Machine Learning Idea of the Week (MLIW) 1

Common machine learning problems include:

- classification: determining what type an object is based on its characteristics (e.g. spam/legitimate email)
- regression / function estimation: what value should be assigned to an object based on its characteristics (e.g. house price, recovery time from cancer)
- clustering: group objects based on how similar they are to each other (e.g. customer spending patterns, news stories)

Machine Learning Idea of the Week (MLIW) 2

Supervised vs Unsupervised learning:

- **supervised learning** is when the algorithm is given a training set of data to build the model. That could be a list of tumour sizes and whether they were cancerous or non-cancerous, a list of pictures and whether they were chihuahuas or blueberry muffins, etc. The key is that the initial information is given ALONG WITH the answer of what type it is. Then the algorithm can build "rules" based on the training set, which hopefully also work for new data.
- **unsupervised learning** on the other hand does not have an initial set with "correct" answers. An example would be a news aggregator which groups news stories together based on similarities of content, without being told what groups to put them in.

MLIW 3: The Naive Bayes' Classifier

Any classifier (which tries to determine which category a particular object belongs to, based on some observable evidence) is essentially using Bayes' rule to find

$$P(\text{category} \mid \text{evidence}) = P(\text{category}) * P(\text{evidence} \mid \text{category}) / P(\text{evidence})$$

Where the denominator can be calculated using the law of total probability as the sum over all categories, $\Sigma\, P(\text{category}_i) * P(\text{evidence} \mid \text{category}_i)$

The simplest type of classifier is called the Naive Bayes' Classifier, which assumes that if there are multiple pieces of evidence, they are conditionally independent (conditional on the category.)

A very common ML example of this is spam detection, where multiple features might be present with different probabilities in Spam or Valid emails. A great example is problem 22 in the end of chapter 4 questions:

Consider the machine learning problem of classifying incoming messages as spam. We define:
A_1 = message fails rdns check (i.e. the "from" domain doesn't match)
A_2 = message is sent to over 100 people
A_3 = message contains a link with the url not matching the alt text

We will assume that the A_i's are independent events, given that a message is spam, and that they are also independent events, given that a message is regular.
We estimate $P(A_1 \mid \text{Spam}) = 0.3$ $P(A_1 \mid \text{Not Spam}) = 0.005$
$P(A_2 \mid \text{Spam}) = 0.2$ $P(A_2 \mid \text{Not Spam}) = 0.04$
$P(A_3 \mid \text{Spam}) = 0.1$ $P(A_3 \mid \text{Not Spam}) = 0.05$
and $P(\text{Spam}) = 0.25$.

Then, in order to find the probability of a message being Spam if it contains all three features, we can use Bayes' Rule:

$$P(\text{Spam} \mid A_1A_2A_3) = \frac{P(\text{Spam})P(A_1A_2A_3 \mid \text{Spam})}{P(\text{Spam})P(A_1A_2A_3 \mid \text{Spam}) + P(\text{Not Spam})P(A_1A_2A_3 \mid \text{Not Spam})}$$

Then we apply conditional independence:

$$= \frac{P(\text{Spam})P(A_1 \mid \text{Spam})P(A_2 \mid \text{Spam})P(A_3 \mid \text{Spam})}{P(\text{Spam})\,P(A_1 \mid \text{Spam})P(A_2 \mid \text{Spam})P(A_3 \mid \text{Spam}) + P(\text{Not})\,P(A_1 \mid \text{Not})P(A_2 \mid \text{Not})P(A_3 \mid \text{Not})}$$

$$= \frac{0.25*0.3*0.2*0.1}{0.25*0.3*0.2*0.1 + 0.75*0.005*0.04*0.05}$$

$$= 0.995$$

The rest of the parts of the problem can be approached similarly. Remember that A_1, A_2, and A_3 are NOT independent! They are only conditionally independent, given the type of email.

MLIW 4: Two models that use the Binomial distribution

These aren't necessarily strictly machine learning applications, but two cool computer science applications of the Binomial distribution.

