How Algorithms Predict Who We Are

Zach Ahmed

1. The Problem

Imagine you are browsing Netflix on a quiet Sunday evening. You have just finished a tense political thriller and you are not sure what to watch next. Within seconds, the platform presents you with a list of recommendations, each one feeling oddly well-suited to your mood. You pick one, enjoy it, and think nothing more of it. But pause for a moment and ask: how did Netflix know? Nobody at the company looked at your viewing history and made a judgement call. No human being decided that, because you enjoyed that particular thriller, you might enjoy this particular drama. A mathematical algorithm did. And the algorithm, it turns out, knew something surprisingly precise about your tastes, inferred entirely from the pattern of what you had watched, when you had watched it, and how long you had stayed before switching off.

The question this raises is not merely one of convenience or clever engineering. It is a deeper and more interesting question. How, precisely, does mathematics extract the person from the data? How does a list of numbers become a prediction about a human being? And what are the fundamental mathematical limits of what such a system can ever know? These are the questions this essay sets out to answer, and the journey from your Netflix queue to the frontiers of linear algebra and probability theory turns out to be a surprisingly short one.

2. A Point in Space

Before any algorithm can predict anything about you, it needs a way of representing you mathematically. The solution that underlies almost all of modern machine learning is both simple and profound: you are a point in space.

Every piece of data about you becomes a number. On a streaming platform, this might be a rating you gave a film, a genre you tend to favour, or the time of day you usually watch. Collect enough of these numbers together and you have a **vector** — an ordered list of numerical values, one for each feature the system is tracking. If the platform tracks two features, say your preference for action films and your preference for comedies, then you can be represented as a point in two-dimensional space. Track a thousand features and you become a point in a thousand-dimensional space.

This sounds abstract, but the geometry that follows from it is concrete and powerful, and it rests on a single key observation: people who are similar to each other occupy

nearby regions of this high-dimensional space. If you and a friend share almost identical tastes, your two points will be close together. If your tastes are entirely different, your points will be far apart.

How do we measure this closeness mathematically? The most natural measure for recommendation problems is the **cosine similarity** between two vectors $\mathbf{u}$ and $\mathbf{v}$, defined as:

$$\cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{u}||\mathbf{v}|}$$

where θ is the angle between the two vectors, $\mathbf{u} \cdot \mathbf{v}$ is their dot product, and $|\mathbf{u}|$ and $|\mathbf{v}|$ are their magnitudes. A cosine similarity of 1 means the two vectors point in exactly the same direction — the two users are, in every measurable sense, identical. A cosine similarity of 0 means the vectors are perpendicular: the two users share nothing in common. A cosine similarity of -1 means they are opposite in every preference.

The entire architecture of recommendation systems, from Spotify to BBC iPlayer to Amazon, rests on this single geometric idea. Find the users nearest to you in this high-dimensional space. Look at what they chose next. Recommend it to you. This approach is called **collaborative filtering**, and its mathematics is nothing more than the geometry of vectors. How elegant that something as human as taste reduces so naturally to an angle.

3. Matrices and Hidden Structure

Collaborative filtering is powerful, but it has a limitation. It requires you to have rated things already. What if you are a new user with no history? And more fundamentally, what if the patterns that predict your behaviour are not obvious features like genre preference, but something hidden — something you never explicitly declared?

This is where a technique called**matrix factorisation**becomes important, and the mathematics behind it is worth understanding carefully.

Picture the ratings of m users across n films collected into a matrix R, where the entry R_{ij} represents the rating that user i gave to film j. In practice, most users have not rated most films, so the matrix is **sparse** — most of its entries are missing. The challenge is to fill in those missing entries, predicting what rating each user would give to each film they have not yet seen. This is precisely the problem Netflix is solving when it recommends something to you.

Matrix factorisation tackles this by assuming that the full ratings matrix R can be ap-

proximated as the product of two smaller matrices:

$$R \approx UV^T$$

where U is an $m \times k$ matrix of **user embeddings** and V is an $n \times k$ matrix of **item embeddings**, and k is a number much smaller than either m or n. Each row of U is a vector of length k representing a user, and each row of V is a vector of length k representing a film. The predicted rating of user i for film j is then simply the dot product of the i-th row of U with the j-th row of V:

$$\hat{R}_{ij} = \mathbf{u}_i \cdot \mathbf{v}_j$$

The algorithm learns the matrices U and V by minimising the difference between the predicted ratings and the actual observed ratings, over all the entries we do know. Formally, it minimises the **reconstruction error**:

$$\min_{U,V} \sum_{(i,j)\text{ observed}} (R_{ij} - \mathbf{u}_i \cdot \mathbf{v}_j)^2$$

What is remarkable about this process is what the k latent dimensions turn out to represent. Nobody tells the algorithm what they mean. Nobody labels one dimension "preference for slow-paced films" or another "tendency to enjoy films made before 1990." The algorithm uncovers these dimensions entirely by itself, because they are the most efficient mathematical explanation of the patterns it observes in the data. The latent structure it finds often corresponds to real and interpretable characteristics of human taste — characteristics that users never explicitly declared, but that are mathematically recoverable from the pattern of what they chose and what they did not.

