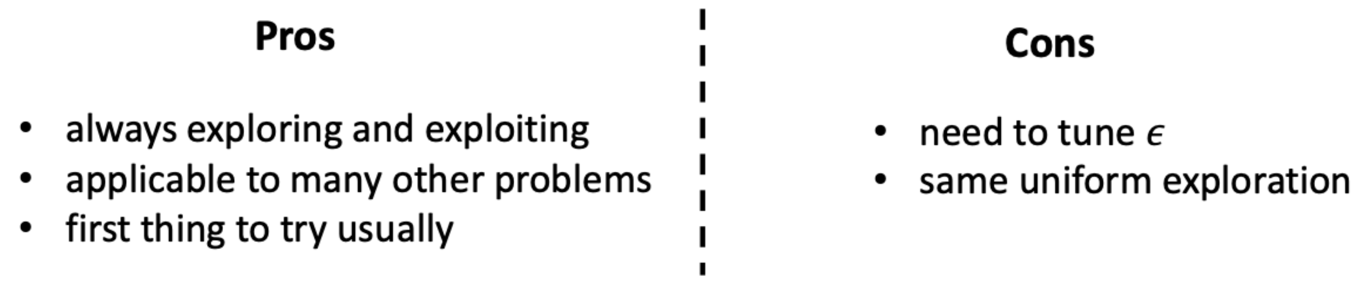
- not clear how to tune the hyper-parameter T_0
- in the exploration phase, even if an arm is clearly worse than others based on a few pulls, it's still pulled T_0/K times
- clearly it won't work if the environment is changing

$\epsilon-$**Greedy: A slightly better algorithm**

$\epsilon-$Greedy Pick each arm once for the first K rounds.

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

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



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?

UCB**: More adaptive exploration**

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

Upper Confidence Bound (UCB) algorithm

For $t = 1, \cdots, T$, pick $a_t = \arg\max_a UCB_{t,a}$ where

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

- $UCB_{t,a}$ initialize all estimates to be same.
- If $n_{t-1,a}$ is small, $UCB_{t,a}$ will be large.
- the first term in $UCB_{t,a}$ represents exploitation, while the second (bonus) term represents exploration.
- the second (bonus)r 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!

Why is it called upper confidence bound?

One can prove that with high probability:

$$\mu_a \leqslant UCB_{t,a}$$

so $UCB_{t,a}$ is indeed an upper bound on the true mean.

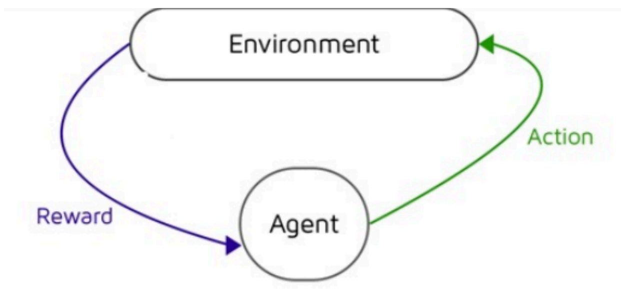
Another way to interpret UCB , "optimism in face of uncertainty":

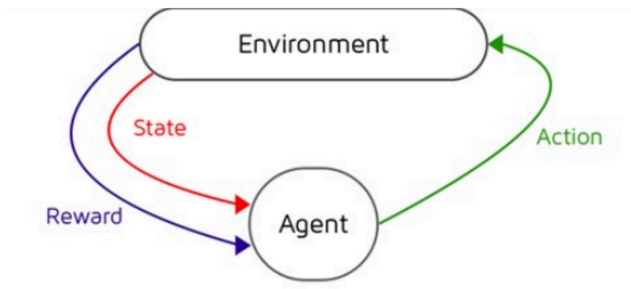
- true environment (best mean) is unknown due to randomness (uncertainty)
- have an upper bound (optimistic guess) on the expected reward of each environment, and pick best one according to upper bound (optimism)

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.