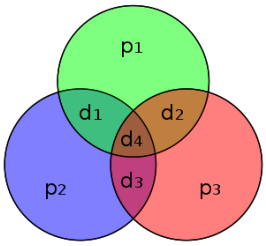
Hamming(4,3) error-correcting code

When sending a message over a "noisy" connection (which may introduce errors), there are lots of ways to improve the probability that the message gets through correctly. One way is to use a "parity bit" which can check whether an error has been introduced. It's not perfect but it can improve the reliability significantly.

Suppose you have a 4-bit message (each bit is 0 or 1) to send, but the connection will "flip" each bit (0 to 1 or 1 to 0) independently with probability 0.1. What is the probability the message is received correctly?

Let X = # of bits flipped.
X ~ Bin(4, 0.1)
We want $P(X = 0) = (4C0)\ 0.1^0\ 0.9^4 = 0.656$

To improve the accuracy, suppose we also send 3 "parity bits" which can allow the receiver to detect and correct up to one error. The way this works uses a 3-event Venn diagram:

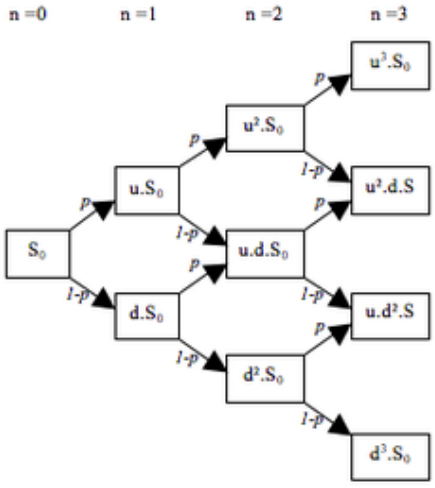


If the message is $d_1d_2d_3d_4$, then parity bits p_1, p_2, and p_3 are set to make the total within each circle even. Then if there is an error in exactly one bit, it can be detected and fixed. (Two or more errors are still a problem.)

Now what is the probability of receiving a correct or correctable message?

Let Y = # of bits flipped.
Y ~ Bin(7, 0.1)
We want $P(Y \leq 1) = (7C0)\ 0.1^0\ 0.9^7 + (7C1)\ 0.1^1\ 0.9^6 = 0.850$

Binomial option pricing model



When pricing a stock option (for example, a call option allows you to purchase a stock at a previously-agreed price called the strike price) the Binomial option pricing model can be used.

We imagine the time period until the option expires is split up into several periods, and in each period we assume the price of the stock can go up by a factor of u with probability p, or down by a factor of d with probability 1-p.

Then the distribution of the final stock price after the last movement is Binomial.

MLIW 5: The Poisson Regression Model

One use of regression (remember that's a machine learning problem that tries to predict a numerical value given a bunch of characteristics) is to predict the count of something, i.e. the number of occurrences of some event in some future time period. For example:

- number of students walking through the SLC expansion per day
- number of cars passing through an intersection per hour
- number of vaccines given out per week
- number of financial transactions per minute

These have in common that they make sense to be modelled by a Poisson distribution: they are non-negative, integer-valued, and have no pre-set maximum. (You could also use the Negative Binomial, or do both and see which one works better.)

Suppose we have some count data for previous time periods - the *training set*. We could estimate the rate λ for the Poisson process over one unit of time, and then we would predict:

$$P(k \text{ events in } 1 \text{ unit of time}) = \frac{e^{-\lambda}\,\lambda^k}{k!}$$

But an assumption of the Poisson process is that the rate λ is constant over time, which we know is not true in many of these cases!

This is where the Poisson Regression Model comes in.

Suppose we have some characteristics that might influence the rate λ, which might be different each day/time period. We call these characteristics the *regressors* (or *explanatory variables*, or *predictors*) and denote them by a vector $\mathbf{x_i}$ for the i^{th} observation. These could be things like:

- day of the week (or just weekend vs weekday)
- weather (category or temperature, precipitation, etc)

We want to write the Poisson rate for day i, λ_i, as some function of $\mathbf{x_i}$. There are lots of choices for the function, but one that works well is $\lambda_i = \exp\{\mathbf{x_i}\,\boldsymbol{\beta}\}$ where $\boldsymbol{\beta}$ is a vector of *parameters* (or regression coefficients.)