This is a genuinely surprising result. By doing nothing more than studying which numbers appear in a matrix of ratings, the algorithm finds hidden dimensions of human personality. You reveal them simply by choosing.

4. Logistic Regression

Matrix factorisation tells us what someone is likely to enjoy.But many of the most consequential algorithmic predictions go further than this: they make a binary decision. Will this patient be readmitted to hospital? Will this loan applicant default? Will this job candidate be invited to interview? These are yes-or-no questions, and answering them requires a different mathematical tool.

The most fundamental such tool is **logistic regression**, and understanding it reveals something important about how algorithms translate numbers into decisions.

Take the problem of predicting whether a student will pass an exam, given features like hours studied and prior grades. We represent each student as a feature vector $\mathbf{x}$, and we want to output a probability between 0 and 1. A natural first attempt might be to use a linear function $\mathbf{w} \cdot \mathbf{x}$, where $\mathbf{w}$ is a vector of weights to be learned. The difficulty is that a linear function can take any value on the real line, not just values between 0 and 1.

The fix is elegant. We pass the linear function through the **sigmoid function** σ, defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

This function accepts any real number as input and squashes it into the interval $(0, 1)$. When z is very large and positive, $\sigma(z)$ approaches 1. When z is very large and negative, $\sigma(z)$ approaches 0. When $z = 0$, we get $\sigma(z) = \frac{1}{2}$. The predicted probability of a positive outcome is therefore:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

The algorithm learns the weight vector $\mathbf{w}$ by finding the values that make the observed training data as probable as possible, a principle called **maximum likelihood estimation**. This turns out to be equivalent to minimising the **cross-entropy loss**:

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{n} \sum_{i=1}^{n} \Big[y_i \log \sigma(\mathbf{w} \cdot \mathbf{x}_i) + (1 - y_i) \log\big(1 - \sigma(\mathbf{w} \cdot \mathbf{x}_i)\big) \Big]$$

It is worth pausing to understand why this loss function has the shape it does. If the true label is $y_i = 1$ and the model assigns probability close to 1, the term $y_i \log \sigma(\mathbf{w} \cdot \mathbf{x}_i)$ is close to $\log(1) = 0$: no penalty at all. If the model assigns probability close to 0 for a genuine positive, the term becomes $\log(0^+)$, which is a very large negative number, producing a very large penalty. The loss function is therefore doing exactly what we want: punishing confident wrong answers severely and rewarding confident correct answers generously.

The minimum of this loss function is found by **gradient descent**: repeatedly adjusting $\mathbf{w}$ in the direction that most steeply reduces $\mathcal{L}$, which is the direction of the negative gradient $-\nabla_{\mathbf{w}} \mathcal{L}$. Beginning from some initial guess $\mathbf{w}_0$, the update rule at each step t

is:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t)$$

where η is the **learning rate**, a small positive number controlling the size of each step. Iterate this long enough and the weights converge to values that draw a boundary in the feature space, separating students predicted to pass from those predicted to fail. Crucially, this boundary was not drawn by a human. It was discovered by mathematics, from nothing more than a table of past results.

5. Feedback and Fairness

Up to this point the algorithm is doing something that, if technically sophisticated, feels broadly reasonable. It learns patterns from data and uses them to make predictions. What makes algorithmic prediction genuinely troubling is what happens when those predictions are used to make decisions, and those decisions then shape the very data on which the algorithm is retrained.

Think about a hiring algorithm trained on a company's historical employment data. The data reflects a past in which, for whatever combination of reasons, fewer women were promoted to senior roles. The algorithm learns that certain patterns — including some that correlate with gender even without gender being an explicit feature — are predictive of promotion. It begins recommending male candidates more often. Those candidates are promoted more often, generating new training data that reinforces the original pattern. The algorithm has not merely observed a historical bias. It has converted it into a self-perpetuating mathematical rule.

This is sometimes called a **performative prediction**: a prediction that influences the outcome it predicts, creating a feedback loop. In the language of dynamical systems, we can describe the state of the system at time $t+1$ as a function of its state at time t:

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t)$$

Whether such a loop amplifies or dampens initial biases depends on the **eigenvalues** of the linearisation of f near its fixed point. If the dominant eigenvalue exceeds 1, small initial biases grow exponentially over time. If it falls below 1, they decay. In the hiring example, the feedback structure places the system firmly in the amplifying regime, and the consequences compound with each retraining cycle.

This brings us to one of the most striking results in the entire field of algorithmic

decision-making. Researchers have proposed several natural mathematical definitions of what it means for an algorithm to be **fair**. Three of the most prominent are:

Demographic parity: the proportion of positive predictions should be equal across groups.

Equalised odds: both the true positive rate and the false positive rate should be equal across groups.

Calibration: a predicted probability of p should correspond to an actual outcome rate of p for every group.

Each of these sounds reasonable in isolation. Remarkably, in 2016, researchers proved that no algorithm can simultaneously satisfy all three definitions whenever the base rates of the outcome differ between groups. This is not a practical limitation. It is a theorem. To see why, take a case where group A has a higher base rate of positive outcomes than group B. Demographic parity requires equal positive prediction rates across both groups. Calibration requires that predictions reflect true outcome rates. These two requirements directly contradict each other the moment the true outcome rates differ. Both cannot be satisfied at once.