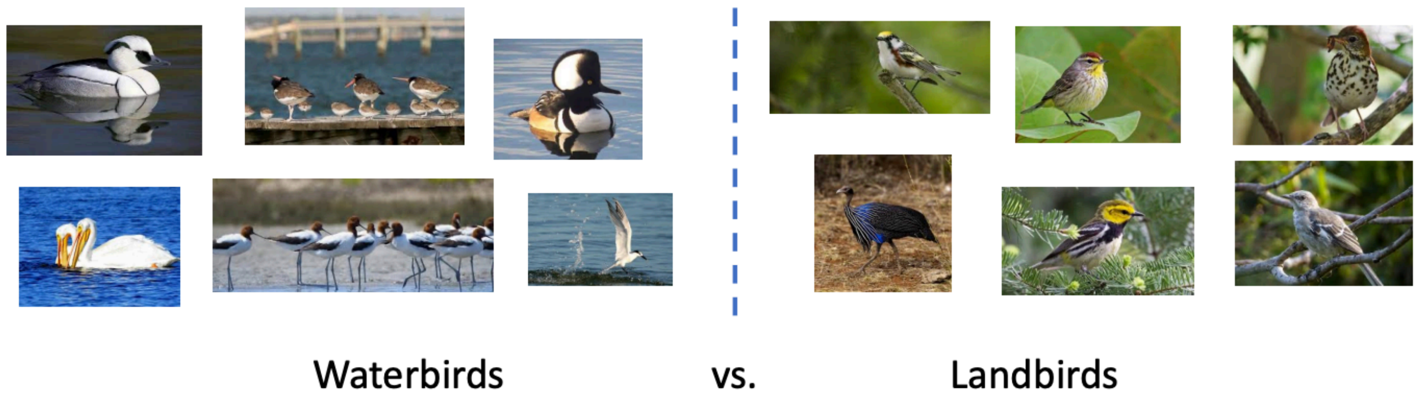
There are many other techniques and models in reinforcement learning (RL) which can deal with this issue.

Responsible ML

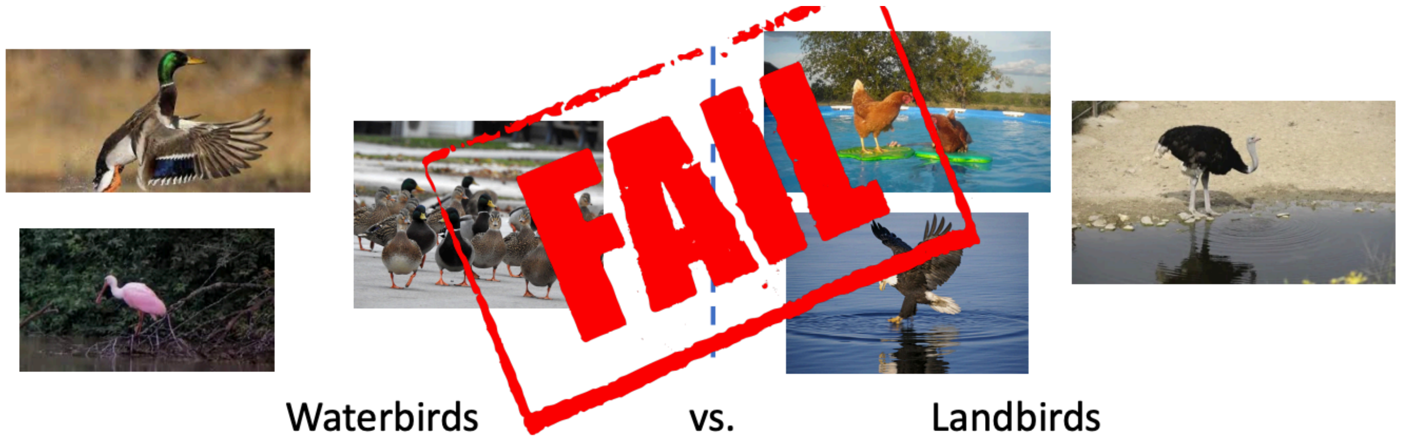
ML models can be very sensitive to changes in the data distribution

ML models can latch onto spurious features to make predictions.