Statistical packages such as R or Python can estimate the optimal values of $\boldsymbol{\beta}$ to give the best possible prediction.*

Then on a future day, you find the value of the regressors ($\mathbf{x_i}$), plug into the function $\exp\{\mathbf{x_i}\,\boldsymbol{\beta}\}$ to get your Poisson rate λ_i, and then you would predict:

$$P(k \text{ events on day } i \mid \text{characteristics } \boldsymbol{x_i}) = \frac{e^{-\lambda_i}\,{\lambda_i}^k}{k!}$$

MLIW 6: Data Bias

We noticed that the sample mean in the thought question was 2.102, which is quite different from the average for all of Canada, when X was the number of kids in a family with kids.

There could be many reasons for this, but likely the most important is that the sample in the class was not representative of the population of Canada, since that includes many young families with one child that may have more, whereas most of the people in the class will not be gaining any new siblings. Other reasons could include:

- Selection bias (if you randomly select people rather than families, people with lots of siblings will be over-represented)
- Different socioeconomic status
- Living in a different area
- Etc

Other types of bias include

- Sampling bias (groups may be more/less likely to reply because of the survey method)
- Survivorship bias (only focusing on the results that "made it" can obscure the info about the ones that didn't)

Any time you're building a machine learning algorithm, it's only as good as the data you build it on. So if the data is biased and does not reflect reality, the predictions from the model will be biased as well. An important concept in machine learning is data stewardship - making sure the data going in is accurate, representative, and appropriate for the purpose of the model.

MLIW 7: The Beta Distribution

In MLIW 5 we talked about using regression to predict a discrete value (count data) – but what if the thing you are trying to predict is a probability itself?

We know probabilities are Real numbers between 0 and 1 so we would want a continuous distribution that is bounded between 0 and 1, otherwise the predictions would not be very good.

One distribution (or rather, a family of distributions) that can be used is the Beta distribution, which has pdf given by:

$$f(x) = cx^{\alpha-1}(1-x)^{\beta-1} \; for \; 0 < x < 1$$

where c is the constant that makes the function integrate to 1 (it depends on α and β.)

Depending on your choices of α and β, this distribution can have many different shapes, but they are all bounded between 0 and 1.

If you were trying to predict a probability and you had no idea what it was, you might choose α = β = 1 to start, which gives a completely uniform distribution on (0,1). But if you got more data, certain values of the probability might become more likely than others, and you could change α and β accordingly.

Let's look at an example

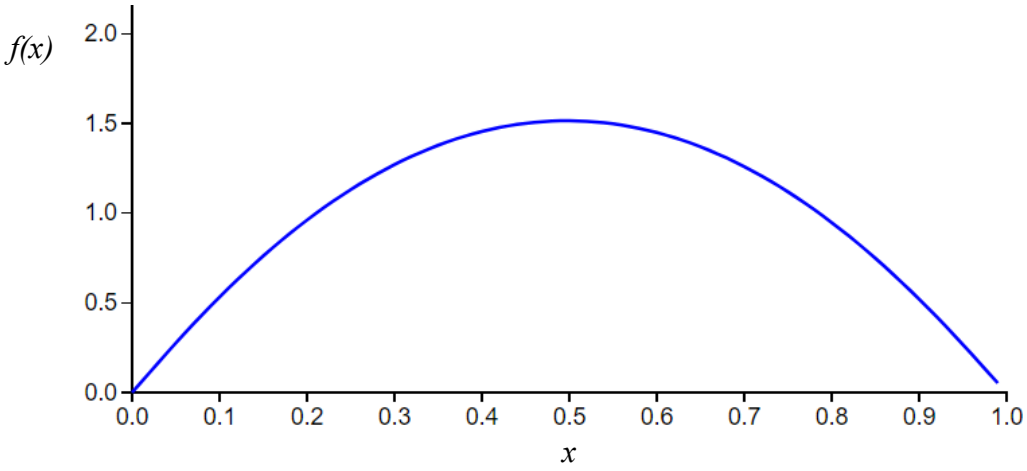
Suppose α and β are both equal to 2. Find *c* and graph the pdf.