Fairness, it turns out, is not a single mathematical object. It is a family of competing constraints, and choosing between them is not a mathematical decision. It is a political one, whether or not the people deploying the algorithm acknowledge it as such.

6. The Fundamental Limit

There is a final and perhaps the most profound mathematical limit to what any prediction algorithm can ever tell us about who we are.

Every algorithm is, at its heart, a compression of historical data. It finds the most efficient mathematical summary of past behaviour and uses that summary to anticipate future behaviour. The mathematical assumption that makes this work is called **stationarity**: the idea that the distribution generating future data is the same as the distribution that generated the training data. For many engineering problems, this is a reasonable assumption. For human beings, it is almost certainly false.

People change. They are changed by the predictions made about them, by the recommendations they receive, by the opportunities they are shown or denied. A person who is shown only a narrow band of content on a streaming platform may develop tastes shaped by that narrowness. A student assigned to a lower predicted grade band may internalise that assessment. The algorithm's prediction and the human's trajectory become entangled in ways that no static model can fully capture.

The deeper point is this. An algorithm does not observe who you are. It observes a **projection** of you — a shadow cast by your data onto the particular set of features the algorithm has been built to measure. The mathematician's term for the ideal version of this is a **sufficient statistic**: a function of the data that captures everything relevant for a particular inference. But sufficient for which inference, and relevant according to whose definition, are questions that lie entirely outside the mathematics.

The algorithm cannot know what it is missing. It cannot represent dimensions of your identity that fall outside its feature space. And the dimensions it does represent, it collapses into a vector: a point in a space that was never designed to hold the full complexity of a human life. The Netflix algorithm knows a great deal about your viewing habits. It knows nothing about why you watch what you watch, what mood you were in when you chose, what you were hoping to feel, or how you felt afterward. Those things do not appear in the data, and so they do not appear in the model.

7. Conclusion

Return, now, to that Sunday evening in front of Netflix. The recommendation that appeared on your screen was the output of a chain of mathematics: cosine similarities computed in high-dimensional space, latent dimensions uncovered by matrix factorisation, probabilities generated by a sigmoid function, weights tuned by gradient descent over millions of training examples. Each step in that chain is, individually, understandable. The mathematics is not mysterious. What is surprising is how much it captures, and equally, how much it does not.

What I find most beautiful about this journey is that it begins with something as mundane as a list of film ratings and ends at the frontier of some of the deepest questions in mathematics and philosophy: what can be inferred from data, what feedback does to a learning system, and whether it is even possible to define fairness precisely. The tools required touch linear algebra, probability theory, optimisation, dynamical systems theory, and the theory of computation. Mathematics has a remarkable habit of finding unexpected unity across seemingly unrelated territories, and the science of algorithmic prediction is one of the most striking modern examples of this tendency.

The algorithm does not know who you are. It knows the coordinates of your point in its feature space, and it makes its best inference from there. Between the vector and the person, there remains a gap that no matrix factorisation, no gradient descent, and no loss function will ever fully close. Understanding the mathematics is not enough to close that gap. But it is the necessary beginning of knowing where the gap lies, and why, in a world that increasingly delegates consequential decisions to these systems, the answer to that question matters more than ever.

References

1. Koren, Y., Bell, R., and Volinsky, C. 'Matrix Factorization Techniques for Recommender Systems.' *IEEE Computer*, 42(8), pp. 30–37, 2009. The foundational paper on matrix factorisation as applied to collaborative filtering, written by the team that won the Netflix Prize competition.

2. Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval*. Cambridge University Press, 2008. Chapter 6 provides a thorough treatment of vector space models and cosine similarity.

3. Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006. Chapters 4 and 5 give a rigorous treatment of logistic regression, maximum likelihood estimation, and gradient descent.

4. Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. Chapter 8 covers gradient descent and its variants in depth. Freely available at `deeplearningbook.org`.

5. Chouldechova, A. 'Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments.' *Big Data*, 5(2), pp. 153–163, 2017. Proves the mathematical incompatibility of calibration and equalised odds when base rates differ across groups.

6. Barocas, S., Hardt, M., and Narayanan, A. *Fairness and Machine Learning: Limitations and Opportunities*. MIT Press, 2023. The most comprehensive treatment of algorithmic fairness and its mathematical foundations. Freely available at `fairmlbook.org`.

7. Perdomo, J., Zrnic, T., Mendler-Dünner, C., and Hardt, M. 'Performative Prediction.' *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020. Formalises the concept of performative prediction and its implications for deployed machine learning systems.

8. Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (eds). *Dataset Shift in Machine Learning*. MIT Press, 2009. The standard reference on distribution shift and stationarity violations in deployed machine learning systems.

9. Fisher, R. A. 'On the Mathematical Foundations of Theoretical Statistics.' *Philo-*

sophical Transactions of the Royal Society A, 222, pp. 309–368, 1922. The original paper introducing the concept of a sufficient statistic.