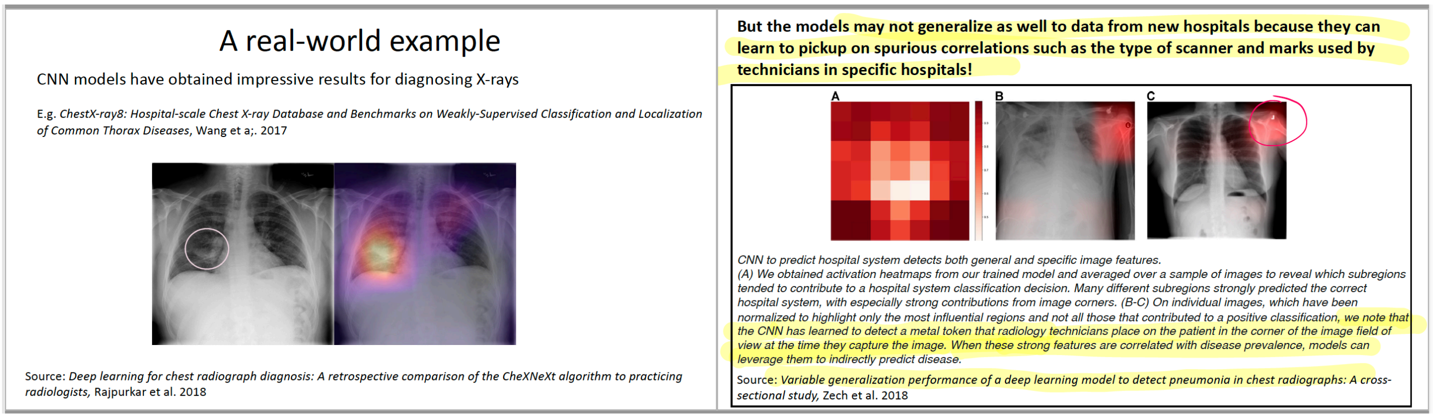
You saw a small example of this in the HW3 Bonus question:



Most images of waterbirds are in water, and landbirds are on land. But this isn’t always true! This is known as failure to distributional shifts.



A real-world example: CNN models have obtained impressive results for diagnosing X-rays. 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!



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!

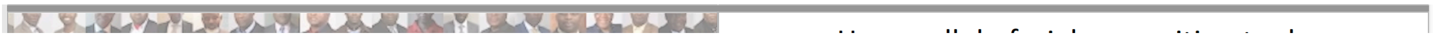
CNN to predict hospital system detects both general and specific image features.
(A) We obtained activation heatmaps from our trained model and averaged over a sample of images to reveal which subregions tended to contribute to a hospital system classification decision. Many different subregions strongly 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

ML models can be biased against certain sub-populations

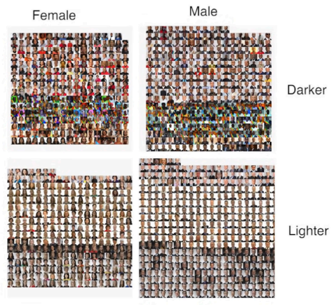
You saw a small example of this in the HW4 word embedding question.

The Gendershades Project. How well do facial recognition tools perform on various demographics? Ans: Not very well.



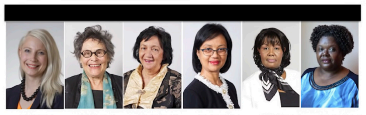
The Gendershades Project

How well do facial recognition tools perform on various demographics?



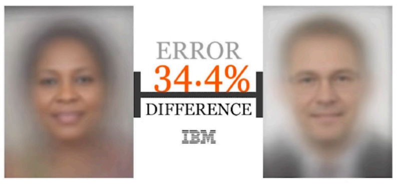
http://gendershades.org/

Ans: *Not very well*

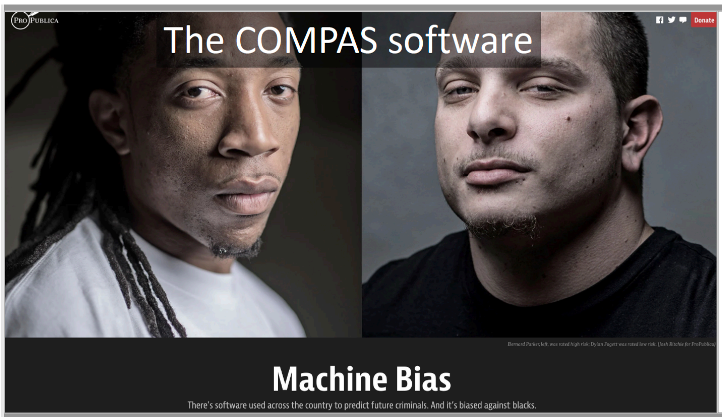


	TYPE I	TYPE II	TYPE III	TYPE IV	TYPE V	TYPE VI
	1.7%	1.1%	3.3%	0%	23.2%	25.0%
IBM	5.1%	7.4%	8.2%	8.3%	33.3%	**46.8%**
FACE++	11.9%	9.7%	8.2%	13.9%	32.4%	**46.5%**

Ans: *Not very well*



The COMPAS software.