$f(x) = cx(1-x), for \; 0 < x < 1$

If we want this to be a valid pdf, we need it to integrate to 1 over the range.

$\int_0^1 cx(1-x)dx = c\int_0^1 x - x^2 dx = c\left[\frac{x^2}{2} - \frac{x^3}{3}\right]_0^1 = \frac{c}{6}$ so $c = 6$.

And the graph looks like this:



You can play around with the values of α and β and see how the shape changes using this (or many other) websites: http://eurekastatistics.com/beta-distribution-pdf-grapher/

MLIW 8: Utility Functions

In Chapter 8 we talk about transforming continuous random variables, i.e. if you have some random variable X with pdf f(x), and a new variable Y = h(X), how can we find the pdf of Y?

We learned the technique using the cdf

1. Write the cdf of Y in terms of the cdf of X
2. Sub in what you know about the cdf of X and differentiate to get the pdf (in either order!)
3. Determine the range of Y

This idea often comes up in Machine Learning by way of something called a "utility function" which assigns a numerical value to the result of an algorithm that represents how "good" it is. How "goodness" is measured depends on the situation, but it's often easy to think about it in terms of how much happiness a certain amount of money would bring.

Let **Y = u(X)** be the utility (happiness) obtained from a wealth of X.
Obviously, u(x) is increasing in x (you would rather have more money than less money!)

So **u'(x) > 0**

But someone with no money at all gaining $10,000 would have much more utility gained than a billionaire gaining $10,000. So the more money you have, the less each additional dollar means to you. (This is known as decreasing marginal utility.)

So **u''(x) < 0**

So if you know a strategy/algorithm will result in a wealth X that has pdf f(x), you can use the technique above to find the distribution of the utility produced, Y.

Additionally, a function with a positive first derivative and a negative second derivative is called a concave function. Concave functions have the property that

E[u(X)] ≤ u(E[X]) for any random variable X (This is called Jensen's inequality.)

So the algorithm should be designed to choose values that maximize the expected utility of an uncertain outcome, E[u(X)], which is not necessarily the same thing as maximizing the expected value of the outcome itself, E[X].

Machine Learning Idea of the Week 9

Setting the threshold for a simple classifier

Let's start with a 1-dimensional problem

Imagine we send a voltage of +2 (for 1) or -2 (for 0) over a connection, to convey a string of bits. The connection is noisy and adds a N(0,1) distributed amount of voltage to whatever signal is sent.

The person receiving the message on the other side must interpret the incoming signal as either a 0 or a 1, based on a threshold c – if the voltage received is above c, it will interpret it as a 1, otherwise a 0. The question is, how should the receiver set the threshold c?

Let's set c = 0.5 (that is, if the voltage received is at least 0.5 they will interpret it as a 1.)

We can find the probability of an error depending on what was actually sent.

If 1 was sent, the voltage received R will be 2 + Z (where $Z \sim N(0,1)$)
So the probability of an error given that a 1 was sent is the probability that the voltage received causes the receiver to interpret it as a 0, i.e. the voltage received is less than 0.5.
$P(R < 0.5) = P(2 + Z < 0.5) = P(Z < -1.5) = 1 – F(1.5) = 1 – 0.93319 = 0.06681$

Similarly, if a 0 was sent, the voltage received will be – 2 + Z and for an error we need
$P(R > 0.5) = P(– 2 + Z > 0.5) = P(Z > 2.5) = 1 – F(2.5) = 1 – 0.99379 – 0.00621$

Because we have c = 0.5 (closer to 2 than to -2), the probability of error is higher for 1's being sent than for 0's.

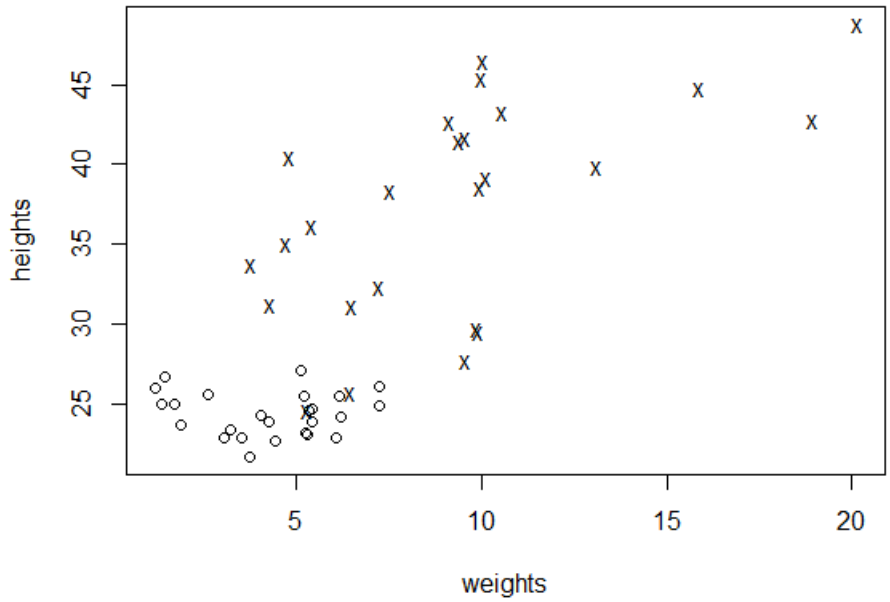
If we wanted the probabilities of error to be equal no matter what the input, we would want to set c to 0 (exactly half way between the signals). Can you see logically why?
That would give an overall probability of error, no matter what signal was sent, of 0.02275. Try to verify that calculation.

But what if we know that we are more likely to receive 1's than 0's, by a factor of (say) 2 to 1 (i.e. the probability of sending a 1 is 2/3 and 0 is 1/3). How could we set c to minimize the overall probability of error?

We want P(error) = P(error|1 sent)*P(1 sent) + P(error|0 sent)*P(0 sent) to be as small as possible.
Using the same process as above, $P(\text{error}|1 \text{ sent}) = P(Z < c-2) = 1 – F(2-c)$ and
$P(\text{error}|0 \text{ sent}) = P(Z > c+2) = 1 – F(2+c)$
We need to solve numerically (or by trial and error) for c, and you'll find the c that minimizes P(error) here with P(1 sent) = 2/3 and P(0 sent) = 1/3 is **c = –0.17** which gives an overall probability of error 0.02121.

Classifiers can also exist in two (or more) dimensions!

Suppose we have the heights and weights of a number of cats and dogs.
First here's a plot of the training set. o's are cats, x's are dogs (this is a supervised machine learning problem since we are given the classifications of the training set.)



We would be interested in creating a rule that would classify an unknown animal into one of the two categories. There are a few ways to do this

- Decision boundaries/trees – determine criteria for each feature e.g. if height < 26 cm and weight < 7, classify as cat
- Support vector machines – determine a line or curve on the graph that best predicts the outcome
- K-nearest neighbours – use the classification of the majority of the k points closest to a new observation

Suppose the weights and heights based on the training set are all normally distributed with the following parameters:

Species	Weight mean	Weight variance	Height mean	Height variance
Dog	9.4	3.6^2	40	5.7^2
Cat	4.1	1.6^2	24	1.5^2

You have a new animal with weight 5 kg and height 27 cm.
(a) Find the probability a cat would be this large (greater than or equal to both).
(b) Find the probability a dog would be this small. What can we conclude?
Answers:
a) 0.28774 x 0.02275 = 0.00655 b) 0.11123 x 0.01130 = 0.00126. About 5 times more likely to be a cat.

Machine Learning Idea of the Week 10

Document content analysis using the "bag of words" algorithm

We can imagine that the words contained in an email are randomly drawn from a "bag of words" with certain words being more or less likely depending on what the email is about.

If the words are theoretically randomly drawn and we know the theoretical probability of all available words, the number of each word in an email will follow a multinomial distribution.

Suppose I got an email that says: "Your account has been limited. Click here to reactivate your account."

Based on a large sample of legitimate and spam emails (a training set), we can find the probabilities of these words are:

Word	**prob if spam**	**prob if legitimate**
account	0.0013	0.0003
been	0.0024	0.0028
limited	0.0008	0.0003
click	0.0084	0.0016
here	0.0076	0.0021
reactivate	0.0011	0.0002
other words	0.9784	0.9927

(a) Find the probability of seeing the distribution of words found in this email if it's spam.