ProPublica

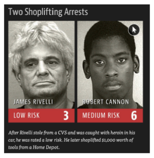
The COMPAS software

Donate

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.

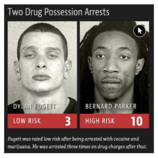


Two Shoplifting Arrests

JAMES RIVELLI LOW RISK 3

ROBERT CANNON MEDIUM RISK 6

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.

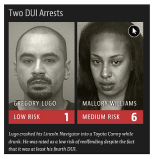


Two Drug Possession Arrests

DYLAN FUGETT LOW RISK 3

BERNARD PARKER HIGH RISK 10

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

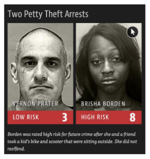


Two DUI Arrests

GREGORY LUGO LOW RISK 1

MALLORY WILLIAMS MEDIUM RISK 6

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.



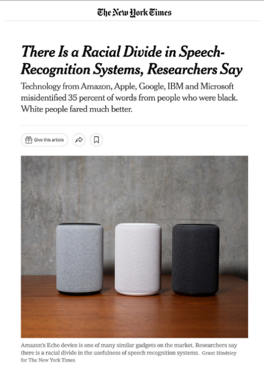
Two Petty Theft Arrests

VERNON PRATER LOW RISK 3

BRISHA BORDEN HIGH RISK 8

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

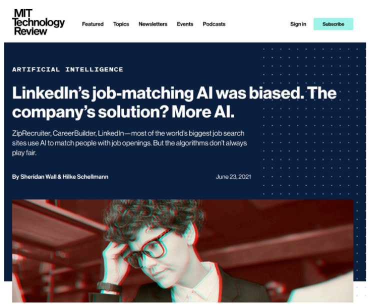
There Is a Racial Divide in Speech-Recognition Systems, Researchers Say

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

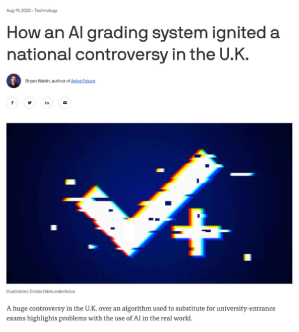
ARTIFICIAL INTELLIGENCE

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

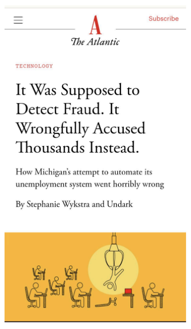
How an AI grading system ignited a national controversy in the U.K.

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



Subscribe

The Atlantic

TECHNOLOGY

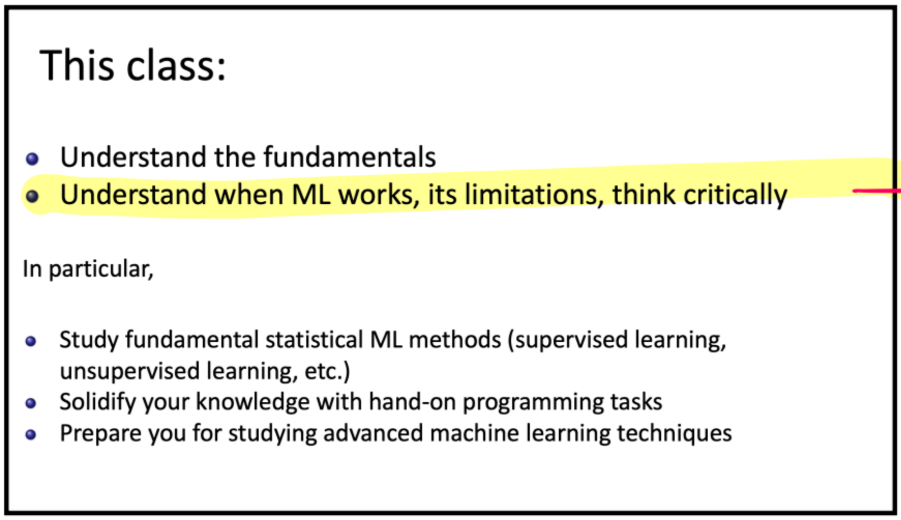
It Was Supposed to Detect Fraud. It Wrongfully Accused Thousands Instead.

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

By Stephanie Wykstra and Undark

Link to article

Conclusion



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.

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