There are 11 words in the email, which include two instances of "account", 1 each of the next 5 words listed above, and 4 instances of other words.

The probability of a Multinomial distribution containing those numbers of observations is

$$f(2,1,1,1,1,1,4) = \frac{11!}{2!4!}0.0013^2 0.0024^1 0.0008^1 0.0084^1 0.0076^1 0.0011^1 0.9784^4 = 3.12 \times 10^{-8}$$

(b) Find the same probability if it's legitimate. What can we conclude?

Similarly, we do the same calculation but using the probabilities for legitimate emails

$$f(2,1,1,1,1,1,4) = \frac{11!}{2!4!}0.0003^2 0.0028^1 0.0003^1 0.0016^1 0.0021^1 0.0002^1 0.9927^4 = 7.27 \times 10^{-10}$$

The word distribution in this email is 42.9 times more likely to have come from the spam bag of words than the legitimate bag. So it's extremely likely this is a spam message.

(c) Find the probability an 11-word spam message has none of those 6 words in it.

This would just be $f(0,0,0,0,0,0,11) = \frac{11!}{11!}0.9784^{11} = 0.786$.

MLIW 11 – Markov Decision Processes

Section 9.3 in the course notes, which is optional for STAT 230 and often left out, is one of my favourite topics: Markov Chains. If you are interested in learning more, I suggest taking STAT 333 or 334, or start by checking out my video about it here: https://www.youtube.com/watch?v=8shnKn5NknI&list=PLOw7dLwplCCx3qw5R7eTtNq7rZ1DQ6TX1&index=13.

Definition
A Markov Chain is a sequence of discrete random variables indexed by time, $X_0,X_1,X_2,...$
The range (possible values) of each variable is called the state space S = {1, 2, …, N}
(but you could have a chain with negative states, or countably infinite states!)
If Xn = i, we say the process is in state i at time n.

It also must satisfy the Markov Property: that the probability of where the chain goes next depends only on the current state, not on any history of the process before that.
Formally, $P(X_{n+1} = j|X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = P(X_{n+1} = j|X_n = i)$ and we call it P_{ij}
The P_{ij}'s can be arranged in a matrix **P** known as the one-step transition probability matrix. **P** is square, all its elements are non-negative, and its rows must sum to 1.

Some Markov Chains keep going forever and may reach an equilibrium distribution, and some may have one or more absorbing states where you can get stuck in the long term.

Examples

1. States are votes for different political parties in US elections
   S = {1,2} $P = \begin{bmatrix} \alpha & 1-\alpha \\ 1-\beta & \beta \end{bmatrix}$ (could estimate α and β for different States)
2. Random walk: Start with $i. On each game you either win $1 (prob p) or lose $1 (prob q = 1 – p). You stop playing when you reach $0 (bankrupt.)
   S = {0,1,2,…} $P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ q & 0 & p & 0 & 0 \\ 0 & q & 0 & p & 0 \\ 0 & 0 & q & 0 & p \\ 0 & 0 & 0 & \ddots & \ddots \end{bmatrix}$
3. Tennis: State 1: A wins, 2: advantage A, 3: deuce, 4: advantage B, 5: B wins
   S = {1,2,3,4,5} $P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.42 & 0 & 0.58 & 0 & 0 \\ 0 & 0.51 & 0 & 0.49 & 0 \\ 0 & 0 & 0.63 & 0 & 0.37 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

Use in reinforcement learning
Attach a "reward" to each state and then you can calculate the total reward for various paths through the Markov Chain, possibly applying a "discount" to steps that are farther in the future.
Then you can determine the optimal choice to make at each step, and the optimal overall policy (set of rules for making decisions at each step) to maximize the total reward.

Machine Learning Idea of the Week 12 – Ethical Use of Data

Weapons of Math Destruction and the Ethical Use of Data
by Diana Skrzydlo

Algorithms are amazing. They can drive your car, suggest what someone might want to watch or buy, track down fraud, keep communications secure, or even predict where the next goose will attack, all with remarkable efficiency.

But an algorithm is only as good as the data it's trained on. If reality is biased, an algorithm trained on that reality will reflect existing inequalities, and worse: perpetuate them with that same remarkable efficiency. The best algorithms get feedback to improve – if a prediction is found to be incorrect, tweaks can be made or additional data included. But many algorithms are black boxes, and end up creating loops that reinforce their own biases rather than correcting them.

Imagine the city of Waterloo is trying to optimize where it sends its police officers to minimize violent crime. One possibility would be to look at data from where violent crimes were committed in the past, and spend more time in those areas. But violent crime is relatively rare, and a model based on that little data might have poor predictive power. So what if they also include data from non-violent, or "nuisance" crimes, such as loitering or bylaw violations? That will give a huge dataset, and would probably reveal more such crimes reported in student areas. So the city would have its police officers spend more time in student areas, where they would probably catch more nuisance crimes, simply by spending more time there. These reports would go back into the algorithm, telling them to spend even more time there, etc. The algorithm would look like a success by catching more "criminals" and validating the high crime areas, but would the city accomplish its goal of reducing violent crime? Probably not! All it would do is catch a lot of minor crimes, while discriminating against students.

Something very similar to this hypothetical example actually happened in many major US cities. Predictive policing algorithms, when fed with "nuisance" crime data, had police spending more and more time in poorer neighbourhoods, which were also highly correlated with racialized neighbourhoods. One result was that while marijuana use is almost equal between white and black teenagers, black teenagers were more than four times as likely to be arrested for possession. The police probably even thought they were being unbiased by "blindly" following the algorithm's recommendations. Unfortunately people believe that algorithms are inherently fair, but they're only as good as their inputs.

Consider another example: the credit score. This started out as a good kind of algorithm - the definition is relatively transparent, the things that help you improve your score (pay bills on time, stop ordering new credit cards, etc.) are actually related to your creditworthiness, and you are entitled to know your information and correct errors if any are found. Sadly, over time it became used as a proxy for things other than the ability to pay back loans.

Insurance companies realized that people with higher credit scores also tended to have fewer claims, and, surmising that people who were conscientious with their finances would be conscientious drivers as well, gave them discounts on car insurance. Employers, looking for a

shortcut to screening employees, began using credit score to influence hiring decisions. The problem is that credit score is also a proxy for socio-economic status. This resulted in people who were poorer being charged more for insurance and being less likely to get hired, which in turn reinforced the cycle of poverty. The issue here is that using credit score as a proxy for things other than your ability to pay bills is invalid. Of course the wealthy are more able to pay back loans, but that doesn't actually make them better drivers or more valuable employees. At a certain point using credit score in this way is just a flimsy excuse to discriminate against poor people. In Canada it is illegal to use credit score as a rating factor in insurance or for hiring decisions, but in the United States it is still used. Punishing those who can least afford it shouldn't be the goal of any financial system, but unless fairness is explicitly accounted for in the algorithms, bias creeps in. Our present society is biased and our systems should seek to correct this as opposed to reinforce it.

Transparency in algorithms is important, but sometimes it can be a problem as well, opening up the door for people and institutions to game the system. This is readily apparent in university rankings. There are many lists of rankings for universities & colleges, both national and international, and they usually have fairly clear criteria. The problem is, a huge amount of time and effort (and money) is spent improving the factors that affect the ranking, but which may or may not have any bearing on actual quality of education. This is the invalid proxy problem cropping up again.

So what can you do? You will be working for companies who make these kinds of decisions one day. It's important to determine whether the data you're using is actually measuring what you want to measure, or is a proxy for something else. Ensure that feedback loops are corrective in nature as opposed to reinforcing bias. Think about whether all the data is actually relevant: sometimes leaving out certain data is the right thing to do. Occasionally it comes down to a moral issue, requiring balance between efficient algorithms and getting non-harmful effects. A small sacrifice in efficiency that prevents a feedback loop of unfairness is a sacrifice you should make.

For more information and lots more examples, please see the book "Weapons of Math Destruction" by Cathy O'Neil. She gave a fascinating talk and workshop at UW in Winter 2018, and most of this article is based on her work.

printed in mathNEWS, volume 137 issue #4